Machine Learning

Written Report

Neitzel, A. W.[1,2], ChatGPT

[1] Instituto de Astrofísica e Ciências do Espaço, Universidade do Porto, CAUP, Rua das Estrelas, 4150-762 Porto, Portugal
e-mail: AndreasWNeitzel@astro.up.pt
[2] Departamento de Física e Astronomia, Faculdade de Ciências da Universidade do Porto, Rua do Campo Alegre, s/n, 4169-007 Porto, Portugal

March 19, 2023

ABSTRACT

Context. This is a report for the Machine Learning module lectured by Jarle Brinchmann for the Doctoral Programme of Astronomy in the University of Porto. Three problems were given to be solved with machine learning and statistical methods.
Aims. The aim is to solve three problem. I. Fit regression machine learning algorithms on photometric data of galaxies to infer the photometric redshift. II. Determine the probabilities associated to neutron star detection, neutron star binaries and merger events based on existing data. III. Determine the number of Gaussians that were used to create a sample of points.
Methods. Supervised and unsupervised machine learning algorithms are used to solve these problems. The supervised machine learning algorithms used were Linear Regression, Random Forest, XGBoost, k-Nearest Neighbors and Extra Trees. Gaussian Mixture Models were used for unsupervised machine learning.
Results. I. 5-fold cross-validation and computation times are provided for the five different supervised machine learning algorithms. II. Neutron star statistics are provided. III. The optimal number of Gaussians are shown according to Akaike and Bayesian Information Criteria, with a more conservative human estimate being provided for comparison.
Conclusions. I. The k-Nearest Neighbours and Extra Trees algorithms are shown to be the superior ones for photometric redshift regression. II. Frequentist and Bayesian approaches need special attention to the assumptions being made. III. It is determined that 5 Gaussians were used to generate the data.

Key words. machine learning – astronomy – neutron stars

1. Introduction

Artificial Intelligence (AI) is the field of computer programs that are capable of autonomously solving complex problems through algorithmic adaptation, a task that has historically been reserved to the ingenuity of the human mind. Within the broad field of AI lies the sub-field of Machine Learning (ML), the purpose of which is the ability to discern patterns from input data, predict outcomes and even assume decisions without being explicitly programmed to do so. In other words, it makes decisions based on the patterns that it infers from the data, as opposed to a pre-defined set of routines. This is called "learning".

The learning procedure of a machine learning algorithm is to take a set of parameters (or "features" as they are called in the field) data θ and infer a function $f = f(\theta)$ which can approximate a desired outcome. Often times this function is dependent on a set of hyperparameters γ. Hyperparameters are distinct from features in that they are tuning parameters of the algorithm in question. An example of this is the k parameter in the k-Nearest Neighbour (kNN) algorithm which, as the name implies, seeks the k nearest neighbours to a given point. The means by which this distance is calculated (Euclidean, Manhattan, among other possible means) is also a hyperparameter.

Different types of problems require different types of approaches. Suppose we have a true value y such that our objective is to find a function $f(\theta, \gamma) \approx y$. Here, y is dubbed the "label" of the features θ and the type of machine learning is considered to be supervised. When we lack access to the label then the machine learning is said to be unsupervised. In this report we shall explore both supervised and unsupervised machine learning algorithms.

Machine Learning algorithms have been first developed several decades ago, but where largely unheard of due to the hardware limitations at the time. It has only been in the past twenty years that technology has advanced enough to provide machines powerful enough to run the heavy computations demanded by the machine learning algorithms. More recently they have seen a great surge in popularity as methods advance and the technology becomes more easily accessible. Libraries such as `sklearn` and `tensorflow` provide powerful optimized algorithms, serving as an essential toolkit for any aspiring data scientist. As technology advances, so too does the data collection in science advance, data which grows to sizes and complexities far too unwieldy for any single human-being to handle alone. At that point, machine learning is invited to assist the scientist, both to handle large amounts of data and to help infer complex patterns that aren't immediately discernible.

The code for this report is available to the public in GitHub[1].

2. Photometric redshifts of galaxies

2.1. Theoretical introduction

In order to map the cosmos and to further our understanding of it, large cosmological surveys are performed to observe and catalog far away objects such as galaxies and Quasi-Stellar Objects (QSOs). Spectroscopy provides a reliable means to determine a galaxy's redshift, however the time and resources required for this practice becomes unfeasible for large surveys and as such more economic options need to be explored. This report explores the usage of machine learning on photometric data to infer the photometric redshift.

2.2. Method

2.2.1. Data

The photometric redshift z_{phot} shall be our fitting function $f(\theta)$, with the features θ coming in the form of apparent magnitudes and colors in the `ugriz` photometric system of several thousands of galaxies. More specifically:

- mag_r: The apparent magnitude in the r band
- u-g: The color u-g
- g-r: The color g-r
- r-i: The color r-i
- i-z: The color i-z

The data is provided in two datasets, A and B, which also contain the spectroscopic redshift z_{spec} of each galaxy. We shall take the spectroscopic redshift and the true redshift and therefore it shall be the label of the supervised machine learning. Dataset A will be used for training and testing purposes, whereas dataset B exists solely for validation purposes. The terms test and validation are often interchanged in the literature, but in the context of this report it is meant to say that the machine learning algorithm learns with dataset A and never ever gets to see a single datum from dataset B.

2.2.2. Feature extraction

Feature extraction is the process of transforming raw input data into a set of features that can be used to train a machine learning model. The goal of feature extraction is to create a set of informative and discriminative features that capture the essential characteristics of the data, making it easier for the model to learn from the data and make accurate predictions. There are several considerations that should be made when carrying out feature extraction:

- Relevance: Features should be relevant to the task at hand. Only include features that are expected to have a strong impact on the output, and avoid including features that are not relevant.
- Discrimination: Features should be able to distinguish between different classes or groups in the data. Features that are too similar for different classes can lead to poor performance.
- Redundancy: Avoid including features that are highly correlated or redundant with other features, as this can lead to overfitting and decreased model performance.
- Scale: Make sure that the features are on a similar scale, as features on different scales can negatively impact model performance.
- Noise: Remove features that are likely to contain noise, as this can negatively impact model performance.
- Complexity: Keep the feature extraction process simple and interpretable, as more complex feature extraction techniques may lead to overfitting and decreased model performance.

Beyond the basics of feature extraction, understanding of the data is key to its adequate treatment. The data is not linear, not strictly Gaussian distributed and outliers are not immediately obvious. Furthermore, it can be inferred from physical principles that multiple processes can be responsible for the same phenomena. The apparent magnitude in the r band is expected to decrease as redshift increases, since high redshifts means that the objects are farther away and therefore dimmer to us. It is also true, however, that different types of galaxies will present different mag_r based on their constitution. Yet another effect to take into consideration is the wavelength shifting effect caused by redshift, which can alter the mag_r. This last phenomena can also be responsible to aberrant behaviors in colors such as g-r, breaking their monotonic behavior and causing them to evolve in the opposite direction

Due to these considerations the preprocessing was kept conservative in order to not risk eliminating data that may be important in characterizing the system. The preprocessing procedure undertaken is as follows:

1. Compare the histograms of datasets A and B to ensure that they both contain identical proportions in the features and labels
2. Apply conservative cuts to tighten the minimum and maximum ranges of the data
3. Scale the features according to their minimum and maximum value such that they range from 0 to 1
4. Perform Principal Component Analysis (PCA) to explore the possibility of dimensionality reduction

In the end no dimensionality reduction was performed since it did not lead to visible improvements in the method. The choice was made to keep the same dimensions with the assumption that all features have important contributions to the output.

2.2.3. Evaluation

The error $E(\theta)$ of the algorithms is defined as:

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

Each algorithm is scored by the mean of their by 5-fold cross-validation of the error defined by Equation 1 and this is then compared with the error of the model trained and tested on database A fitted to database B. If the latter is significantly greater than the former then it is said that the model is overfitted.

2.3. Algorithms

2.3.1. Linear Regression

A series of different machine learning regression algorithms from the `sklearn` library in `Python` are tested. As a preliminary

[1] https://github.com/AndreasWNeitzel/ML2023-Project

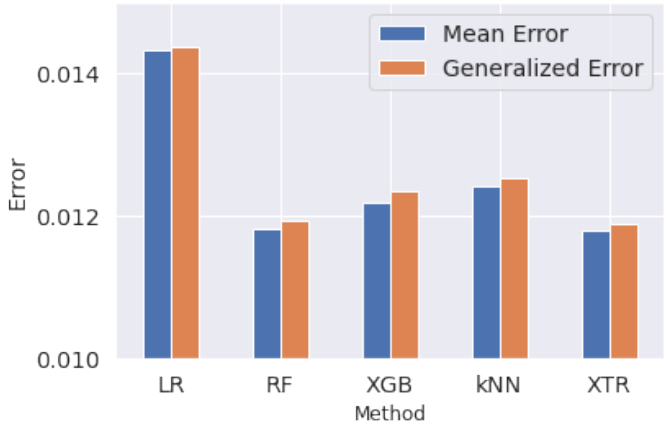

Fig. 1. Barplot of the mean 5-fold cross-validation error and the generalized error.

step, Linear Regression (LR) was performed on un-processed data first in order to verify the accuracy that this method provides. Ridge and LASSO Regression, two popular and more complex algorithms, were not chosen since their power comes in the form of trading accuracy in exchange for a decreased risk in overfitting. Changing their hyperparameters can only results in an accuracy equal to or worse than LR, thereby being redundant for this purpose. LR provided a 5-fold cross-validation error of $E(\theta) = 0.01458 \pm 0.00034$ and a generalized error of $E(\theta) = 0.01463$, thereby showing no signs of overfitting.

Motivated by the goal of better results, ideally $E(\theta) < 0.01$, conservative data preprocessing was performed and non-linear machine learning algorithms are tested. These are:

- LR: Linear Regression (second trial)
- RF: Random Forest
- XGB: XGBoost
- kNN: k-Nearest Neighbours
- XTR: Extra Trees

2.4. Results

The mean error E_A and the generalized error E_B comparisons are provided in Figure 1. These values, alongisde the total computation time, are given in Table 1. In terms of error, all algorithms are robust to overfitting, have negligible variance and do not differ greatly from one another, with the exception of LR which underperforms in comparison to the rest. In terms of computation time, tree based algorithms are computationally expensive, taking much longer than the others but achieve slightly better results. Two algorithms therefore stand out in terms of their efficiency and speed. These are kNN and XTR, both of which have the potential to outperform RF and XGB in the determination of photometric redshift.

3. The likelihood of gravitational wave neutron stars

Given a dataset containing the masses of 68 neutron stars, it is asked to estimate the probability of finding a neutron star with mass $M \geq 1.8\mathrm{M}_\odot$. There are three ways to go about this. First is the frequentist approach, which is the divide the number of valid events by the total number of events:

$$P(M \geq 1.8\mathrm{M}_\odot) = \frac{\#\ \mathrm{Stars}(M \geq 1.8\mathrm{M}_\odot)}{\#\ \mathrm{All\ stars}} \tag{2}$$

Method	$E_A(\theta)$	$E_B(\theta)$	Time (s)
LR	0.0143±0.0003	0.0144	0.059
RF	0.0118±0.0005	0.0119	49.979
XGB	0.0122±0.0005	0.0123	48.361
kNN	0.0124±0.0004	0.0125	1.762
XTR	0.0118±0.0005	0.0119	13.441

Table 1. Errors and total computation time of each algorithm.

Second is the modified frequentist approach, which is to add 1 to the numerator and denominator. This is done to argue against the idea of probabilities being zero for an event that isn't necessarily impossible.

$$P(M \geq 1.8\mathrm{M}_\odot) = \frac{\#\ \mathrm{Stars}(M \geq 1.8\mathrm{M}_\odot) + 1}{\#\ \mathrm{All\ stars} + 1} \tag{3}$$

Third is the Bayesian approach, which is to take the area of the normalized probability density function of the neutron star mass that corresponds to $M \geq 1.8\mathrm{M}_\odot$:

$$P(M \geq 1.8\mathrm{M}_\odot) = \frac{\int_{1.8\mathrm{M}_\odot}^{+\infty} \rho(M) dM}{\int_{-\infty}^{+\infty} \rho(M) dM} \tag{4}$$

Each method has its own caveat. Figure 2 shows the histogram and Gaussian Kernel Density Estimate for the mass distribution of neutron stars. Evidently we do not have enough data to make rigorous arguments. The Bayesian method, for example, requires us to be in possession of a probability density function, ρ. Since our data is insufficient, we need to construct one from the data. Either we can use a kernel density estimate or we can generate a PDF by adding the probability density functions of the mass of each neutron star and then normalizing it. The former method depends on the assumptions made for the kernel and the bandwidth, whereas the latter assumes we have access to the probability density functions of the mass of each neutron star and that this would be sufficient. Unfortunately we are only in possession of asymmetric uncertainties. Furthermore, the ranges covered by Gaussians extend between $[-\infty, +\infty]$ $\mathrm{M}_\odot$. Neutron stars cannot have negative masses nor can these be infinite. Nonetheless, concessions must be made in order to obtain a rough estimate.

If we want to estimate mass ranges we can follow the Bayesian procedure. In Table 2 we have the frequentist and Bayesian probabilities. The Bayesian sum probabilities were obtained by summing over the set of Gaussian distributions generated by assuming a mean μ equal to the tabulated mass and a standard deviation σ equal to the mean of the upper and low uncertainties around the mass, divided by the integral of the resulting function in order to normalize it, as described by the following equation:

$$\rho(M) = \frac{\sum_i \exp\left\{-\frac{(\mu - x)^2}{2\sigma^2}\right\}}{\int_{-\infty}^{\infty} \sum_i \exp\left\{-\frac{(\mu - x)^2}{2\sigma^2}\right\} dx} \tag{5}$$

The computed distributions are illustrated in Figures 3 and 4.

4. Misterious peaks

A 1D dataset was provided, containing in it points that were drawn from a distribution with an unknown number of Gaus-

Method	$P(M \geq 1.8M_\odot)$	$P(1.36 \leq M \leq 2.26M_\odot)$	$P(0.86 \leq M \leq 1.36M_\odot)$
F	0.16	-	-
FM	0.17	-	-
BSum	0.20	0.56	0.34
BKDE	0.16	0.59	0.36

Table 2. Probabilities of different masses assuming different strategies. F: Frequentist. FM: Frequentist modified. BSum: Bayesian sum of estimated Gaussian probability density functions. BKDE: Bayesian Kernel Density Estimate.

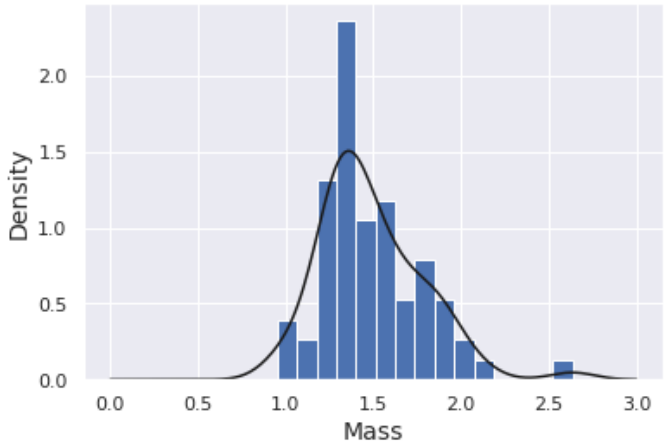

Fig. 2. Histogram and Gaussian Kernel Density Estimate for the mass distribution of neutron stars.

sians. The objective is to discover how many Gaussians there are in this distribution. This was done by performing a Gaussian Mixture Model (GMM) and varying the hyperparameter of the number of Gaussians. The Akaike Information Criteria (AIC) and the Bayesian Information Criteria (BIC) were deduced for anywhere between 2 and 14 Gaussians. The evolution of these criteria is shown in Figure 6.

AIC determined that 12 Gaussians is the ideal number, whereas BIC determined that 7 is the ideal number. The argument can be made that, by analyzing the behavior of the score curves, the most conservative estimate would be 5 Gaussians, since there is a sharp drop at $n = 5$ Gaussians that then more or less plateaus afterward. The histograms and GMM fits are plotted for AIC, BIC and the conservative 5 Gaussian estimate in Figures 6, 7 and 8 respectively.

Figure 6 has 12 peaks and appears overfitted. Figure 7 has 7 peaks and appears to very realistically fit the histogram. Figure 8 has 5 peaks and look less tight, but is nonetheless quite powerful at describing the histogram despite the small amount of peaks. As a result, it is concluded that 5 Gaussians is the optimal guess.

5. Special thanks

Another use of machine learning is to serve as an assistant. `ChatGPT` was consulted multilpe times throughout the development of this report in order to help write code and to give inspiration for writing material. Although `ChatGPT` is non-human, it can be thought of as the personification of millions of real people who have devoted countless quality information that the device was then able to train itself on. As such, by recognizing `ChatGPT` as a co-author of this report I recognize the contributions of millions of other people towards the advancement of human knowledge.

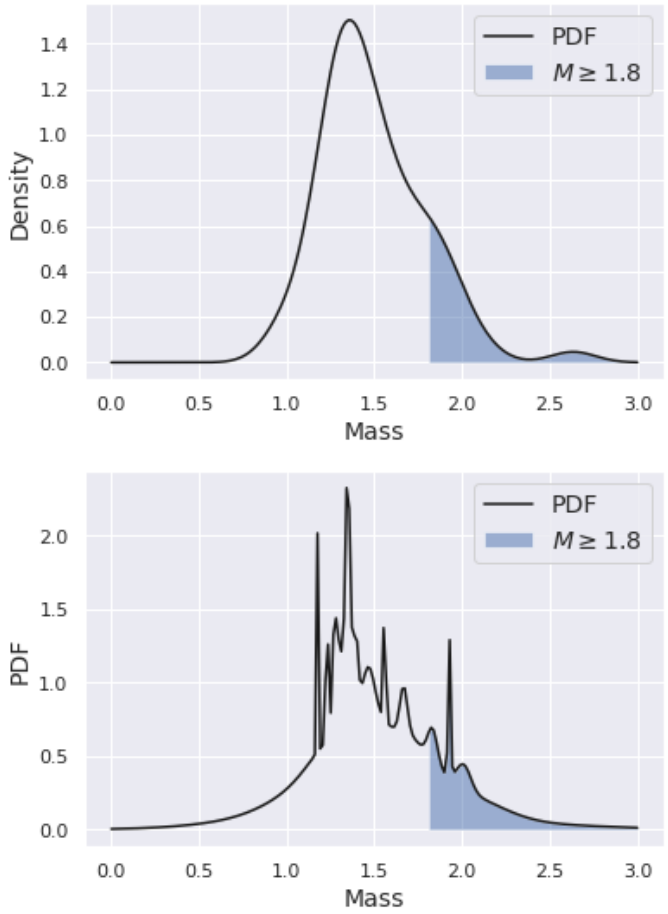

Fig. 3. Mass distribution of neutron stars. Shaded areas represent probabilities. Top: sum of probability density functions. Bottom: kernel density estimate.

References

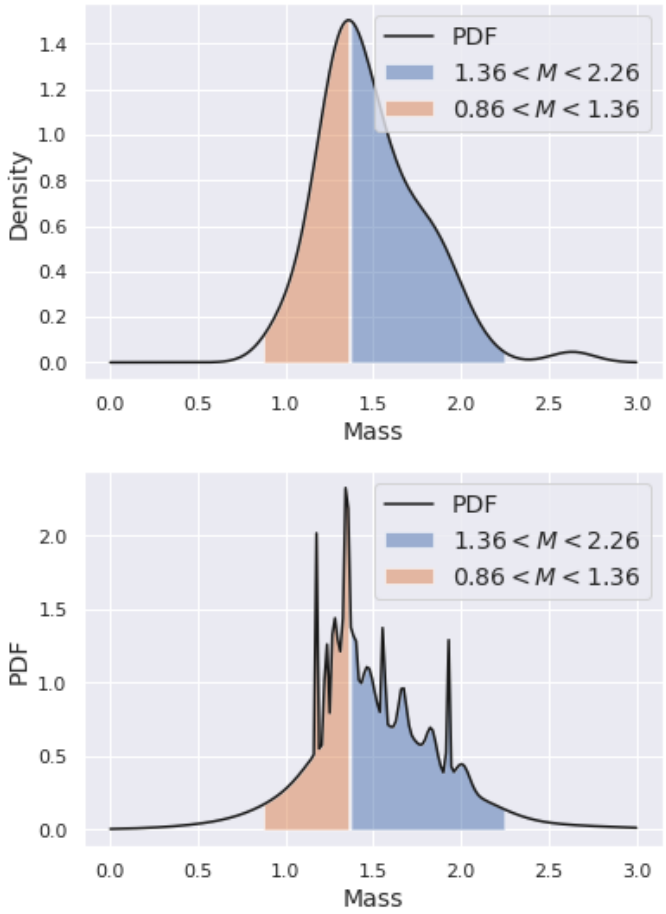

Fig. 4. Mass distribution of neutron stars. Shaded areas represent probabilities. Top: sum of probability density functions. Bottom: kernel density estimate.

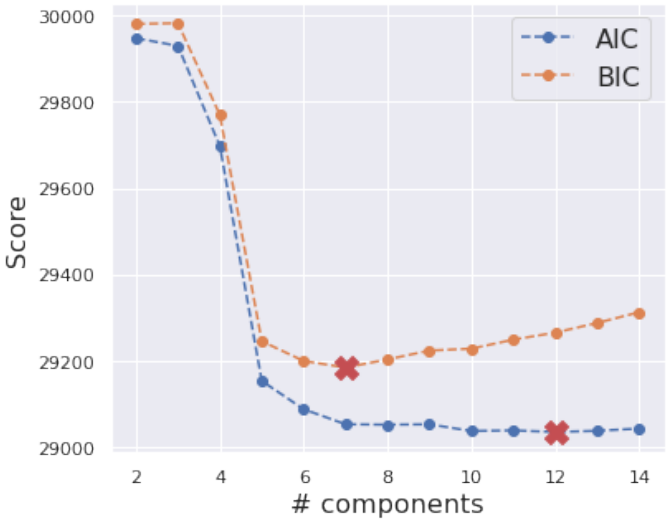

Fig. 5. Scores of AIC and BIC as a function of the number of Gaussians.

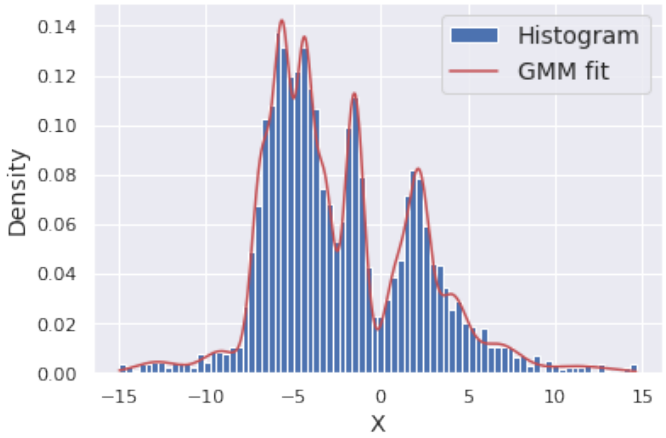

Fig. 6. Histogram of the data with the optimal number of peaks as determined by AIC.

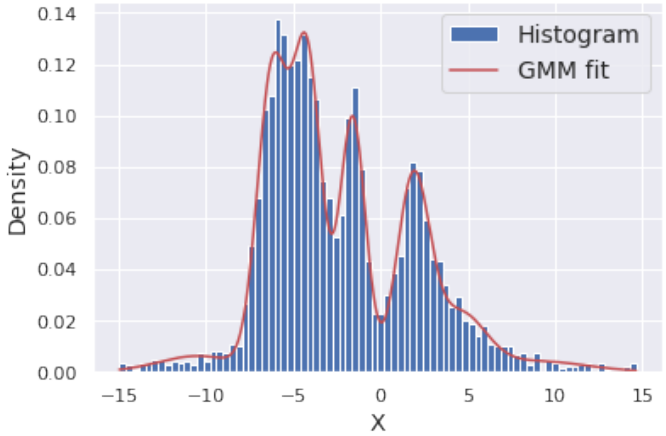

Fig. 7. Histogram of the data with the optimal number of peaks as determined by BIC.

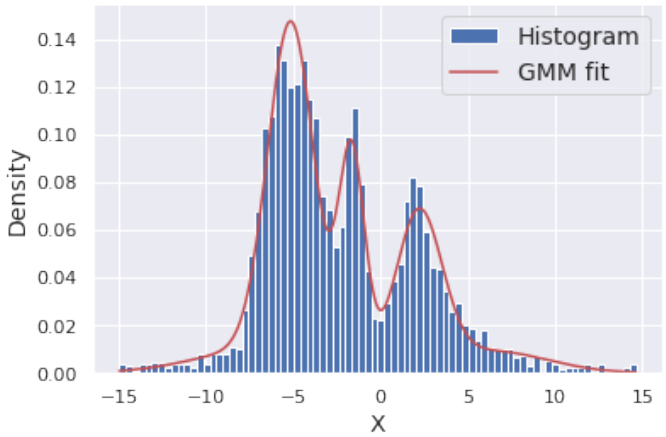

Fig. 8. Histogram of the data with a conservative estimate of 5 peaks.