Evaluating the performance of GPU classification based on memory type: A study of three algorithms

Akshay Chikhalkar
Department of Electrical Engineering and Computer Science
Technical Hocschule Ostwestfalen-Lippe - University of Applied Sciences and Arts
Lemgo, Germany
akshay.chikhalkar@stud.th-owl.de

***Abstract*—This research paper presents a study on the classification of Graphics Processing Units (GPUs) based on memory type. The motivation for this study arose from personal experience as a tech enthusiast, where I recognized the challenge of providing accurate recommendations for technology products in light of the complexity and rapid evolution of the technology domain. The study aims to automate this process by using machine learning algorithms to classify GPUs based on memory type. Three different machine learning algorithms were employed in this study. The performance of these algorithms was evaluated using a confusion matrix and cross-validation. The results of this study demonstrate the effectiveness of using these algorithms for classifying GPUs based on memory type, and provide insights into which algorithm is the best fit for this task.**

***Index Terms*—classifier, model, random forest (RFC), decision tree (DTC), support vector machine (SVM), GPU**

I. Motivation

GPU classification is a powerful technique that allows for faster and more efficient processing of large amounts of data. For tech enthusiasts, the ability to quickly and accurately classify data can open up new possibilities for research and experimentation. With the increasing amount of data being generated by devices and applications, the ability to process this data in real-time is becoming increasingly important. GPU classification allows for the creation of more sophisticated models and algorithms, enabling new insights and discoveries. Additionally, the use of GPU classification can significantly reduce the time and resources required for data processing, allowing for more efficient use of resources and cost savings. Overall, GPU classification is a valuable tool for anyone looking to push the boundaries of what is possible with data analysis and machine learning.

The current study was motivated by the desire to address the challenge of providing technology recommendations based on multiple factors. I, as a tech enthusiast, noticed that family members, friends and colleagues often sought advice on technology products, particularly in the realm of computer technology. The complexity of the technology domain, as well as the increasing number of products being released, makes it difficult to keep track of all options and determine the best fit for individual needs.

To address this challenge, I proposed a classification solution that utilizes computer processing to classify technology products based on relevant features. The initial focus of the study was on laptop computers and the classification of their Graphics Processing Units (GPUs) based on memory type. Three Machine Learning (ML) algorithms were employed and the script was written in Python programming language. The goal of this study is to not only classify laptop computers, but also to expand this solution to other technology products.

The study aimed to classify GPUs based on memory type, as memory plays a crucial role in the performance of a GPU. Different types of memory, such as DDR3, DDR4, GDDR5, and GDDR6, have different characteristics and can impact the overall performance of a GPU. By classifying GPUs based on memory type, the study aimed to provide a more accurate and comprehensive evaluation of the products available in the market.

II. State of the Art

A. What is ML?

Machine learning is a subfield of computer science that aims to allow computers to ”learn” without being explicitly programmed [1]. It has its roots in the 1950s artificial intelligence movement and emphasises practical goals and applications, particularly prediction and optimization. In machine learning, computers ”learn” by improving their performance at tasks through ”experience” [2].

B. Why ML?

A branch of computing algorithms called machine learning is constantly developing and aims to replicate human intelligence by learning from the environment. In the new era of ’big data,’ they are regarded as the workhorse. Machine learning methods have been effectively used in a variety of industries, including banking, entertainment, biomedicine, pattern recognition, computer vision, spacecraft engineering, and computational biology. In addition, it has readily available libraries to perform tasks like clustering, classification etc. Hence, ML is the perfect solution to classification.

C. Machine Learning Overview

Here is a brief overview of some of the most popular machine learning algorithms (ML) [4].

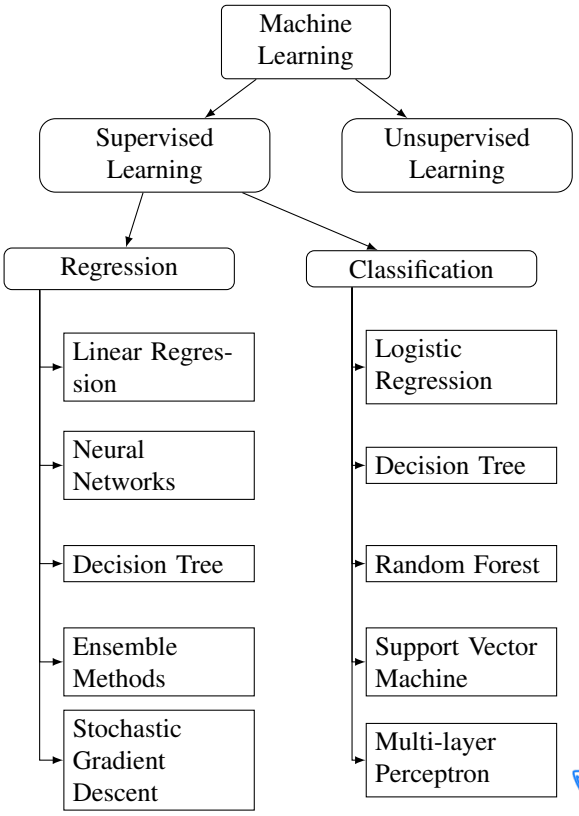


Fig. 1: Types of supervised and unspervised algorithms

Machine learning is classified into Supervised Learning and Unsupervised Learning along with others like Semi-Supervised Learning, Reinforcement Learning, Multi-task Learning, Ensemble Learning, Neural Networks, and Instance-Based Learning [4].

Unsupervised learning is a form of machine learning technique that unearths obscure patterns or data clusters without the assistance of a human [6].

In this paper, supervised learning was studied. It is machine learning algorithms can develop broad patterns and hypotheses by using examples from outside sources to predict the results of incoming examples. The goal of supervised machine learning classification algorithms is to classify data based on existing knowledge (Labelled data) [5]. Supervised Learning is further categorised into two parts i.e. Regression and Classification.

Regression is frequently used for forecasting and prediction, two areas where machine learning and their application have a lot in common.

Supervised classification is one of the functions that so-called intelligent systems carry out most frequently [7]. The goal of supervised learning is to create a precise model of the distribution of class labels in terms of predictor features. When the values of the predictor characteristics are known but the value of the class label is unknown, the resulting classifier is used to give class labels to the testing cases. In this paper, we focus on the classification of GPU based on memory type (Target variable).

Classification is a data mining technique used to predict data instance group membership [8]

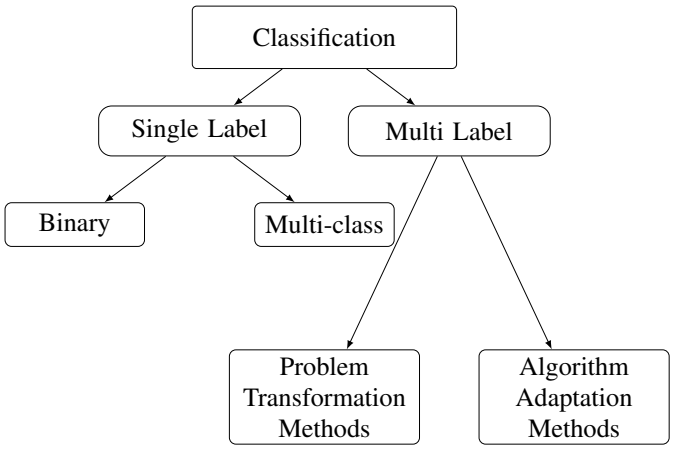


Fig. 2: Machine learning classification tree fiagram

Most classification problems assign a single class to each example or instance. However, there are many classification tasks in which each instance can be assigned to one or more classes. This set of problems falls under the category of multi-label classification [9]. Multi-label classification is further categorised into two types, Problem Transformation Methods and Algorithm Adaptation Methods.

Binary classification is a supervised learning algorithm that categorizes new observations into one of two classes [10].

A multiclass classification task is a machine learning classification task with more than two classes or outputs [11]. It assumes that each sample has only one label and employs the Bayes theorem to predict the class of unknown datasets [11].

A study presents a network kernel function called Multi-field packet classification, which classifies and routes packets on a GPU using a set of rules [12]. The study utilizes a shapelet discovery algorithm for time series classification, which identifies useful subsequences from a set of time series [13]. The study notes that with recent advancements in high-performance computing techniques, such as GPU, large-scale deep learning models are now possible for machine learning applications [14]. However, there is a lack of research on CPU classification. To address this gap, the study conducts GPU classification using different types of graphics memory and evaluates and compares three classification algorithms (Random Forest, Support Vector Classification and Decision Tree classifier).

III. Solutions

The proposed solution is to use machine learning algorithms to classify Graphics Processing Units (GPUs) based on memory type. This would help to automate the process of product recommendations for technology enthusiasts and make it easier to keep track of the ever-demanding technology market. The proposed solution takes advantage of machine learning algorithms such as Random Forest, Decision Tree and

Support Vector Machine to classify GPUs based on memory type. The implementation for these algorithm is given below.

A. Classifiers

*1) **Random Forest:*** In 2001, Leo Breiman of the University of California proposed the Random Forest [19]. It is a classifier composed of a collection of tree-structured classifiers with identically distributed independent random vectors, with each tree casting a unit vote for the most popular class at input x [20]. An upper bound for Random Forests is extracted to get the generalisation error in terms of two parameters Exactitude and interdependence of individual classifiers [18]. Fig.3 depicts the entire classification process based on the random forest.

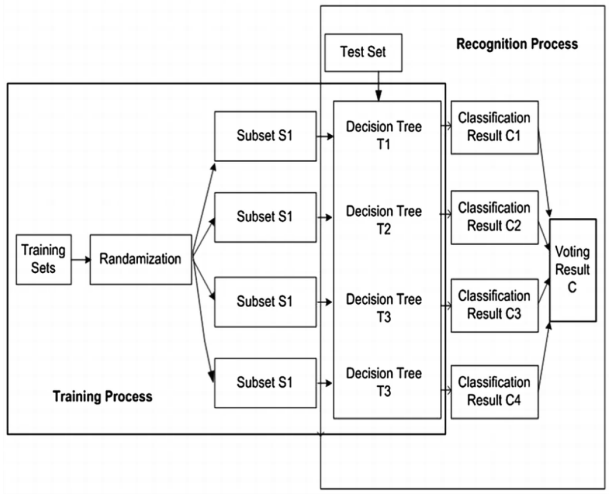


Fig. 3: Conceptual framework of random forest classifier [17].

A random forest is a type of ensemble learning algorithm that builds small decision trees with a limited number of features. The random forest categorises data points with each tree, and the average of these decision trees becomes the prediction model.

Formalization: The algorithm takes a set of elements as input.

$$(N, n, e) \rightarrow \frac{1}{n}\sum_{i=1}^{n} f_n(x, e) \rightarrow f(x)$$

Where, N – training data, e – Entropy, n – Number of decision trees

*2) **Decision Tree:*** A decision tree is the most commonly used decision-making tool. To accomplish this, create a decision tree with various branches and leaves. These branches and leaves should represent all of the various aspects of a given situation [21]. A decision tree functions similarly to a decision support tool. It employs a tree-like graph of decisions and their potential outcomes, such as resource costs, event outcomes, and utility. It is one method of displaying an algorithm [21]. Various types of decision trees can be used depending on the situation and desired outcome.

Algorithm 1: Random forest

Input: Data N
Output: Find differnent classes of objects

Function `RandomForestClassifier`(X, y, n, e)**:**
  $DF \leftarrow \phi$;
  for $x \in X$ **do**
    $x_i \leftarrow\!\!\prec BTS(X)$;
    $f \subset F$;
    $d_i \leftarrow f_n(x, f, e)$;
    $DF \leftarrow DF \cup d_i$;
  end
  return DF;

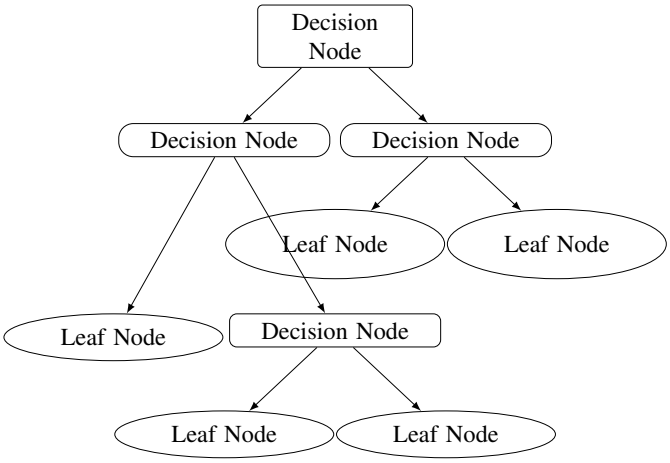


Fig. 4: Decision tree training flow chart

A decision tree algorithm is a supervised learning method used for classification and regression problems. The algorithm works by recursively splitting the data into subsets based on the values of input features, creating a tree-like structure. Each internal node of the tree represents a feature, and each leaf node represents a class label or a predicted value. Where, E represents the set of examples, A represents the set of attributes, PE represents the set of parent examples, T represents the decision tree, A' is the most promising attribute and importance(a, E) is a function to find the importance of attribute an over examples E.

*3) **Support Vector Machine:*** The C-support vector classification (C-SVC) method is a data mining extension of the SVM (support vector machine) that has many applications in data classification, regression, and others [15]. The letter 'C' in C-SVC stands for the regularisation parameter, which affects the margin of separation between classes. A higher value of C leads to a narrower margin, while a lower value

Algorithm 2: Decision tree

Input: Examples: E, Attributes: A, Parent Examples: PE
Output: Decision Tree: T

Function `DecisionTreeClassifier`(E, A, PE)**:**
  Result: Decision Tree: T
  if $|E| = 0$ **then**
    return *pluralityValue(PE)*
  end
  if $|A| = 0$ **then**
    return *pluralityValue(E)*
  end
  if $\forall e \in E, e$ *classifies the same* **then**
    return *the classification*
  end
  $A' = \text{argmax}_{a \in A}(importance(a, E))$;
  $T = newTree(root = A')$;
  for $v \in A'$ **do**
    $exs = \{e \in E | e.A' = v\}$;
    $subtree = decisionTreeLearning(exs, A - A', E)$;
    $T.addSubtreeAsBranch(subtree, label = (A', v))$;
  end
  return T

Algorithm 3: Support vector machine

Input: Data N
Output: Find differnent classes of objects

Function `SVC`(X, y, k)**:**
  $V \leftarrow \phi$;
  for $x \in X$ **do**
    $x_k = k(X)$;
    $v = v - x_k$;
    $V \leftarrow V \cup v$;
  end
  return V;

of C leads to a wider margin [16].
The implementation is built on top of libsvm. Fit time scales at least quadratically with sample number and may be impractical beyond tens of thousands of samples[1].
A support vector machine is a supervised learning algorithm that determines the decision boundary between two classes of objects. It is a linear or non-linear classifier that distinguishes between the two types of data points. It employs the kernel function to determine a decision boundary between data points for non-linear data points.
Formalization: The algorithm take a set of an element as input.

$$(N, K) \rightarrow \sum_{i=1}^{n} \alpha_i \quad y_i \quad K(x_i, x) + b \rightarrow f(x)$$

Where, $N-$ training data, $K-$ kernel function

*4) **Performance Metrics**:* A Confusion Matrix and Cross-Validation are both commonly used performance matrices for evaluating the effectiveness of classification algorithms in machine learning.

Confusion matrix:
A Confusion Matrix is a commonly used performance matrix for classification algorithms. It is a table that is used to compare the predicted values of a model with the actual values, and it is particularly useful for classification problems with multiple output classes. The matrix is organized as an N*N table, where N represents the number of output classes. The four different combinations of predicted and actual values are represented in the matrix and are used to calculate other performance metrics such as accuracy, precision, recall, and F1-score [22].

The Confusion matrix typically contains the following four combinations of predicted and actual values:

- True Positives (TP) - the number of observations that are correctly predicted as positive
- False Positives (FP) - the number of observations that are incorrectly predicted as positive
- True Negatives (TN) - the number of observations that are correctly predicted as negative
- False Negatives (FN) - the number of observations that are incorrectly predicted as negative

These values are used to calculate various metrics such as accuracy, precision, recall and F1-score which gives a holistic view of the performance of the model [23].

Accuracy is defined as the ratio of correctly predicted observations to total observations [23].

$$Accuracy = \frac{(TP + TN)}{(TP + FP + TN + FN)}$$

Precision is defined as the ratio of true positive values to total correctly predicted values [23].

$$Precision = \frac{TP}{(TP + FP)}$$

The number of correctly predicted values from all positive classes is defined as recall. It is also known as sensitivity [24].

$$Recall = \frac{TP}{(TP + FP)}$$

The $\mathbf{F_1}$ score is the weighted average of precision and recall[2].

[1]https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

[2]https://en.wikipedia.org/wiki/F-score

$$F_1 = 2\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{2\text{tp}}{2\text{tp} + \text{fp} + \text{fn}}$$

5) Cross-validation - StratifiedKFold: Cross-validation is a technique that is used to assess the performance of a model by splitting the data into several partitions, training the model on one partition and testing it on the other partition [25]. This is done multiple times, each time using a different partition for testing, and the results are then averaged to give an overall estimate of the model's performance [25].
StratifiedKFold is a variation of K-fold cross-validation that ensures each fold has a similar proportion of samples from each class. It was used with 10 folds for better cross-validation and executed multiple times to account for variations in execution time.

IV. Experiments and Results

Data Preparation: A Kaggle dataset named Data Mining UNDIP was chosen for this classification simulation. It contained a total of 16 features, out of which 5 features were considered to train the models (Dependent Variable - Memory Type and Independent Variables - Memory Size, GPU Clock, Memory Clock, Memory Bus Width). Interpolation was used for generating and filling in missing values in the dataset. The dataset was split into 80-20% for training and testing respectively.
Algorithm Selection: Three different machine learning algorithms were employed in this study: Random Forest, Decision Tree, and Support Vector Machine. These algorithms were chosen as they have been widely used in similar classification tasks and have been shown to produce accurate results.
Algorithm Training: Each algorithm was trained using the selected features from the dataset, and the trained models were used to classify the GPUs based on the memory type.
Evaluation: The performance for three classifiers (Random Forest, Decision Tree and Support Vector Machine) was evaluated using two techniques, Confusion Matrix and Cross-validation. In cross-validation, along with accuracy, execution time was recorded by performing cross-validation multiple times.
Note 1: The performance evaluation for all three algorithms was measured without parameter optimisation.
Note 2: The entire benchmarking was performed using macOS Ventura 13.1 (22C65) running on Apple Macbook Air, Apple M1 (8-Core CPU), LPDDR4X 16GB Ram.

A. Confusion matrix results

In the Confusion Matrix performance evaluation, Accuracy, Precision, Recall and F1 score were recorded on a scale of 0 to 1 (0 is worst and 1 is best).

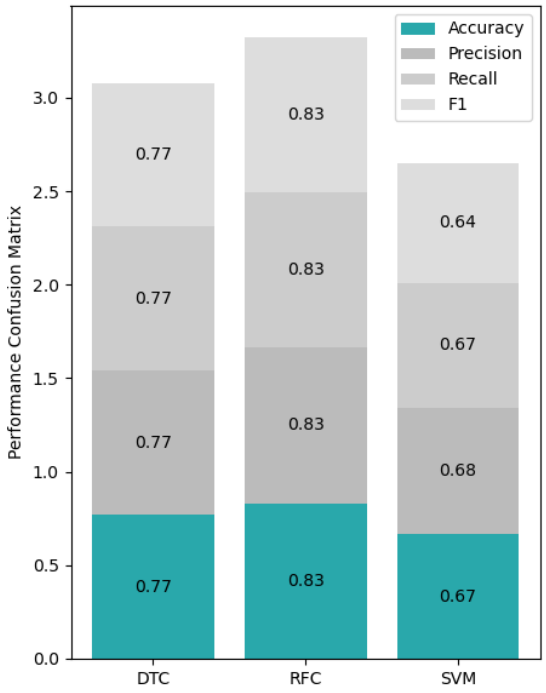


Fig. 5: Performance based on confusion matrix results

RFC outperforms other classifiers (DTC and SVM) in each aspect. It has the highest sectional and overall score, refer to Fig.1. However, DTC perform second best and SVM was last. Clearly from the performance vice Random Forest comes out as the best classification algorithm in comparison with Decision Tree and Support Vector Machine. Refer to TABLEI for confusion matrix results.
Based on these results, it appears that the Random Forest

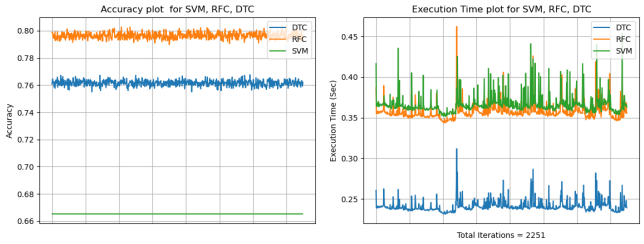
Classifier Name	Accuracy	Precision	Recall	F_1
DTC	0.769	0.770	0.769	0.767
RFC	0.830	0.834	0.830	0.829
SVC	0.666	0.676	0.666	0.639

TABLE I: Confusion matrix results

Classifier (RFC) has the highest accuracy, precision, recall, and F1-score among the three algorithms. The Decision Tree Classifier (DTC) also performed well in terms of accuracy, precision, and recall. However, the Support Vector Machine (SVC) had the lowest performance among the three algorithms in all the four metrics.

B. Cross-validation results

All three algorithms (DTC, RFC, and SVM) showed consistent performance after 2000 cross-validation tests. SVM was particularly stable with an accuracy of 0.67. DTC's



(a) Accuracy plot (b) Execution time plot

Fig. 6: Accuracy and execution time plot

accuracy varied between 0.79 and 0.80, while RFC's accuracy ranged from 0.75 to 0.77, refer to Fig.6.

Note: The average was taken from over 2000 results for each classifier.

In terms of execution time, DTC outperformed RFC and SVM by taking the lowest time with 0.24 sec. RFC and SVM performed approximately the same with a difference of 0.1 sec difference between them.

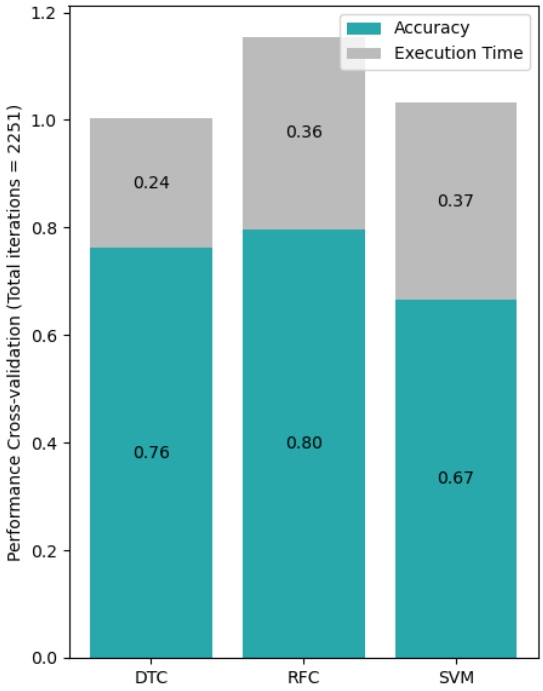


Fig. 7: Performance based on cross-validation results

All three algorithms (DTC, RFC and SVM) showed consistent performance after 2000 cross-validation tests. SVM was particularly stable with an accuracy of 0.67. DTC's accuracy varied between 0.79 and 0.80, while RFC's accuracy ranged from 0.75 to 0.77. Refer to TABLEII for cross-validation results.

Classifier Name	Accuracy	Execution Time	Iterations
DTC	0.761	0.241	183
RFC	0.796	0.358	183
SVM	0.665	0.366	183

TABLE II: Cross-validation results

The results of the study indicate that the Random Forest Classifier (RFC) performed the best among the three algorithms considered in this study. The Random Forest Classifier achieved the highest score in cross-validation and also had the highest accuracy, precision, recall, and F1-score among the three algorithms as per confusion matrix. Additionally, the Decision Tree Classifier (DTC) also performed well in terms of score and accuracy, precision, and recall but Random Forest Classifier performed better than DTC. However, the Support Vector Machine (SVM) had the lowest performance among the three algorithms in all the evaluation metrics. The execution time for the Random Forest classifier is slightly higher than the Decision Tree Classifier, while Support Vector Machine has the highest execution time among the three algorithms.

V. Summary and Outlook

The results of the study showed that the Random Forest algorithm had the highest accuracy in classifying the GPU based on memory type, followed by Decision Tree and Support Vector Machine. The study also evaluated the performance of the algorithms based on precision, recall and F1-score, which are commonly used metrics for evaluating the performance of classification models.However, it should be noted that the study did not include any parameter optimization, and the performance could potentially be improved through hypermeter tuning.

The study concludes that the proposed classification solution, which utilizes computer processing and machine learning algorithms, can effectively classify GPUs based on memory type. The results of this study can be used to provide more accurate recommendations for technology products and can be extended to other technology products as well. This research can be used as a foundation for further research in the field of technology product classification, and it can be useful for consumers, manufacturers and retailers.

References

[1] Samuel, A.L., 1959. Some stuies on Machine Learning Using the Game of Checkers. IBM Journal on Research and Development, 3, pp.210-229.

[2] Mitchell, T.M. and Mitchell, T.M., 1997. Machine learning (Vol. 1, No. 9). New York: McGraw-hill.

[3] Tao Tao, Chen Sun, Zhaoyang Wu, Jian Yang, Jing Wang, Deep Neural Network-Based Prediction and Early Warning of Student Grades and Recommendations for Similar Learning Approaches, Applied Sciences, 10.3390/app12157733, 12, 15, (7733), (2022).

[4] Batta Mahesh Machine Learning Algorithms - A Review International Journal of Science and Research (IJSR) ISSN: 2319-7064 ResearchGate Impact Factor (2018): 0.28 — SJIF (2018): 7.426

[5] A. Singh, N. Thakur and A. Sharma, "A review of supervised machine learning algorithms," 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 2016, pp. 1310-1315.

[6] Usama, M., Qadir, J., Raza, A., Arif, H., Yau, K.L.A., Elkhatib, Y., Hussain, A. and Al-Fuqaha, A., 2019. Unsupervised machine learning for networking: Techniques, applications and research challenges. IEEE access, 7, pp.65579-65615.

[7] Kotsiantis, S.B., Zaharakis, I. and Pintelas, P., 2007. Supervised machine learning: A review of classification techniques. Emerging artificial intelligence applications in computer engineering, 160(1), pp.3-24.

[8] Kesavaraj, G. and Sukumaran, S., 2013, July. A study on classification techniques in data mining. In 2013 fourth international conference on computing, communications and networking technologies (ICCCNT) (pp. 1-7). IEEE.

[9] de Carvalho, A.C. and Freitas, A.A., 2009. A tutorial on multi-label classification techniques. Foundations of computational intelligence volume 5, pp.177-195.

[10] Bellinger, C., Sharma, S. and Japkowicz, N., 2012, December. One-class versus binary classification: Which and when?. In 2012 11th international conference on machine learning and applications (Vol. 2, pp. 102-106). IEEE.

[11] Aly, M., 2005. Survey on multiclass classification methods. Neural Netw, 19(1), p.9.

[12] Zhou, S., Singapura, S.G. and Prasanna, V.K., 2014, September. High-performance packet classification on GPU. In 2014 IEEE High Performance Extreme Computing Conference (HPEC) (pp. 1-6). IEEE.

[13] Chang, K.W., Deka, B., Hwu, W.M.W. and Roth, D., 2012, December. Efficient pattern-based time series classification on GPU. In 2012 IEEE 12th International Conference on Data Mining (pp. 131-140). IEEE.

[14] Zhang, Q., Bai, C., Liu, Z., Yang, L.T., Yu, H., Zhao, J. and Yuan, H., 2020. A GPU-based residual network for medical image classification in smart medicine. Information Sciences, 536, pp.91-100.

[15] D. Rahmawati and Y. -P. Huang, "Using C-support vector classification to forecast dengue fever epidemics in Taiwan," 2016 International Conference on System Science and Engineering (ICSSE), Puli, Taiwan, 2016, pp. 1-4, doi: 10.1109/ICSSE.2016.7551552.

[16] Novakovic, J. and Veljovic, A., 2011, September. C-support vector classification: Selection of kernel and parameters in medical diagnosis. In 2011 IEEE 9th international symposium on intelligent systems and informatics (pp. 465-470). IEEE.

[17] Parmar, A., Katariya, R. and Patel, V., 2019. A review on random forest: An ensemble classifier. In International Conference on Intelligent Data Communication Technologies and Internet of Things (ICICI) 2018 (pp. 758-763). Springer International Publishing.

[18] Yigin, B.O., Algin, O. and Saygili, G., 2020. Comparison of morphometric parameters in prediction of hydrocephalus using random forests. Computers in Biology and Medicine, 116, p.103547.

[19] Breiman, L., 2001. Random forests. Machine learning, 45(1), pp.5-32.

[20] Reis, I., Baron, D. and Shahaf, S., 2018. Probabilistic random forest: A machine learning algorithm for noisy data sets. The Astronomical Journal, 157(1), p.16.

[21] Navada, A., Ansari, A.N., Patil, S. and Sonkamble, B.A., 2011, June. Overview of use of decision tree algorithms in machine learning. In 2011 IEEE control and system graduate research colloquium (pp. 37-42). IEEE.

[22] Patel, H.H. and Prajapati, P., 2018. Study and analysis of decision tree based classification algorithms. International Journal of Computer Sciences and Engineering, 6(10), pp.74-78.

[23] Farnaaz, N. and Jabbar, M.A., 2016. Random forest modeling for network intrusion detection system. Procedia Computer Science, 89, pp.213-217.

[24] Azar, A.T., Elshazly, H.I., Hassanien, A.E. and Elkorany, A.M., 2014. A random forest classifier for lymph diseases. Computer methods and programs in biomedicine, 113(2), pp.465-473.

[25] Artur, M., 2021. Review the performance of the Bernoulli Naïve Bayes Classifier in Intrusion Detection Systems using Recursive Feature Elimination with Cross-validated selection of the best number of features. Procedia Computer Science, 190, pp.564-570.