Journal of Engineering and Sustainable Development
Vol. 30, No. 01, January 2026
https://jeasd.uomustansiriyah.edu.iq/index.php/jeasd
ISSN 2520-0917
Research Article
https://doi.org/10.31272/jeasd.2998

Implementing and Validating a Data Mining System Based on Decision Trees

Qutaiba Humadi Mohammed[1*], Chaitanya Konda[2], Anupama Namburu[3]

[1]Media Technology and Communications Engineering Department, College of Engineering, University of Information Technology and Communications, Baghdad, Iraq
[2]Department of CSE, Dr.YSR ANU College of Engineering and Technology, Acharya Nagarjuna University, Nagarjuna Nagar, Guntur, Andhra Pradesh 522510, India
[3]School of Engineering, Jawaharlal Nehru University, New Delhi 110067, India

[*]Email: Qutaiba.humadi@uoitc.edu.iq

Article Info

Received	14/09/2024
Revised	28/10/2025
Accepted	12/11/2025

Abstract

Data mining is concerned with revealing valuable patterns and insights out of big data sets, and machine learning forms a critical part in streamlining this effort. In any supervised classification task, decision tree algorithms become particularly beneficial since they provide understandable models and can be used to evaluate the significance of features. This paper assesses the work of YaDT as an example of a decision-making algorithm that uses the famous Titanic dataset. Provided a stringent statistical analysis to explore how the model behaves in different evaluation conditions. The findings reveal that YaDT builds a highly organized decision tree with strong generalization abilities, hence showing high classification ability. However, there was also a significant reduction in predictive reliability in some classes with very limited amounts of predictable data, indicating poor representation in the dataset. Additional evaluation of model performance using ROC analysis, based on cross-validation results, demonstrated comparable discriminative ability. Overall, it can be concluded that the findings favor the hypothesis that YaDT is a resilient, interpretable classifier, and as such, it can be competitive with the alternative machine-learning algorithms.

Keywords: Data Mining, Decision Trees, Machine Learning, Statistical Analysis, Titanic Dataset.

1. Introduction

Detection and resolution of a problem will begin with a proper connection to relevant information, data, and context [1]. Given that statisticians or data analysts often lack domain-specific expertise in the application area, collaboration with subject-matter experts is essential to the knowledge-discovery process. A formal, clear expression of a problem's characteristics, accessible to non-statisticians, helps create a common conceptual system among professionals from different fields [2].

In the context of data mining, classification is a supervised learning paradigm where a category of prediction inferring the categorical classification of new examples is done through a collection of examples that have been previously labeled. The difficulty is amplified by the occurrence of an imbalanced dataset, where the number of times individual classes appear in the dataset does not match the same number of times as the other classes within the same sample. This phenomenon is often seen in a wide range of practical settings: in financial analytics, a small percentage of the population of highly risky borrowers can represent a small part of the dataset, whereas in educational settings, most students are capable of achieving success, thus creating a natural community of the majority. Such discrepancies have been widely recorded in previous studies, which highlights the fact that accurate forecasting of the minority group may prove to be of utmost significance even in a case where the group is underrepresented in the training sample [1]-[3].

Exploratory data mining supports initial research-level investigations by allowing analysts to examine datasets without the constraints imposed by preconceived hypotheses. The decision-making queries at the high level give the users the ability to discover the emergent patterns and optimize their analytical attention as new information is discovered [3]. Interactive exploration tools play an important role in enhancing the analytical process through dynamic filtering, variable emphasis, and hierarchical visualization of results.[4]. This exploratory facility counteracts the excess burden that is inherent in defining the problem too narrowly at first.

Academically, data mining is a paradigmatic case of an interdisciplinary venture, relying on the methodologies of statistics, the algorithms of machine learning, the cognitive models of artificial intelligence, and the distributed systems of cloud computing. Fundamental to this area of disciplinary intersection is a keen understanding of the issue of security and privacy, which should be infused into each phase of the data lifecycle, including data acquisition and preprocessing, model implementation, and post hoc auditing [5]. Decision Trees (DTs) remain among the most popular data-mining methods because they generate understandable, rule-based models, are easily visualized, and can handle both numerical and nominal variables [6]. Machine learning (ML) extends this capability by building models that identify patterns from prior experience. Since both ML and DM are based on many conceptual similarities, it is normal to use them together, which in turn would accelerate the process of knowledge discovery and increase the accuracy of the analytical process. [7]. Without the input of machine learning, as Zhou has noted, the data mining world would be similar to a needle picking one needle in a haystack.[8], Subsequent empirical studies have shown that machine learning has significantly contributed to the optimization of data mining techniques [9]. Specifically, decision trees remain among the most studied and used data-mining algorithms [10], primarily because they express learned knowledge in a comprehensible way and support a wide range of variable structures [11].

Based on this background, the purpose of the present work is to use the already gained knowledge about decision trees to develop a machine learning framework that could learn salient features and classify the data using the YaDT (Yet Another Decision Tree) algorithm.

1.1. Algorithm C4.5

The decision trees (DTs) are still among the most popular machine learning paradigms studied within data-mining courses due to their intuitive interpretability, their ability to handle the heterogeneous data typologies, and their relatively efficient training regimes, in comparison with other more complex constructs like the artificial neural networks [5]. The popularity of decision trees has therefore spawned a long lineage of algorithms, including C4.5, and commercial offshoots C4.8 (also known as J4.8) and C5.0. These versions include advanced pruning heuristics designed to reduce overfitting and thereby improve the model's generalization. Exemplary algorithms: The most well-known error-adjustment formalisms, such as pessimistic pruning, reduce the effects of optimism bias on training data by balancing the risks associated with individual tree nodes, thereby refining the tree structure based on weighted errors on a case-by-case basis [12]-[15]. These developments encourage a consistent academic effort to augment decision-tree strengths, especially in the presence of noisy or incomplete data, a challenge for which Abe [16] is a notable proponent of using numerical property metrics.

Contemporary academic research has highlighted the practical value of decision tree methods in applied settings, where their inherent transparency enhances decision support. As an example, the decision tree framework played a fundamental role in directing the activities of the nursing cohorts, which brought tangible results of operational efficiency. These empirical applications are the best examples of how explicable designs can be used to substantively improve operational decision-making procedures, potentially justifying the use of decision trees to support well-organized classification initiatives [17].

Though much of what has existed in the scholarly literature up to this point offers a significant amount of insight into the development and implementation of decision trees, the literature that has been published up to now exhibits a set of focal points that do not directly address the issues of practical implementation of decision-tree-based data mining on structurally organized datasets where supervised classification is required. The work on pruning schemes, noise-reduction schemes, or domain-specific adaptations provides a valuable methodological contribution, but these schemes do not meet the requirement of a single, rigorously validated data-mining pipeline, making data preparation, feature engineering, model construction, and performance evaluation the same algorithmically consistent set. This gap, in turn, underscores the need to implement and empirically justify an extensive decision-tree DM system, supported by systematic preprocessing and model testing.

2. Methodology

The primary purpose of this work is to design and rigorously evaluate a machine learning model based on the C4.5 decision tree algorithm and to assess its predictive performance in supervised classification. The C4.5 algorithm is widely recognized as a standard classification tool and has served as a cornerstone for the subsequent development of other algorithms [16]. Our process is based on a carefully structured machine learning framework that includes data selection, preprocessing, construction, and evaluation. The first step involved identifying a supervised dataset and curating it for use in the analysis. Missing values, duplicate records, and inconsistent attribute formatting, common manifestations of so-called noise that can affect learning results, were mitigated by the preprocessing stage. After the data were prepared, the model was implemented using the YaDT C4.5, which was selected for its reliability and interpretability.

A training-validation cycle was performed iteratively to assess both the model's stability and its generalizability. The data were separated using k-fold cross-validation; a decision tree was trained for each fold; error estimates and a confusion matrix were generated to evaluate performance. After all folds were completed, the total accuracy and error rates were summed to assess the model's performance. The experience of these iterations provided information on how to optimize the dataset by performing specific feature engineering and information on how to modify the model architecture. A detailed description of the entire workflow, including data preparation, preprocessing, algorithm selection, and validation methodology, is provided in the following subsections.

2.1. Data Collection

The choice of the datasets to use in this work was informed by the fact that the datasets should be able to be used to perform supervised classification, that is, each instance should have a predetermined target attribute and the descriptive features of the target in the instance. Two datasets were selected after considering several candidates to address various methodological objectives. The initial dataset was selected to validate the machine learning model and to assess the efficiency of the C4.5 decision tree algorithm. To this end, the Titanic dataset, widely used on Kaggle, has been extensively examined in the literature, with a range of machine learning methods applied [18]. It is well known in the scientific community and therefore applicable to benchmarking the performance of the YaDT C4.5 implementation [16].

The second, obtained with the assistance of the Ministry of Agriculture, represents a real-world classification problem involving the identification of potentially harmful substances in beef samples [19]. The data provided provides an opportunity to apply the C4.5 algorithm in practice and identify salient patterns, e.g., seasonal peaks in specific contaminants or changes in detection frequency. The combination of the two datasets will yield a balanced evaluation: the former will assess the algorithm's reliability on a strong benchmark, whereas the latter will demonstrate that the model can derive practical insights from real operational data.

2.2. Data Pre-Processing

Once the data has been selected for training the decision tree algorithm, it is necessary to ensure that the data is consistent and appropriate to prevent errors in model construction. Data preprocessing is an important step in enabling a machine learning model to extract meaningful patterns and produce reliable classifications [20]. In this step, inconsistencies, such as invalid records and structural deficiencies in the data, have been detected and corrected to ensure that the algorithm is fed with accurate input. It is necessary to ensure that the data entered into the system strictly conforms to the algorithm's specifications and is of high quality.

A preprocessing data workflow was adopted as shown in Fig.1. The first two phases were implemented before the model was put into action, and their purpose was to remove noise, delete unnecessary records, and deal with unfinished observations. The last step was discretisation, performed in-house using the C4.5 algorithm, which automatically partitions continuous variables into intervals for decision-rule induction [21]. This systematic approach enabled the resulting data to serve as a strong foundation for subsequent training, pruning, and validation.

Data cleaning and selection
Treatment of missing features
Discretization of continuous characteristics

Figure 1. Pre-processing stage data processing.

2.2.1. Data cleaning and selection

The data cleaning and selection procedures were applied systematically in order to distinguish and cut noise, which can be described as the discrepancies in the characteristics of certain occurrences or complete records which stand out as inflexible to the domineering structure of the information set. Such irregularities are necessary to be eliminated so that the models will maintain the accuracy when defining new classes. Because noise detection often depends on subtle interpretations of data provenance, the availability of domain-specific knowledge is necessary to interpret aberrant behaviours correctly. In line with this, a literature review relevant to the dataset's field was conducted carefully before cleaning to ensure that valid yet rare cases were not excluded. This contextual understanding is necessary to classify instances as noisy, thereby removing valid data and ultimately harming the learning process by feeding the algorithm an improperly cleaned training dataset.

The processes that were implemented during this phase are:

- Identification of possible discrepancies in the data.
- Identify possible typographical errors.
- Check possible inconsistencies in numerical values.
- Check possible inconsistencies in categories.
- Deal with missing features.

Typographical errors are common in real-world datasets and can significantly affect model performance if not addressed appropriately [22]. When it comes to categorical attributes, a single typographical error may create an anomalous, non-existent type, so that the creation of decision trees can be compromised. As a remedy to this, listed all possible values of each discrete attribute systematically and examined the distribution of the instances in terms of these categories and thus detected and marked out aberrant entries. In most empirical datasets, numerical inconsistencies also pose a similar challenge. To reveal such differences, the concept of graphical diagnostics was employed: a graphical representation of the entire range of numerical values for each attribute. These plots helped identify outliers that deviated significantly from the expected distribution, and these outliers frequently indicated faulty or unrealistic data points [23]. On each discrepancy being identified, inspect that discrepancy, making the necessary corrections or deleting spurious records. The final and probably the most important aspect of the cleaning stage is done with respect to the treatment of missing features. This is a process that requires careful methodological consideration, where poor conduct may result in bias or loss of integrity in the latter model learning.

2.2.2. Treatment of missing features

This issue of missing features in empirical data is endemic when training predictive modeling systems without specifically addressing it [24]. The percentage of missing records in the sample used herein was substantial, particularly in the Titanic dataset, with the proportion of missing values approaching 20%. Therefore, a crude method of selecting non-complete samples, such as tempting, would not be appropriate; it would select a large portion of the observations and thus create artificial heterogeneity [25].

Therefore, in disregarding features, the selection of attributes was based on the relevance and meaning of each feature. The first assessment was conducted to determine whether the absence of a datum was semantically significant. To illustrate, in case a registrant is asked to provide a driver number, the intentional leave of this field can mean deliberateness as opposed to an unintentional oversight. Where there is no signal, this is intentional in such situations and not a non-measurable condition. In the places where it was determined in the analysis that the missing datum actually represented a meaningful category, a special category was coined to represent this blank, named “Unknown” in the case of categorical variables and coded within the numeric range 0 to 999 in the case of quantitative characteristics.

In situations where the absence of a value is not semantically significant, a different approach is used: the missing attribute is set to the most common value in the training set. This does not alter the distribution of the attribute. C4.5 algorithm also supports the eventuality of the missing values by sharing the instances that lack the attributes over all the branches of the concerned decision node in the same proportion to the likelihood of the observed labels. Its J48 implementation incorporates additional mechanisms, including handling missing values, pruning decision trees, managing continuous attributes, and deriving classification rules.

2.2.3. Discretization of numerical values

Despite supporting continuous features, the Decision Tree algorithm cannot operate directly on them, because the values tested at the decision nodes and the predictions produced by the algorithm must be discrete [26]. Therefore, the classification of continuous attributes into discrete intervals is necessary. This process, known as discretization, is performed internally by the C4.5 algorithm. Therefore, performing the discretization directly in the dataset is unnecessary.

The discretization process carried out by the C4.5 algorithm [27] consists of dynamically defining an interval threshold by calculating the Information Gain (or Entropy). This calculation is performed to determine the threshold that yields the most significant information gain, i.e., the interval that covers the most significant number of instances within a subset. Once this range is defined, the decision node only performs a Boolean comparison: whether the continuous value of the instance is greater than or less than/equal to the calculated threshold.

2.3. Developing the ML Model

After determining the dataset for model evaluation, the model creation stage began using the C4.5 algorithm [28]. The following sections present the rationale for selecting the algorithm and the main tools that will assist in using and evaluating the system.

2.3.1. Algorithm C4.5

The model used in this work is the C4.5 Decision Tree algorithm, selected for its robustness, ability to handle diverse data types, and strong interpretability—all of which align to build an explainable and reliable classification model. Decision-tree techniques are an essential part of current data-mining practice, renowned for their fast training, the intuitively clear construction of a rule set, and their ability to use both categorical and numeric feature spaces (Witten and Frank, 2000). One of them, the C4.5 algorithm, holds a central place, as it has become a standard in classification and is the ancestor of numerous other tree-based algorithms [29]. Its skillful management of categorical attributes also eliminates elaborate encoding plans, thereby preventing unnecessary dimensional proliferation and facilitating the development of more parsimonious models. In addition, C4.5 has internal mechanisms to handle missing values, which ensure the integrity of the training process and, consequently, the reliability of the model's predictive performance [30]. Although other machine learning methods can handle categorical inputs and missing values, they are less useful in the data-mining setting, where interpretation of underlying patterns is preferable because of their intrinsic lack of interpretability [31]. Alternatively, decision trees generate clear, rule-based models that are easily interpretable as learned structures in the data and are thus particularly useful for measuring feature relevance, diagnosing the causes of misclassification, and subsequent model improvement [32]. This interpretability is important in the present work because it is necessary not only to fit the instances correctly but also to derive significant value from the underlying dataset.

2.3.2. Language and development environment

The first idea was to implement the C4.5 algorithm manually in C++, a language well-suited to object-oriented programming and for which the authors are familiar with its syntax and tools. However, as the primary objective of this work is to demonstrate the effectiveness of decision trees in extracting significant patterns in datasets, rather than to develop a new implementation, it would have been wiser to rely on a proven and tested system. In this regard, YaDT, another Decision Tree (YaDT), as discussed by Ruggieri (2004), was adopted [33]. YaDT is a polished C++ version of C4.5, called EC4.5 (Enhanced C4.5), and provides several improvements over traditional implementations, particularly in computational speed and memory management. Both C++ and Python were used to facilitate key processes, such as data cleaning, feature preprocessing, and handling missing attributes [34]. C++ was used to optimize performance, whereas Python was chosen for its extensive libraries for handling tabular datasets, such as .xls and .csv files. The development environment was a Linux Debian Wheezy 7.0 (64-bit) system comprising Sublime Text 2.0 as the primary code editor, G++ 4.7.2 as the compiler, and Python 2.7.3 for task implementation.

2.3.3. Development tools

In addition to the development environment, several assistant tools were used to facilitate the work. For version control, the GitHub platform was used. GitHub is an online service used by Git version-control system users [35]. In order to create visualizations, the GnuPlot utility was used. This tool was primarily used in the Data Pre-Processing steps, where scatter plots of numerical variables were generated to identify outliers and potential anomalies, and in model verification.

2.4. ML Model Validation

After the decision tree model was established, there was an urgent need to assess its predictive performance through an independent test set. Decision trees are prone to overfitting, i.e., absorbing idiosyncratic trends in the training data, thereby impairing their external validity. Therefore, it was necessary to employ a stringent assessment methodology that yields a realistic estimate of performance. The current evaluative phase is thus aligned with the overall objective of the work, namely, the development of a reliable and widely generalizable machine learning model for supervised classification [36].

2.4.1. Cross-validation

K-fold cross-validation was used to ensure that the model's performance was not dependent on a single train-test split. This approach provides a solid evaluation of generalization and appropriately completes the preceding preprocessing and feature engineering steps, which aim to reduce noise and improve dataset integrity. The overall dataset was used for both training and testing. The selected datasets were fully annotated. The data were then divided into k folds of approximately equal cardinality. In cases where perfect partitioning was not possible, the remaining elements were shared between the folds. The folds were assigned distinct numbers from 1 to k. The validation process included k repetitions. Each time, one fold was selected as the test set, and the remaining k-1 folds were used for training. After training on a model, the classification error on the test fold was reported. When all iterations were completed, the final error estimate was computed as the average fold error. Such a strategy provided a statistically sound estimate of the model's predictive effectiveness and ensured that the test was closely aligned with the overall objective: to evaluate the decision tree model's performance across a range of data subsets and contextual variations.

2.4.2. Analysis of results

In addition to cross-validation, the confusion matrix was used to evaluate the algorithm's predictive accuracy critically. Cross-validation only evaluates the error estimate computed for all instances (i.e., the number of instances incorrectly classified). Hence, the Confusion Matrix provides a more detailed estimate of misclassification between classes. Based on the Confusion Matrix, the True Positive Rates (TPR) and False Positive Rates (FPR) were calculated, metrics used to plot the ROC curve [37].

2.5. ML Model Optimization

First, the ML model was trained and evaluated on the complete dataset, using all instance-level characteristics. Once the error estimates and the confusion matrix for this iteration are obtained, the data will be analyzed to determine whether the algorithm's predictive performance can be improved. This optimization can be achieved in two ways: by applying the FE technique to the dataset or by improving the algorithm. Dimensionality reduction was applied to enhance the quality of the training and test datasets by removing less informative features during Decision Tree construction.

Feature engineering (FE) is a process by which the dataset's characteristics are worked on so that the ML model has improved performance. The technique used to perform FE on the system dataset was dimensionality reduction. Although humans are not limited to three dimensions, higher dimensionality makes it more difficult for ML models to process the dataset [38]. This technique aims to reduce the number of features the ML model will handle during training and evaluation, thereby improving the algorithm's performance. The Dimensionality Reduction technique employed is Feature Selection. Again, the greedy algorithm will be used to select the optimal subset from the complete set. This approach is known as reverse Attribute Selection: starting with the full feature set, the worst feature within that set is eliminated.

3. Results

3.1. Dataset - Titanic

The first dataset analysed is provided by Kaggle [39]. On this well-known platform, several challenges in analysis and DM are proposed, in the search for the best predictive model for a given problem. This dataset, provided in September as part of one of these challenges, comprises information on the passengers aboard the Titanic. The analyzed dataset contains 1309 instances, whose characteristics comprise data about passengers who were on the boat on its first and only trip when it sank after hitting an iceberg. The prediction class is a categorical variable indicating whether the passenger survived (1) or not (0). Each instance contains 13 characteristics, in addition to the target characteristic (which indicates whether the passenger survived or not): class the passenger was in, name, gender, age, number of siblings/wives on board, number of parents/children on board, ticket number, ticket fare, cabin, location where the passenger boarded, which lifeboat the passenger was on, whether the passenger was able to board any of the available boats when the ship sank, body identification number, and passenger's destination location. Of these, 10 features are categorical, and 4 are numeric. The percentage of missing features is approximately 20.1%. The distribution is: 61.8% of passengers did not survive the accident, and 38.2% survived.

3.2. Data Cleaning and Selection

Before creating the AM model, the dataset was preprocessed by cleaning and Selecting Data. As previously described, this phase is conducted to ensure that the dataset is sufficiently consistent for model development and validation, and to remove potential inconsistencies that could hinder data mining. Data. Preprocessing of the dataset was performed using a Python script. The Python Data Analysis Library (pandas) was used to manipulate the dataset as a data frame, and NumPy, in conjunction with the Open-Source Library of Scientific Tools (SciPy), was used to apply mathematical functions.

YaDT works with datasets in a specific format, so it was necessary to address some problems present in this dataset. In this case, the algorithm does not handle names with quotation marks, so it is essential to remove them from the name and $home_dest$Characteristics. To do this, the $replace()$Function was applied to each of the elements present in the list, referring to each characteristic,w ith the help of *the map*() and *lambda*() functions. In addition to removing the quotation marks, it was necessary to remove the commas within the characteristics, as they would cause issues when reading YaDT.

Furthermore, it was decided to divide the passenger's full name into two distinct columns: name and surname. This division was performed by executing the *split*() function, dividing the first and last name using the token passed as a parameter (in this case, the hyphen).

3.3. Identification of Possible Inconsistencies in Numerical Values and Categories

To identify inconsistencies in the data, they performed a systematic review of categorical and numerical attributes. For categorical attributes, the listed categories were related to each attribute, ensuring that no case deviated from the expected groups. The features with a small number of distinct values (such as *sex and sibsp*) were examined alongside the standard categorical fields because they exhibit limited variability, which allows validation to proceed without complications. Python code that uses NumPy's unique and ravel functions was used to isolate and tabularly examine the unique values of each categorical variable.

Upon analysis, the following features (surname, ticket number, cabin name, and boat identifier) showed an excess of individual entries, grouping passengers into small clique-like groups. As a result, the input of these features into the predictive model was considered to be not generalizable enough, and it was re-evaluated in the process of feature engineering to avoid introducing extra variability.

To assess the numerical attributes, a set of scatterplots was constructed to determine whether extreme values fall within realistic ranges, thereby identifying possible outliers or errors in measures such as age and fare. An additional script was used to extract and sort the values of these variables, producing ordered and unordered datasets and informing the subsequent choice of data cleaning or transformation. This methodological procedure helped verify numerical consistency and informed subsequent decisions regarding data cleaning or transformation. The only missing values were those that warranted further remediation, as no significant inconsistencies were found.

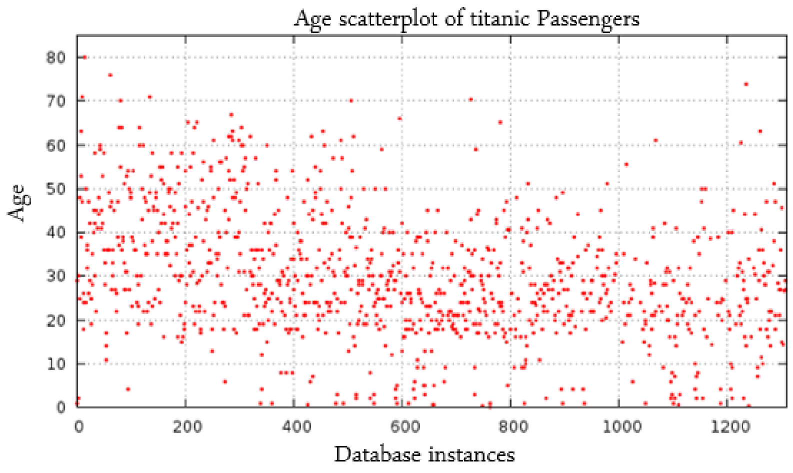

Figure 2. Plotting the age of the passengers with the number of occurrences

The age distribution shown in Fig. 2 indicates that passenger ages are widely distributed across the dataset, with infants on one end and 80-year-olds on the other. The appearance of values such as 0.16666 is used to demonstrate internal consistency and the absence of errors, as zero age in the dataset encodes infants under one year. The above findings indicate that the age attribute is valid and requires no correction.

In examining the *fare* feature, classifying values by passenger class yields significant structural features. In the first-class group, as shown in Fig. 3, most fares are distributed in the expected middle range, and only a few extreme values exceed £512. Archival sources indicate that these extremes pertain to members of the Cardoza family and their servants, who occupied one of the most expensive suites on the Titanic; therefore, these outliers are true high-fare cases and not anomalies [40].

In the second class, as shown in Fig. 4, fares are largely concentrated between £10 and £40, with a small cluster of higher values. These higher fares correspond to families or travel groups sharing a single ticket, which explains the increased cost.

Third-class fares, as shown in Fig. 5, exhibit the most significant variability, but the elevated values are attributable to group ticket purchases rather than anomalous data entries. Historical passenger records confirm that several large families shared tickets priced significantly above the average.

Overall, the visual analyses confirmed that the observed outliers reflect genuine historical circumstances rather than data errors. Therefore, the primary preprocessing requirement for this attribute is to handle missing or zero-valued entries, rather than to modify valid extreme values.

3.4. Treatment of Missing Features

The Titanic dataset has many missing features. More precisely, 7 of the 14 features present have missing values, with some features absent in more than 70% of instances. YaDT supports missing features, but it is still necessary for these features to be treated before feeding the algorithm.

3.5. Feature Engineering

Upon completion of Data Cleaning and Selection, the Dimensionality Reduction technique was applied to the dataset to remove features via Feature Selection, as described previously. To achieve this, some features were analysed and removed, as their Information Gain was too low or their presence would render the model overly simple for pattern discovery. Before E.C. was applied, a test decision tree was generated using the “raw” dataset, i.e., all features were included. The tree generated from this base is shown in Fig. 6.

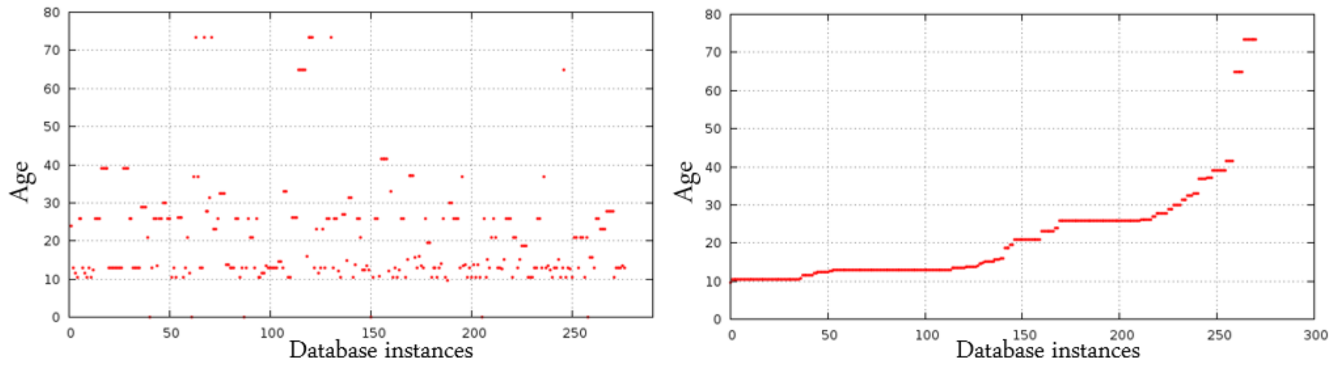

Figure 3. Graph of the fare paid by 1st class passengers

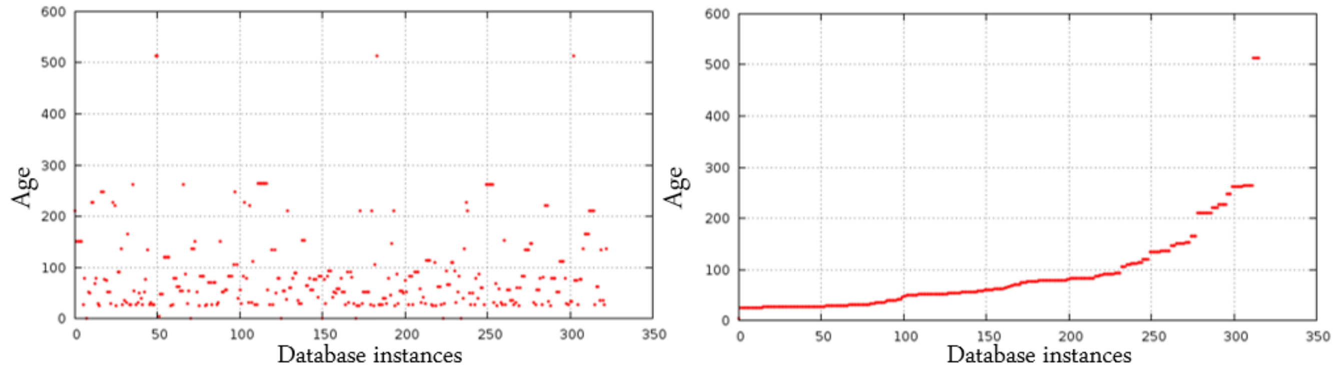

Figure 4. Graph of the fare paid by 2nd-class passengers

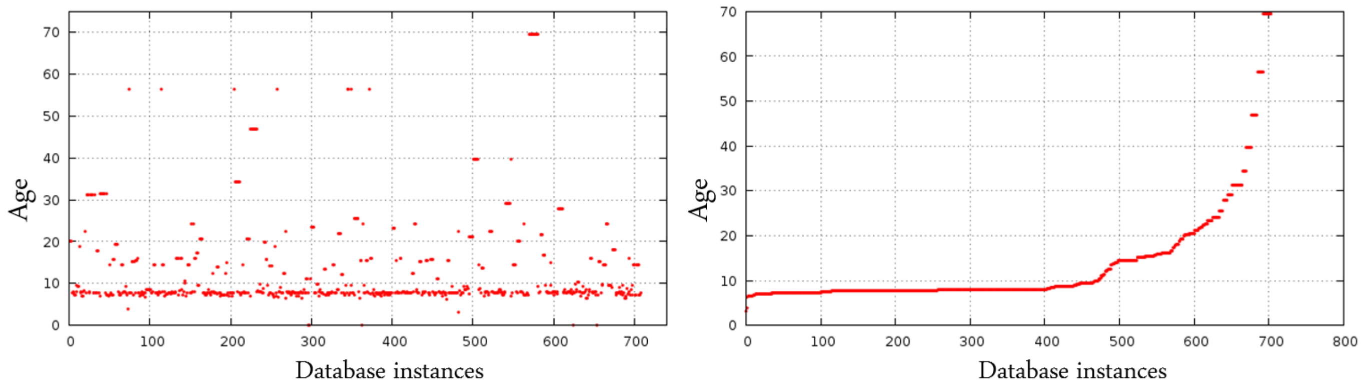

Figure 5. Graph of the fare paid by 3rd-class passengers

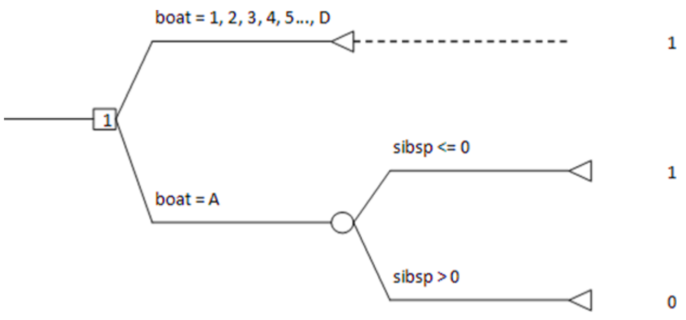

Figure 6. A decision tree was generated with all data from the Titanic dataset.

As you can see, the constructed decision tree is inferior; if you were in the lifeboat, you would have survived. If he wasn't, he died. Despite achieving an excellent accuracy on the test set, this tree does not help in discovering new patterns for DM. When these two characteristics are analysed together, it is evident why a tree like this was created: of 500 survivors, 477 were in an identified lifeboat, and of the 809 who lost their lives, only 9 were in lifeboats.

Analyzing a little deeper, it is possible to notice another dangerous feature: the body. Despite having only 121 valid values, it can significantly reduce the tree size. Whenever it encounters a valid value, the tree correctly classifies the instance as 0, since only dead people have a valid identifier.

Therefore, the boat and body are two characteristics that make the tree very weak for DM. Consequently, they were removed in the first round of the E.C. process. In addition, for the opposite reason, three other characteristics were removed: name, surname, and ticket. Because they have many categories, these characteristics serve as identifiers of the instance, making them unsuitable for the model's expected generalization.

3.6. Model Creation

After the Dataset Pre-Processing and Feature Engineering steps were completed, the first ML model was finally created. However, before viewing the model generated by the C4.5 algorithm, a brief description of how the YaDT system works is necessary.

In addition to the dataset, the system requires a second file containing the dataset's "metadata", that is, the description of each characteristic of the instances that make up the dataset. This metadata must be in the following format:

name, data,_type, column_type

$$name, data\_type, column\_type$$

Since the name is the name of the characteristic that will be presented in the generated model, the datatype is the *data type* (integer, float, string, or null), and *column_type* is the type of the characteristic (numeric, categorical, or class). YaDT supports additional characteristics, such as weights, but in this work, these types are optional. The file containing the metadata used to generate the model was as follows:

- Class, integer, discrete
- Sex, string, discrete
- Age. Float, continuous
- Sibsp, integer, continuous
- Patch, entire, continuous
- Ticket, string, discrete
- Fare, float, continuous
- Cabin, string, discrete
- Embarked, string, discrete
- Home_dest, string, discrete
- Survived, integer, class

In this case, discrete corresponds to categorical characteristics, and continuous to numerical characteristics; the distinction is merely nomenclatural.

YaDT has a GUI; however, it is recommended that the system be run via the command line to provide greater control over C4.5's adjustable parameters.

As shown in the command line, two factors are used to build the tree: *CONFIDENCE* and *SPLIT*. Recommend using the default value for the confidence factor, 25% [14], [41].

To determine an appropriate **SPLIT** parameter, a sensitivity analysis was conducted in which 498 trees were generated using the same dataset while varying SPLIT values from 2 to 500 (with CONFIDENCE fixed at 0.25). The error rates on the training set indicate that the training set performs well at lower SPLIT values but then declines as SPLIT increases. The minimum error occurred at SPLIT = 6, with values above 10 producing progressively higher misclassification rates. When SPLIT approached 500, tree error exceeded 36%, confirming that significant SPLIT thresholds lead to overfitting and reduced generalization.

Using SPLIT = 6 as the optimal value, the **CONFIDENCE** parameter was evaluated similarly by generating 20 trees with CONFIDENCE ranging from 0.05 to 1.0 in increments of 0.05. This analysis allowed identification of the CONFIDENCE range that minimizes pruning error while avoiding over-pruning. The results provided the basis for selecting the CONFIDENCE level used in the final model configuration, as shown in Figs. 7 and 8.

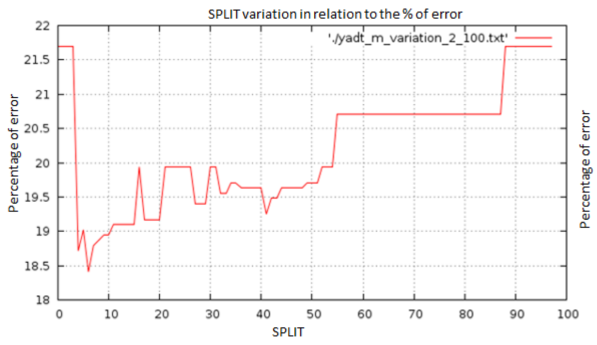

Figure 7. SPLIT variation from 2 to 100.

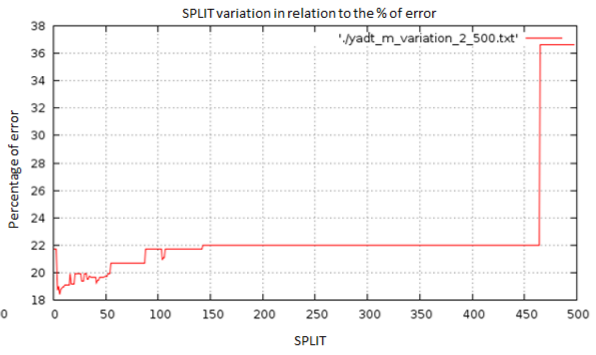

Figure 8. SPLIT variation from 2 to 500.

Fig. 9 shows that the home dest feature is missing from the model because its large number of categories reduced its Information Gain, potentially pruning it from the tree. Sex provided the most Information Gain, so it served as the tree's root node. The remaining features were used to create decision nodes.

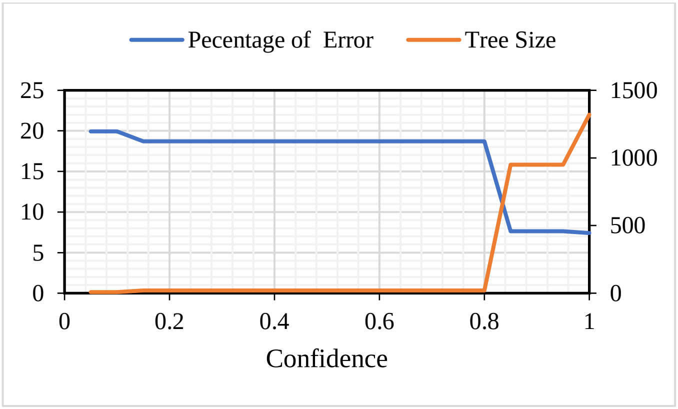

Figure 9. Tests for the optimal CONFIDENCE value - Titanic

When analysing CONFIDENCE values, consider the generated tree size and the error percentage. The smallest trees were obtained with CONFIDENCE less than 10%, but at a higher error rate than those with CONFIDENCE between 15% and 80%. Excellent error percentages were achieved from 85% onwards, at the expense of a large tree. In other words, an overgrown tree is caused by improper pruning. It was discovered that these error percentages were caused by the creation of new branches for all ticket categories or home dest that were not pruned, resulting in a large tree.

Thus, the AM model was created with CONFIDENCE = 0.25, a standard value with favourable test results, and SPLIT = 6. Finally, the first model was developed.

3.6.1. Model validation

The error percentage calculated by YaDT was 18.71%. In Table 1, the confusion matrix for this model is presented.

Table 1. Confusion matrix of the generated model - Titanic

TP = 765	FP = 201
FN = 44	TN = 299

Correct classifications were made for 765/809 cases as zero and for 299/500 cases as 1. About False Positive and Negative rates, 201/500 cases were mistakenly classified as 0, or 40% of the total, and 44/809 cases were mistakenly classified as 1, or 5.4% of the total. All of the model's male instances perished as a result of algorithm pruning at the decision node, creating a significant discrepancy and damaging classification. The model classifies female instances with greater accuracy.

3.6.2. Evaluation design and metrics

Precision: This metric measures the proportion of correct positive predictions. It's (1) is as follows:

$$\textbf{Precision} = \frac{TP}{TP+FP} \tag{1}$$

Recall (Sensitivity): While precision focuses on the proportion of correct positive predictions, recall concerns the proportion of actual positives that the model correctly identifies. It's (2) is as follows:

$$\textbf{Recall} = \frac{TP}{TP+FN} \tag{2}$$

F1 Score: Since both precision and recall are essential for data analysis but are often in tension (i.e., improving one may reduce the other), the F1 score provides a balanced metric that accounts for both. It is the harmonic mean of precision and recall and provides a comprehensive measure of the model's quality in identifying both classes. It's (3) is as follows:

$$\textbf{F1 Score} = \frac{2*\text{Precision}*\text{Recall}}{T\text{Precision}+\text{Recall}} \tag{3}$$

Cross-validation and the confusion matrix were used to validate the model. The k-partitions technique was applied. YaDT error estimates are approximate, so the objective of this validation is to obtain a more accurate model error estimate.

To determine the optimal k value before figuring out the best values for SPLIT and CONFIDENCE. To accomplish this, the following C++ code was written: The algorithm is given both the desired k and the entire dataset. The instances are then stored as Instance objects, creating a vector after the dataset has been read. Next, divide this vector into k parts and shuffle it in a loop. In the end, k test bases (number of instances not in the equivalent test base) and k training bases (number of instances not in the test base) will be created, totalling 2k datasets.

You can find the complete code at <http://github.com/hialo/decisiontreebuilder>. The Instance class is created in the snippet above to hold base instances. Since the two bases used in this work yield integer results, the classification variable is of type int. Next, using k ranging from 0 to the base size minus one, or 1308 (Leave One Oct, a well-liked validation technique) [42], a Shell Script was written to carry out the Cross Validation algorithm.

The script will create all k possible trees for each k and store the results using the k-partition method. After producing all feasible outcomes within the range, a Python script computes the average error percentage for each algorithm iteration by reading the error percentage for each k-model from YaDT's matrix.txt file [43].

After the algorithm was run and the points were obtained, the following graph was obtained:

The first few k values exhibit the most significant fluctuations before the graph stabilizes. An optimal value of k cannot be determined because the training and test datasets differ significantly. The lowest error percentage may no longer hold in a subsequent cross-validation iteration. K is therefore 10 because, as Witten and Frank (2000) and Mitchell (1997) stated, this is the literature standard. Because the algorithm has a low computational cost, it requires only a small amount of processing power. Error percentage increases linearly with k in Fig.10.

Subsequently, 10 training and 10 test datasets were created, and the error was computed for each training and test set. In the end, the average of these errors was obtained. The average error obtained via cross-validation was 20.75%, approximately 2% higher than the error obtained when generating the model, as expected. The code used to read all error percentages generated during the Cross-Validation iterations is the same as that used to read the error percentages to determine the optimal k, except that it modifies the base addresses.

Furthermore, with the help of the Confusion Matrices generated by each model during the execution of the k-partition technique, the values referring to their True Positive Rates (TPR) and False Positive Rates (FPR) were calculated for each model generated as (4) and (5).

$$TPR = TP/(TP + FN) \tag{4}$$

$$FPR = 1 - TN/(FP + TN) \tag{5}$$

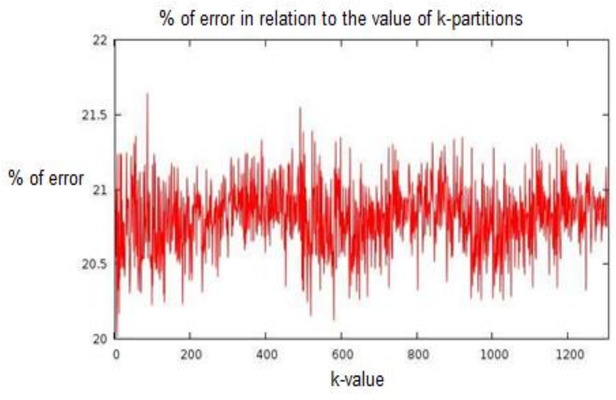

Figure 10. The percentage of error in the value of k

3.7. Dataset – PNCRC/Cattle

3.7.1. Dataset description

MAPA (Ministry of Agriculture, Livestock and Supply) provided the dataset for analysis at this stage. This dataset was collected by the PNCRC (National Waste and Contaminant Control Plan) [44], a federal inspection program for food production chains. Its purpose is to monitor production system controls and the quality and safety of animal and vegetable products for commercial and consumer use.

The PNCRC's two large areas each have their own set of regulations. In FIS-registered businesses, several samples are collected using a statistical plan. All FIS-registered slaughterhouses participate in weekly sample collection draws. A computerized system, SISRES (Waste and Contaminant Control System) [45], manages sample forwarding and laboratory results, plan information, and a random selection of establishments from which Ministry of Agriculture inspectors will collect samples. On average, 52 weekly draws are held nationwide from January 1 to December 31.

Following analysis, if the MRL (Maximum Residue Limit) is exceeded or a prohibited drug is used, the Ministry of Agriculture takes immediate regulatory action to keep the contaminated product from reaching the market. In addition, property actions are taken to determine the cause of the infraction.

The dataset contained analyses of bovine samples from 2002 to 2014. This dataset includes only laboratory-analysed substances, such as antiparasitic. The prediction class is a categorical variable with three possible classifications: no detection, detection with values below the MRL, and detection with values above the MRL. Each of the 32,114 instances in the analysed dataset possesses 23 characteristics. Table 2 summarizes the base instances.

Table 2. Description of the PNCNR bovine sample dataset

Feature	Description	Type	Non-null values
YEAR	Year in which the analysis was carried out	categorical	32114
N_ANALISE	Analysis ID No.	categorical	32114
FIS	Code Federal Inspection Service	categorical	32114
NOME_PROPR	Property Name in which the sample was obtained	categorical	31255
IF	Property UF	categorical	31302
COD_PROPR	Identification code of the property	categorical	698
PROPRIET_NAME	Owner's name	categorical	31362
MUN_NAME	Name of the municipality	categorical	31355
Zip code	First 5 digits of the Zip code of the property	categorical	30315
CEP2	Last 3 digits of the Zip code of the property	categorical	24543
COD_SPECIE	Species code Sample Group ID	categorical	31472
COD_TIPAN	Which tested substance does it belong to	categorical	31472
QTY_ANIMALS	Number of animals in the tested batch	numerical	31212
WEEK	Week of analysis	categorical	31472
END_PROPR	Address of the property	categorical	14197
PERSON_TYPE		categorical	14348
UF_PROPRIET	Owner's UF	categorical	14197
MUN_PROPRIET	Municipality of origin From the owner	categorical	14181
CEP_PR	Owner's zip code	categorical	14197
COD_RESIDUO	Substance code	categorical	31472
RESULT	Test Result of the substance	numerical	13190
SIT_VIOLAC	If there was a violation (1) Sample or Not (0)	categorical	20763
STATUS	Class that describes the Action That Will Be Taken	categorical	31472

3.7.2. Data cleaning and selection

As with the Titanic dataset, the dataset was preprocessed before being fed into the algorithm. As described in the following sections, some characteristics were examined with respect to instance-classification errors and the treatment of missing values. These steps were repeated using a Python script that employed the pandas and SciPy modules.

- **Identification of possible typographic inconsistencies:** As was done in the Titanic dataset, at this stage, possible discrepancies that could interfere with the reading of data carried out by YaDT were removed.

The numerical characteristics YEAR, N_ANALYSE, FIS, COD_TIPAN, CEP, and QTY_ANIMALS, due to the SISRES filter, included commas in values above 1000 when saved in the table, in addition to quotation marks. Again, using the $replace()$ function in conjunction with the $map()$ function to apply the function to all data frame elements, and commas and quotation marks were removed from such characteristics.

Furthermore, the CEP2 characteristic was 0. These values were replaced by their correct representation, 000, again using the map () functions and $lambda$ ().

- **Treatment of Missing Features:** The findings indicate that 16 of 23 dataset characteristics have missing values, with some missing more than 80%. Some base characteristics would be removed during the E.C. stage, making further treatment unnecessary.

This stage addressed UF, NAME_MUN, CEP, CEP2, QTY_ANIMALS, and RESULT. The characteristics UF, NAME_MUN, CEP, and CEP2 were addressed first. These characteristics are complementary because they pertain to the properties from which the samples were collected. Thus, one characteristic's value can be deduced from another. Table 2 shows that MUN_NAME has the fewest missing values (119).

When MUN_NAME was known, UF and CEP values were estimated. This enabled the accurate entry of 64 states and ZIP codes. The number of missing values in CEP2 was approximately 5000 higher than in CEP. When CEP existed, but CEP2 did not, CEP2 was 000, according to CEP data analysis. These values were filled in accordingly. Approximately 120 instances had missing values for these characteristics; these cases could not be deduced because UF, NAME_MUN, CEP, and CEP2 were missing, and, following the standard procedure, missing values were replaced with Unknown.

Filled in missing values in QTY_ANIMALS using the average of all instances, just as we did with the Titanic dataset's age characteristic. This mean was calculated, and missing values were filled with NumPy's $mean()$ function. According to the Ministry of Agriculture, the missing RESULT values were caused by SISRES filter errors and should have been 0. Thus, all of the missing values for this characteristic were filled.

3.7.3. Feature engineering

This phase started with a substitute. Specific instance characteristics were swapped for a meaningless mapped value because the Ministry considers them confidential.

These characteristics are FIS and PROPER_NAME, which pertain to the establishment of the sample's collection. An additional table was created, in which each FIS and its property were mapped to an index, FIS_INDEX, and the property was replaced with this index to preserve the privacy of the sample suppliers.

Owner data removal is also requested. NAME_PROPRIET, END_PROPRIET, TYPE_CURRENCY, UF_PROPRIET, MUN_PROPRIET, and CEP_PR were eliminated attributes. They did not examine the categories in the prior topic as a result. The removal of these features was safe because over 50% of them had multiple characteristics.

The removal of COD_PROPR resulted from this. As with the name of the Titanic, this characteristic occurs 712 times (2% of occurrences). Additionally, the COD_SPECIE characteristic was eliminated because it is exclusive to the dataset, which contains only beef-sample tests. CEP was created by combining the features of CEP and CEP2. The substance name from the current code was substituted for the COD_RESIDUO and COD_TIPAN values to aid in visualising the patterns in the generated model.

3.7.4. Model generation

After completing the Data Pre-Processing and E.C. stages, the AM model was generated using YaDT. As in the Titanic dataset, the best values for SPLIT and CONFIDENCE were first calculated before developing the definitive model using all instances of the dataset.

The methodology was the same: first, to determine the optimal SPLIT value by varying it from 2 to 500, using the default CONFIDENCE value of 0.25. The calculated value of SPLIT is then used to determine the optimal value of CONFIDENCE.

In Fig. 11, the variation in the error percentage with respect to the SPLIT value is shown. Unlike the Titanic dataset, the error rate remained relatively stable, with no variation from approximately 0 to approximately 70. Then, up to approximately 120, the error rate increases considerably before stabilizing again. As you can see, obtaining an optimal SPLIT value was also impossible. Therefore, the value used to generate the model was the minimum SPLIT value, 2.

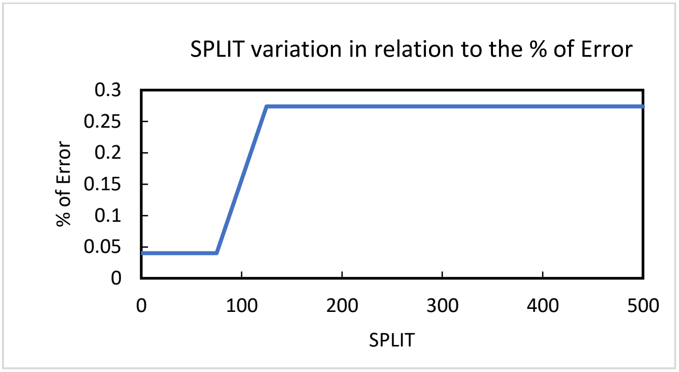

Figure 11. SPLIT variation from 2 to 500

Using the SPLIT value, the optimal CONFIDENCE value for model generation was determined. The same procedure was adopted: varying the percentage from 5 to 5%, all error percentages were obtained within the range of 5 to 100%, the maximum allowed. Fig.12 presents the results obtained.

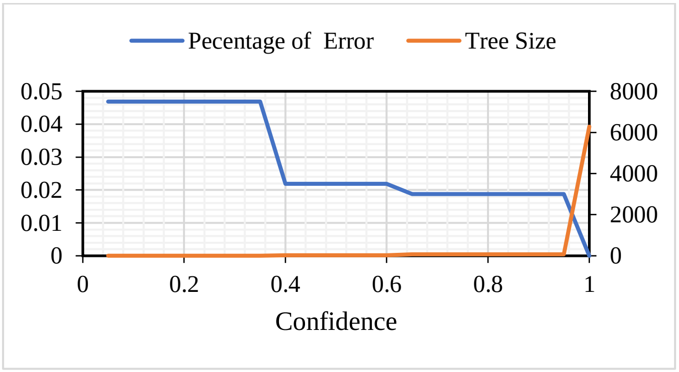

Figure 12. Tests for the optimal CONFIDENCE value - PNCRC/Cattle

When CONFIDENCE was selected, the tree size was taken into account. Among the trees grown, 5-35% were simple and had only one feature. The trees in the 40%-60% range were more complex, but even though they were simpler, they had a lower error rate. The error rate between 65 and 95% was slightly lower, but the trees were much bigger than the others. With 100% CONFIDENCE, the tree has serious overfitting problems. It is 100% accurate, but it depends on this dataset, which makes the results for test cases we know very little about.

The results were identical; therefore, SPLIT = 2 and CONFIDENCE = 0.40 were selected. This means there was no best SPLIT, like with the Titanic base. Used the right size tree, a reasonable error rate, and the same confidence level. The following model was made with the values given in Fig.13.

A small tree was trained for pattern extraction, achieving high accuracy: the model classified only 7 instances incorrectly out of 32114, yielding an error rate of 0.02%.

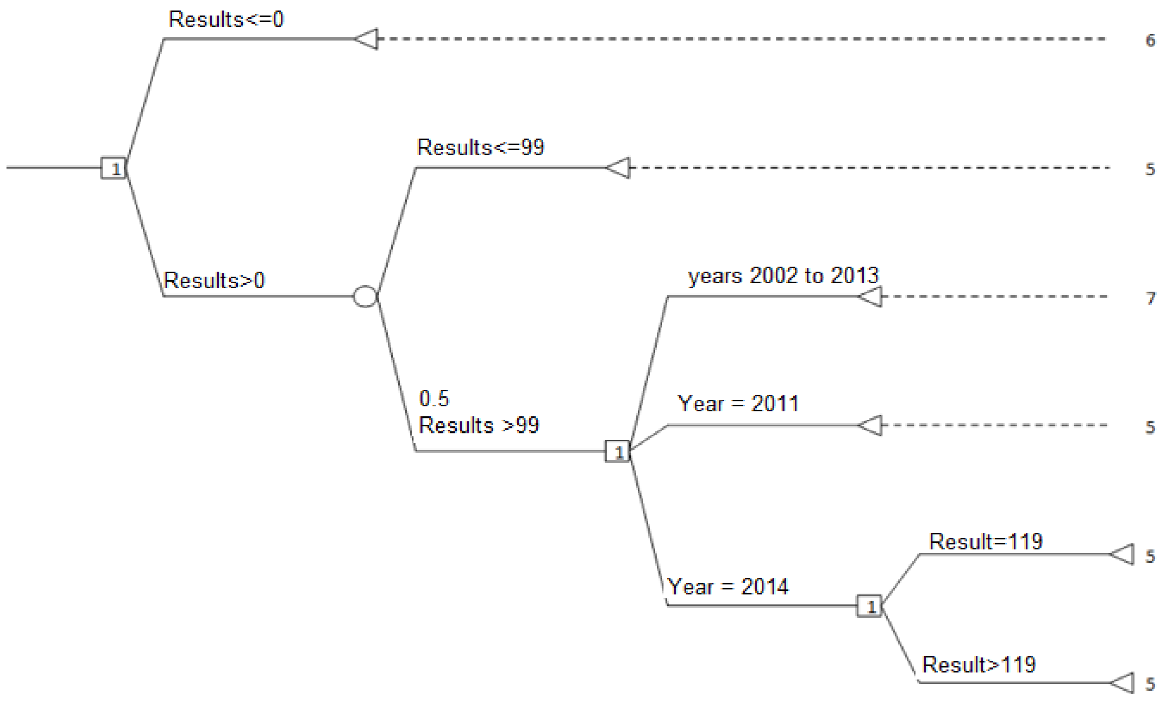

Figure 13. Decision Tree generated with data from the PNCRC/Bovines dataset.

This value differs from the previous model, which estimated 18% using the Titanic dataset. This suspiciously low value indicates overfitting, but this issue is addressed in the subsequent section on model validation.

3.7.5 Model validation

As previously stated, the model's error rate on the dataset was very low, approximately 0.02%. Table 3 shows the confusion matrix for the model in question:

As shown, the generated model did not miss any instances whose expected class was non-detection (STATUS = 6), i.e., samples that returned zero on the test. It is worth noting that the only issue the model encountered was in classifying instances with detection values more significant than the LMR allowed with STATUS = 5, but not equal to 7. The samples that did not exceed the LMR were correctly classified. The number of instances misclassified as 5 was 7/68. The C4.5 Decision Trees model was applied in this work for analysis. The precision, recall, and F1 score results are detailed and explained below.

- Precision: 72.68%
- Recall: 72.33%
- F1 score: 72.87%

The precision of the studied C4.5 Decision Trees indicates the proportion of correct predictions among all the predictions made. A precision of 72.68% means that the model correctly classified 72.33% of the instances. The recall (sensitivity) reflects the model's ability to identify positive instances correctly; in this case, 72.33%. The F1 score is a metric that combines precision and recall into a single figure, 72.87% in this case, indicating a balance between the two.

Table 3. Confusion matrix of the generated model - PNCRC/Cattle

Expected/Obtained			
	STATUS = 5	**STATUS = 6**	**STATUS = 7**
STATUS = 5	814	0	7
STATUS = 6	0	31127	0
STATUS = 7	0	0	6

When asked to classify new test data, cross-validation was used again to assess the model's performance. Because the algorithm used to generate the training and test datasets was random, the test used to determine the optimal k varied widely; therefore, k = 10 was chosen again. Thus, chose the literature-consensus value. An error was calculated for ten training and ten test datasets. The mean of these errors was computed.

Using cross-validation, the model's average error was 0.06%, which remains negligible. In this case, the model correctly identified all instances; however, some class 5 iterations contained errors, indicating imperfect generalization. This behavior is revealed by inspecting the ROC curve for this model.

In addition to cross-validation, the ROC curve was used to assess model performance. The ROC curve is important in this context due to the information trade-off. TFP lowers DVT, as evidenced by confusion matrices and ROC curves. This trade-off must be handled with caution in real life. Sometimes trading precision for false positives is worthwhile, and vice versa. In the PNCRC, it is extremely serious for a contaminated sample whose tested substance exceeds the MRL to be classified as healthy; therefore, any model that could make this error must be avoided at all costs, even if it requires less accuracy. According to Brink and Richards (2014), there is no free lunch.

The model's ROC curve was then generated using the one-versus-all method. Only binary classifiers can use the ROC curve technique, which plots the positive and negative class rates. However, this technique enables adaptation of the ROC curve for multi-class classifiers [46]. The one-versus-all technique creates a curve by labelling one of the model's n classes as positive and the remaining classes as unfavourable. In the end, a graph will have n curves, each with one positive class and the rest negative. It resembles cross-validation. This model's ROC curve comprises 30 points: three curves of 10 points each, computed from the confusion matrices of the k cross-validation iterations.

Fig.14 shows the model's ROC curve. You can see that the curve was very close to the ideal classifier, which produces only True Positives and no False Positives. It was so close that the figure had to be magnified to show the model's points. The black line depicts the class 5 curve, the red line represents the class 6 curve, and the correct green line depicts the class 7 curve. The points depict the created ML model, with each colour representing a curve. Because of how the graph was created, the points were extremely close to the graph threshold, making them difficult to see.

Fig.15 shows an expanded ROC curve. The X-axis ranges from 0 to 0.002, while the Y-axis spans from 0.5 to 1. This graph does not include the point labelled "class 6) because it is outside this range. The class 7 curve performed the worst when compared to the Y axis. This means it has the lowest True Positive values, which is understandable given that the class makes the most mistakes.

These are the findings of the study that examined two very different datasets. The results presented in this section are discussed in greater detail below, as well as suggestions for future work.

4. Discussion

4.1. First Stage

This first stage of the proposed work enabled the author to develop a high-quality theoretical framework for implementing the project in the second stage.

4.2. Second Stage

The second evaluation stage of the Decision Trees algorithm for DM took place between July and December 2014. C4.5 was implemented using YaDT, as previously stated. It produced better results than others, but the algorithm must be validated before use. This is the primary reason for working on these two datasets: the Titanic dataset, which is well-known and studied by DM experts, provided an excellent opportunity to test the algorithm's effectiveness using global results and the ship's history. As the primary evaluation metric, the algorithm's Kaggle classification was used. This was then applied to a real-world problem from the PNCRC/Bovine dataset.

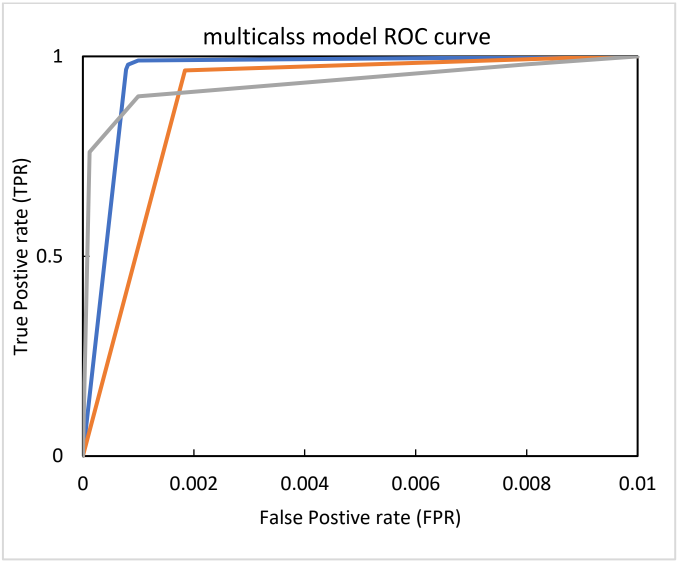

Figure 14: ROC curve of the multi-class model

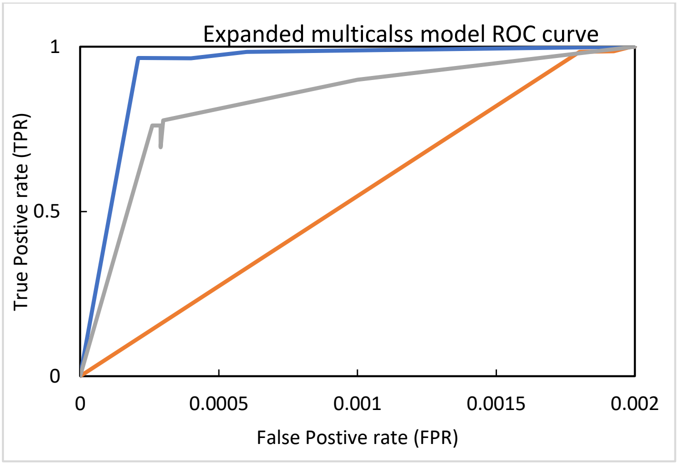

Figure 15. ROC curve of the expanded multi-class model

4.2.1. Results-titanic

YaDT produces a consistent ML model, as evidenced by its acceptable results. The model's accuracy was 79%-81%, ranking it among the top 300 on Kaggle (2012), where the dataset was obtained. During tree construction, patterns supporting the Titanic story were discovered. Sex was the most prominent feature on the tree, followed by class. Titanica (2013) reported that 81% of male passengers died, compared to 28% of female passengers. In other words, pruning the tree based on the instance dying if it is male keeps the model's error rate constant, as 19% of male instances are classified incorrectly. Accident survivors backed up the claim that 97% of first-class women and 90% of second-class women were saved. Only 46% of the women in the third class survived, so the tree continued to grow to protect the model's generalization. Also discovered that YaDT supports various tree-construction adjustments, including the confidence value used in the C4.5 algorithm's pessimistic pruning and the minimum number of leaf-node cases [47]. This enables finer-tuned model construction for a specific case, such as identifying optimal variable values. In conclusion, the algorithm and model performed admirably, justifying their use in the second scenario, a real-world application.

4.2.2. Results-PNCRC/Cattle

Positive outcomes were anticipated when comparing the algorithm's output with the Titanic dataset, given that it uncovered several intriguing patterns associated with the ship's history. However, this differed from the model generation results. Despite its precision, the algorithm-generated tree can help validate temporal, regional, or substance-specific patterns. To develop a reliable classifier using a tree with only two attributes, one of which (RESULT) is a class indicator. However, this only accomplishes the primary objective. It is important to emphasize that this is not the algorithm's fault; YaDT produced a nearly flawless tree by considering the requirements of an effective classifier, yielding commendable generalizability [48]. The inefficiency of the results is thought to have been caused by the base model itself: the algorithm was unable to verify patterns within the scant data to be mined due to the small number of truly relevant instances (instances with class 5 represented approximately 2% of the base, and instances with class 7 represented only about 0.2%). In other words, additional representation is required in the dataset to locate the patterns that MAPA seeks [49].

5. Conclusions

The findings of this work demonstrate that the methodological objectives were successfully met. The original validation, conducted on the Titanic dataset, showed strong, well-differentiated performance and, as such, validates the effectiveness of the YaDT implementation and the C4.5 algorithm's stability in a well-structured experimental environment. The observed accuracy levels are further supported by historical reliability and extensive benchmarking that preceded this work, lending credence to the soundness of the approach's methodology. Conversely, in the application to the PNCRC/Cattle dataset, the model exhibited severe limitations, including an inadequate sample size and an uneven distribution, largely attributable to the minority classes (5 and 7). Such an imbalance also necessarily limited the algorithm's ability to identify meaningful behavioural patterns that could help MAPA identify important substances. As a result, to improve predictive performance in that domain, the dataset should incorporate improved sampling methods, particularly a more balanced representation of the key classes. Although the PNCRC/Cattle data do not provide optimal conditions for demonstrating the full capabilities of the decision trees in data mining, our endeavours nevertheless demonstrate the salient attributes of the algorithm, including its interpretability and consistent classification behaviour. Further work should include a deeper analysis of MAPA's data structure and data collection methods, with the objective of identifying features that enable more efficient pattern discovery and, thus, enhance the appropriateness of decision trees or other models in this particular setting.

Acknowledgements

The authors would like to thank the Journal of Engineering and Sustainable Development.

Conflict of interest

The authors confirm that there is no conflict of interest.

Author Contribution Statement

Qutaiba Humadi Mohammed proposed the problem statement, developed the theory, performed the computations, and determined the objective.

Chaitanya Konda & Anupama Namburu verified the analytical methods and investigated and supervised the findings of this work.

All authors discussed the results and contributed to the final manuscript.

References

[1] R. Ragavi, B. Srinithi, and V. S. Anitha Sofia, “Data Mining Issues and Challenges: A Review,” *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 7, no. 11, pp. 118–121, Nov. 2018, doi: https://doi.org/10.17148/ijarcce.2018.71125.

[2] P. Kordjamshidi, D. Roth, and K. Kersting, “Declarative Learning-Based Programming as an Interface to AI Systems,” *Frontiers in Artificial Intelligence*, vol. 5, Mar. 2022, doi: https://doi.org/10.3389/frai.2022.755361.

[3] A. Zia, M. Aziz, I. Popa, S. A. Khan, A. F. Hamedani, and A. R. Asif, “Artificial Intelligence-Based Medical Data Mining,” *Journal of Personalized Medicine*, vol. 12, no. 9, p. 1359, Aug. 2022, doi: https://doi.org/10.3390/jpm12091359.

[4] D. Wang, T. Miwa, and T. Morikawa, “Big Trajectory Data Mining: A Survey of Methods, Applications, and Services,” *Sensors*, vol. 20, no. 16, p. 4571, Aug. 2020, doi: https://doi.org/10.3390/s20164571.

[5] J. R. Niaf, A. K. Kadhim, Q. H. Mohammed, H. K. Hoomod, and M. M. Salman, “New Cloud Computing Authentication Based on Secure Hash Algorithm (SHA-3) and Lightweight Sosemanuk Algorithm,” *Lecture notes in networks and systems*, pp. 207–221, Jan. 2025, doi: https://doi.org/10.1007/978-981-97-7603-0_19.

[6] L. Meng, B. Bai, W. Zhang, L. Liu, and C. Zhang, “Research on a Decision Tree Classification Algorithm Based on Granular Matrices,” *Electronics*, vol. 12, no. 21, pp. 4470–4470, Oct. 2023, doi: https://doi.org/10.3390/electronics12214470.

[7] F. Kibrete, T. Trzepieciński, H. S. Gebremedhen, and D. E. Woldemichael, “Artificial Intelligence in Predicting Mechanical Properties of Composite Materials,” *Journal of Composites Science*, vol. 7, no. 9, p. 364, Sep. 2023, doi: https://doi.org/10.3390/jcs7090364.

[8] Z.-H. Zhou, “Three perspectives of data mining,” *Artificial Intelligence*, vol. 143, no. 1, pp. 139–146, Jan. 2003, doi: https://doi.org/10.1016/s0004-3702(02)00357-0.

[9] H. M. Ibrahim, M. A. Shyaa, A. N. Yousif, and A. J. Ouda, “Data Mining Technique And Evaluation In Iraqi Named Crime Documents,” *International Journal of Psychosocial Rehabilitation*, vol. 24, no. 6, p. 2020, https://www.psychosocial.com/index.php/ijpr/article/download/7389/6625/13275.

[10] J. M. Barrios and P. E. Romero, “Decision Tree Methods for Predicting Surface Roughness in Fused Deposition Modeling Parts,” *Materials*, vol. 12, no. 16, p. 2574, Aug. 2019, doi: https://doi.org/10.3390/ma12162574.

[11] A. V. de Oliveira, M. C. S. Dazzi, A. M. da R. Fernandes, R. L. S. Dazzi, P. Ferreira, and V. R. Q. Leithardt, “Decision Support Using Machine Learning Indication for Financial Investment,” *Future Internet*, vol. 14, no. 11, p. 304, Oct. 2022, doi: https://doi.org/10.3390/fi14110304.

[12] Z. Xiaoliang, Y. Hongcan, W. Jian, and W. Shangzhuo, “Research and application of the improved algorithm C4.5 on Decision tree,” *2009 International Conference on Test and Measurement, Hong Kong*, , pp. 184-187.2009. doi: https://doi.org/10.1109/ICTM.2009.5413078.

[13] R. K. Amin, Indwiarti, and Y. Sibaroni, “Implementation of decision tree using C4.5 algorithm in decision making of loan application by debtor (Case study: Bank Pasar of Yogyakarta Special Region),” 2015 3rd International Conference on Information and Communication Technology (ICoICT), May 2015, doi: https://doi.org/10.1109/icoict.2015.7231400.

[14] I. H. Witten and E. Frank, “Data mining,” *ACM SIGMOD Record*, vol. 31, no. 1, p. 76, Mar. 2002, doi: https://doi.org/10.1145/507338.507355.

[15] W. Ibrahim, S. Abdullaev, H. Alkattan, O. A. Adelaja, and A. A. Subhi, “Development of a Model Using Data Mining Technique to Test, Predict and Obtain Knowledge from the Academics Results of Information Technology Students,” *Data*, vol. 7, no. 5, p. 67, May 2022, doi: https://doi.org/10.3390/data7050067.

[16] K. Abe, "Data Mining and Machine Learning Applications for Educational Big Data in the University," *2019 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*, Fukuoka, Japan, 2019, pp. 350-355, doi: https://doi.org/10.1109/DASC/PiCom/CBDCom/CyberSciTech.2019.00071.

[17] A. Cherfi, K. Nouira, and A. Ferchichi, “Very Fast C4.5 Decision Tree Algorithm,” *Applied Artificial Intelligence*, vol. 32, no. 2, pp. 119–137, Mar. 2018, doi: https://doi.org/10.1080/08839514.2018.1447479.

[18] N. Farag and G. Hassan, “Predicting the Survivors of the Titanic Kaggle, Machine Learning from Disaster,*” Proceedings of the 7th International Conference on Software and Information Engineering - ICSIE’18*, 2018, doi: https://doi.org/10.1145/3220267.3220282.

[19] M.Hakem, Z. Boulouard, and M. Kissi, “Classification of Body Weight in Beef Cattle via Machine Learning Methods: A Review,” *Procedia Computer Science*, vol. 198, pp. 263–268, 2022, doi: https://doi.org/10.1016/j.procs.2021.12.238.

[20] A. M. Rahmani et al., “Machine Learning (ML) in Medicine: Review, Applications, and Challenges,” *Mathematics*, vol. 9, no. 22, p. 2970, Nov. 2021, doi: https://doi.org/10.3390/math9222970.

[21] M.-W. Huang, C.-F. Tsai, S.-C. Tsui, and W.-C. Lin, “Combining data discretization and missing value imputation for incomplete medical datasets,” *PloS one*, vol. 18, no. 11, pp. e0295032–e0295032, Nov. 2023, doi: https://doi.org/10.1371/journal.pone.0295032.

[22] G. Gledec, M. Horvat, M. Mikuc, and B. Blaskovic, “A Comprehensive Dataset of Spelling Errors and Users’ *Corrections in Croatian Language,” Data*, vol. 8, no. 5, pp. 89–89, May 2023, doi: https://doi.org/10.3390/data8050089.

[23] F. Mauriello, A. Montella, M. Pernetti, and F. Galante, “An Exploratory Analysis of Curve Trajectories on Two-Lane Rural Highways,” *Sustainability*, vol. 10, no. 11, p. 4248, Nov. 2018, doi: https://doi.org/10.3390/su10114248.

[24] S. Tufail, H. Riggs, M. Tariq, and A. I. Sarwat, “Advancements and Challenges in Machine Learning: A Comprehensive Review of Models, Libraries, Applications, and Algorithms,” *Electronics*, vol. 12, no. 8, p. 1789, Jan. 2023, doi: https://doi.org/10.3390/electronics12081789.

[25] P. Linardatos, V. Papastefanopoulos, and S. Kotsiantis, “Explainable AI: A Review of Machine Learning Interpretability Methods,” *Entropy*, vol. 23, no. 1, p. 18, Dec. 2020, doi: https://doi.org/10.3390/e23010018.

[26] S. Jun, “Evolutionary Algorithm for Improving Decision Tree with Global Discretization in Manufacturing,” *Sensors*, vol. 21, no. 8, p. 2849, Apr. 2021, doi: https://doi.org/10.3390/s21082849.

[27] S.-W. Lin and S.-C. Chen, “Parameter determination and feature selection for C4.5 algorithm using scatter search approach,” *Soft Computing*, vol. 16, no. 1, pp. 63–75, May 2011, doi: https://doi.org/10.1007/s00500-011-0734-z.

[28] S. Sathyadevan and R. R. Nair, “Comparative Analysis of Decision Tree Algorithms: ID3, C4.5 and Random Forest,” *In Computational Intelligence in Data Mining-Volume 1: Proceedings of the International Conference on CIDM*, 20-21 December 2014, pp. 549-562. New Delhi: Springer India, 2014, doi: https://doi.org/10.1007/978-81-322-2205-7_51.

[29] L. Zhao, S. Lee, and S.-P. Jeong, “Decision Tree Application to Classification Problems with Boosting Algorithm,” *Electronics*, vol. 10, no. 16, p. 1903, Jan. 2021, doi: https://doi.org/10.3390/electronics10161903.

[30] S.-J. Lee, Z. Xu, T. Li, and Y. Yang, “A novel bagging C4.5 algorithm based on wrapper feature selection for supporting wise clinical decision making,” *Journal of Biomedical Informatics*, vol. 78, pp. 144–155, Feb. 2018, doi: https://doi.org/10.1016/j.jbi.2017.11.005.

[31] S. Alvarez-Rodríguez and F. G. Peña-Lecona, “Artificial Neural Networks with Machine Learning Design for a Polyphasic Encoder,” Sensors, vol. 23, no. 20, pp. 8347–8347, Oct. 2023, doi: https://doi.org/10.3390/s23208347.

[32] E. O. Kiyak, G. Tuysuzoglu, and D. Birant, “Partial Decision Tree Forest: A Machine Learning Model for the Geosciences,” Minerals, vol. 13, no. 6, pp. 01–15, Jun. 2023, doi: https://doi.org/10.3390/min13060800.

[33] S. Ruggieri, “YaDT: yet another decision tree builder,” 16th IEEE International Conference on Tools with Artificial Intelligence, Boca Raton, FL, USA, 2004, pp.260-265, doi: https://doi.org/10.1109/ictai.2004.123.

[34] N. Maaroof, A. Moreno, A. Valls, M. Jabreel, and M. Szeląg, “A Comparative Study of Two Rule-Based Explanation Methods for Diabetic Retinopathy Risk Assessment,” Applied Sciences, vol. 12, no. 7, p. 3358, Mar. 2022, doi: https://doi.org/10.3390/app12073358.

[35] J. D. Blischak, E. R. Davenport, and G. Wilson, “A Quick Introduction to Version Control with Git and GitHub,” *PLOS Computational Biology*, vol. 12, no. 1, p. e1004668, Jan. 2016, doi: https://doi.org/10.1371/journal.pcbi.1004668.

[36] H. K. Hoomod, M. A. Al-Hamami, A. K. Kadhim, Q. H. Mohammed, and Jolan Rokan Niaf, “MQTT Routing Optimizing Based Intrusion Detection for Internet of Things Using Hybrid Machine Learning,” *2024 International Conference on Decision Aid Sciences and Applications (DASA)*, Manama, Bahrain, 2024, pp. 1-4, doi: https://doi.org/10.1109/dasa63652.2024.10836235.

[37] C. S. Hong and T. G. Oh, “TPR-TNR plot for confusion matrix,” *Communications for Statistical Applications and Methods*, vol. 28, no. 2, pp. 161–169, Mar. 2021, doi: https://doi.org/10.29220/csam.2021.28.2.161.

[38] N. Ahmad and A. B. Nassif, “Dimensionality Reduction: Challenges and Solutions,” ITM Web of Conferences, vol. 43, p. 01017, 2022, doi: https://doi.org/10.1051/itmconf/20224301017.

[39] A. Dasgupta, V. P. Mishra, S. Jha, B. Singh, and V. K. Shukla, “Predicting the Likelihood of Survival of Titanic’s Passengers by Machine Learning,” *2021 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE), Dubai, United Arab Emirates, 2021, pp. 52-57*, https://ieeexplore.ieee.org/document/9410757.

[40] J. Bier, “Bodily circulation and the measure of a life: Forensic identification and valuation after the Titanic disaster,” Social Studies of Science, vol. 48, no. 5, pp. 635–662, Sep. 2018, doi: https://doi.org/10.1177/0306312718801173.

[41] T. M. Mitchell, *Machine learning*. New York: Mcgraw-Hill, 1997. Available: https://www.cs.cmu.edu/~tom/files/MachineLearningTomMitchell.pdf.

[42] G. I. Webb et al., "Leave-One-Out Cross-Validation," *n: Sammut, C., Webb, G.I. (eds) Encyclopedia of Machine Learning. Springer, Boston, MA.* ISBN: 9780387307688, pp. 600–601, 2011, doi: https://doi.org/10.1007/978-0-387-30164-8_469.

[43] Y. Ma, D. Liu, and L. Cai, "Deep Learning-Based Upper Limb Functional Assessment Using a Single Kinect v2 Sensor," *Sensors*, vol. 20, no. 7, p. 1903, Mar. 2020, doi: https://doi.org/10.3390/s20071903.

[44] A. A. Obaid, I. A. Rahman, I. J. Idan, and S. Nagapan, "Construction Waste and its Distribution in Iraq: An Ample Review," *Indian Journal of Science and Technology*, vol. 12, no. 17, pp. 1–10, May 2019, doi: https://doi.org/10.17485/ijst/2019/v12i17/144627.

[45] L. Al-Taie, N. Al-Ansari, S. Knutsson, and R. Pusch, "Hazardous Wastes Problems in Iraq: A Suggestion for an Environmental Solution," *Journal of Earth Sciences and Geotechnical Engineering*, vol. 3, no. 3, pp. 1792–9660, 2013, Available: https://www.scienpress.com/journal_focus.asp?main_id=59&Sub_id=IV&Issue=795.

[46] J. S. Aguilar-Ruiz and M. Michalak, "Multiclass Classification Performance Curve," *IEEE Access*, vol. 10, pp. 68915–68921, 2022, doi: https://doi.org/10.1109/ACCESS.2022.3186444.

[47] E. Indra, K. Ho, Arlinanda, R. Hakim, D. Sitanggang, and O. Sihombing, "Application of C4.5 Algorithm for Cattle Disease Classification," *Journal of Physics: Conference Series*, vol. 1230, p. 012070, Jul. 2019, doi: https://doi.org/10.1088/1742-6596/1230/1/012070.

[48] N. Lin, D. A. Noe, and X. He, "Tree-Based Methods and Their Applications," *In: Pham, H. (eds) Springer Handbook of Engineering Statistics. Springer Handbooks. Springer, London*, pp. 551–570, Jan. 2006, doi: https://doi.org/10.1007/978-1-84628-288-1_30.

[49] S. Kumar, A. Rai, and A. Kumar, "Decision Tree Based Models for Classification in Agricultural Ergonomics," Statistics and Applications, vol. 12, pp. 21–33, 2014, Accessed: Aug. 09, 2024. [Online]. Available: https://ssca.org.in/media/3Sadhu.pdf.