Black Sea Journal of Engineering and Science
doi: 10.34248/bsengineering.1517904

Open Access Journal
e-ISSN: 2619 – 8991

Research Article
Volume 7 - Issue 5: 960-970 / September 2024

PERFORMANCE COMPARISON OF SUPERVISED MACHINE LEARNING METHODS IN CLASSIFYING CELESTIAL OBJECTS

Maide Feyza ER[1]*, Turgay Tugay BİLGİN[2]

[1]Bandirma Onyedi Eylul University, Faculty of Engineering and Natural Sciences, Department of Software Engineering, 10200, Balıkesir, Türkiye

[2]Bursa Technical University, Faculty of Engineering and Natural Sciences, Department of Computer Engineering, 16350, Bursa, Türkiye

Abstract: In recent times, astronomy has entered a new era with rapidly growing data sources and advanced observation techniques. The construction of powerful telescopes has enabled the collection of spectral data from millions of celestial objects. However, the increasing number and variety of data have made it challenging to categorize these celestial objects. This study employs machine learning methods to address the fundamental problem of classifying stars, galaxies, and quasars in astronomy. The dataset underwent detailed preprocessing to identify effective features for classification. KNIME Analytics Platform was used for data analysis and visualization, facilitating rapid and efficient data analysis through its drag-and-drop interface. Among the machine learning methods used in our study—Decision Trees, Random Forest, and Naive Bayes—the highest accuracy rate of 97.86% was achieved with the Random Forest model. Notably, despite its lower overall performance compared to other models, the Naive Bayes classifier exhibited superior performance in distinguishing the STAR class, which is one of the study's interesting findings. Future studies aim to enhance model accuracy by using larger and more diverse datasets and exploring different machine learning algorithms. Additionally, the impact of deep learning methods on classification performance will be investigated.

Keywords: Machine learning, Classification, Decision tree, Naive Bayes, Random forest

***Sorumlu yazar (Corresponding author):** Bandirma Onyedi Eylul University, Faculty of Engineering and Natural Sciences, Department of Software Engineering, 10200, Balıkesir, Türkiye
E mail: mer@bandirma.edu.tr (M. F. ER)
Maide Feyza ER https://orcid.org/0000-0003-2580-1309
Turgay Tugay BİLGİN https://orcid.org/0000-0002-9245-5728

Received: July 18, 2024
Accepted: September 03, 2024
Published: September 15, 2024

Cite as: Er MF, Bilgin TT. 2024. Performance comparison of supervised machine learning methods in classifying celestial objects. BSJ Eng Sci, 7(5): 960-970.

1. Introduction

Nowadays, astronomy has entered a new era with rapidly growing data sources and advanced observation techniques. With the construction of more powerful telescopes, spectral data from millions of celestial objects are being collected. Additionally, it is anticipated that the volume of data generated by next-generation telescopes will significantly increase in the future (Hughes et al., 2022). Consequently, it is becoming increasingly difficult for astronomers to manually examine and label the ever-growing astronomical data. In situations where large quantities of data are available, as well as detailed data, distinguishing the source type is time-consuming and often impractical. Therefore, understanding this vast amount of data and conducting large-scale analyses to classify galactic and extragalactic sources has become a challenging task. For such large datasets, machine learning methods have emerged as useful and valuable tools for analyzing and classifying data.

Machine learning provides powerful tools for extracting meaningful information from large and complex datasets, enabling more precise and efficient classification. Consequently, in recent years, the use of machine learning methods in the classification of quasars, stars, and galaxies has become increasingly widespread. Machine learning methods perform well in identifying and characterizing different types of galactic objects through techniques such as modeling large datasets and feature extraction.

Quasars, stars, and galaxies are fundamental building blocks of the universe, and accurately classifying these cosmic objects is crucial for astronomical research. Quasars are highly luminous active galactic nuclei (Hughes et al., 2022). The identification of quasars emerged following the detection of radio emissions from star-like sources with high redshift values (Clarke et al., 2020). Even with larger optical telescopes at their disposal, astronomers find it extremely difficult to distinguish between a star and a quasar since both appear as bright points of light. Machine learning methods are particularly advantageous in this domain due to their potential to overcome the limitations of human vision and traditional techniques.

Machine learning algorithms can recognize complex patterns in large datasets and classify these objects with high accuracy. These algorithms process spectral features, brightness variations, and other astronomical data to distinguish subtle differences between quasars, stars, and galaxies. Additionally, machine learning provides astronomers with more precise and rapid classification capabilities. In this context, a study by Omat et al. (2022) highlighted the importance of feature engineering techniques in classifying galaxies, stars, and quasars using machine learning methods. By employing machine learning methods such as Decision Trees, K-Nearest Neighbors, Multinomial Logistic Regression, Naive Bayes Classifier, Support Vector Classification, and Random Forest, their study found that Random Forest performed best with an accuracy of 98%. Furthermore, Random Forest was found to correctly classify all instances labeled as stars in the dataset.

Mehta et al. (2022) conducted a study on a dataset consisting of 100,000 observations, investigating the classification of stellar features into Galaxy and Star categories using machine learning algorithms. The study aimed to explore the effectiveness of various machine learning algorithms, including K-Nearest Neighbors, Support Vector Classifier, Random Forest, Logistic Regression, Decision Tree, and Naive Bayes, in classifying stars based on their spectral features. The findings revealed that the Support Vector Classifier demonstrated the highest accuracy among the tested models, highlighting its effectiveness in star classification, while the Decision Tree model showed the lowest accuracy.

In their study, Kumar and Gharat (2023) presented a novel approach to classify stars as binary or exoplanet candidates using deep learning techniques. The proposed model is designed to accept two different types of inputs to enhance both accuracy and generalization. One input layer receives pre-computed statistical features, while the other input layer takes time series data from light curves. The proposed method demonstrated a strong performance with a test accuracy of 81.17%.

Haghighi (2023) examined various machine learning algorithms, including Linear Regression, Logistic Regression, Naive Bayes, SVM, Decision Trees, and Neural Networks. Some of the proposed models were applied to a dataset consisting of variable and non-variable stars from the SDSS Survey Stripe 82. The findings showed that Decision Trees provided the best accuracy and F1 score.

In their study, Huichaqueo and Orrego (2022) presented a machine learning-based method for the automatic spectral classification of stars using data from the SDSS database. They developed a Random Forest model to extract the spectral class of observed stars, training the model considering three data usage scenarios: the use of original data, undersampling, and oversampling techniques. Their study found that the model trained with augmented data outperformed the other scenarios. Furthermore, experimental results showed that the combinatorial use of data as an input model contributed to the improvement of prediction scores across all data usage scenarios (Huichaqueo and Orrego, 2022).

In his study, Brice (2019) utilized standard classification methods such as K-Nearest Neighbors, Random Forest, and Support Vector Machine to automatically classify spectra using data from the SDSS. The study focused on reducing the high dimensionality of stellar spectrum data through Feature Selection methods, including Chi-Square and Fisher score, as well as incorporating domain-specific astronomical knowledge to enable classification in a lower-dimensional space. The research highlighted the potential of machine learning to automate the classification of stellar spectra, offering a more efficient alternative to traditional, observation-based methods, which can be time-consuming.

Lastly, Savyanavar et al. (2023) compared the performance of traditional machine learning classifiers with a proposed CNN model to classify star and galaxy images. Their study aimed to improve classification accuracy by leveraging the feature extraction capabilities of CNNs in star-galaxy classification. They proposed a novel CNN architecture consisting of three sub-blocks for feature extraction, enhancement, and noise reduction. While traditional machine learning algorithms achieved a maximum accuracy of 78%, the proposed CNN model outperformed them, achieving an accuracy of 92.44% on the star-galaxy dataset.

The similarity of quasars to both galaxies and stars makes their differentiation challenging. Therefore, this study aims to classify stars, galaxies, and quasars based on their spectral features and to determine which classifier performs better for this problem. Decision Trees, Random Forest, and Naive Bayes classifiers were employed for this purpose, and performance metrics were evaluated.

The classification of galactic objects using machine learning methods is a well-studied topic in the literature. However, determining which classifier performs better in classifying specific classes such as quasars, stars, and galaxies is the focus of this study. The KNIME Analytics Platform was employed for data analysis, visualization, and classification in the study. Its user-friendly drag-and-drop interface facilitates rapid and effective analysis of data, requiring minimal technical expertise.

2. Materials and Methods

The dataset used in the study aims to classify stars, galaxies, and quasars based on their spectral features. The data, comprising a total of 100,000 space observations, was obtained by the Sloan Digital Sky Survey (SDSS) and is available on the Kaggle website under the title ‘Stellar Classification Dataset - SDSS17’ (Fedesoriano, 2022). Each data point consists of 17 feature columns and one class column that categorizes the data into stars, galaxies, or quasars. The names and descriptions of the features are provided in Table 1.

The analysis, model training, performance evaluation, graphs, and visual figures in the study were conducted using the KNIME Analytics Platform. The KNIME Analytics Platform is an open-source software that enables accessing, analyzing, and visualizing data without any coding requirements. KNIME provides a visual programming environment with an intuitive interface that integrates various technologies (Fillbrunn et al., 2017).

Table 1. Attributes and descriptions of the dataset.

Attribute	Description
obj_ID	Unique value that identifies the object
alpha	Right ascension angle
delta	Deviation angle
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 photometric system
z	Infrared filter in photometric system
run_id	Run Number used to identify the specific scan
rereun_id	Rerun number to indicate how the image was processed
cam_col	Camera column that defines the scan line within the study
field_id	Field number to identify the field
spec_obj_id	Unique identification used for optical spectroscopic objects
class	Object class
redshift	Redshift value depending on the increase in wavelength
plate	License plate ID that identifies each license plate in SDSS
mjd	Modified Julian Date used to indicate when a particular piece of SDSS data was received
fiber_id	Fiber ID, which identifies the fiber directing light to the focal plane in each observation

In KNIME, a workflow is constructed with nodes. Data is passed between nodes through connections throughout the workflow. Each node can perform various tasks such as reading and writing files, transforming data, training models, or generating visuals. Depending on their tasks, nodes have specific settings that can be configured in their configuration dialogs.

2.1. Dataset Overview

The KNIME Data Explorer node was used to examine the dataset broadly and to observe the characteristics of its attributes. The Data Explorer node provides a variety of options to interactively view the properties of input data through a table. It was checked from the resulting table whether there were any missing, empty, or NaN (Not-a-Number) values in the dataset, and none of these values were found.

The table also displayed the statistical properties of numerical columns, including their minimum, maximum, mean, standard deviation, and variance. Upon inspection, notably high standard deviation and variance values for certain attributes (especially u, g, and z) provided a significant indication that outliers might be present.

Another important clue for identifying outliers is provided by the histogram of the attribute, which visually represents the frequency of values in the dataset. The histogram graph was obtained using the Statistics node in KNIME. Similar to the Data Explorer node, the Statistics node displays statistical properties but produces histogram graphs in a less complex and more easily readable format compared to the Data Explorer node. Therefore, histogram graphs were obtained using the Statistics node.

Upon examining the histogram graphs of all attributes in the dataset, it was observed that the attributes u, g, z, field_ID, and redshift contain outliers. Additionally, the attribute rerun_ID was found to have the same value for all records. The histogram graphs of these attributes are provided in Figure 1.

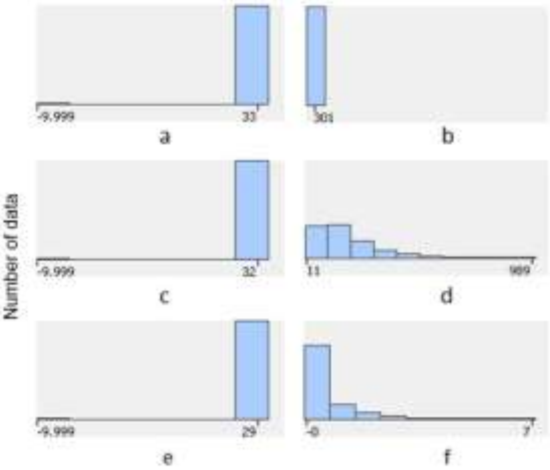

Figure 1. Histogram graphs of u (a), rerun_ID (b), g (c), field_ID (d), z (e) and redshift (f) attributes obtained with the Statistics node.

Standard deviation, variance, or histogram alone are not sufficient to accurately identify outliers in all attributes. For a more detailed analysis, KNIME's Box Plot node was used. Box Plot node; Displays statistical parameters such as minimum, lower quartile, median, upper quartile and maximum. The Box Plot node provides a quick overview of the outliers of a dataset.

Box Plot graphics were examined for all attributes of the data set and it was seen that in addition to the u, g, z, field_ID and redshift attributes, the r and i attributes also had outliers. An example graphic for this is given in Figure 2 for the r and i attributes.

In addition to missing and outlier values, another important issue to check in the data set is whether class labels are distributed fairly. The chart in which the number of data according to classes is obtained using KNIME's Bar Chart node is given in Figure 3. When the graph is examined, it is seen that the class labels are unevenly distributed.

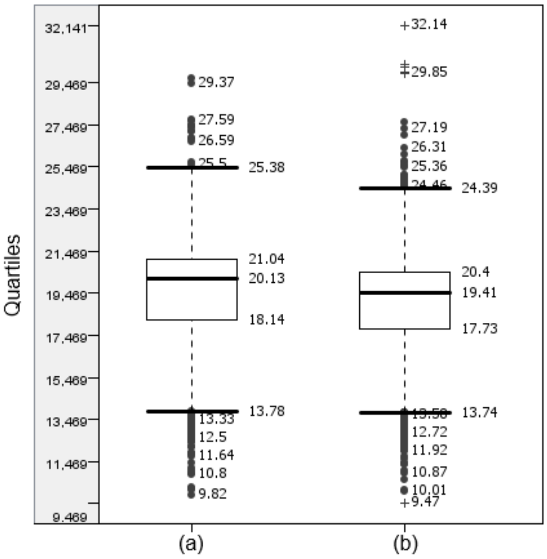

Figure 2. Representation of r and i attributes with box plot.

When the data set is examined with KNIME's visualization tools; It was determined that there were outliers for some attributes and class labels were unevenly distributed. The methods used in the study to overcome these problems are explained in detail under the heading of Data Preprocessing.

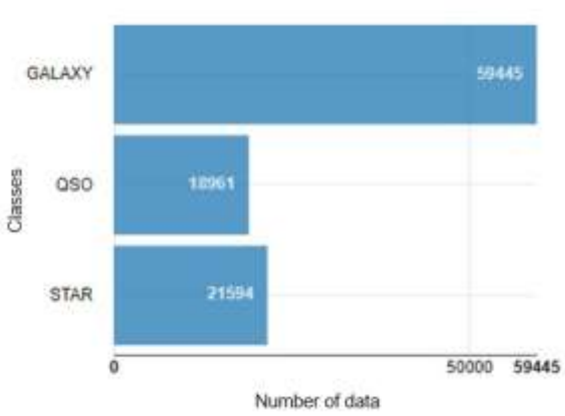

Figure 3. Number of data by classes.

2.2. Data Preprocessing

KNIME's Numeric Outliers node was used to remove outliers in the data set. The Numeric Outliers node detects and processes outliers separately for each of the selected columns via the interquartile range (IQR).

KNIME's Numeric Outliers node first calculates the IQR value to detect outliers of a particular column. IQR calculation is as in equation 1. The formula used to calculate whether an entry is an outlier, such that the interquartile distance factor k≥0, is given in equation 2. Accordingly, if the R value falls outside its calculated range, it is marked as an outlier.

$$IQR = Q_3 - Q_1 \qquad (1)$$

$$R = [Q_1 - k(IQR),\ Q_3 + k(IQR)] \qquad (2)$$

In the study, k = 1.5 value was used for the Numeric Outliers node, and the configuration of the node was made to convert these values into missing data when outlier treatment was found.

After applying the Numeric Outliers node to the dataset, the outliers turned into missing data. KNIME's Missing Value node was used to fill in the missing data. The configuration of the Missing Value node has been set to use the Linear Interpolation method to fill in the missing data. Linear Interpolation is a mathematical technique used to estimate the value between two known data points on a line. This method assumes a linear relationship between data points. The formula of Linear Interpolation is as in equation 3. Where y_1 and y_2 are two known points, x_1 and x_2 are the x-coordinates of the known points, y is the estimated value between y_1 and y_2.

$$y = y_1 + \frac{(y_2-y_1)\times(x-x_1)}{x_2-x_1} \qquad (3)$$

After filling in the missing data, KNIME's SMOTE node was used to eliminate the imbalance in class labels. SMOTE is a technique based on the k-Nearest Neighbor (kNN) algorithm. Creates new synthetic samples using sampling data from the minority class. These synthetic samples increase the representation of the minority class in the dataset while considering the immediate neighbors of existing samples, thus reducing class imbalance (Li et al., 2021). The graph showing the number of data in the classes after sampling the minority class with SMOTE is given in Figure 4.

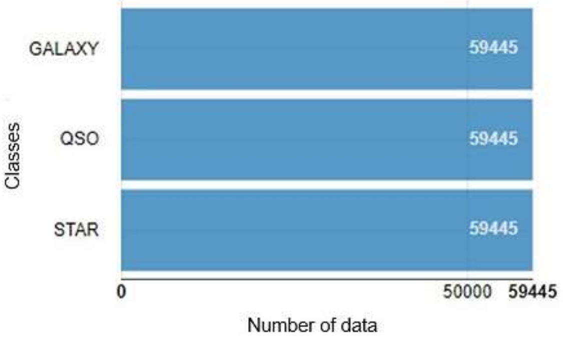

Figure 4. Data distribution according to classes after sampling with SMOTE.

2.3. Feature Selection

After the data preprocessing phase, feature selection was made. Feature selection reveals which features are more important and add valuable information in training the classifier. In addition to revealing the features that distinguish classes from each other, it also enables the removal of features that are not decisive in classification, thus training the model faster. For this purpose, two different feature selection methods were used in the study. The first of these is correlation analysis.

Correlation analysis is a method used to understand the relationship between variables and evaluate the connection between data. Correlation measures the direction and strength of the relationship between two variables, whether positive or negative.

KNIME's Linear Correlation node was used to perform correlation analysis of the features in the data set. This node calculates the correlation coefficient for each selected pair of columns. Which correlation measure to apply depends on the type of variables. The Linear Correlation node uses the Pearson Correlation Coefficient method for numerical data and Pearson's Chi Square Test for nominal data.

Since all values in the data set, except the class attribute, consist of numerical variables, the Pearson correlation coefficient method was used. The linear correlation analysis graph of the attributes in the relevant data set is given in Figure 5. Accordingly, the value of the measurement varies between -1 (strong negative correlation) and +1 (strong positive correlation). A value of 0 indicates that there is no linear correlation.

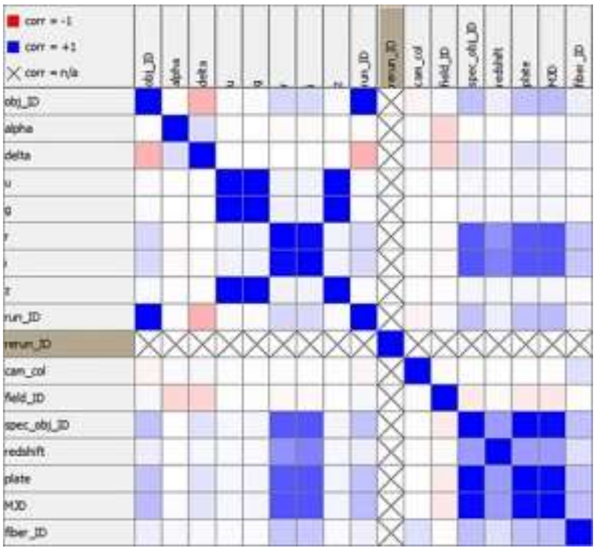

Figure 5. Linear correlation analysis graph of attributes in the data set.

When the linear correlation analysis given in Figure 5 is examined, it is seen that the rerun_ID attribute is not related to other attributes. It can be seen that the u, g and z attributes have a positive correlation with each other. While spec_obj_ID, plate, and MJD have a strong positive correlation with each other, the positive correlation with the redshift attribute is relatively weak. Likewise, it is seen that these attributes have a positive correlation, albeit weak, with the r and i attributes.

When the correlation analysis was examined, it was seen that the rerun_ID attribute was not positively or negatively correlated with other attributes in the data set. However, it was observed from the histogram of the attribute in Figure 1 that it gave the same value for all records. Based on these analyses, it is understood that the attribute does not play a decisive role in the formation of classes. Therefore, the rerun_ID attribute is removed from the data set as it will not help in model training.

Another method used for feature selection is Kernel Density Estimation (KDE). It was used to reveal at which values the attributes were concentrated.

Kernel Density Estimation is given in equation 4. Accordingly, $K(x)$ is called the Kernel function. The kernel function is symmetrical, like the Gaussian distribution, it increases as it gets closer to the data point and decreases as it gets further away. KDE basically applies the Kernel function to each X_i in the sample, with X_i being the data point. Thus, each data point represents X_i as small density bumps and then sums all these small bumps to get the final density estimate (Chen, 2018).

$$\widehat{p_n}(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{X_i - x}{h}\right) \qquad (4)$$

where h>0 in the equation, it is the bandwidth that controls the amount of smoothing. Bandwidth h plays an important role in the quality of KDE. When the bandwidth h is very small, the density curve has a very wavy and convoluted structure. On the other hand, when h is very large, fusion of the bumps occurs. This excessive smoothing causes important values to be hidden (Chen, 2018). Therefore, choosing the right bandwidth is extremely important.

Kernel Density Estimation was used in the study to reveal at what values the attributes in the data set concentrated or differed on a class basis. KNIME's 1D Kernel Density Plot node was used for this. Gaussian was chosen as the kernel estimation method. To select the appropriate bandwidth, the configuration of the node was made to use the Silverman Approach, which is the most practical and easiest to calculate method, considering the size of the data set. Kernel Density Estimation graphs of the features drawn with KNIME's 1D Kernel Density Plot node are given in Figure 6, Figure 7 and Figure 8.

When the graphs in Figure 6 and Figure 7 are examined, it is seen that some values of the attributes are effective in determining the classes. When Figure 6(f), Figure 7(a) and Figure 7(d) are examined, where the MJD, *plate* and *spec_obj_ID* attributes are given in the KDE graph according to classes, respectively, it is seen that the attributes have values that clearly distinguish all three classes. It has been observed that the attributes *g* and *i* in Figure 6(d) and Figure 6(e) and the *r* and *u* attributes in Figure 7(b) and Figure 7(e) clearly stand out in determining the QSO class.

The *alpha* features in Figure 6(a) and *delta* in Figure 6(b) are not as prominent as other features in reflecting the characteristics of the classes. However, for some values of the attributes, it has values that slightly distinguish the GALAXY and QSO classes.

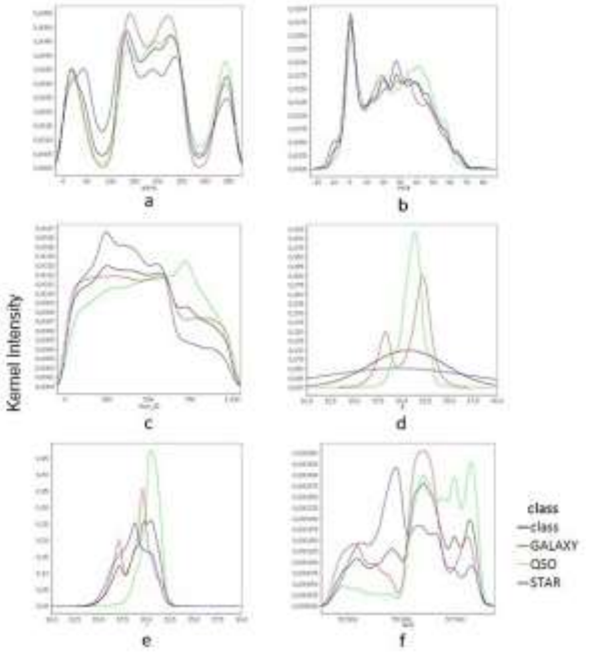

Figure 6. Kernel Density Estimation graphs of alpha (a), delta (b), fiber_ID (c), g (d), i (e) and MJD (f) attributes.

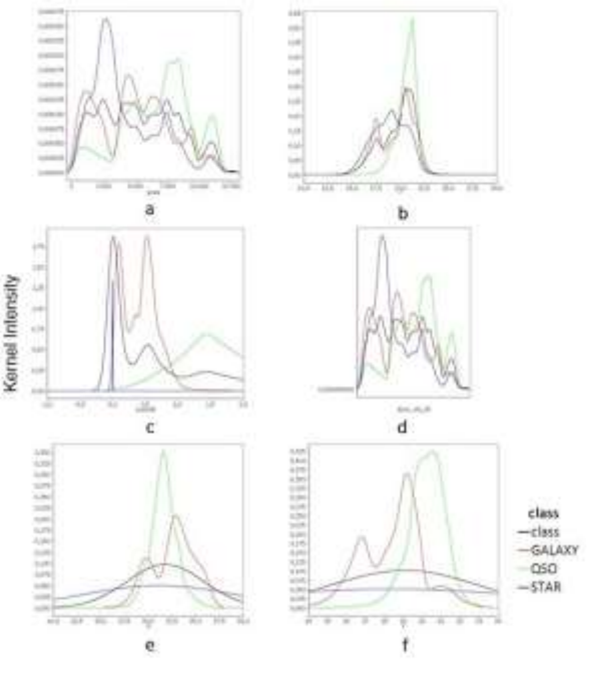

Figure 7. Kernel Density Estimation graphs of plate (a), r (b), redshift (c), spec_obj_ID (d), u (e) and z(f) attributes.

On the other hand, when the KDE graphs in Figure 8 are examined, it is seen that the attributes do not contain values that distinguish the classes. For almost all values, the distribution showed approximately the same characteristic. Therefore, since these features will not contribute to determining classes, they are removed from the data set before model training.

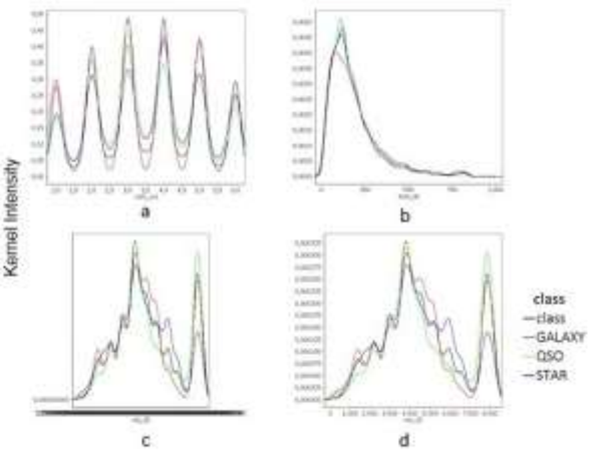

Figure 8. Kernel Density Estimation graphs of cam_col (a), field_ID (b), obj_ID (c) and run_ID (d) attributes.

2.4. Classifier Selection and Model Training

In the study, three different machine learning methods were used for model training. These; Decision Trees, Random Forest and Naive Bayes.

When the characteristics of the data set are taken into account, Random Forest, Decision Trees and Naive Bayes appear to be good machine learning methods. Naive Bayes works under the assumption that the characteristics of the data set are independent and generally performs well on small or medium-sized data sets. When there are independent features in your data set, Naive Bayes will evaluate this situation better and perform well. Random Forest and Decision Trees can work better on large data sets. Because these algorithms can capture the variations and relationships in the data set more effectively.

However, Naive Bayes is quite advantageous in terms of the transparency and simplicity of the model. In Decision Trees, it is quite interpretable because it clearly shows which feature makes which decision at each step.

Finally, in terms of the complexity of the algorithm, Naive Bayes is a relatively simpler algorithm and requires less computational power, which is advantageous especially when fast results are needed. Random Forest and Decision Trees require hyperparameter adjustment, which can increase the performance of the model.

2.4.1. Decision trees

Decision trees are an effective method used in the field of machine learning and data mining to solve classification and regression problems. Decision trees are basically a tree-structured classifier consisting of a root node, branches, internal nodes and leaf nodes in a hierarchical manner. It is a structure where internal nodes represent features of a dataset, branches represent decision rules, and each leaf node represents the outcome (Thomas et al., 2020). This hierarchical structure of decision trees is given in Figure 9.

Decision trees divide the data set into smaller subsets and determine a decision rule in each split. Each rule is based on a specific feature of the data set. With these rules, data points are classified or predicted by following different paths in the branches of the tree. It uses the divide and conquer strategy by performing greedy search to determine the most suitable split points within a tree (Thomas et al., 2020). It provides a graphical representation of all possible solutions to a problem based on given conditions. It stands out as a simple, understandable and high-performance model. Decision trees have a wide range of usage as they provide successful results in large data sets and complex problems.

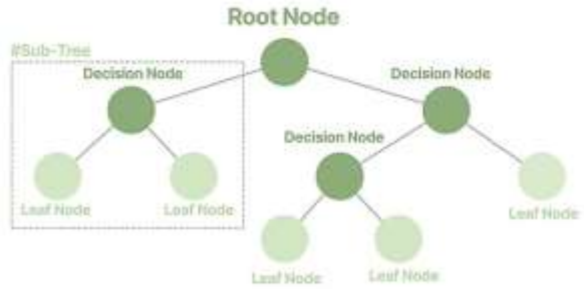

Figure 9. Hierarchical tree structure of decision trees.

2.4.2. Random Forest

Random forests are an ensemble learning method formed by combining multiple decision trees (Mouchel-Vallon and Hodzic, 2023). Each tree is trained using random samples of the dataset and runs independently on different subsets. This way, each tree makes its own prediction and combines its results to get majority votes for the average. This reduces overlearning and increases overall reliability (Mouchel-Vallon and Hodzic, 2023). An example random forest representation is given in Figure 10.

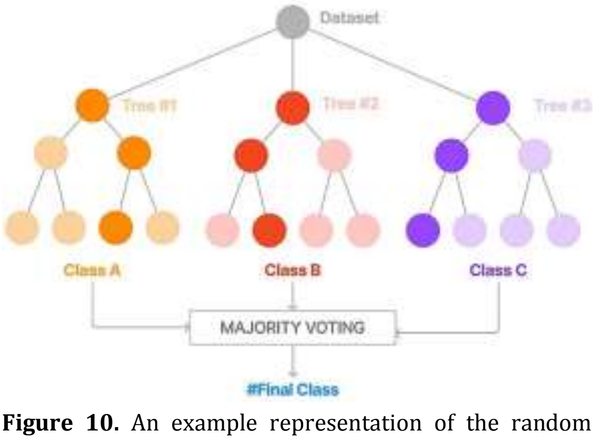

Figure 10. An example representation of the random forest structure consisting of a combination of decision trees.

The building blocks of the random forest algorithm are decision trees. The main difference between decision tree and random forest algorithm is that in random forest, the creation of root nodes and the separation of nodes are done randomly. While decision trees are trained using the entire dataset, random forests use random samples of the dataset in training each tree. This sampling process allows each tree to operate independently on different subsets.

Random forests can produce more stable and generalizing results because they consist of multiple trees. They also reduce overlearning and are better able to handle noise in the dataset. Random forests can operate at high speed on large data sets and provide high performance in classification and regression problems. Therefore, random forests have a wide range of applications.

2.4.3. Naive Bayes

Naive Bayes is a probability-based algorithm used to solve classification problems in the field of machine learning. Basically, it is based on the principle of Bayes theorem (Ramana, 2022). Naive Bayes is called "naive" because it assumes that the features observed in the classification process are independent of each other. This assumption means that each feature is evaluated independently without affecting the class.

Naive Bayes enables the classification of a new sample based on a predetermined class label. The classification process is based on calculating the probability of each class. After determining the relationship of the attributes in the data set with the class label, classification is made using Bayes' theorem. This theorem uses the probability of the class and the probabilities of the properties being observed in the given class to calculate the probability that the data point at which the properties are observed belongs to a class. Bayes' theorem is given in equation 5.

$$p(A|B) = \frac{p(A).p(B|A)}{p(B)} \tag{5}$$

In the equation, P(A) represents the probability of event A occurring and P(B) represents the probability of event B occurring. P(A|B) represents the probability of event A occurring if event B occurs, and P(B|A) represents the probability of event B occurring if event A occurs.

Naive Bayes is an algorithm that is often used successfully in areas such as spam filtering and text classification. Because it can provide effective results in large data sets with its simple structure and fast calculation ability (Ramana, 2022). However, the assumption that all features are independent is usually not valid in real life. Therefore, it may achieve low accuracy in cases where there are strong dependencies between features.

2.4.4. Building models with KNIME

80% of the data set was used for training and 20% for testing. KNIME's Partitioning node was used to divide the data set. This node divides the input table into two row-wise sections. The two sections it separates are delivered to two output ports. Learner nodes are connected to the output where 80% of the data is transmitted, and Predictor nodes are connected to the output where 20% of the data is transmitted. The classifiers used for model training in the study and the KNIME workflow diagram are given in Figure 11.

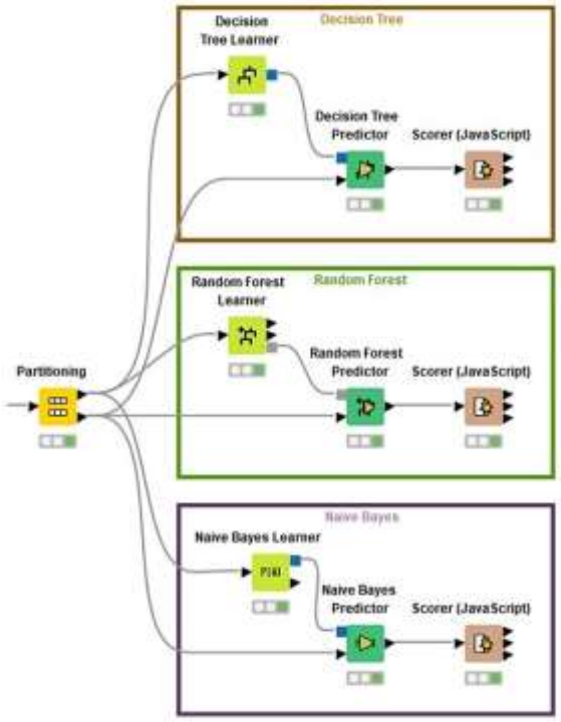

Figure 11. Classifiers used in model training and KNIME workflow diagram.

KNIME's Decision Tree Learner and Decision Tree Predictor nodes were used to build the Decision Trees model. The Decision Tree Learner node creates a decision tree in main memory for classification. The target attribute must be nominal. Other attributes used to make decisions can be nominal or numerical. Numerical splits are always binary, dividing the domain into two parts at a given split point. The algorithm provides two quality measures for the division calculation; Gini index and earnings rate. Additionally, a post pruning method is available to reduce tree size and increase prediction accuracy. The Decision Tree Predictor node was used to predict the class value of the model.

KNIME's Random Forest Learner and Random Forest Predictor nodes were used to build the Random Forest model. The Random Forest model created with Random Forest Learner consists of a selected number of decision trees. Each decision tree model is built with a different set of records, and a randomly chosen set of columns is used for each split within a tree. Rowsets for each decision tree are created by bootstrapping and have the same size as the original input table. The feature set for an individual split in a decision tree is determined by randomly selecting sqrt(m) features from the available features, where m is the total number of learning columns. The output model identifies a random forest and passes it to the Random Forest Predictor node for prediction.

KNIME's Naive Bayes Learner and Naive Bayes Predictor nodes were used to establish the Naive Bayes model. The Naive Bayes Learner node creates a Bayesian model from the given training data. Calculates the number of rows per attribute value per class for nominal attributes and the Gaussian distribution for numeric attributes. The created model can be used as the Naive Bayes Predictor node to predict the class membership of unclassified data.

Models were trained with the three different classifiers described above. Performance outputs were obtained with KNIME's Scorer node, which was connected to the output of the Predictor node of the models. The Scorer node uses performance metrics such as accuracy, error, sensitivity, precision, specificity and F-measure to measure the performance of models and presents the results in an interactive table. The values obtained with the Scorer node are given in the Findings section and the results are discussed.

2.5. Performance Evaluation of Models

Machine learning performance metrics are measures used to evaluate how well a machine learning model works. These metrics help evaluate how accurate the model's predictions are, misclassification rates, the model's ability to generalize, and other performance characteristics. In the study, accuracy, error, sensitivity, precision, specificity and F1-measure metrics were used to evaluate the performance of the models along with the complexity matrix.

2.5.1. Complexity Matrix

A complexity matrix is a table used to evaluate the classification performance of a machine learning model. The complexity matrix helps analyze the model's accuracy and error types by comparing predicted class labels with actual class labels.

The complexity matrix consists of four main components: true positive, false positive, false negative and true negative. True Positive (TP) is the number of samples that are actually positive that are correctly predicted as positive. False Positive (FP) is the number of samples in which samples that are actually negative are wrongly predicted as positive. False alarm situations are examples of FP. False Negative (FN) is the number of samples in which samples that were actually positive were incorrectly predicted as negative, while true negative (TN) is the number of samples that were actually negative were correctly predicted as negative. The complexity matrix contains the numerical values of these four components and allows to analyze the performance of the model in detail. This matrix can be used to calculate various performance metrics.

2.5.2. Performance Metrics

Table 2. Performance metrics, descriptions and formula used to measure the performance of the model (Erickson and Kitamura, 2021)

Metric	Description	Formula
Accuracy	It represents the proportion of samples that the model predicted correctly.	$\frac{(TP\ +\ TN)}{(TP\ +\ FP\ +\ FN\ +\ TN)}$
Sensitivity	It indicates how many of the truly positive samples were correctly detected.	$\frac{TP}{(TP\ +\ FN)}$
Precision	It refers to the ratio of samples predicted to be positive to samples that are actually positive.	$\frac{TP}{(TP\ +\ FP)}$
Specificity	It is a performance metric that measures a model's ability to correctly predict negative examples that are actually negative in classification problems	$\frac{TN}{(TN\ +\ FP)}$
F1-Score	It is the harmonic average of sensitivity and sensitivity metrics. It is used to achieve a balance between precision and sensitivity in unbalanced data sets.	$\frac{2TP}{(2TP\ +FP\ +\ FN)}$

3. Results and Discussion

Results from performance metrics were obtained by connecting the Scorer node to the output of the Predictor nodes of the models. Therefore, model performances were compared only on test data.

Complexity matrices obtained from Decision Tree, Random Forest and Naive Bayes models are given in Table 3, Table 4 and Table 5, respectively.

Table 3. Complexity matrix of the Decision Tree model

		Predicted		
		GALAXY	QSO	STAR
Actual	GALAXY	11464	404	21
	QSO	499	11389	1
	STAR	24	0	11865

Table 4. Complexity matrix of the Random Forest model

		Predicted		
		GALAXY	QSO	STAR
Actual	GALAXY	11536	232	121
	QSO	406	11482	1
	STAR	3	0	11886

Table 5. Complexity matrix of the Naive Bayes model

		Predicted		
		GALAXY	QSO	STAR
Actual	GALAXY	8740	2975	174
	QSO	1233	10654	2
	STAR	0	0	11889

When the complexity matrices given in Table 3, Table 4 and Table 5 are examined, it is seen that the number of False Positives and False Negatives for GALAXY and QSO classes is high in all three models. The models show the same tendency to misclassify and repeat certain errors. This indicates that the GALAXY and QSO classes are more difficult to recognize accurately and the characteristics of these classes may be more ambiguous than the STAR class.

Performance values of the models are given in Table 6. Accuracy values for Decision Trees, Random Forest and Naive Bayes models were obtained as 97.34%, 97.86% and 87.71%, respectively, while loss values were found to be 2.66%, 2.14% and 12.29%, respectively. These results show that the Random Forest method has the highest accuracy rate. Although Decision Trees and Naive Bayes algorithms have also achieved high accuracy values, it is clear that Random Forest provides superior performance.

When the sensitivity, precision and specificity values of the GALAXY and QSO classes, which are most similar to each other, are examined, it is seen that the Random Forest model obtains higher values than Decision Trees. However, the Naive Bayes model, which is weaker than other classifiers, achieved the highest sensitivity value by correctly predicting all true positive examples in the STAR class.

It is seen that the STAR class has the highest sensitivity, sharpness, specificity and f1-measure values in all three classifiers. The fact that the star class has the highest values in all four metrics shows that the model is more successful than other classes in correctly classifying stars. Additionally, this shows that the features of the star class are more distinct and distinctive than other classes, so the classifiers learn the star class better.

In terms of F1-measure, very high values for all class labels are seen especially in the Random Forest model and Decision Trees. This shows that the models generally exhibit a balanced performance. These findings show that the Random Forest model can perform effective classification on this data set and performs well.

Table 6. Results obtained from the performance metrics of the models

Classification Method	Accuracy	Loss	Classes	Sensitivity	Precision	Specificity	F1-measure
Decision Trees	97.34%	2.66%	GALAXY	96.43%	95.64%	97.80%	96.03%
			QSO	95.79%	96.57%	98.30%	96.18%
			STAR	99.80%	99.81%	99.91%	99.81%
Random Forest	97.86%	2.14%	GALAXY	97.03%	96.58%	98.28%	96.80%
			QSO	96.58%	98.02%	99.02%	97.29%
			STAR	99.97%	98.98%	99.49%	99.48%
Naive Bayes	87.71%	12.29%	GALAXY	73.51%	87.64%	94.81%	79.96%
			QSO	89.61%	78.17%	87.49%	83.50%
			STAR	100%	98.54%	99.26%	99.27%

4. Conclusion

Within the scope of this study, we evaluated the performance of three different machine learning algorithms: Decision Trees, Random Forest and Naive Bayes to classify GALAXY, QSO and STAR classes. Our study was carried out on the K-NIME platform. The results obtained show that all three algorithms exhibit superior performance in classifying the STAR class. Decision Trees and Random Forest models attracted attention with their high accuracy (97.34% and 97.86%) and low error rates (2.66% and 2.14%). Similarly, in the study of Omat et al. (2022) Random Forest showed the best performance with an accuracy rate of 98%. Especially for the STAR class, the sensitivity, precision, specificity and F1-measure values of these models were found to be almost perfect. Haghighi (2023), who tested various classifiers, showed that Decision Trees provided the best accuracy and F1 score.

Although the Naive Bayes algorithm had lower overall accuracy than the other two models (87.71%), it still showed high performance in the STAR class. However, one of the interesting results of the study is that the Naive Bayes classifier can distinguish the STAR class in the best way. As a result, it has been observed that the STAR class can be successfully classified by all algorithms thanks to its distinct and distinctive features, but more advanced and complex models (such as Random Forest) give better results in separating the GALAXY and QSO classes. While the study helps us understand which algorithm will perform better on a given data set, the findings highlight the importance of choosing the right algorithm in classification problems. At the same time, these findings provide important clues in the selection of machine learning algorithms used in classifying astronomical objects.

In future studies, it is aimed to increase classification performance by using larger data sets and more complex models. Improvements can be added to make the model better recognize and distinguish GALAXY and QSO classes. Different analysis techniques can be applied to reveal the superior ability of the Naive Bayes classifier in determining the STAR class. A more comprehensive comparison can be made using different data sets and algorithm parameters, and more detailed results can be obtained by examining different metrics.

Author Contributions

The percentage of the authors contributions is presented below. All authors reviewed and approved the final version of the manuscript.

	M.F.E.	T.T.B.
C	70	30
D	70	30
S	40	60
DCP	40	60
DAI	50	50
L	50	50
W	70	30
CR	40	60
SR	80	20

C=Concept, D= design, S= supervision, DCP= data collection and/or processing, DAI= data analysis and/or interpretation, L= literature search, W= writing, CR= critical review, SR= submission and revision.

Conflict of Interest

The authors declared that there is no conflict of interest.

Ethical Consideration

Ethics committee approval was not required for this study because of there was no study on animals or humans.

References

Brice MJ. 2019. Classification of stars from redshifted stellar spectra utilizing machine learning. MSc Thesis, Central Washington University, Computational Science, Washington, US, pp: 73.

Chen YC. 2018. Lecture 6: Density Estimation: Histogram and Kernel Density Estimator. URL= http://faculty.washington.edu/yenchic/18W_425/Lec6_hist_KDE.pdf (accessed date: May 10, 2024).

Clarke AO, Scaife AMM, Greenhalgh R, Griguta V. 2020. Identifying galaxies, quasars, and stars with machine learning: A new catalogue of classifications for 111 million SDSS sources without spectra. Astronomy Astrophys, 639: A84.

Erickson BJ, Kitamura F. 2021. Magician's corner: 9. Performance metrics for machine learning models. Radiol Artif Intel, 3(3): e200126.

Fedesoriano. 2022. Stellar Classification Dataset-SDSS17. URL= https://www.kaggle.com/fedesoriano/stellar-classification-dataset-sdss17 (accessed date: May 15, 2024).

Fillbrunn A, Dietz C, Pfeuffer J, Rahn R, Landrum GA, Berthold MR. 2017. KNIME for reproducible cross-domain analysis of life science data. J Biotechnol, 261: 149-156.

Haghighi MHZ. 2023. Analyzing astronomical data with machine learning techniques. arXiv Preprint, arXiv: 2302.11573.

Hughes AC, Bailer-Jones CA, Jamal S. 2022. Quasar and galaxy classification using Gaia EDR3 and CatWise2020. Astronomy Astrophys, 668: A99.

Huichaqueo MO, Orrego RM. 2022. Automatic spectral classification of stars using machine learning: An approach based on the use of unbalanced data. Machine Learn Appl, 9(4): 01-16.

Kumar A, Gharat S. 2023. Star cassification: A deep learning approach for ıdentifying binary and exoplanet stars. arXiv Preprint, arXiv: 2301.13115.

Li J, Zhu Q, Wu Q, Zhang Z, Gong Y, He Z, Zhu F. 2021. SMOTE-NaN-DE: Addressing the noisy and borderline examples problem in imbalanced classification by natural neighbors and differential evolution. Know Based Syst, 223: 107056.

Mehta T, Bhuta N, Shinde S. 2022. Experimental analysis of stellar classification by using different machine learning algorithms. 2022 International Conference on Industry 4.0 Technology (I4Tech), September 23-24, Pune, India, pp: 1-8.

Mouchel-Vallon C, Hodzic A. 2023. Toward emulating an explicit organic chemistry mechanism with random forest models. J Geophys Res Atmospheres, 128(10): e2022JD038227.

Omat D, Otey J, Al-Mousa A. 2022. Stellar objects classification using supervised machine learning techniques. International Arab Conference on Information Technology (ACIT), November 22-24, Abu Dhabi, United Arab Emirates, pp: 1-8.

Ramana PV. 2022. Naïve Bayes to machine learning approach for structural dynamic complications. ASPS Conf Proc, 1(4): 1283-1291.

Savyanavar AS, Mhala N, Sutar SH. 2023. Star-galaxy classification using machine learning algorithms and deep learning. Int J Info Technol Secur, 15(2): 87-96.

Thomas T, Vijayaraghavan P, Emmanuel A, Thomas S, Vijayaraghavan TP, Emmanuel S. 2020. Applications of decision trees. Machine Learn Appr Cyber Secur Analyt, 2020: 157-184.