Analysis of selected algorithms for the classification of space objects[*]

Radosław Jędrzejczyk[1,*,†], Katarzyna Kłeczek[1,†]

[1]*Faculty of Applied Mathematics, Silesian University of Technology, Kaszubska 23, 44100 Gliwice, POLAND*

Abstract
Along with the rise of available astronomical data, captured from numerous facilities from around the world, a need for faster and more sophisticated data analysis methods emerges. Data captures from numerous observation of large quantities of object in the sky can reach large volumes very quickly, making it impossible for scientist to analyse by hand. This rises the need for fast and reliable automated methods of data processing, which can be found in computer science research. Leveraging algorithms used in different areas of research is crucial for processing information about celestial bodies. In this work, we apply machine learning methods from computer science domain into an astronomy problem. We lay out three different machine learning algorithms, along with their inner workings, and show how they can be applied to astronomy problems. We show how those algorithms can be used to speed up processing of large volumes of data, and how they can help scientists in classification of celestial bodies. We investigate how each algorithm performs and try to find the best performing one in the problem of classification of different objects, based on their characteristics.

Keywords
knn, naive bayes, decision trees

1. Introduction

In modern astronomy, increasing number of data is becoming an ever-growing problem and opportunity. Formulation and validation of many theories require scientists to go through huge databases, which have become impossible to do by hand. At the same time, increasing capabilities of earth-based observatories and space telescopes are providing us with many sky surveys containing petabytes of quality data [1, 2]. This data-intensive situation encourages the investigation of new methodologies, big data tools and techniques, therefore providing a great environment for astroinformatics development [3].

Machine learning has a significant impact on this new reality [4, 5, 6]. It provides many tools that can be used to swiftly classify huge amounts of data, which we will try to explore in this paper. We will go through algorithms such as Decision Tree [7], Naive Bayes [8] and K-Nearest Neighbors [9] and analyse their accuracy to distinguish between different objects.

[*]*IVUS2024: Information Society and University Studies 2024, May 17, Kaunas, Lithuania*
[1,*] Corresponding author
[†] These author contributed equally.

rj303537@student.polsl.pl (R. Jędrzejczyk); katakle387@student.polsl.pl (K. Kłeczek)
0009-0008-5295-0143 (R. Jędrzejczyk); 0009-0009-3366-8289 (K. Kłeczek)

2. Methodology

In the beginning, we will need to transform our data into a convenient form. In the case of the non-numerical data, we will simply map it to one by associating separate numbers for each value. On the other hand, numerical data will be rescaled using min-max normalization.

We will compare performance of different algorithms, given the task of classification of stellar objects. For the comparison, we have chosen:

- KNN (K-Nearest Neighbors) classification.
- Decision tree model.
- Naive Bayes.

Mathematical Model for K-Nearest Neighbors (K-NN)

If we assume we have a training dataset consisting of N data points:

$$\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\} \tag{1}$$

where x_i is the feature vector for the i-th point, and y_i is the class label (for classification) or value (for regression).

Then we can calculate a distance metric, typically using the Euclidean distance d between two points x and z defined as:

$$d(x, z) = \sqrt{\sum_{j=1}^{m} (x_j - z_j)^2} \tag{2}$$

where x and z are feature vectors of dimension m.

To classify a new point x, we compute the distances between x and all points in the training set, then select K nearest neighbours and assign a class label based on the majority.

The parameter K is a crucial hyperparameter in the KNN algorithm. A small K can lead to overfitting, while a large K can lead to underfitting. The optimal value of K is often selected using cross-validation methods.

```
Algorithm 1: KNN Algorithm
  Data: Training data x_t, training classes y_t, class to be classified label_searched, test data x, algorithm constant 'k'
        neighbours_number
  Result: Predictions
1 for each tested in x do
2     distances ← distance between tested and each in x_t;
3     closest_indexes ← indexes of neighbours_number closest neighbours;
4     labels ← classes of closest neighbours;
5     result ← dominating label in labels;
6     Add result to prediction list;
7 Create data structure with predictions, by choosing indexes of the test data; return Predictions
```

Mathematical Model for Decision Tree

If we assume we have a training dataset consisting of N data points:

$$\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\} \tag{3}$$

where x_i is the feature vector for the i-th point, and y_i is the class label (for classification) or value (for regression). Then a decision tree is a tree-like model where internal nodes represents a test on a feature, branches represents outcomes of those tests and leaf node represents a class label.

To build a decision tree, we recursively split the data at each node. The choice of split is based on a criterion that maximizes the separation of the classes or reduces the prediction error. Common criteria include:

Gini Index:

$$Gini(D) = 1 - \sum_{k=1}^{K} p_k^2 \tag{4}$$

where p_k is the proportion of instances of class k in the dataset D.

Information Gain:

$$IG(D, A) = Entropy(D) - \sum_{v \in \text{Values}(A)} \frac{|D_v|}{|D|} Entropy(D_v) \tag{5}$$

where $Entropy(D)$ is given by:

$$Entropy(D) = -\sum_{k=1}^{K} p_k \log_2 p_k \tag{6}$$

and D_v is the subset of D where attribute A has value v.

Mean Squared Error (MSE):

$$MSE(D) = \frac{1}{|D|} \sum_{i=1}^{|D|} (y_i - \bar{y})^2 \tag{7}$$

where $\bar{y}$ is the mean of the values in the dataset D.

Mathematical Model for Naive Bayes

Assume we have a training dataset consisting of N data points:

$$\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\} \tag{8}$$

where $x_i = (x_{i1}, x_{i2}, \ldots, x_{im})$ is the feature vector for the i-th point, and y_i is the class label from a set of classes $\{C_1, C_2, \ldots, C_K\}$.

The Naive Bayes algorithm is based on Bayes' Theorem:

$$P(C_k \mid x) = \frac{P(x \mid C_k) P(C_k)}{P(x)} \tag{9}$$

Algorithm 2: Decision Tree Algorithm

```
Data: Training dataset $x_i$, set of attributes $A$, class attribute $y_i$
Result: Decision tree
1  begin
2      Create a root node $t$;
3      if all instances in $x_i$ belong to the same class $y$ then
4          Label $t$ as leaf node with class $y_i$
5      else
6          if $A$ is empty then
7              Label $t$ as leaf node with majority class in $x_i$
8          Choose attribute $a$ from $A$ that best classifies instances in $D$;
9          Label node $t$ as attribute $a$;
10         Remove $a$ from $A$;
11         for each value $v$ of $a$ do
12             Add a branch to $t$ corresponding to $v$;
13             Let $x$ be the subset of instances in $x_i$ with value $v$ for attribute $a$;
14             if $x$ is empty then
15                 Label the corresponding branch with the majority class in $x_i$;
16             Label the corresponding branch using Decision Tree Algorithm($x$, $A$, $y_i$);
17     ReturnDecision tree ;
```

where: $P(C_k \mid x)$ is the posterior probability of class C_k given feature vector x, $P(x \mid C_k)$ is the likelihood of feature vector x given class C_k, $P(C_k)$ is the prior probability of class C_k, $P(x)$ is the evidence or marginal likelihood of feature vector x.

The "naive" assumption is that the features are conditionally independent given the class label:

$$P(x \mid C_k) = \prod_{j=1}^{m} P(x_j \mid C_k) \tag{10}$$

The goal is to predict the class label $y^\wedge$ for a new instance x by maximizing the posterior probability:

$$\hat{y} = \arg\max_{C_k} P(C_k \mid x) \tag{11}$$

Using Bayes' Theorem and the naive assumption, we can write:

$$\hat{y} = \arg\max_{C_k} \left(P(C_k) \prod_{j=1}^{m} P(x_j \mid C_k) \right) \tag{12}$$

The probabilities $P(C_k)$ and $P(x_j \mid C_k)$ need to be estimated from the training data and the prior probability of class C_k is estimated as:

$$P(C_k) = \frac{N_k}{N} \tag{13}$$

where N_k is the number of instances in class C_k. For continuous features, a common approach is to assume a Gaussian distribution:

$$P(x_j \mid C_k) = \frac{1}{\sqrt{2\pi\sigma_{jk}^2}} \exp\left(-\frac{(x_j - \mu_{jk})^2}{2\sigma_{jk}^2}\right) \tag{14}$$

where μ_{jk} and σ_{jk}^2 are the mean and variance of the feature x_j for class C_k.

Algorithm 3: Naive Bayes Algorithm

```
Data: Training dataset $x$, class attribute $y$;
Result: Classifier model
1 begin
2     for each class $y$ in $y$; do
3         Calculate prior probability $P(y)$;
4         for each attribute $a$ do
5             Calculate conditional probability $P(a|y)$;
6     return Classifier model;
```

Additionally, we will look for the best number of neighbours for KNN classifier. We will use a few libraries to handle our operations: Sklearn [10]- will provide us with algorithm implementations, saving us a lot of time and ensuring we will be able to go through relatively big databases in reasonable time. Pandas [11] - will provide us with data structure (DataFrame). Seaborn [12] and Matplotlib [13]- will be used for visualizations, graphs, etc.

In order to find the best constant for KNN, we will launch classification in a simple loop, looking for the best solution. Generally speaking, when this number will increase our accuracy should decrease, therefore this approach is reasonable and should not take too much time.

Algorithm 4: Loop for finding the best constant for KNN

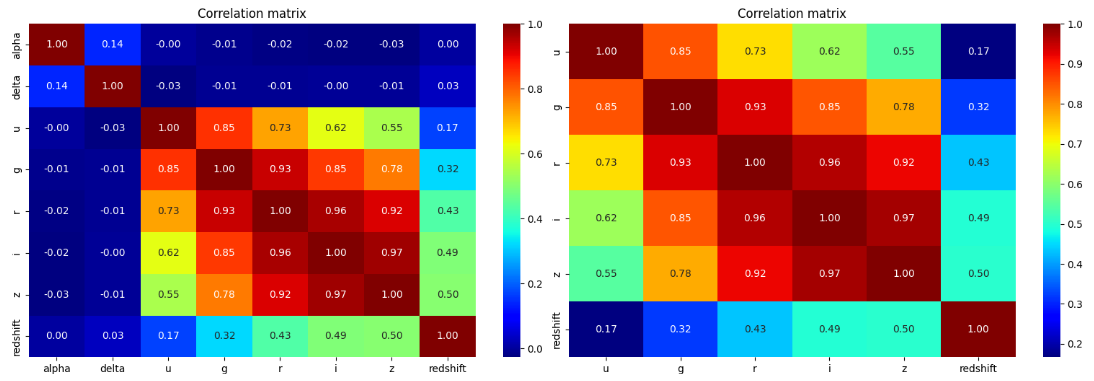
```
Data: Training data $x\_t$, training classes $y\_t$, class to be classified label_searched, test data $x$
Result: Best constant
1 begin
2     Feed KNN algorithm with $x\_t$ and $y\_t$ data.;
3     Set KNN constant as 1.;
4     while KNN constant is lower than significant number do
5         Classify $x$ using KNN.;
6         Check accuracy $n$ and add it to the list $N$.;
7         Increase KNN constant by 1.;
8     $n_{max}$ = index of biggest $n$ in $N$.;
9     return $n_{max}$
```

In the end, we present the confusion matrix for each of our solutions, and we will consider only two metrics:

- Accuracy (Equation 15) - to measure how many correct classifications we get.

- False categorization - in order to check if any of the classes are more often confused with others.

$$\text{Accuracy} = \frac{\text{Correct classifications}}{\text{All classifications}} \tag{15}$$



(a) Correlation matrix for SDSS-IV data.

(b) Final correlation matrix for SDSS-IV data.

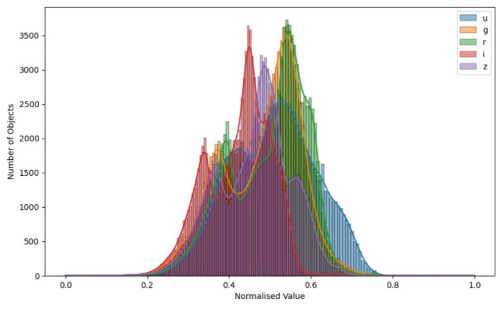
Figure 1: Comparison of correlation matrices for SDSS-IV data.

3. Experiments

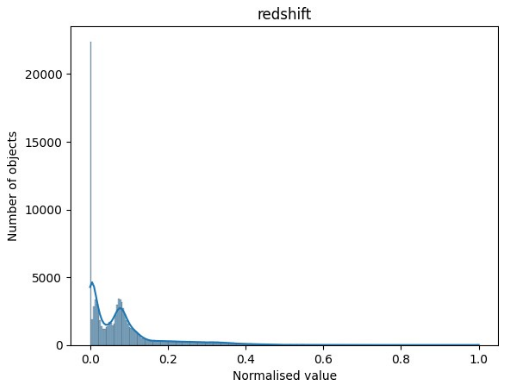
For our dataset, we have chosen data from Sloan Digital Sky Survey DR17 [14] (it was accessed from [15]). Which was the fourth phase of the Sloan Digital Sky Survey (we will call it SDSS-IV from now on). It contains 100000 observations, each containing (qouting [16]):

- obj_ID = Object Identifier, the unique value that identifies the object in the image catalogue used by the CAS
- alpha = Right Ascension angle (at J2000 epoch)
- delta = Declination angle (at J2000 epoch)
- u = Ultraviolet filter in the photometric system
- g = Green filter in the photometric system
- r = Red filter in the photometric system
- i = Near Infrared filter in the photometric system
- z = Infrared filter in the photometric system
- run_ID = Run Number used to identify the specific scan
- rereun_ID = Rerun Number to specify how the image was processed
- cam_col = Camera column to identify the scanline within the run
- field_ID = Field number to identify each field
- spec_obj_ID = Unique ID used for optical spectroscopic objects (this means that 2 different observations with the same spec_obj_ID must share the output class)
- class = object class (galaxy, star or quasar object)

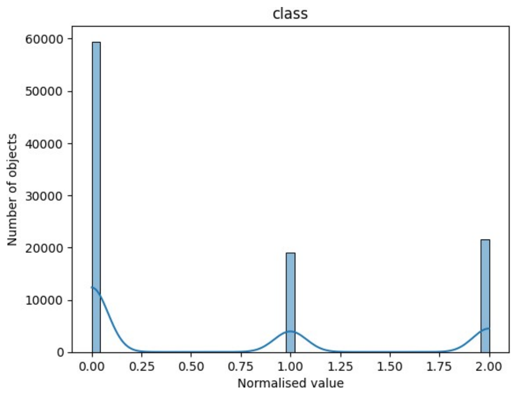
- redshift = redshift value based on the increase in wavelength
- plate = plate ID, identifies each plate in SDSS
- MJD = Modified Julian Date, used to indicate when a given piece of SDSS data was taken
- fiber_ID = fiber ID that identifies the fiber that pointed the light at the focal plane in each observation



(a) Histograms for all normalized data, excluding redshift.



(b) Histogram for normalised redshift.



(c) Number of different objects (0 - galaxies, 1 - QSOs, 2 - stars).

Figure 2: Various histograms showing data distributions and object counts.

Some of that information will not be used for our classification, as they are contained in SDSS-IV for cataloguing purposes (such as object identifiers). We will focus on: coordinates alpha and delta; data from filtered channels u, g, r, i and z; class, which is the aim of our classification efforts.

After mapping and normalising our data in ultraviolet, green and infrared presented strange pattern, where basically all data is accumulated near value 1.0. Upon further inspection it turns out that one of the observed objects have some abnormal values (equals to -9999), we will remove it from our dataset and then proceed.

Now we will have a look at the correlation matrix (figure 1a) and address some of the relations:

- Coordinates have neutral relations with all the other data.
- Ultraviolet and green relation- green light is a part of spectrum of many stars similar to the Sun (G-type main-sequence stars). Those stars also happens to emit significant part of their radiation as ultraviolet. An additional effect, that can also explain moderate relation with infrared and near infrared light is absorption and re-emission of different by interstellar gas, which then re-emits in those wavelengths (heat radiation) [17].
- Infrared, near-infrared and red data have strong relation - red stars are typically colder, but they still emit a lot of infrared radiation. Additional factor - absorption and re-emission of light was mentioned above.
- Moderate relation of red, near infrared and infrared light with redshift can by explained by many objects detected as red having their colour shifted due to phenomenons as Doppler effect. This relation might be absent from other detectors, as light of stars different from infrared might have been cut off by stardust or shifted strong enough to not be detected at all [17].

In general, it is easy to notice strong relations with red and infrared light. This phenomenon might be related to extinction of light in the space, which is more explicit for shorter wavelengths. The coordinates of our objects are mostly related to each other (but it is still very weak relation). It also has a pure neutral relation with most of the data from detectors, therefore we are going to drop this one. Our final correlation matrix is shown for the sake of clarity in figure 1b.

Additionally, we will provide histograms for SDSS-IV data, we will plot them on to one histogram, excluding redshift, which will be shown separately for clarity (figures 2a and 2b). We will also have a look at a number of each of the individual objects in our data (figure 2c) - we can notice significant dominance of galaxies. Galaxies and quasars are similar in number, with a small margin for stars.

We will split our data with at train and test set with ratio of 0.2. After running the calculation mentioned in chapter before, we get:

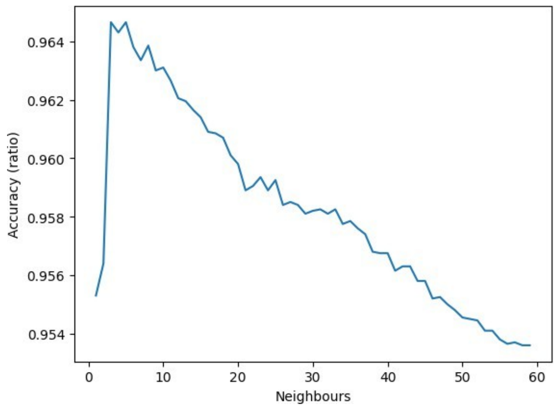
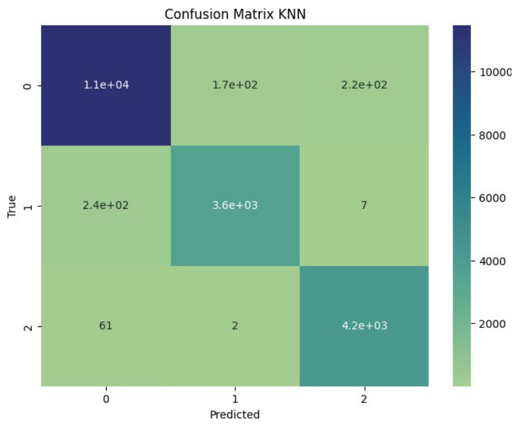
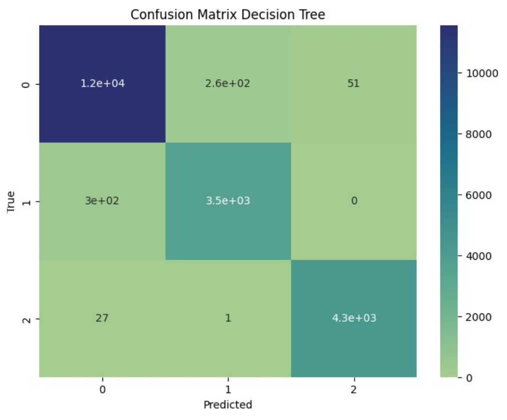


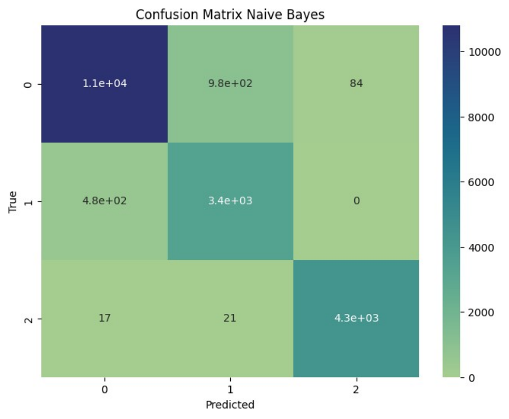
Figure 3: Accuracy of KNN.



(a) Confusion matrix for KNN



(b) Confusion matrix for ID3



(c) Confusion matrix for Naive Bayes

Figure 4: Comparison of confusion matrices for KNN, ID3, and Naive Bayes Algorithms (0 - galaxies, 1 - QSOs, 2 - stars).

- For KNN we get 96.465% accuracy, which was best for numbers of neighbours equal to 3 as shown in figure 3 with confusion matrix as in figure 4a.
- Decision tree have achieved 96.78% accuracy (confusion matrix in figure 4b).
- Naive Bayes have achieved the lowest accuracy of 92.11% (confusion matrix in figure 4c)

4. Conclusion

Behaviour of KNN accuracy was, as expected, decreasing with relation to its constant. On the other hand, all the analysed algorithms achieved good accuracy (above 90%). Bayes algorithms turned out to have had some problems distinguishing between galaxies and quasars (almost 1000 wrongly classified galaxies), although two others algorithms also struggled there. KNN seems to deal the best with this problem, recognising even so slightly more QSO objects than others but have more mismatches, recognising some of the galaxies as stars. None of the algorithms have any problems recognising stars and rarely ever mismatches them.

References

[1] B. Becker, M. Vaccari, M. Prescott, T. Grobler, Cnn architecture comparison for radio galaxy classification, Monthly Notices of the Royal Astronomical Society 503 (2021) 1828–1846.

[2] A. Iess, E. Cuoco, F. Morawski, C. Nicolaou, O. Lahav, Lstm and cnn application for core-collapse supernova search in gravitational wave real data, Astronomy & Astrophysics 669 (2023) A42.

[3] Y. Z. Yanxia Zhang, Astronomy in the big data era, Data Science Journal (2015). doi:10.5334/dsj-2015-011.

[4] L. Xu, J. Wang, X. Li, F. Cai, Y. Tao, T. A. Gulliver, Performance analysis and prediction for mobile internet-of-things (iot) networks: a cnn approach, IEEE Internet of Things Journal 8 (2021) 13355–13366.

[5] M. Woźniak, J. Szczotka, A. Sikora, A. Zielonka, Fuzzy logic type-2 intelligent moisture control system, Expert Systems with Applications 238 (2024) 121581.

[6] D. Połap, K. Kęsik, A. Winnicka, M. Woźniak, Strengthening the perception of the virtual worlds in a virtual reality environment, ISA transactions 102 (2020) 397–406.

[7] M. Woźniak, D. Połap, Soft trees with neural components as image-processing technique for archeological excavations, Personal and Ubiquitous Computing 24 (2020) 363–375.

[8] I. Wickramasinghe, H. Kalutarage, Naive bayes: applications, variations and vulnerabilities: a review of literature with code snippets for implementation, Soft Computing 25 (2021) 2277–2293.

[9] N. Ukey, Z. Yang, B. Li, G. Zhang, Y. Hu, W. Zhang, Survey on exact knn queries over high-dimensional data space, Sensors 23 (2023) 629.

[10] Package of scikit-learn, https://scikit-learn.org/stable/, 2024. Accessed: 2024-05-17.

[11] Pandas library, https://pandas.pydata.org/, 2024. Accessed: 2024-05-17.

[12] Seaborn library, https://seaborn.pydata.org/, 2024. Accessed: 2024-05-17.

[13] Matplotlib library, https://matplotlib.org/, 2023. Accessed: 2024-05-17.

[14] Original source of data release 17 from sloan digital sky survey, https://www.sdss4.org/dr17/, 2022. Accessed: 2024-05-18.

[15] Source of our data at kaggle.com, https://www.kaggle.com/datasets/fedesoriano/stellar-classification-dataset-sdss17, 2022. Accessed: 2024-03-29.

[16] Fedesoriano, Stellar classification dataset - sdss17, 2022. Retrieved May 18, 2024 from https://www.kaggle.com/fedesoriano/stellar-classification-dataset-sdss17.

[17] Article about infrared imaging, https://www.skyatnightmagazine.com/space-science/infrared-astronomy, 2024. Accessed: 2024-05-18.