An Analysis of Support Vector Machines and Decision Trees in Handwritten Digits Recognition

KABIRU LABARAN BALA[1], DR M.S ARGUNGU[2], DR D. GABI[3], AMINU LABARAN[4], ABUBAKAR ABDURARUAF[5]

[1,2,3,5] Department of Computer Science, Abdullahi Fodio University of Science and Technology, Aliero

[4] Department of Computer Science, Federal Polytechnic Kaura Namoda

Abstract- A Support Vector Machines (SVMs) and Decision Trees in handwritten digits recognition. The primary research aim is to find performance and suitability of these algorithms. Most of the time, the physically constructed digits are not inside the domains of proportionate size, due to the similarities between the numbers, such as 1 and 7, 5 and 6, 3 and 8, 2 and 5, 2 and 7. The performance and accuracy of machine learning algorithms heavily depend on the quality and diversity of the dataset used for training and testing. This research limited by the availability of suitable handwritten digit datasets (MNIST), which impact the generalizability of the results. Using MNIST dataset consists of a large number of 28x28 pixel grayscale images of handwritten digits (0 through 9) and 60,000 Training Images and 10,000 Testing. In this study, different machine learning methods, which are SVM and Decision Trees architectures are used to achieve high performance on the digit string recognition problem. In these methods, images of digit strings are trained with the SVM and Decision Trees model methods structure by sliding a fixed size window through the images labeling each sub-image as a part of a digit or not. The research ultimately revealed that Support Vector Machines was the classifier, with 98% accuracy. The lowest score of accuracy goes to Decision Tree with 96% of accuracy and we Recommend the Use of SVM as the Best Algorithms for the handwritten digits recognition but also we suggest that combination of difference Datasets for more accuracy of the recognition MNIST dataset a most of U.S written style..

Index Terms- Machine Learning, Support Vector Machine, Decision Tree, MNIST, digit recognition and Data Set.

I. INTRODUCTION

Two of the best available statistical classification, Support Vector Machines (SVMs) and Decision Trees (DTs), are examined through both necessary modifications as well as traditional implementations to explore the inherent structure of digit appearance in the way of maximal scarification along with necessary complexity. Various techniques also addressed the presence of dynamic information in captured images. The results obtained from competing visual digit databases demonstrate that SVM and DT classifiers jointly could skillfully predict new digit templates by representing the structure at static level and usage of proper resources at capturing level simultaneously. Decision Trees provide an elaborated way to incorporate the dynamic information within static ones that subsequent of such an incorporation leads to one of the best available digit digitization approaches using various distances. Also, it concludes that capturing information present within the static image is inevitable for a practical prediction. Capturing dynamic information over a single static image could lead to a quantifiable level of prediction improvement (Reddy et al., 2020).

Handwriting digits recognition, a challenging task in pattern recognition, has been automatized, and used in many practical applications such as user identification, user verification, and mail sorting (Chakraborty & Bhattacharjee, 2020). Excellent practical performance has already been obtained by using various statistical classification methods. Substantial deviations in extracted characteristic information of digit appearance, as a result of rapid variations in writing style and size, sometimes lead to misclassifications by existing statistical methods. These deviations are primarily due to the ineffective expression of dynamic information over the captured images and under-fit in a poorly expressive functional space. To address these concerns, we perform concurrent analysis on static and dynamic information and with a functional space that contains only information necessary by significantly sparsifying the functional space (Keysers, 2007).

In finally, the background of the research highlights the importance of accurate handwriting digits recognition in various applications and the challenges associated with this task. The comparison analysis of SVMs and Decision Trees aims to contribute to the advancement of this field by providing valuable insights into the performance and suitability of these algorithms for recognizing handwritten digits.

II. RESEARCH METHODOLOGY

In this section, the research describes the various methods, tools, datasets used, how the models are created and how the models were trained are tested. In this section, chapters discuss how algorithms used and presented the block diagram of the proposed system:

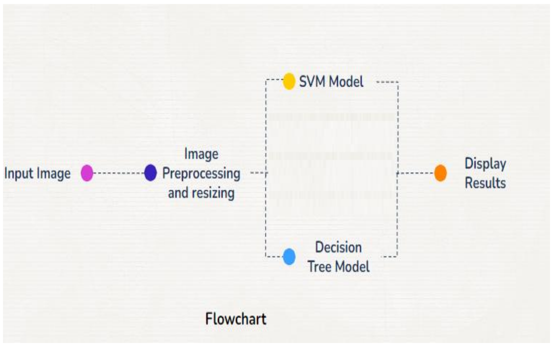


Figure 2.1 Flowchart

2.1 Data Collection

The MNIST dataset, which stands for the Modified National Institute of Standards and Technology dataset, is a widely used dataset in the field of machine learning and computer vision. It was created for the purpose of training and testing various machine learning algorithms for handwritten digit recognition. Here's how the MNIST dataset was collected and obtained (Geetha & Malathi 2023).

A. Data Source: The MNIST dataset consists of a large number of 28x28 pixel grayscale images of handwritten digits (0 through 9). These digits were collected from two primary sources:

a. NIST Special Database 3: This database originally contained binary images of handwritten digits collected from Census Bureau employees and high school students. These images were converted into the MNIST format.

b. NIST Special Database 1: This database contained handwritten characters, including digits and letters, from Census Bureau employees. The digits from this database were also used to create the MNIST dataset.

B. Data Preprocessing: The original images from the NIST databases underwent preprocessing to ensure consistency and usability in machine learning experiments. This preprocessing included normalizing the images to have a fixed size of 28x28 pixels and scaling the grayscale values to a range between 0 and 1.

C. Dataset Split: The MNIST dataset is typically split into two subsets: a training set and a test set. The training set is used to train machine learning models, while the test set is used to evaluate their performance. The dataset also includes labels indicating the correct digit corresponding to each image.

D. Accessibility: The MNIST dataset is made publicly available and can be downloaded from various sources, including the official MNIST website (http://yann.lecun.com/exdb/mnist/), machine learning libraries and data repositories.

It's important to note that while MNIST was a groundbreaking dataset for many years, it is relatively small by modern standards, with only 60,000 training images and 10,000 test images. As a result, it has been largely superseded by larger and more challenging datasets for benchmarking machine learning algorithms. Nonetheless, MNIST remains a valuable resource for educational purposes and for testing and prototyping algorithms in the field of computer vision.

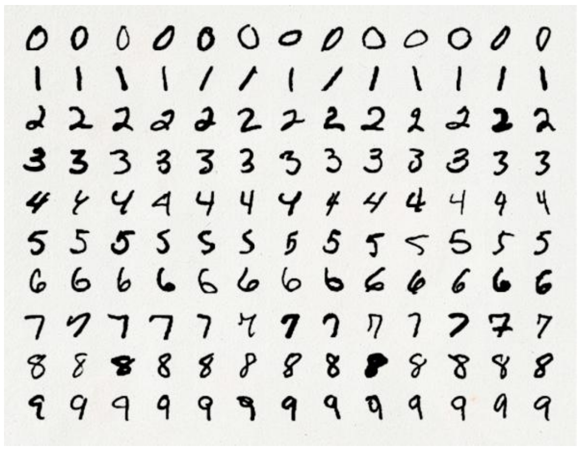


Figure. 2.2 MNIST Data Set Visualization

In addition to the MNIST dataset and our custom dataset, we have also incorporated a supplementary dataset into our program. This additional dataset addresses a specific limitation of MNIST, which primarily consists of American-style numbers, making it challenging to classify isolated numbers, particularly those like "1" and "7".

2.2 Model Implementation

2.2.1 Decision Tree Implementation

A decision tree is a supervised machine learning algorithm used for both classification and regression tasks. It's a hierarchical structure that resembles an upside-down tree, where each node represents a decision or a test on an attribute, each branch represents an outcome of the decision, and each leaf node represents a class label or a predicted value. Decision trees are easy to understand and interpret, making them valuable for both data analysis and decision support (Mendes Gil, & Ferreira 2014).

Components of a Decision Tree:

I. Parent Node: Think of it as the big boss node. It's at the top and tells others what to do.
II. Child Node: These are like the workers or helpers. They listen to the big boss node.
III. Root Node: This is where everything starts. It's like the very first boss who doesn't have anyone above.
IV. Leaf Node/Leaf: These are like the end results. They don't have anyone else to tell what to do.
V. Internal Nodes/Nodes: These are in the middle. They have a boss above them and workers below them.
VI. Splitting: It's like breaking a big task into smaller tasks so it's easier to handle.
VII. Decision Node: When the big boss decides what tasks to split into smaller ones, it's called a decision node.
VIII. Pruning: If the big boss thinks some tasks are not needed anymore, they remove them. It's like cleaning up unnecessary work.
IX. Branch/Sub-tree: It's like looking at just one part of the big task, not the whole thing. Like focusing on one group of workers instead of the whole company.

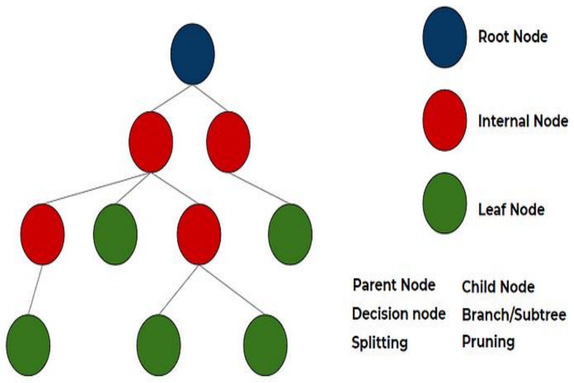


Figure 2.3 Decision Tree Structure

2.2.2 SVM Implementation:

Support Vector Machine (SVM) is a machine learning algorithm used for classification and regression tasks. SVM can be used to separate data into different classes or predict numerical values. One of the key aspects of SVM is its ability to handle both linear and nonlinear problems through the use of kernel functions.

Here's an explanation of SVM in both linear and nonlinear contexts (Rajput, P, & Kumar, S. 2021.):

I. Linear SVM:

- Objective: In linear SVM, the algorithm aims to find a hyperplane that best separates two classes of data points with the largest possible margin between them. This hyperplane is referred to as the "maximum margin hyperplane."

- Linear Separability: Linear SVM is suitable when the data is linearly separable, meaning a straight line (in two dimensions) or a hyperplane (in higher dimensions) can cleanly separate the classes.
- Equation of the Hyperplane: In a two-dimensional space, the equation of the hyperplane can be represented as: w^T * x + b = 0, where w is the weight vector, x is the input feature vector, and b is the bias term.
- Support Vectors: The data points closest to the hyperplane, known as "support vectors," are crucial for defining the margin and the decision boundary. These support vectors help determine the orientation and position of the hyperplane.
- C Parameter: The regularization parameter 'C' in linear SVM controls the trade-off between maximizing the margin and minimizing the classification error. A smaller 'C' value allows for a wider margin but may tolerate some misclassifications, while a larger 'C' value tries to minimize misclassifications at the cost of a narrower margin.

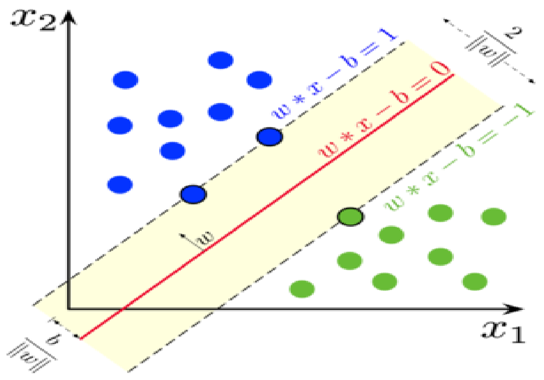


Figure 2.4 SVM Linear

II. Nonlinear SVM:

- Objective: Nonlinear SVM is used when the data is not linearly separable, meaning a straight line or hyperplane cannot cleanly separate the classes in the original feature space.
- Kernel Trick: To address nonlinear problems, SVM employs a technique known as the "kernel trick." Instead of directly transforming the data into a higher-dimensional space, which can be computationally expensive, SVM applies a kernel function to the data in the original feature space. The kernel function computes the dot product between data points in the higher-dimensional space without explicitly transforming them.
- Common Kernels: There are various types of kernel functions, including:
- Polynomial Kernel: Used to capture polynomial relationships in the data.
- Radial Basis Function (RBF) Kernel: Suitable for capturing complex, nonlinear patterns in the data.
- Sigmoid Kernel: Can model sigmoidal (S-shaped) decision boundaries.
- Kernel Parameters: Depending on the kernel chosen, there may be additional hyperparameters to fine-tune, such as the kernel width in the RBF kernel.
- ‘Effectiveness: Nonlinear SVM is effective in capturing intricate relationships within the data and is capable of handling complex, nonlinear separations.

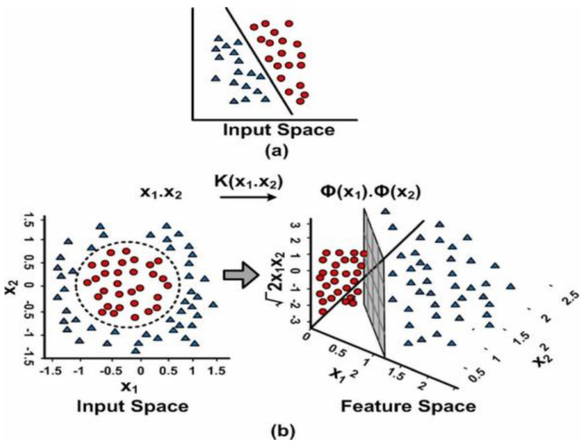


Figure 2.5 SVM Non Linear

2.2.3Programming Environment:

Python is a high-level, interpreted, and versatile programming language known for its simplicity, readability, and broad applicability. It is characterized by its (Rathore, L., & Yadav, R. 2023.):

I. Readability: Python emphasizes clean and readable code through indentation.
II. Interpreted Nature: Code is executed without prior compilation, making development fast and accessible.
III. General-Purpose: Python can be used for a wide range of applications, from web development to scientific computing and artificial intelligence.

IV. Cross-Platform: Python runs on multiple operating systems.
V. Extensive Standard Library: Python comes with a rich set of built-in modules for various tasks.
VI. Dynamic Typing: Data types are inferred at runtime, enhancing flexibility.
VII. Community and Ecosystem: Python has a large and active developer community, with a rich ecosystem of third-party libraries and frameworks.
VIII. Object-Oriented: Python supports object-oriented programming.
IX. Indentation: It uses whitespace for defining code blocks, promoting a consistent coding style.
X. Open Source: Python is open-source and freely available.

Below are the required libraries. (Rathore, L., & Yadav, R. 2023).

I. pandas as pd: The pandas library is used for data manipulation and analysis. It provides data structures like DataFrames and Series, which are helpful for handling structured data, such as tabular data. The alias pd is a common convention used to make it easier to reference functions and objects from the pandas library throughout the code.
II. seaborn as sns: Seaborn is a data visualization library that is built on top of Matplotlib. It provides a high-level interface for creating informative and attractive statistical graphics. The alias sns is commonly used to reference seaborn functions and settings.
III. from sklearn.model_selection import train_test_split: This import statement is used to import the train_test_split function from the model_selection module of the Scikit-Learn library (abbreviated as sklearn). This function is often used to split a dataset into training and testing subsets for machine learning model development and evaluation.
IV. from sklearn import: This statement is a wildcard import of all submodules within Scikit-Learn. While wildcard imports are generally discouraged, this particular import is used to simplify the code for educational purposes. In practice, it's better to import only the specific modules or classes you need.
V. import numpy as np: The numpy library is essential for numerical and array-based operations in Python. It provides support for multi-dimensional arrays and mathematical functions. The alias np is a common convention used to reference numpy functions and objects throughout the code.
VI. import matplotlib.pyplot as plt: Matplotlib is a widely-used library for creating static, animated, and interactive visualizations in Python. The pyplot submodule is commonly imported as plt to simplify plotting commands.
VII. from sklearn.preprocessing import StandardScaler: Scikit-Learn provides various preprocessing techniques, and StandardScaler is a class used for standardizing or scaling features in a dataset. It helps ensure that features have similar scales, which is often important for machine learning algorithms.
VIII. from sklearn.svm import SVC: This import statement is used to import the SVC class, which stands for Support Vector Classification. It's part of Scikit-Learn and is used for implementing Support Vector Machine (SVM) classification models.
IX. from sklearn.pipeline import Pipeline: Scikit-Learn's Pipeline class allows you to create a sequence of data preprocessing and model training steps that can be executed in a structured and automated way.
X. from sklearn import metrics: This import statement brings in various metrics for evaluating machine learning models, such as accuracy, precision, recall, etc. It's used to assess the performance of models.
XI. from sklearn.metrics import accuracy_score: This specific import brings in the accuracy_score function, which is used to compute the accuracy of classification models by comparing predicted labels to actual labels.
XII. import pickle: The pickle module is used for serializing and deserializing Python objects. It's often used for saving and loading machine learning models.
XIII. from joblib import dump, load: Joblib is another library for serializing Python objects, often used for saving and loading large machine learning models more efficiently than

pickle. dump is used to save objects, and load is used to load them.

XIV. from sklearn import tree, metrics: This line imports two modules/classes, tree and metrics, from Scikit-Learn (Sharma, D., Singh, I., & Pandey, U. 2022.).

1. tree: This module provides the necessary components for working with decision trees, including building decision tree classifiers and regressors.
2. metrics: This module contains various metrics and functions for evaluating machine learning models' performance, such as accuracy, precision, recall, etc.

XV.from sklearn.tree import DecisionTreeClassifier: This line specifically imports the DecisionTreeClassifier class from the tree module within Scikit-Learn. The DecisionTreeClassifier is a machine learning algorithm used for classification tasks. It builds a decision tree model based on the training data and uses it to make predictions.

2.3 Model Training and Testing

Training Procedure: handwritten digit recognition using machine learning algorithms like Support Vector Machines (SVM) and Decision Trees, it's crucial to divide the dataset into training and testing subsets to evaluate the models' performance accurately. The MNIST dataset is a commonly used dataset for this task, containing 28x28 pixel grayscale images of handwritten digits from 0 to 9. Here's how we divide the MNIST dataset for training and testing (Gomathy, C. K., & Jaya Sairam, K. 2021):

I. Dataset Description: Start by obtaining the MNIST dataset, which typically consists of a large number of labeled examples (e.g., 60,000 training images and 10,000 testing images).
II. Data Preprocessing: Before splitting the data, we preprocess it. This includes tasks such as normalization, feature extraction, and data augmentation (optional). For MNIST, you might simply normalize pixel values to be in the range [0, 1] by dividing by 255.
III. Splitting the Dataset:

a. Training Set: The training set is used to train the machine learning models. It's crucial for learning the patterns and features in the data. You typically reserve a significant portion of the dataset for training, often around 80% of the data. For MNIST with 60,000 samples, this would be approximately 60,000 samples.
b. Testing Set: The testing set is used to evaluate the model's performance on unseen data. You should set aside a smaller portion of the dataset for testing, typically around 20% of the data. In the case of MNIST, this would be approximately 10,000 samples.
c. Validation Set (Optional): In some cases, you might create a validation set in addition to the training and testing sets. The validation set helps in tuning hyperparameters and preventing overfitting. You can typically use around 10-20% of the training data for validation.

IV. Implementing the Split:
a. Since we are working with Python programming language, we can use libraries such as NumPy, scikit-learn, or TensorFlow/Keras to efficiently split the dataset. For instance, in scikit-learn:

```
python
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split
(X, y, test_size=0.2, random_state=42)
```

Here, X represents the feature matrix (image data), and y represents the corresponding labels.

V. Model Training and Evaluation: After splitting the dataset, we can proceed to train your machine learning models (e.g., SVM and Decision Tree) on the training set and evaluate their performance on the testing set. We can use metrics like accuracy, precision, and confusion matrices to assess the model's performance.

III. RESULTS AND DISCUSSIONS

A model's performance is evaluated based on different metrics such as accuracy, and precision score. The different evaluation metrics will allow data analysts to understand a model's performance based on its strengths and weaknesses. For this project, two models, CNN, MLP and SVM will be evaluated based on certain metrics and a decision will be made to choose the best algorithm for handwritten digit recognition.

3.1 Processing Result

To obtaining good analytic results, the pre-processing results contained the 10 nominal Digits. Figure 4.1 displays the Distribution of 10 Digits in MNIST Training Set that was utilized in this research and Figure 4.2 Show the Sample Digits of the Data Set.

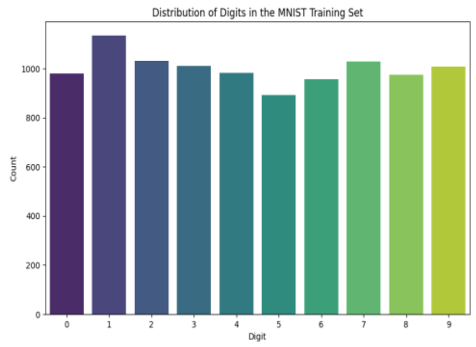


Figure 3.1 Distribution of Digits in the MNIST Training Set

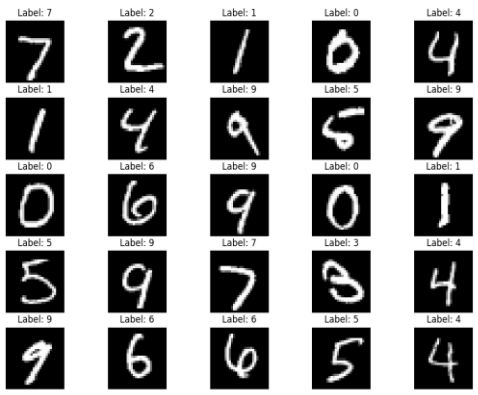


Figure 3.2 Sample Digits of the Dataset

3.2 Performance Results of the Classifiers
Here are the experimental results that were driven by the performance metrics that were covered in the Methodology.

In the final experimentation of the models which include training and testing the models, Support Vector Model has the accuracy of 98 while Decision Tree has the accuracy of 94%. The results of the algorithms are clearly summarized in Table 3.

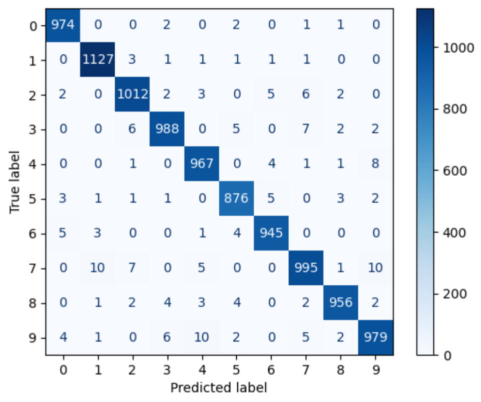


Figure 3.3 Confusion matrix using SVM on test dataset

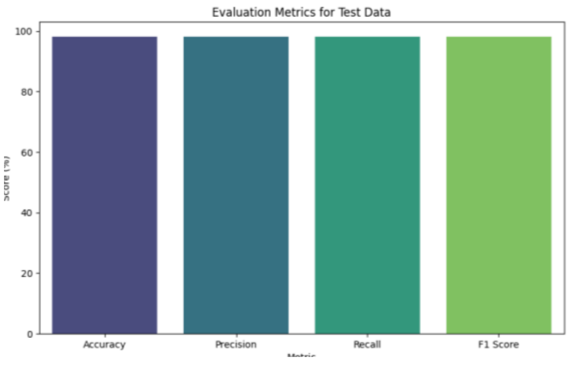


Figure 3.4 : SVM Testing result.

Figure 3.3 and figure.3.4 above show Confusion matrix using SVM on test dataset and Models Result Respectibily. The Final result of the SVM Model after the Implementation; the model had an Testing accuracy of 98%, Precision Score of 98%, Recall Score of 98% and F1 Score of 98%.

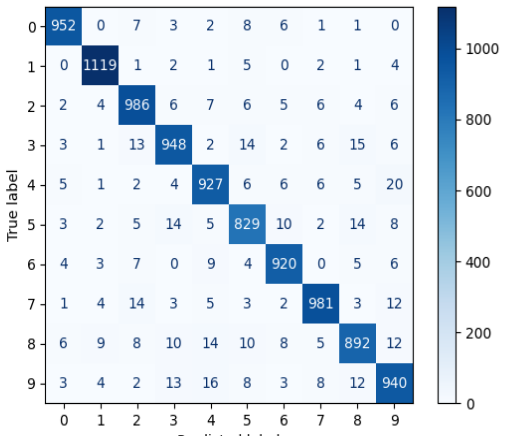


Figure 3.5 Confusion matrix using Decision Tree on test dataset

Figure 3.5 show the Confusion Matrix of Decission Tree were the result show that '1' as thas the lowest Correct Predicat while 5 has the Lowest.

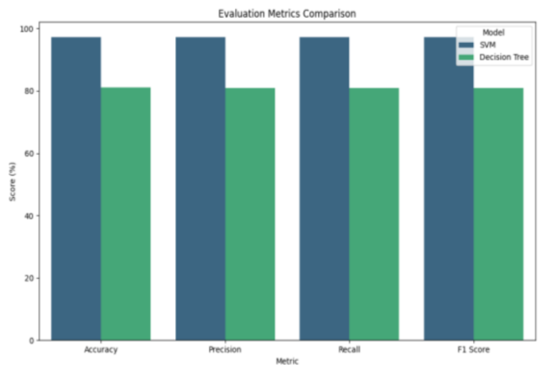


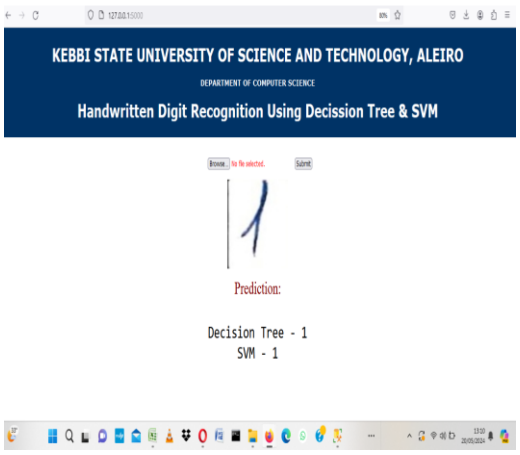
Figure 3.6 Evaluate SVM and Decision Tree

The Figure 3.6 Above Shows that Graphtical Represention of the Result were we can identify that the SVM Model is presented in a bar of blue colour while the Decision tree is presentation in bar of green colour and also the bar Chart show clearly that in all Eveluation Support Vector Machines (SVM) has the highest Performance than the Decision Tree.

Table 3.1 Shows the Result of the Evaluation below.

	Metric	SVM	Decision Tree
0	Accuracy	98.37	95.88
1	Precision	98.366593	95.826053
2	Recall	98.350036	95.817291
3	F1 Score	98.357676	95.817488

Table 3.1 Result


KEBBI STATE UNIVERSITY OF SCIENCE AND TECHNOLOGY, ALEIRO

DEPARTMENT OF COMPUTER SCIENCE

Handwritten Digit Recognition Using Decission Tree & SVM

Prediction:

Decision Tree - 1
SVM - 1

Figure 3.7 GUI Design.

Figure 3.7 Above Shows the Web Interface of the models where we upload the image of digits "1" and the Predicted Result of each algorithm is "1" which is correct.

CONCLUSION

In this Research handwritten digits from MNIST database are trained and tested using two Machine Learning algorithms such as Decision Tree, Support Vector Machine. 60000 samples are used for training the model and 10000 samples are used for testing the model. The models are compared based on their accuracy, precision, recall and f1 score

REFERENCES

[1] Aruna Jyothi, & Saideep Reddy. (2021). Handwritten digit recognition using double convolution neural network. *Ijcrt*, *2320-2882*.

[2] Chakraborty, P., Jahanapi, S. S., & Choudhury, T. (2022). Bangla handwritten digit recognition. In J. M. R. S. Tavares et al. (Eds.), *Cyber intelligence and information retrieval* (Lecture Notes in Networks and Systems, Vol. 291, pp. page numbers of the relevant chapter). Springer Nature Singapore Pte Ltd. https://doi.org/10.1007/978-981-16-4284-5_14

[3] Geetha, S., & Malathi, K. (2023). SVM vs decision tree algorithm: Cost effective comparison to enhance crime detection and prevention. *Journal of Survey in Fisheries Sciences*, *10*(1S), 2814-2822.

[4] Gomathy, C. K., & Jaya Sairam, K. (2021). The handwritten digits recognition. *International Journal of Engineering Applied Sciences and Technology*, *6*(7), 203-206. Retrieved from http://www.ijeast.com

[5] Kumar, C. V., & Sriramya, P. (2022). Comparison of SVM algorithms with decision trees for accurate recognition to handwritten digits to improve the accuracy value. *Baltic Journal of Law & Politics*, *15*(4), 295-303. https://doi.org/10.2478/bjlp-2022-004031

[6] Kusetogullari, H., Yavariabdi, A., Hall, J., & Lavesson, N. (2021). DIGITNET: A deep handwritten digit detection and recognition

method using a new historical handwritten digit dataset. *Big Data Research*, *23*, 100182.

[7] Mendes Gil, & Ferreira. (2014). Handwritten digit recognition using SVM binary classifiers and unbalanced decision trees. *International Publishing Switzerland*.

[8] Nader, L., Mohamed, A., Nazir, M., & Awadalla, M. (2018). Identification of writer's gender using handwriting analysis. *International Journal of Scientific and Research Publications*, *8*(10), 728. https://doi.org/10.29322/IJSRP.8.10.2018.p8288

[9] Priya, Singh, R., & Changlani, S. (2017). Review on handwritten digit recognition. *International Journal of Novel Research and Development (IJNRD)*, *2*(4), 91.

[10] Raj, A., Sharma, S., Singh, J., & Singh, A. (2023). Revolutionizing data entry: An in-depth study of optical character recognition technology and its future potential. *International Journal for Research in Applied Science & Engineering Technology (IJRASET)*, *11*(2), 645.

[11] Rajput, P., Nahar, J., & Kumar, S. (2021). Handwritten digit recognition using Python [Review of handwritten digit recognition using Python]. *Ijcrt*, *2320-2882*(2320-2882).

[12] Rathore, L., & Yadav, R. (2023). Advancements in handwritten digit recognition: A literature review of machine learning and deep learning approaches. *International Journal for Research in Applied Science & Engineering Technology (IJRASET)*, *11*(8), 1879

[13] Rzayeva, L., Myrzatay, A., Abitova, et al. (2023). Enhancing LAN failure predictions with decision trees and SVMs: Methodology and implementation.

[14] Sharma, D., Singh, I., & Pandey, U. (2022). Handwritten digit recognition. *International Research Journal of Engineering and Technology (IRJET)*, *09*(06), 1185.

[15] Singh, P., Pawar, P., & Raj, N. (2022). Handwritten digit recognition. *International Journal for Research in Applied Science & Engineering Technology (IJRASET)*, *10*(5), 75.

[16] Chakraborty, P., Jahanapi, S. S., & Choudhury, T. (2022). Bangla handwritten digit recognition. In J. M. R. S. Tavares et al. (Eds.), *Cyber intelligence and information retrieval* (Lecture Notes in Networks and Systems, Vol. 291, pp. page numbers of the relevant chapter). Springer Nature Singapore Pte Ltd. https://doi.org/10.1007/978-981-16-4284-5_14

[17] Rani, S., Ur Rehman, A., Yousaf, B., Tayyab Rauf, H., Abouel Nasr, E., & Kadry, S. (2022). Recognition of handwritten medical prescription using signature verification techniques. *ncbi.nlm.nih.gov*

[18] Pashine, S., Dixit, R., & Kushwah, R. (2021). Handwritten digit recognition using machine and deep learning algorithms

[19] Hamdi, Y., Akouaydi, H., Boubaker, H., & Alimi, A. M. (2020). Handwriting quality analysis using online-offline models

[20] Liu, X., Guo, L., Wang, H., Guo, J., Yang, S., & Duan, L. (2022). Research on imbalance machine learning methods for MR[Formula: see text] WI soft tissue sarcoma data. *ncbi.nlm.nih.gov*

[21] Gornale, S., Kumar, S., Patil, A., & Hiremath, P. S. (2021). Behavioral biometric data analysis for gender classification using feature fusion and machine learning. *ncbi.nlm.nih.gov*

[22] Singh, A., Sharma, A., Ahmed, A., Sundramoorthy, A. K., Furukawa, H., Arya, S., & Khosla, A. (2021). Recent advances in electrochemical biosensors: Applications, challenges, and future scope. *ncbi.nlm.nih.gov*

[23] Dinges, L., Al-Hamadi, A., Elzobi, M., & El-etriby, S. (2016). Synthesis of common Arabic handwritings to aid optical character recognition research. *ncbi.nlm.nih.gov*

[24] Saha, D., & Manickavasagan, A. (2021). Machine learning techniques for analysis of hyperspectral images to determine quality of food products: A review. *ncbi.nlm.nih.gov*

[25] Sharma, A. (2012). Handwritten digit recognition using support vector machine