I.J. Modern Education and Computer Science, **2025, 6, 127-145**
Published Online on December 8, 2025 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijmecs.2025.06.09

Dynamic Data Mining: Using Dynamic ID3 Algorithm to Solve Any Problem that Needs Decision Tree Support

Amir Amjad Gharbi*
Syrian Virtual University, Damascus, Syria
Email: amirgharbi995@gmail.com
ORCID iD: https://orcid.org/0009-0007-2911-2128
*Corresponding Author

Bassel Alkhatib
Syrian Virtual University, Damascus, Syria
Al-Sham Private University, Damascus, Syria
Email: t_balkhatib@svuonline.org, b.alkhatib.foit@aspu.edu.sy
ORCID iD: https://orcid.org/0000-0001-7589-2021

Received: 09 December 2024; Revised: 18 February, 2025; Accepted: 24 April, 2025; Published: 08 December, 2025

Abstract: Most programmers and users resort to find individual solution per problem depending on the data and nature of problem, that will lead to solve a specific problem using an algorithm without the ability of this solution to solve a new problem. This variance comes from the difference in algorithm parameters from one problem to another, as these parameters related to data nature, its size, and values it carried that can affect the way algorithm work. Individual solutions lead to increase in time cost and effort spent on solving a new problem, which the new problem requires to work on programming new criteria for algorithm solution. That is prompted us to highlight necessaries to develop main components for algorithms used in practical life, such as data mining algorithms so that a solution designed for one problem can be more easily adapted to new problems with different data structures, within the general scope of decision tree applicability. These algorithm components need control mechanism settings, so when using component to solve problem, there is no need to develop algorithm settings again, regardless data size and data structure. We found that the dynamic solution saves effort and time needed to solve problems with same algorithm. In this paper, we present our methodology for using ID3 decision tree algorithm to mine data dynamically, and the mechanism used to achieve the dynamic solution, that provides a flexible and reusable solution for a wide range of problems that require decision tree support, reducing the need to redesign or reimplement models for each new task The proposed model was tested on three datasets. The proposed model achieved an accuracy of 97%, 97%, and 93% on the breast cancer, heart disease, and diabetes datasets, respectively.

Index Terms: Dynamic Data Mining, Decision Tree ID3, Algorithm Parameters, Stored Dataset Criteria, Performance Measures

1. Introduction

Data mining techniques began to develop in the 1990s. During the initial phase, data information was available in computers, disks and tapes. Data mining has become an emerging technology in the digital age due to its efficiency in handling big databases with multi-processor computers. Data mining uses advanced techniques to discover patterns from data, with many tasks: association rules, classification, clustering, and outlier discoveries. Classification is an important and effective task in data mining to solve complex real-world problems, such as accuracy and risk prediction [1]. Most of these studies rely on specific features of the data set and a standard algorithm, such as defining fields and numeric categories for each feature, without the ability of this study to solve the problem of dynamic changes in the data that can occur in the real world and may lead to incorrect decisions [2, 3]. The goal of this study is to find a dynamic solution to ID3 decision tree algorithm [4, 5] that can solve a variety of classification problems and adapt to multiple decision support scenarios without rebuilding the decision support system from scratch.

Unlike many standard incremental decision tree algorithms that rely on fixed statistical triggers or stream-only data assumptions (e.g., Hoeffding Trees), the system presented here is designed as a dynamic, reusable platform. It offers user-level configuration of attribute importance, handles both categorical and numerical fields flexibly, and allows real-time reprocessing without hardcoded thresholds. This design makes it suitable for both streaming and batch data in diverse application domains.

First, we talk about some works similar to working with dynamic data mining. In the second section, we talk about the mechanism to achieve dynamic work, and ID3 decision tree algorithm and a practical example of how it works. In the third section, we talk about the methodology to achieve ID3 algorithm in a dynamic way and solve all the issues. In the last section, we talk about results we reached by applying the dynamic system to three decision-making problems: breast cancer, heart disease, and diabetes.

1.1 Research Hypothesis and Objective

This research is guided by the hypothesis that a user-configurable, dynamic ID3-based system can provide competitive classification accuracy while improving generalizability, reusability, and adaptability across different problem domains without retraining or refactoring from scratch. We aim to demonstrate that dynamic feature formation, numerical domain generation, and real-time tree construction enable the creation of flexible and scalable decision trees.

Traditional ID3 algorithms operate on static datasets and feature spaces. The goal of this research is to introduce a dynamic layer based on supervised learning theory, where the importance of features and the segmentation of numerical features are externally generated. This aligns with direct human learning models and allows the algorithm to continuously adapt to dynamic changes in real-world data.

2. Related Works

Christophe Marsala et al. [6] This research defines the ID3 algorithm mechanism that deals with dynamic data, where the training set is built gradually, when a new protected dataset is presented, a new rank will set for each feature and an independent tree will draw again. This strategic is very common for processing a big data. These incremental algorithms are based on windowing technology, where the tree is built several times, so that recording a new dataset can cause changes in decision tree form, these changes occur gradually from the tree root towards the leaves improving inner nodes values. The dynamic data that constitute evolving training sets can have several forms: temporal data that comes from time series or data result from periodic measurements of phenomena that develop over time, this leads to an increase in training set, and this should be taken into consider that the new data is more important than the old data that is already exist in training set.

While Marsala et al. [6] propose a fuzzy dynamic tree model that gradually adjusts with incoming data, their work focuses on window-based techniques and fuzzy logic, which are not easily applicable in general-purpose classification systems. Our approach differs in that it maintains decision tree adaptability through attribute-level control, user-defined numerical ranges, and real-time reconfiguration without requiring domain-specific tuning or fuzzy logic systems

George Mathew et al. [7] has performed a research that demonstrates a dynamically built prediction model using patients' data from multiple hospitals that can be used as a decision support tool. This tool processes attributes-based queries, that helps to build a decision support model from many hospitals when there are not enough local patients' information to draw conclusions. This study builds distributed prediction models using patients' data and analyzes 3 years patients' samples, this model can be used to help collaboratively predictive models build for better conclusions. Using 261 attributes to build model, show that hospitals collaboration with fewer than 100 hospitalizations was able to achieve a good improvement in hospitalization prediction accuracy collectively using a distribute model compared to local models.

Mathew et al. [7] presented a distributed learning model that focuses on federated data and hospital collaboration, primarily addressing data availability and privacy. In contrast, our work concentrates on dynamic behavior at the algorithmic level rather than distribution. We propose a solution that adapts the core structure of the ID3 algorithm for incremental updates, user-configurable logic, and system-wide reuse.

Chen et al. [8] introduced GBDT-IL, an incremental learning approach based on Gradient Boosting Decision Trees (GBDT) tailored for detecting botnet traffic in IoT environments. This method addresses the challenges posed by evolving botnets by adapting to dynamic IoT data using incremental learning. To enhance model performance and prevent overfitting due to redundancy in the incremental learning process, a tree-pruning mechanism is incorporated. While GBDT-IL focuses on botnet detection, its incremental learning strategy aligns with our system's emphasis on adaptability to evolving data streams.

Louren ço et al. [9] proposed DFDT, a novel algorithm designed for energy-efficient, memory-constrained data stream mining on edge devices. DFDT enhances the efficiency of Hoeffding tree growth by dynamically adjusting parameters such as grace periods, tie thresholds, and split evaluations based on incoming data. It incorporates stricter evaluation rules and adaptive expansion modes, allowing for more computation on frequently visited nodes while conserving energy on others. This approach resonates with our system's goal of dynamic adaptability, particularly in resource-constrained environments. Our system builds on the dynamic principle but provides greater customization,

allowing users to manually define attribute significance and generate decision trees suited to a broader range of static and streaming datasets.

Daghero et al. [10] explored the deployment of dynamic decision tree ensembles, such as Random Forests and Gradient Boosting Trees, on multi-core low-power IoT devices. Their approach adjusts the number of executed trees based on latency and energy targets, as well as the complexity of the processed input, to balance computational cost and accuracy. This dynamic ensemble method aligns with our system's emphasis on adaptability and efficiency, especially in edge computing scenarios. Although their work succeeds in time-sensitive environments, it is optimized for stream processing and lacks user-level control over attribute structures or thresholds. Our system builds on the dynamic principle but provides greater customization, allowing users to manually define attribute significance and generate decision trees suited to a broader range of static and streaming datasets.

In contrast to fully automated models like DFDT or GBDT-IL, our system emphasizes transparency and user agency. Attribute behavior, discretization strategy, and target field selection are all defined interactively, allowing users to reuse the framework across datasets without modifying the core algorithm. This balance between adaptability and configurability is a key distinction.

3. Methods

This research builds on existing work in supervised learning, particularly classification methods rooted in decision tree models. While classification is a core task in data mining, our focus is on making the tree construction process dynamic and adaptable, addressing a gap in most existing implementations of the ID3 algorithm. The core mechanism behind the system is the ID3 algorithm, which selects splitting attributes based on information gain. Rather than using it in its static form, we extend it with a dynamic layer that allows real-time updates and user-driven control over attribute weighting and discretization.

It is a recursive classification algorithm with decreasing data, and the result is a decision tree consisting patterns rules set from the tree root to the leaves through which can predict new data outcome [11]. It relies on dividing the problem principle, that solves each problem part separately, then gathers solutions. Decision tree is created by choosing best attribute, that the tree depth is reduced and data is classified correctly. The best gain attribute is calculated by:

- Calculating entropy H(S) for a dataset S as shown in equation (1).
- Calculate entropy for each category based on each attribute using equation (1)
- Calculate information gain (IG) when splitting a dataset D on a feature A can be described in equation (2).
- Select the top gain as the root node

If final decision (positive or negative) is not reached, it repeats the same steps with decreasing in data and until reach final leaves. Entropy is a homogeneity measure of dataset for binary classification (positive or negative).[12]

$$H(s) = -\sum_{i=1}^{n} pi(\log_2(pi)) \quad (1)$$

Where n is the number of classes and Pi is the proportion of class i instances within the dataset.

$$IG(D, A) = H(D) - \sum_{v \in Values(A)} \frac{|D_v|}{|D|} \cdot H(D_v) \quad (2)$$

Where:

- $H(D)$ is the entropy of the dataset D.
- $Values(A)$ is the set of all possible values that feature A can take.
- D_v is the subset of D for which feature A has value v.
- $|D_v|$ is the number of examples in $|D_v|$.
- $|D|$ is the total number of examples in D.
- $H(D_v)$ is the entropy of D_v, calculated in the same way as $H(D)$.

4. System Infrastructure

The proposed system is designed to enable the construction of a dynamic decision tree by extending the traditional ID3 algorithm to an adaptive data-driven framework. At its core, the system relies on the principle of continuous learning, the ability to process and incorporate new data samples over time without having to retrain the entire model from scratch. When new data is introduced, the system uses code from the scikit-multiflow and River libraries—which we added to the proposed system to introduce dynamic decision tree input—which enables the system to evaluate the consistency of this new data with existing decision paths and selectively recalculates entropy and information gain for

only affected branches. This design ensures that only relevant parts of the tree are updated, maintaining computational efficiency and avoiding unnecessary recalculations for unaffected nodes.

One of the most important features enabling this system is the modular architecture for managing datasets and attributes. Users can create new datasets or modify existing ones at any time, determine or modify the importance of attributes, define target variables, and set numerical thresholds. These configurations are not hard-coded, but rather stored dynamically, allowing the system to be reused across multiple problem domains. In particular, numeric attributes are managed using flexible boundary generation techniques, either by equal-width aggregation or a distributional properties-based arithmetic method, enabling the tree to dynamically interpret continuous data ranges with minimal user intervention.

The system is also designed to handle datasets of varying structures and sizes. Whether the inputs are small and class-specific or large with complex numeric attributes, the infrastructure adapts seamlessly. As new data flows in, it is assigned unique row identifiers and integrated into the existing dataset, triggering local modifications to the tree. This behavior mimics an online learning environment, where the tree structure evolves gradually, learning from patterns in real time.

To support secure and personalized access, the infrastructure includes OAuth-based user authentication, allowing each user to manage only their own datasets. Import and export tools facilitate integration with external sources, ensuring compatibility with real-world workflows. Together, these components create a robust and flexible environment in which the ID3 algorithm operates not as a static classifier, but as a living, evolving model capable of continuous learning, intelligent adaptation, and supporting decision-making in diverse and dynamic scenarios.

Fig. 1. shows Dataset Attributes Management system.

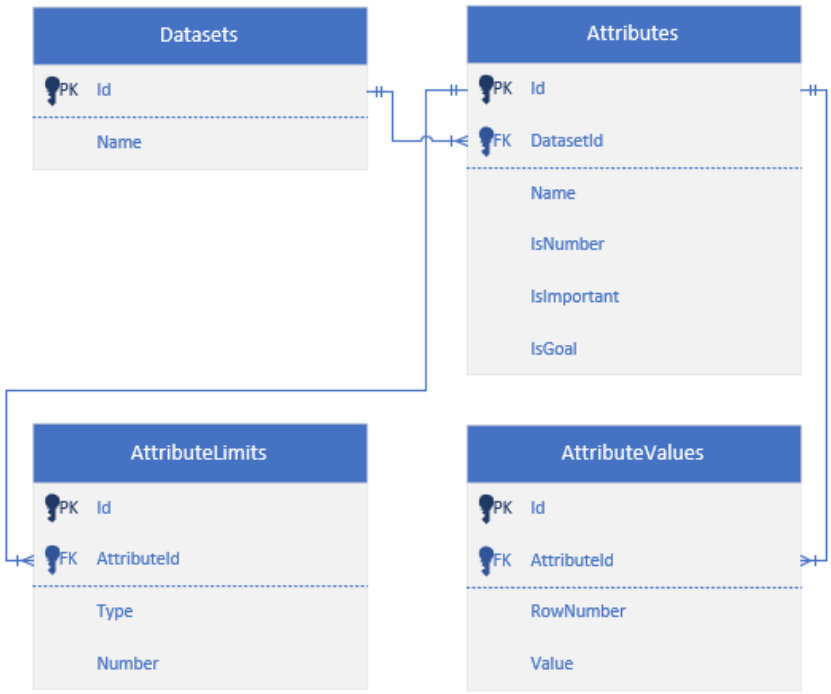

Fig. 1. Dataset Attributes Management system

4.1 Manage Datasets

It includes add a new dataset edit saved dataset, defines dataset name, attributes and specify attributes setting.

4.2 Manage Dataset Attributes

It includes add a new attribute and edit saved attribute (see Fig.2), defines attribute name, type, important to use it in ID3 algorithm, target, and limits for numerical attributes, it includes some restrictions:

- Attribute name is not repeated for dataset
- Target attribute must exist at most for dataset
- Important attribute must exist at least for dataset
- Numerical attribute must contain numerical limits

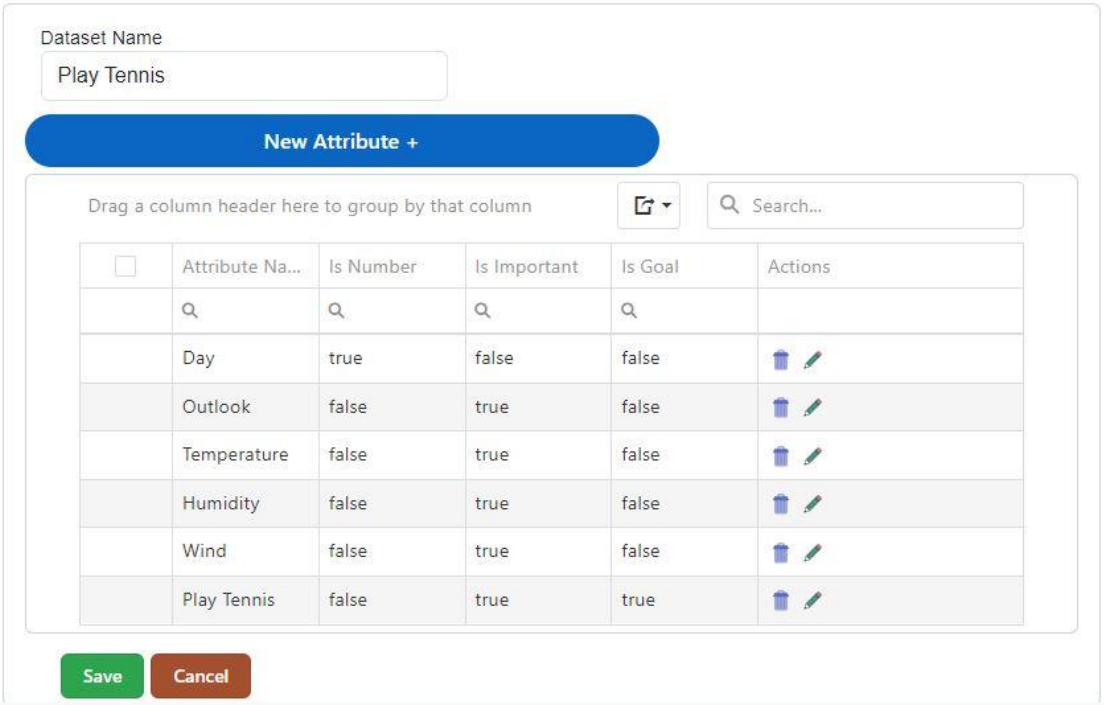

Fig. 2. Manage Dataset Attributes

It includes add a new limit for numerical attributes, type (larger, greater or equal to, smaller, smaller or equal to) and limit number. System sort limits based on limit type then based on limit number in order to form numerical fields during algorithm execution, with following conditions:

- Two limits with same type must not repeated
- Two limits with numbers must not intersect

4.3 Manage Dataset Items

It includes add dataset new item and edit saved item, fetches all dataset attributes and place each attribute with its value so that the following conditions are met: (see Fig. 3)

- All-important attributes values must be filled in
- Target attribute is Yes or No and must be filled in
- Numerical attribute filled with numerical values

Fig. 3. Manage Dataset Items

During process, attributes values have been stored and row number has been added to represent the data item, and its value is max (row number) + 1. When dataset information displayed in the table, all dataset attributes columns added to the dynamic table with attribute specifications such as type and name. Then we go over attributes values and rely on attribute row number and attribute primary key to add attribute value to the dynamic table. If the row number exists, attribute value set to the row number line at the attribute column. If the row number does not exist, a new line created and added to the dynamic table and attribute value set at the attribute column as shown in Table 1:

Table 1. Manage Dataset Items

Day	Outlook	Temperature	Humidity	Wind	Play	Actions
1	Sunny	Hot	High	Weak	No	
2	Sunny	Hot	High	Strong	No	
3	Overcast	Hot	High	Weak	Yes	
4	Rain	Mild	High	Weak	Yes	
5	Rain	Cool	Normal	Weak	Yes	
6	Rain	Cool	Normal	Strong	No	
7	Overcast	Cool	Normal	Strong	Yes	
8	Sunny	Mild	High	Weak	No	
9	Sunny	Cool	Normal	Weak	Yes	
10	Rain	Mild	Normal	Weak	Yes	
11	Sunny	Mild	Normal	Strong	Yes	
12	Overcast	Mild	High	Strong	Yes	
13	Overcast	Hot	Normal	Weak	Yes	
14	Rain	Mild	High	Strong	No	

4.4 Drawing a dataset decision tree (Dynamic ID3 algorithm)

It includes node structure and ID3 procedure details. To draw dataset decision tree (as can be seen if Fig. 4), we need to fetch dataset attributes setting: (name, type, important and target). Then fetch dataset attributes values with row number, and create list that contains all row numbers, and its attributes values dictionary. Then define the target attribute, determine important attributes that will be included in ID3 algorithm, and calculate negative cases and positive cases. The following pseudocode shows the detailed procedure of the proposed dynamic ID3 algorithm. While our system incorporates elements of incremental learning, it is not an online learner in the statistical sense. Instead, it provides a dynamic layer over the ID3 algorithm that adapts structurally to user-defined changes in dataset configuration or data growth—rebuilding only affected parts of the tree and retaining all manually assigned constraints.

Pseudocode: Dynamic ID3 algorithm

```
FUNCTION BuildTree(dataset, attributes, target, level, parent, parentValue):

  node ← CreateNode(level, parent, parentValue)
  classCounts ← CountClasses(dataset, target)

  IF all records belong to one class OR attributes is empty:
    node.isLeaf ← TRUE
    node.class ← MajorityClass(classCounts)
    RETURN node

  bestAttr ← NULL
  bestGain ← -∞

  FOR attr IN attributes:
    values ← (IsNumerical(attr)) ? GenerateDynamicRanges(attr, dataset)
                                 : GetDistinctValues(dataset, attr)
    gain ← ComputeInformationGain(dataset, attr, values, target)
    IF gain > bestGain:
      bestGain ← gain
      bestAttr ← attr

  node.attribute ← bestAttr
  values ← GetValues(bestAttr)

  FOR value IN values:
    subset ← Filter(dataset, bestAttr, value)
    IF subset IS EMPTY:
      child ← CreateLeaf(MajorityClass(classCounts))
    ELSE:
      newAttrs ← attributes - {bestAttr}
      child ← BuildTree(subset, newAttrs, target, level + 1, node, value)
    node.children.ADD(child)

  RETURN node
```

Attribute: Outlook	Positive: 9	Negative: 5	
Sunny >> Attribute: Humidity	Positive: 2	Negative: 3	
Value: High	Positive: 0	Negative: 3	Goal: false
Value: Normal	Positive: 2	Negative: 0	Goal: true
Value: Overcast	Positive: 4	Negative: 0	Goal: true
Rain >> Attribute: Wind	Positive: 3	Negative: 2	
Value: Weak	Positive: 3	Negative: 0	Goal: true
Value: Strong	Positive: 0	Negative: 2	Goal: false

Fig. 4. Play tennis decision tree result

The full system architecture is presented in Fig. 5

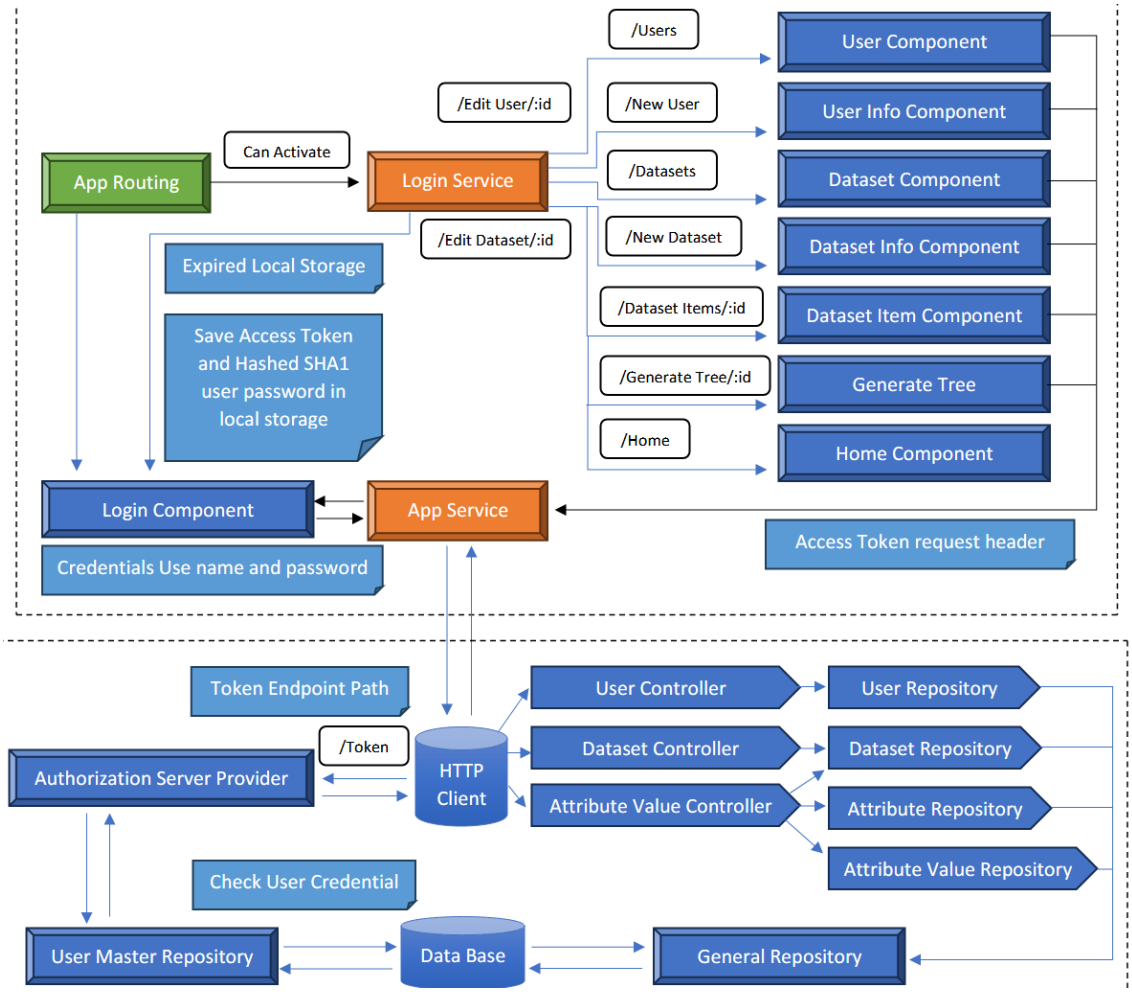

Fig. 5. The full system architecture

4.5 Performance metrics

Every learning model must be evaluated before trusting it for making decisions in real applications, here we have to find Confusion Matrix (see Fig. 6), which is a table used to show learning model efficiency that views different possible cases of right and wrong, where expected data from the learning model is compared with the real data, consists four sections:

1. True Positive: number of true positive cases
2. False Positive: number of false positive cases
3. False Negative: number of false negative cases
4. True Negative: number of true negative cases

- Accuracy measure

It is correct classifications percentage, which is the result of dividing the examples count that the classification is correct by dataset total count:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{3}$$

- Precision measure

It is correct classifications ratio of correct classified positive cases count to the examples count that were classified as positive:

$$Pr\,e\,cision = \frac{TP}{TP+FP} \tag{4}$$

- Recall measure

It is correct classifications ratio of correct classified positive cases count to the examples count that should be classified as positive:

$$Re\,c\,all = \frac{TP}{TP+FN} \tag{5}$$

- F1_Score measure

It is a measure that balances Precision and Recall in one value, if this measure is high, it means that both precision and recall are good, it means that measure can accept low precision with high recall, or high recall with low precision:

$$F1\_Score = 2 \times \frac{Pr\,ecision \times Re\,call}{Pr\,ecision + Re\,call} \tag{6}$$

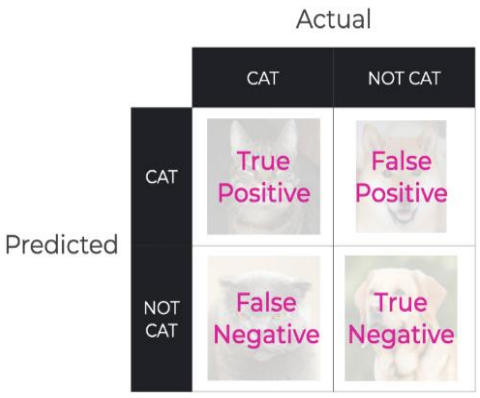

Fig. 6. Confusion Matrix (TP, FP, FN, TN)

5. Discussions

5.1 Breast Cancer dataset

- Diagnoses

We have imported a dataset of 569 patients, that contains nine attributes for breast cancer diagnosis[13]:

1) Radius: Distances average from the mass center to distances on the mass perimeter
2) Texture: Standard deviation of gray scale values

3) Perimeter: Primary tumor average value
4) Area: Cancer cells areas average value
5) Smoothness: Cells difference average within radius
6) Compactness: Estimating perimeter and area avg
7) Concavity: Shape concave parts intensity
8) Concave Points: Shape perimeter concave parts
9) Diagnoses: Malignant and Benign

- Attributes Setting

The limits for all numerical attributes shown in table 2:

Table 2. Breast cancer numeric attributes setting

Attributes\Limits	Minimum	First Limit	Second Limit	Maximum
Radius	6.98	12.94	15.49	28.11
Texture	9.71	17.74	20.93	39.28
Perimeter	43.7	83.4	101.6	188.5
Area	143.5	539.9	793.3	2501
Smoothness	0.052	0.09	0.101	0.163
Compactness	0.019	0.084	0.124	0.345
Concavity	0	0.058	0.124	0.426
Concave Points	0	0.032	0.068	0.2012

- Resulting Decision Tree

We note that diagnosis related to shape contour concave parts count, which represents the root attribute, and then decision tree divided into 3 branches based on (Concave Points) root attribute numerical limits fields, and every numeric field has max gain attribute. Fig. 7, Fig. 8, Fig. 9, Fig 10, Fig. 11 show breast cancer characteristics.

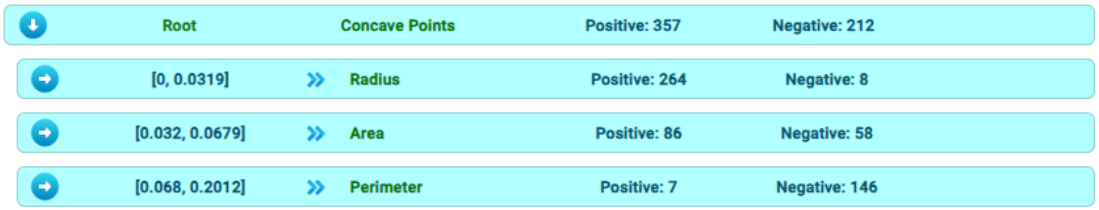

Fig. 7. Breast cancer decision tree root branches

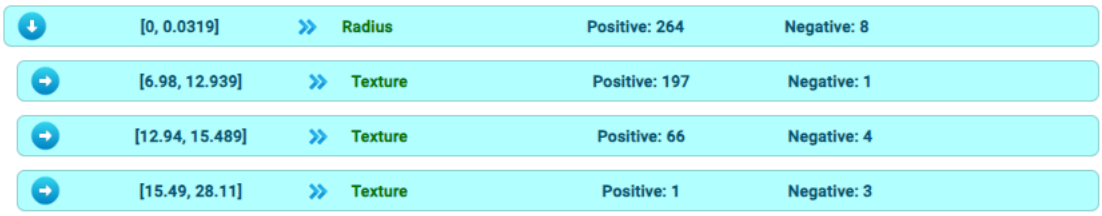

Fig. 8. Breast cancer - Concave Points [0, 0.319] - Radius

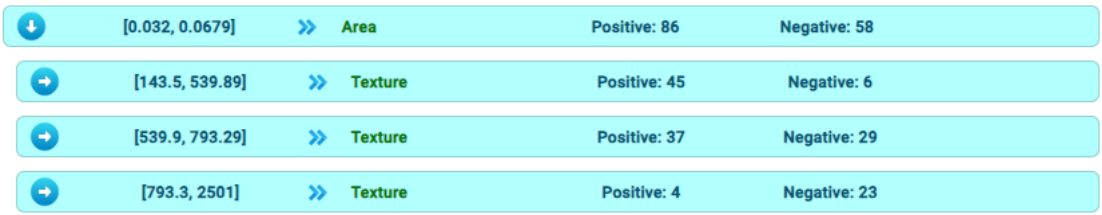

Fig. 9. Breast cancer - Concave Points [0.032, 0.0679] - Area

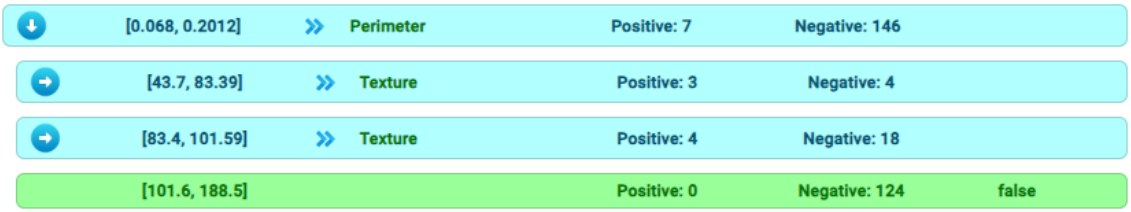

Fig. 10. Breast cancer - Concave Points [0.068, 0.2012] – Perimeter

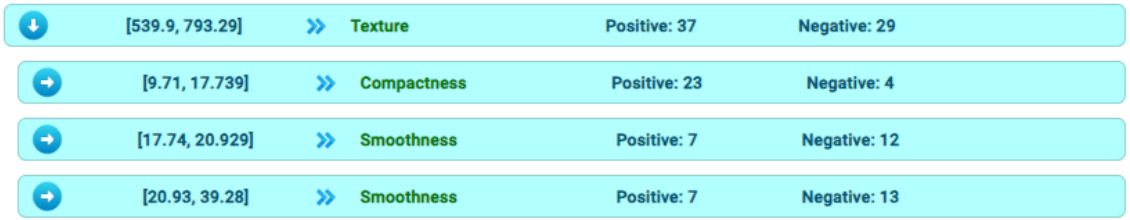

Fig. 11. Breast cancer decision tree - divergent branch

After tree rules patterns creation from root attribute to leaves, we notice that there are differences in positive and negative cases at the second branch of tree when:

1. Concave Points value between 0.032 and 0.068
2. Area value is between 539.9 and 793.29
3. Number of positive elements is 37
4. Number of negative elements is 29

- Performance Measures

To calculate performance measures, we import an excel file similar to the dataset from which decision tree was drawn, but without the target attribute values, which in our example is diagnosis (Test Data). System predicts diagnosis values based on resulting decision tree and calculates performance measures by comparing diagnosis prediction results with actual diagnosis values. Breast cancer prediction performance measures are presented in table 3.

Fig 12 shows the confusion matrix resulted from breast cancer dataset

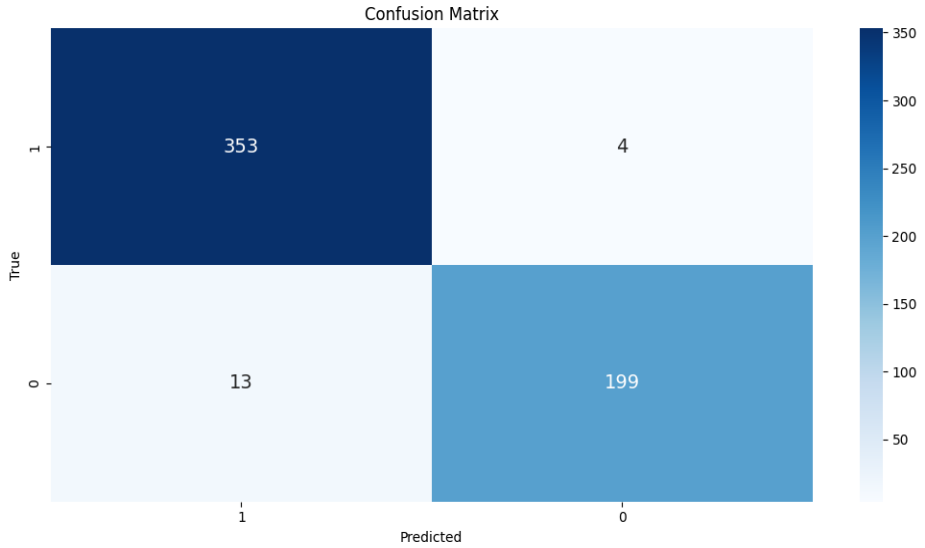

Fig. 12. The confusion matrix resulted from breast cancer dataset

Majority of errors are false positives (13) predicting cancer when benign.

Only 4 false negatives, which is favorable since missing cancer cases is riskier.

The reason misclassification is that some leaves of resulting decision tree contain both positive and negative diagnoses, after recursion with ID3 algorithm reaches leaves, and there are no more important attributes to find attribute that achieves the greatest gain, or there is no longer data that fulfills data categories tree branches as can be seen if Fig. 13.

The model appears conservatively biased due to tending to overlap in borderline cases.

Table 3. Breast cancer prediction performance measures

Accuracy	Precision	Recall	F1_Score
97.012	96.448	98.879	97.648

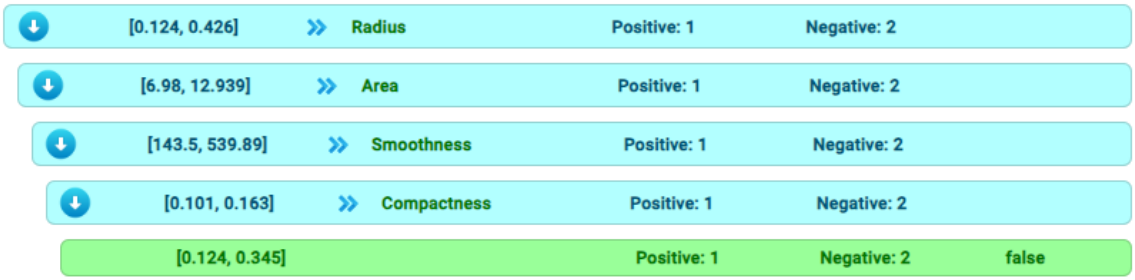

Fig. 13. Leaf node contains positive and negative diagnose

5.2 Heart Disease

There are several types of heart disease, and the most common type is coronary artery disease, that decrease blood flow to the heart, which can cause a heart attack, its symptoms include chest pain, shortness of breath, coughing or wheezing, legs swelling, poor blood supply to the extremities, and fast or irregular heartbeat. After diagnosis, the patient can make lifestyle changes to reduce the risk of developing health problems.

- Diagnoses

We have imported a dataset of 303 patients [14], that contains many attributes to diagnose heart disease [15]:

1) Age: Patient age in years
2) Sex: Patient gender (1 Man and 0 Female)
3) Chest Pain Type (CP)
4) Rest Blood Pressure (T Rest BP)
5) Cholesterol: (Chol in mg/dL)
6) Fasting Blood Sugar (FBS)
7) Resting Electrocardiographic (Rest ECG)
8) Maximum Heart Rate (T Al Ach)
9) Exercise Angina (Ex Ang)
10) Old Peak: ST stress test during exercise compared to rest (for specific part of the heart)
11) Slope: The slope of heartbeats number during peak exercise (to a specific part of the heart)
12) CA: Major vessels stained by micro endoscopy
13) Thalassemia: People with thalassemia have a large amount of iron in their bodies, and too much iron can lead to heart damage.
14) Target: Which gives the final diagnosis of whether the patient has heart disease or not

- Attributes Setting

The limits for all numerical attributes shown in Table 4:

Table 4. Heart disease prediction performance measures

Attributes\Limits	Minimum	First Limit	Second Limit	Maximum
T Rest BPS	94	125	138	200
Cholesterol	126	228	266	564
Max Heart Rate	71	140	158	202
ST Old Peak	0	0.6	1.5	6.2

- Resulting Decision Tree

We note that diagnosis is related to Thalassemia, which represents the root attribute, then the decision tree is divided into 4 branches based on Thalassemia attribute categories (normal, fixed defect, reversable defect, no). Fig. 14, Fig. 15, Fig. 16, Fig 17, Fig. 18 and Fig. 19 show heart disease characteristics.

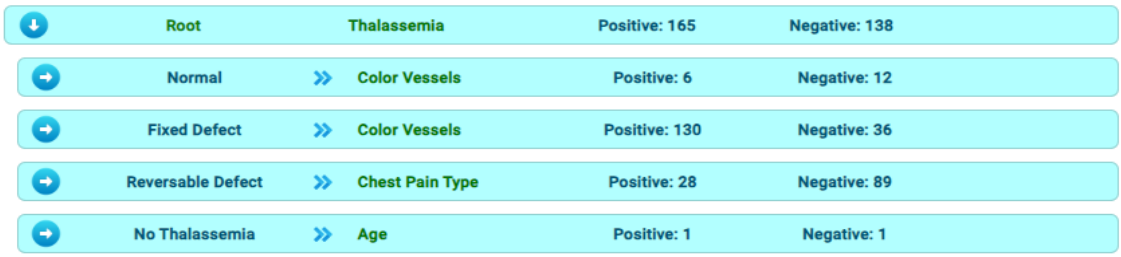

Fig. 14. Heart disease decision tree root branches

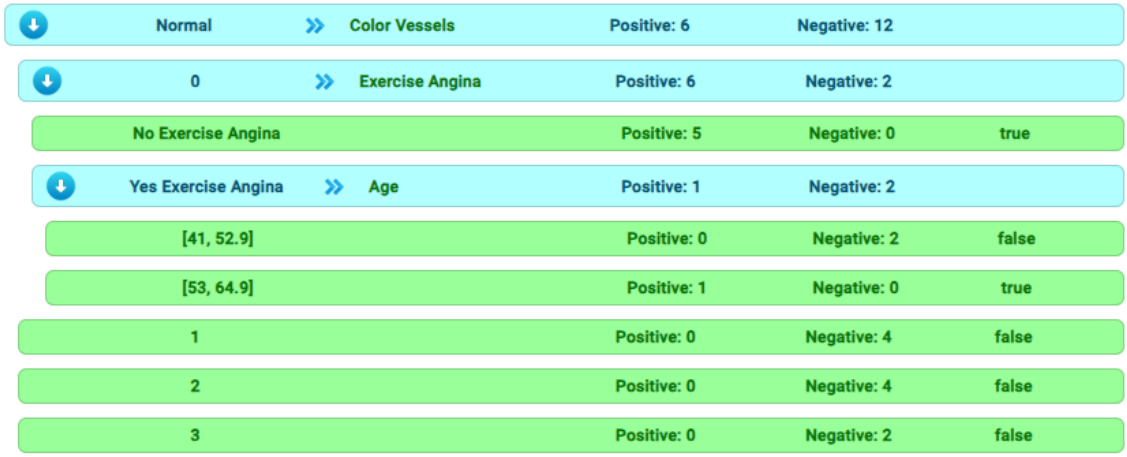

Fig. 15. Heart disease - Thalassemia normal - Color Vessels

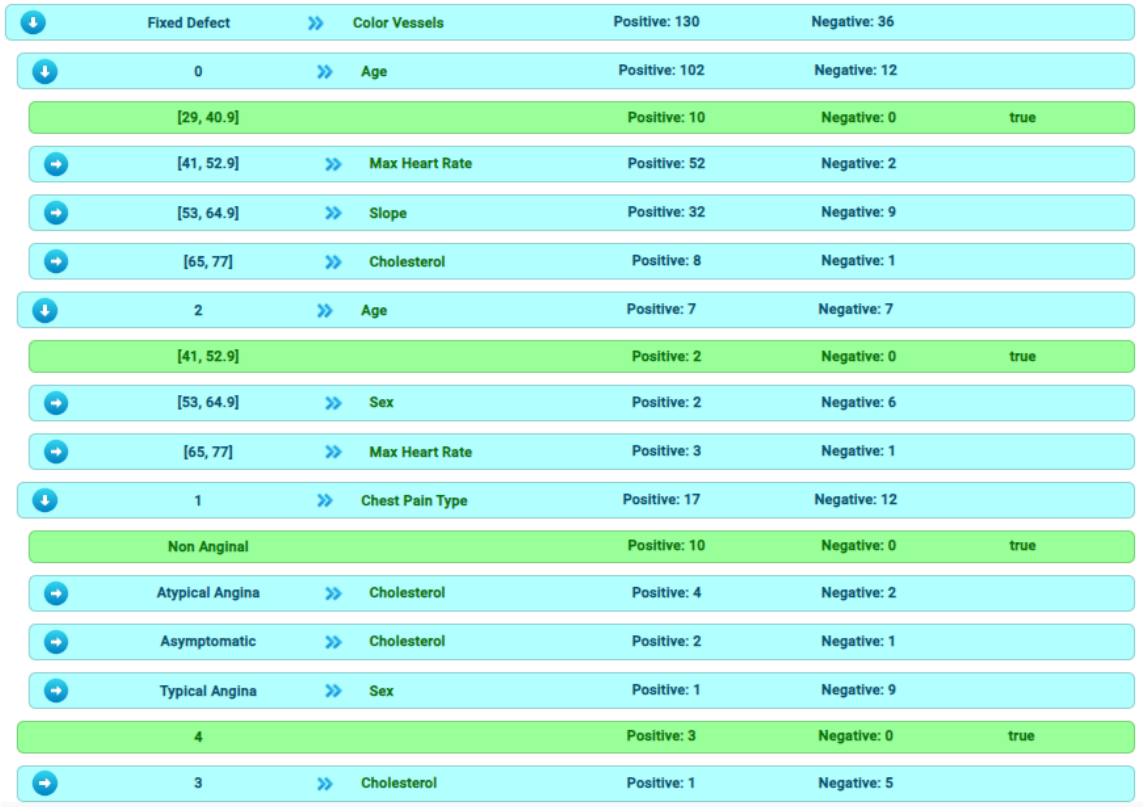

Fig. 16. Heart disease - Thalassemia fixed defect - Color Vessels

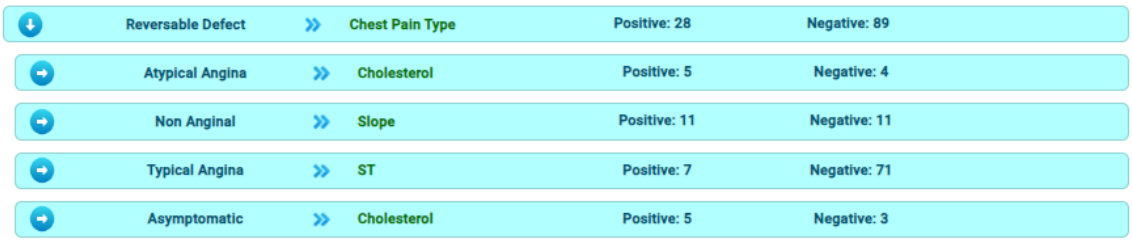

Fig. 17. Heart disease - Thalassemia reversable defect - Chest Pain Type

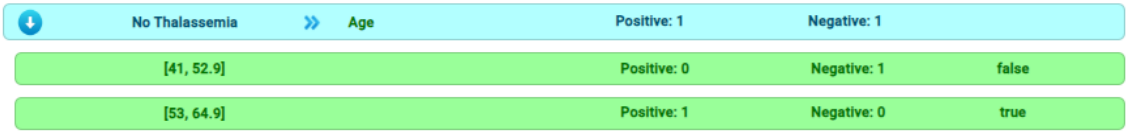

Fig. 18. Heart disease - Thalassemia: no thalassemia - Age

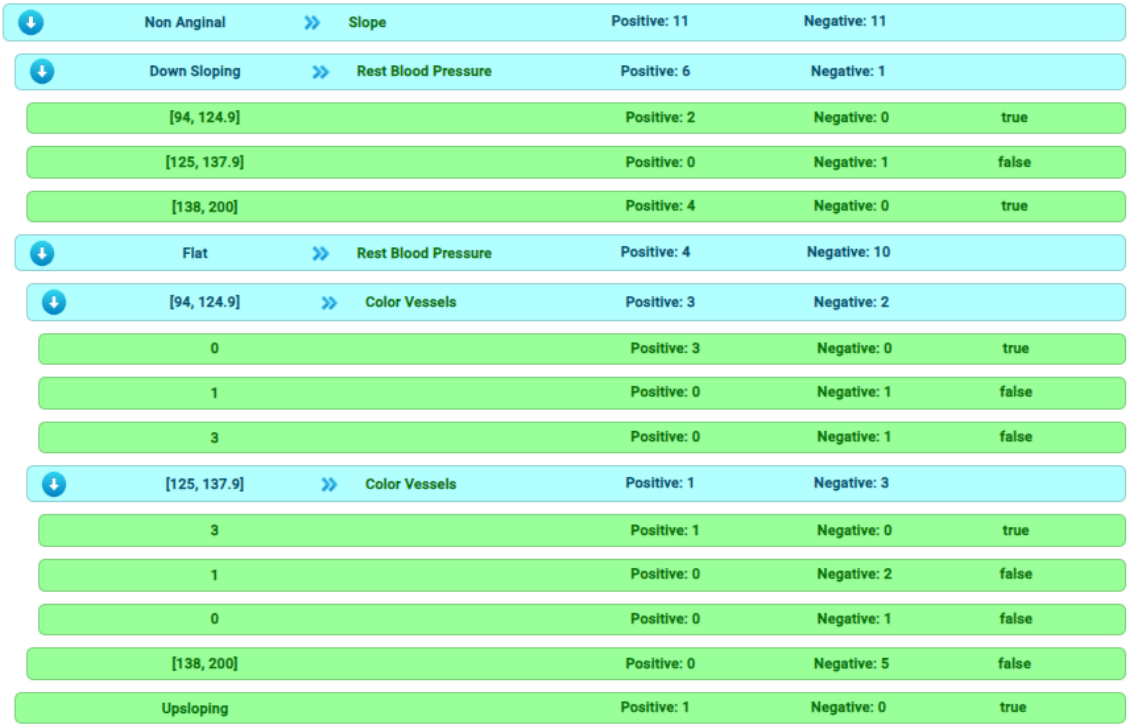

Fig. 19. Heart disease decision tree - divergent branch

After tree rules patterns creation from root attribute to leaves, we notice that there are differences in positive and negative cases at the third branch of tree when:

- Thalassemia: Reversable Defect
- Chest Pain Type: Non Anginal
- Number of positive elements is 11
- Number of negative elements is 11

- Performance Measures is presented in Table 5.

Fig 20 shows the confusion matrix resulted from heart disease dataset

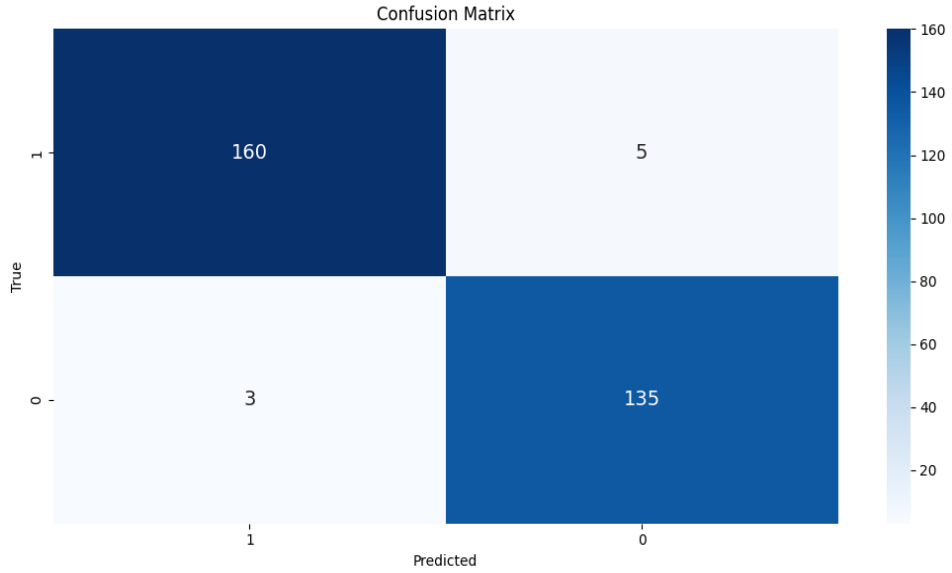

Fig. 20. The confusion matrix resulted from heart disease dataset

False Negatives (5): Heart disease present but predicted as absent.
False Positives (3): Predicted disease when there was none.

These cases may be due to patient data near the threshold boundaries or a lack of distinct separation in certain features.

Table 5. Heart disease prediction performance measures

Accuracy	Precision	Recall	F1_Score
97.359	98.159	96.969	98.156

Fig 21 shows the heart disease all leaves

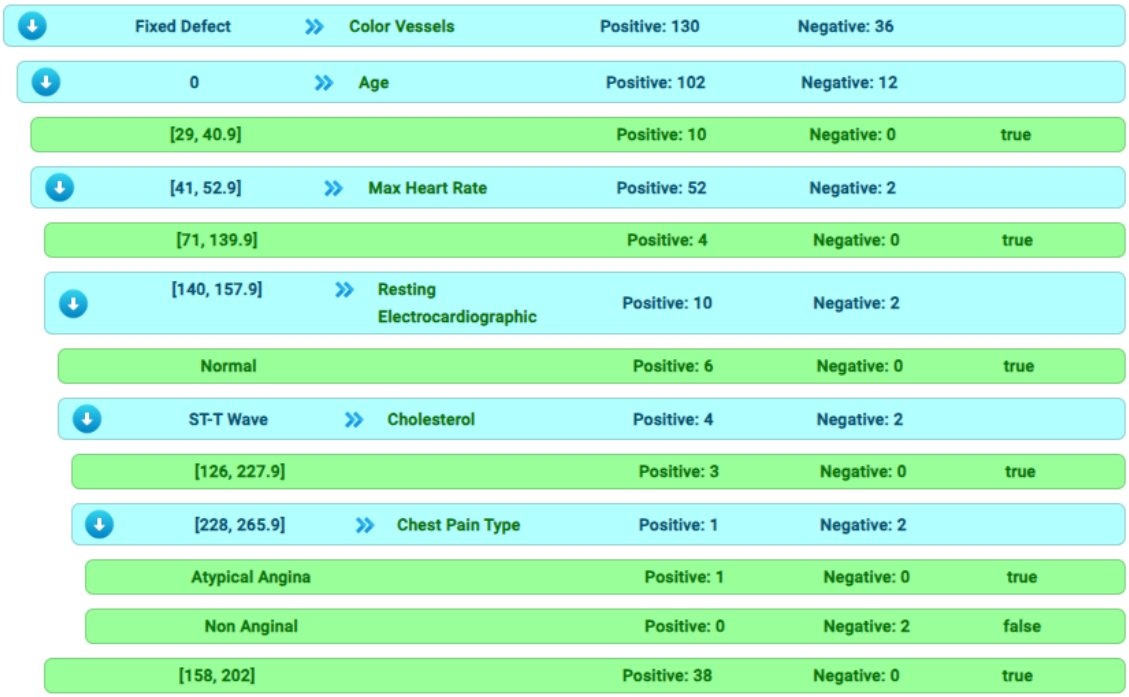

Fig. 21. Heart disease all leaves

5.3 Diabetes

Insulin is hormone that regulates blood glucose levels. Diabetes is a chronic disease that occurs either when the pancreas does not produce enough insulin or when the body cannot effectively use the insulin it produces.

- Diagnoses

We have imported a dataset of 768 patients [16], that contains nine Attributes to diagnose diabetes [17]:

1) Pregnancies: Times number pregnancy occurs
2) Glucose: Blood sugar concentration two hours after an oral glucose tolerance test
3) Diastolic Blood Pressure (BP) (in mmHg)
4) Skin: thickness of three layers of skin (in mm)
5) Insulin: Insulin delivery for 2 hours (µh/mm)
6) Body Mass Index (BMI)
7) (weight in kilograms / height in meters)
8) Diabetes Pedigree Function
9) Age: Patient age in years
10) Outcome: it gives the final diagnosis of whether the patient has diabetes or not

- Attributes Setting:

Table 6 shows the limits for all numerical attributes except age attribute with four numerical fields: Age: Divided into four equal numerical fields: [21 – 36][36 - 51] [51 – 66] [66 – 81]

Table 6. Diabetes numeric attributes setting

Attributes\Limits	Minimum	First Limit	Second	Maximum
Pregnancies	0	3	6	17
Glucose	0	108	132	199
Blood Pressure	0	62	73	122
Skin Thickness	0	15	27	99
Insulin	0	54	130	846
Body Mass Index	0	29	34.4	67.1
Diabetes Pedigree	0.078	0.362	0.617	2.42

- Resulting Decision Tree

We note that the diagnosis is related to blood sugar concentration two hours after oral glucose tolerance test, which represents the root attribute, then the decision tree branches into 3 branches based on glucose attribute numerical fields. Fig. 22, Fig. 23, Fig. 24, Fig 25 and Fig. 26 show diabetic characteristics.

Fig. 22. Diabetes decision tree root branches

Fig. 23. Diabetes - Glucose [0, 107.9] - Bad Mass Index

Fig. 24. Diabetes - Glucose [108, 131.9] - Age

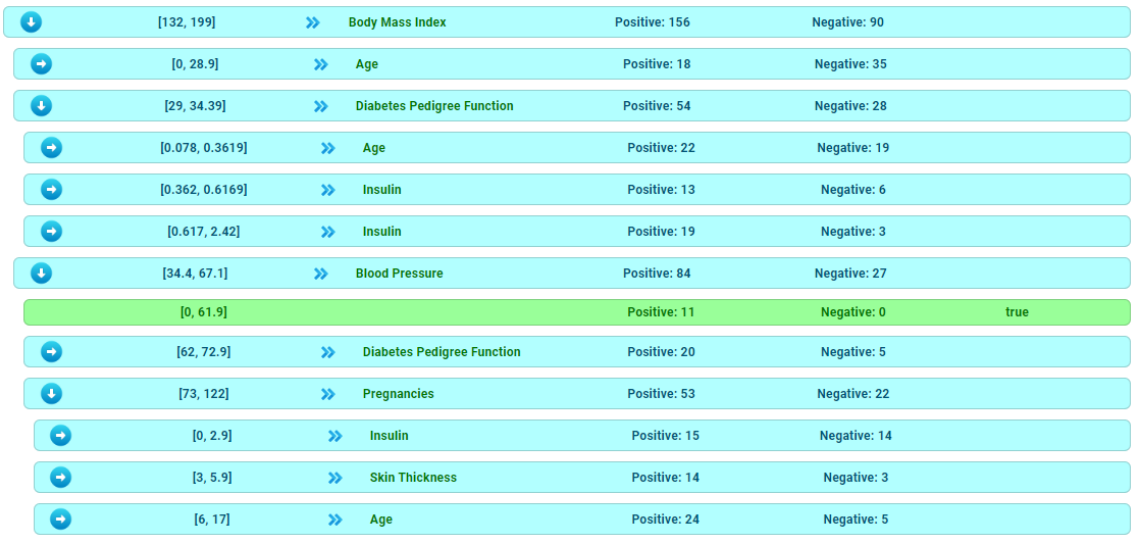

Fig. 25. Diabetes - Glucose [132, 199] - Bad Mass Index

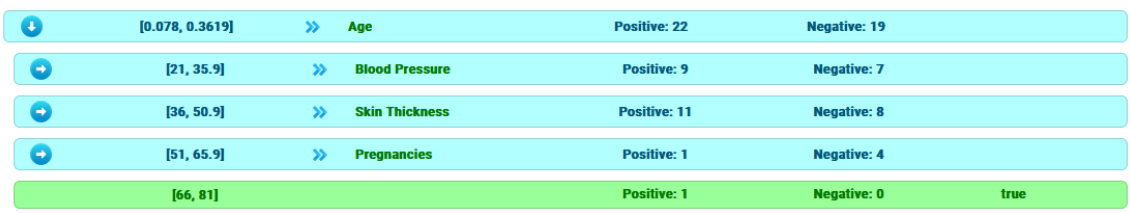

Fig. 26. Diabetes decision tree - divergent branch

After tree rules patterns creation from root attribute to leaves, we notice difference in positive and negative cases at the third branch of the tree when:

1) Glucose: [132, 199]
2) Body Mass Index: [29, 34.39]
3) Diabetes Pedigree Function: [0.078, 0.3619]
4) Number of positive elements is 22
5) Number of negative elements is 19

- Performance Measures are presented in Table 7.

Table 7. Diabetes prediction performance measures

Accuracy	Precision	Recall	F1_Score
93.098	86.44	95.149	90.585

Fig 27 shows the confusion matrix resulted from diabetes dataset

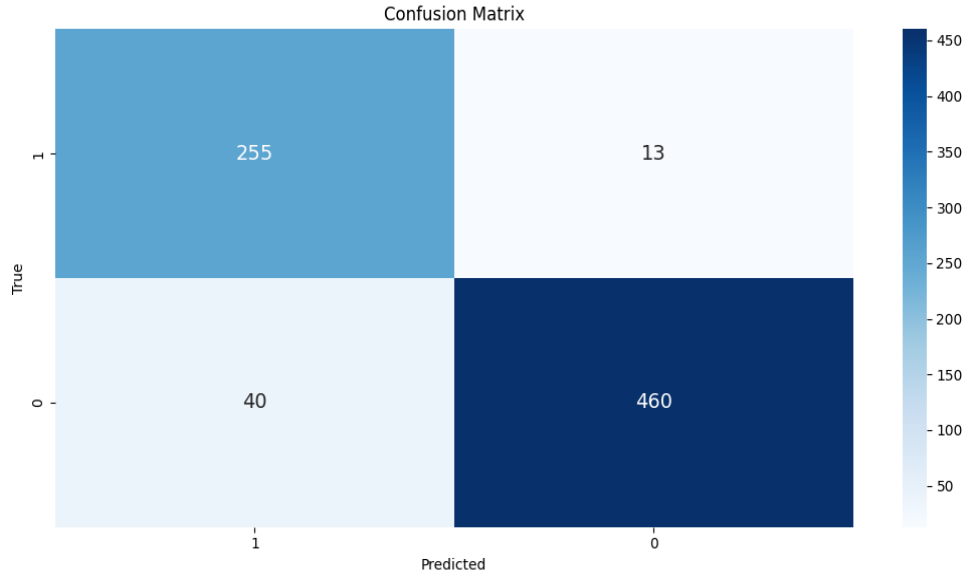

Fig. 27. The confusion matrix resulted from diabetes dataset

The high number of false positives (40) and 13 false negatives indicates that the model tends to overpredict diabetes.

This may be due to the lack of reflection of the true thresholds due to the relatively small dataset size.

The reason misclassification is that some leaves of resulting decision tree contain both positive and negative values; as can be seen in Fig. 28 and Fig. 29.

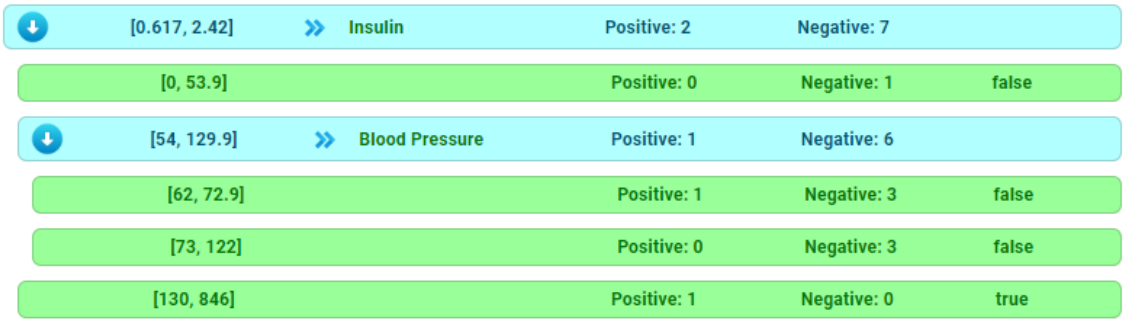

Fig. 28. Leaf node contains positive and negative items

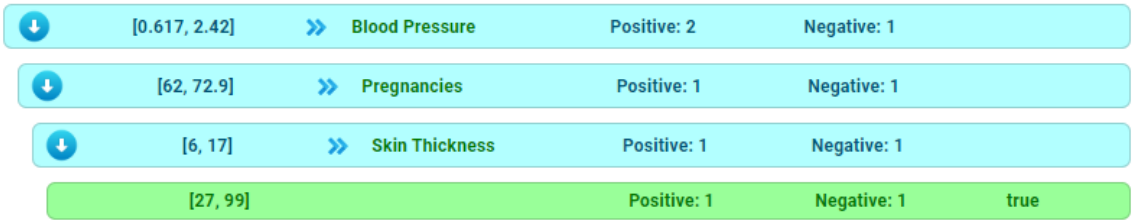

Fig. 29. Leaf node contains positive and negative item

6. Conclusion

In this research, we proposed a dynamic implementation of the ID3 decision tree algorithm aimed at addressing the limitations of traditional, static decision tree models. By allowing users to configure attribute settings, dynamically handle numerical fields, and incrementally grow the tree as new data is introduced, the system provides a flexible approach to building decision trees across diverse datasets. Our focus was not on replacing every static model, but rather on designing a general-purpose, adaptable framework that can be reused without needing to redesign the algorithm for each new problem.

While our system incorporates elements of incremental learning, it is not an online learner in the statistical sense. Instead, it provides a dynamic layer over the ID3 algorithm that adapts structurally to user-defined changes in dataset configuration or data growth—rebuilding only affected parts of the tree and retaining all manually assigned constraints.

We evaluated the system across three different medical datasets—breast cancer, heart disease, and diabetes—to demonstrate its ability to adapt to various attribute types, data structures, and classification goals. While the results were encouraging, we acknowledge that further testing on more varied domains and under real-time data stream conditions would be needed to fully validate its dynamic capabilities.

This study focused on demonstrating the adaptability and generalizability of the proposed system using three well-known medical datasets. While these experiments showed promising results, we acknowledge that broader validation using larger and more diverse datasets, formal cross-validation protocols (e.g., 5-fold or 10-fold CV), and statistical testing are necessary to rigorously assess generalization performance. Furthermore, future work will include comparison with standard dynamic decision tree models such as Hoeffding Trees and GBDT-IL to better contextualize performance metrics.

In the current implementation, numeric attribute values are partitioned into ranges either using equal-width division or by calculating the arithmetic mean between the minimum and maximum. While this method provides a simple and fast way to generate the range, it may not always reflect the actual distribution of the data. Incorrectly choosing thresholds can distort class boundaries and negatively impact model generalization. Future improvements to the system will include more effective partitioning techniques, such as entropy-based classification and quantile classification. Combining these techniques will allow the system to better capture class-relevant partitions in numeric attributes while maintaining scalability.

References

[1] A. Rajeshkanna and K. Arunesh, "Role of Decision Tree Classification in Data Mining," *International Journal of Pure and Applied Mathematics*, vol. 119, no. 15, pp. 2533-2543, 2018. [Online]. Available: https://acadpubl.eu/hub/2018-119-15/2/265.pdf.

[2] H. Mahmud, A. K. M. N. Islam, S. I. Ahmed, and K. Smolander, "What influences algorithmic decision-making? A systematic literature review on algorithm aversion," *Technological Forecasting and Social Change*, vol. 175, p. 121390, 2022/02/01/ 2022, doi: https://doi.org/10.1016/j.techfore.2021.121390.

[3] I. H. Sarker, "Machine Learning: Algorithms, Real-World Applications and Research Directions," *SN Computer Science*, vol. 2, no. 3, p. 160, 2021/03/22 2021, doi: https://doi.org/10.1007/s42979-021-00592-x.

[4] A. Rajeshkanna and K. Arunesh, "ID3 Decision Tree Classification: An Algorithmic Perspective based on Error rate," in *2020 International Conference on Electronics and Sustainable Communication Systems (ICESC)*, Coimbatore, India, 2-4 July 2020 2020: IEEE, pp. 787-790, doi: https://doi.org/10.1109/ICESC48915.2020.9155578.

[5] E. E. Ogheneovo and P. A. Nlerum, "Iterative Dichotomizer 3 (ID3) Decision Tree: A Machine Learning Algorithm for Data Classification and Predictive Analysis," *International Journal of Advanced Engineering Research and Science*, vol. 7, no. 4, pp. 514-521, 2020, doi: https://doi.org/10.22161/ijaers.74.60.

[6] C. Marsala, "Fuzzy decision trees for dynamic data," in *2013 IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS)*, Singapore, 16-19 April 2013 2013, pp. 17-24, doi: https://doi.org/10.1109/EAIS.2013.6604100.

[7] G. Mathew and Z. Obradovic, "Dynamic distributed predictive learning models that preserve privacy for hospitals with insufficient labeled data," *Network Modeling Analysis in Health Informatics and Bioinformatics*, vol. 2, no. 4, pp. 245-255, 2013/12/01 2013, doi: https://doi.org/10.1007/s13721-013-0041-y.

[8] R. Chen, T. Dai, Y. Zhang, Y. Zhu, X. Liu, and E. Zhao, "GBDT-IL: Incremental Learning of Gradient Boosting Decision Trees to Detect Botnets in Internet of Things," *Sensors*, vol. 24, no. 7, p. 2083, 2024, doi: https://doi.org/10.3390/s24072083.

[9] A. Lourenço, J. Rodrigo, J. Gama, and G. Marreiros, "DFDT: Dynamic Fast Decision Tree for IoT Data Stream Mining on Edge Devices," *arXiv preprint*, vol. arXiv.2502.14011, 2025, doi: https://doi.org/10.48550/arXiv.2502.14011.
[10] F. Daghero, A. Burrello, E. Macii, P. Montuschi, M. Poncino, and D. J. Pagliari, "Dynamic Decision Tree Ensembles for Energy-Efficient Inference on IoT Edge Nodes," *IEEE Internet of Things Journal*, vol. 11, no. 1, pp. 742-757, 2024, doi: https://doi.org/10.1109/JIOT.2023.3286276.
[11] S. Yang, J.-Z. Guo, and J.-W. Jin, "An improved Id3 algorithm for medical data classification," *Computers & Electrical Engineering*, vol. 65, pp. 474-487, 2018/01/01/ 2018, doi: https://doi.org/10.1016/j.compeleceng.2017.08.005.
[12] T. Jyothirmayi and S. Reddy, "ID3 Algorithm: An Algorithm for better decision tree," *International Journal on Computer Science and Engineering*, vol. 2, no. 9, pp. 2827-2830, 2010. [Online]. Available: http://www.enggjournals.com/ijcse/doc/IJCSE10-02-09-050.pdf
[13] M. F. Ak, "A Comparative Analysis of Breast Cancer Detection and Diagnosis Using Data Visualization and Machine Learning Applications," (in eng), *Healthcare (Basel, Switzerland)*, vol. 8, no. 2, Apr 26 2020, doi: https://doi.org/10.3390/healthcare8020111.
[14] C. Kakaraparthi. "Prediction of Heart Disease by ML Techniques." Kaggle; Data science company. https://www.kaggle.com/code/charankakaraparthi/prediction-of-heart-disease-by-ml-techniques (accessed 21/November/2024, 2024).
[15] H. R. Marateb and S. Goudarzi, "A noninvasive method for coronary artery diseases diagnosis using a clinically-interpretable fuzzy rule-based system," (in eng), *Journal of research in medical sciences : the official journal of Isfahan University of Medical Sciences*, vol. 20, no. 3, pp. 214-23, Mar 2015. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC4468223/.
[16] K. Karagöz. "Pima Indians Diabetes Db to Classification." Kaggle; Data science company. https://www.kaggle.com/code/kursatkaragoz29/pima-indians-diabetes-db-to-classification (accessed 21/11/2024, 2024).
[17] V. Chang, J. Bailey, Q. A. Xu, and Z. Sun, "Pima Indians diabetes mellitus classification based on machine learning (ML) algorithms," (in eng), *Neural computing & applications*, pp. 1-17, Mar 24 2022, doi: https://doi.org/10.1007/s00521-022-07049-z.

Authors' Profiles

Amir Amjad Gharbi, he is an MSc student specializing in software engineering, with a robust background in various development environments and programming languages. He holds a degree in Software Engineering from the Faculty of Information Technology at Damascus University, completed in 2018. Amir's technical skills encompass a range of programming languages such as Java, JavaScript, C++, and C#, as well as proficiency in development environments like ASP.NET, Angular, and Oracle. He is also well-versed in database management with MySQL and XML technologies. Amir Gharbi is committed to advancing his knowledge and skills in software development, making significant contributions to the field through both his academic pursuits and practical applications.

Bassel Alkhatib, PhD, he is the web sciences master director at the Syrian Virtual University and the head of Artificial Intelligence department at Information Technology Faculty at Damascus University. He holds PhD degree in computer science from the University of Bordeaux-France, 1993. Dr. Alkhatib supervises many PhD students in web mining, and knowledge management. He also leads and teaches modules at both BSc and MSc levels in computer science and web engineering in Syrian Virtual University, Damascus University, and Al-Shem Private University.

How to cite this paper: Amir Amjad Gharbi, Bassel Alkhatib, "Dynamic Data Mining: Using Dynamic ID3 Algorithm to Solve Any Problem that Needs Decision Tree Support", International Journal of Modern Education and Computer Science(IJMECS), Vol.17, No.6, pp. 127-145, 2025. DOI:10.5815/ijmecs.2025.06.09