

PROFESSIONAL TRAINING REPORT - II

entitled

Efficient Machine Learning Algorithm Recommendation System

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering with
specialization in Artificial Intelligence and Robotics

by

Balu Monish (41612009)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

CATEGORY -1 UNIVERSITY BY UGC

**Accredited with Grade "A++" by NAAC | 12B Status by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119**

APRIL 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

CATEGORY-I UNIVERSITY BY UGC

Accredited "A++" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Professional Training Report is the bonafide work of Mr. Balu Monish (41612009) who carried out the project entitled "Efficient Machine Learning Algorithm Recommendation System" under my supervision from January 2024 to April 2024.

Internal Guide

Dr. SONIA JENIFER RAYEN, M.Tech, Ph.D.,

Head of the Department

Dr. S. VIGNESHWARI, M.E., Ph.D.,

Submitted for Viva voce Examination held on 7th May 2024

Internal Examiner

External Examiner

DECLARATION

I, Balu Monish (41612009), hereby declare that the Professional Training Report-II entitled "Efficient Machine Learning Algorithm Recommendation System" done by me under the guidance of Dr. Sonia Jenifer Rayen, M.Tech, Ph.D., is submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering with specialization in Artificial Intelligence and Robotics.

DATE: 7th May 2024

PLACE: Chennai



SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph.D., Dean, School of Computing, Dr. S.Vigneshwari M.E., Ph.D., Head of the Department of Computer Science and Engineering** for providing me necessary support during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Internal Guide **Dr. Sonia Jenifer Rayen, M.Tech, Ph.D.**, for her valuable guidance, suggestions and constant encouragement which paved way for the successful completion of my phase-2 Professional Training.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

COURSE CERTIFICATE



Introduction to Career Skills in Data Analytics

Course completed by Balu Monish
Mar 10, 2023 at 01:40PM UTC • 2 hours 21 minutes

Top skills covered

Data Analytics

Tech Career Skills

Head of Content Strategy, Learning

Certificate ID: 9513e0d73514e519a36474c0c5b16c6fdfe951b41c675f2aec035e07bda49f8c



Career Essentials in Data Analysis by Microsoft and LinkedIn

Learning Path completed by Balu Monish
Mar 12, 2023 at 12:24PM UTC • 9 hours 13 minutes

Top skills covered

Data Analysis

Data Visualization

Data Analytics

Head of Content Strategy, Learning

Certificate ID: 27a7c3d33d44078fb82141ad20cc04d3e8c8f6c8be5d100d638afdbceb7e5267



ABSTRACT

As the domain of machine learning continues to evolve, there is an increasing demand for understanding the efficiency and effectiveness of various algorithms across different tasks. This research paper presents a comprehensive investigation into the efficiency of diverse machine learning algorithms, aiming to provide insights into their performance metrics, computational demands, and scalability. Through a systematic empirical analysis, we evaluate the efficiency of prominent machine learning algorithms including but not limited to decision trees, support vector machines, neural networks, k-nearest neighbors, and ensemble methods. The evaluation encompasses multiple dimensions such as model training time, prediction speed, memory consumption, and scalability to large datasets. Furthermore, we explore the impact of algorithmic parameters, dataset characteristics, and hardware configurations on algorithm efficiency. Our findings shed light on the relative strengths and weaknesses of different machine learning algorithms in terms of efficiency, offering valuable guidance to practitioners and researchers in selecting appropriate algorithms for specific applications. Additionally, we discuss potential avenues for enhancing the efficiency of machine learning algorithms through algorithmic optimizations, parallel computing techniques, and hardware acceleration. Overall, this research contributes to the advancement of machine learning by providing empirical evidence and practical recommendations for optimizing algorithmic efficiency, thereby facilitating the development of scalable and resource-efficient machine learning systems.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	vi
	LIST OF FIGURES	viii
1	INTRODUCTION	1
	1.1 Overview	
2	LITERATURE SURVEY	4
	2.1 survey	
3	REQUIREMENTS ANALYSIS	
	3.1 Objective	7
	3.2.1 Hardware Requirements	
	3.2.2 Software Requirements	8
4	DESIGN DESCRIPTION OF PROPOSED PRODUCT	12
	4.1.1 Ideation Map/Architecture Diagram	12
	4.1.2 Various stages	14
	4.1.3 working principles	15
	4.2.1 Novelty of the Project	16
	4.3 Results	17
5	CONCLUSION	19
	REFERENCES	20
	APPENDIX	
	A. Research Paper	23
	B. Source Code	29

LIST OF FIGURES

Figure No.	Figure Name	Page No.
4.1	Algorithmic Flow of Program	12
4.3.1	Results of the Experiment	17
4.3.2	Regular vs K-Fold vs Grid search	18
4.3.3	Algorithm Predicted vs Observed Graph	18

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Machine learning (ML) is a rapidly evolving field that has gained significant prominence due to its ability to solve complex problems across various domains. As the deployment of ML algorithms becomes more widespread, a comprehensive understanding of their efficiency and effectiveness across diverse tasks is crucial. This overview examines the key machine learning algorithms included in the study, focusing on their principles, how they work, and their use cases.

Decision trees are based on a hierarchical, tree-like structure that splits data into groups based on feature values. Each decision node represents a feature, and each branch from the node corresponds to a possible feature value, directing the data down different paths. The algorithm works by recursively dividing the data into increasingly homogeneous subsets based on feature values, continuing until leaf nodes are reached, representing final decisions or classifications. Decision trees are widely used in classification tasks such as credit risk assessment and medical diagnosis, as well as in regression tasks like estimating house prices.

Support Vector Machines (SVM) aims to find an optimal separating hyperplane between different classes in a feature space. This hyperplane maximizes the margin between data points of different classes, allowing for robust classification. The algorithm operates by transforming the data into a higher-dimensional space using kernel functions (e.g., linear, polynomial, radial basis function) to find a decision boundary that best separates the data. SVM is commonly used in text classification, image classification, and bioinformatics for tasks such as disease diagnosis based on genetic data.

Neural networks are composed of interconnected layers of artificial neurons that process input data to produce outputs. These networks can capture complex relationships within the data by adjusting the weights of the connections between neurons. The algorithm involves forward propagation of data through the network and backpropagation of errors to adjust the weights. Activation functions such as sigmoid, ReLU, and tanh help model non-linearities in the data. Neural networks are highly versatile and are used in a range of applications, including image recognition, natural language processing, and autonomous vehicles.

k-Nearest Neighbors (k-NN) is a non-parametric, instance-based algorithm that classifies a query point based on the majority vote or weighted average of its k nearest neighbors. The algorithm calculates distances (e.g., Euclidean, Manhattan) between the query point and all data points in the training set, then identifies the k closest neighbors. Classification or regression is determined by the average or majority class of these neighbors. k-NN is useful in recommendation systems, anomaly detection, and tasks requiring a measure of similarity, such as customer segmentation.

Naive Bayes is a probabilistic algorithm based on Bayes' theorem, with the assumption that features are independent given the class. This allows for efficient calculation of class probabilities. The algorithm calculates the likelihood of the input data belonging to each class and chooses the class with the highest probability. It uses different distributions (e.g., Gaussian, Bernoulli, Multinomial) depending on the nature of the data. Naive Bayes is effective in spam filtering, document classification, and sentiment analysis due to its simplicity and ability to handle high-dimensional data.

Principal Component Analysis (PCA) is a dimensionality reduction technique that identifies the most significant features of a dataset by transforming it into a new coordinate system. This transformation preserves as much variance as possible while reducing the number of dimensions. The algorithm calculates the covariance matrix of the data and extracts its eigenvalues and eigenvectors. The eigenvectors with the highest eigenvalues are selected as the principal components, representing the most important features. PCA is useful in preprocessing data, reducing noise, and visualizing high-dimensional data. It can improve the efficiency of machine learning algorithms and is commonly used in data compression and feature selection.

L1 (Lasso) and L2 (Ridge) regularization are techniques used to prevent overfitting in machine learning models by adding penalties to the model's complexity. L1 regularization adds a penalty proportional to the absolute value of the coefficients, encouraging sparsity in the model. L2 regularization adds a penalty proportional to the square of the coefficients, providing stability to the model's weight values. Both regularization methods are commonly used in linear and logistic regression, as well as in other models such as neural networks. They help improve model generalizability and robustness, making them valuable in various prediction and classification tasks.

Ensemble methods combine multiple models to improve performance and reduce variance. Techniques such as bagging, boosting, and stacking leverage the strengths of different models to create a more robust model. Bagging (Bootstrap Aggregating) involves training multiple instances of the same model on different subsets of the data and averaging their predictions. Boosting involves sequentially training weak learners and adjusting their weights based on previous errors. Stacking combines the predictions of different models through a meta-learner. Ensemble methods are known for their accuracy and robustness and are widely used in classification and regression tasks such as credit scoring, financial modelling, and medical diagnosis.

The study of the efficiency of these machine learning algorithms across different tasks highlights the trade-offs between their operational mechanisms and performance metrics. When selecting an appropriate algorithm for a specific application, practitioners should consider the data characteristics, task requirements, and computational resources available. Further research into optimizing these algorithms through parallel computing and hardware acceleration may lead to advancements in the field and the development of more efficient ML systems.

The study of the efficiency of these machine learning algorithms across different tasks highlights the trade-offs between their operational mechanisms and performance metrics. When selecting an appropriate algorithm for a specific application, practitioners should consider the data characteristics, task requirements, and computational resources available. Further research into optimizing these algorithms

through parallel computing and hardware acceleration may lead to advancements in the field and the development of more efficient ML systems.

A rigorous assessment of the efficiency of machine learning algorithms necessitates the consideration of several critical testing parameters to fully understand their performance in practical applications. Time complexity, which measures the duration required by an algorithm to train a model and make predictions, is vital for determining the feasibility of deploying the algorithm in time-sensitive scenarios. Space complexity assesses the memory requirements of an algorithm, which can significantly impact its usability when working with large-scale datasets or in resource-limited environments.

Furthermore, accuracy score serves as a quantitative measure of an algorithm's effectiveness in correctly predicting outcomes. This metric provides insight into the model's precision and reliability across various tasks. Processor usage evaluates the computational demands placed on the system by the algorithm, which can influence its scalability and compatibility with available hardware infrastructure.

By systematically evaluating these testing parameters, researchers and practitioners can make well-informed decisions regarding the selection and implementation of machine learning algorithms for specific use cases. This thorough analysis ensures the optimization of performance, resource efficiency, and compatibility with existing systems, ultimately contributing to the development of robust and scalable machine learning solutions.

CHAPTER 2

LITERATURE REVIEW

2.1 SURVEY

A comprehensive review of the recent literature reveals a variety of extensive investigations into the efficiency and performance of machine learning (ML) algorithms across various domains such as biomedical research, energy efficiency, manufacturing, DNA sequence classification, and banking operations. These studies demonstrate the significant potential of ML algorithms in providing robust solutions for complex data analysis tasks and in achieving sustainable advancements across these application areas.

A study investigating the performance and efficiency of ML algorithms for analyzing rectangular biomedical data undertook a detailed comparison of decision trees (DT), random forests (RF), artificial neural networks (ANN), and support vector machines (SVM) for classifying multi-category outcomes. Researchers applied these algorithms to a large dataset of female breast cancer cases, noting comparable levels of accuracy across the algorithms, thereby establishing the reliability of these approaches in classifying complex biomedical datasets. This study also emphasized the value of dimension reduction based on information gain to enhance model efficiency while maintaining classification accuracy.

In the context of pursuing greener AI, a study evaluated the performance and energy efficiency of decision tree algorithms, exploring the computational performance and energy consumption across various datasets. This research probed the relationships between energy efficiency and predictive performance, as well as the identification of energy hotspots within the algorithms. These insights contribute to the optimization of ML algorithms for sustainable computing practices, supporting the growing field of green AI.

A study exploring the integration of ML algorithms with Data Envelopment Analysis (DEA) for measuring and predicting the efficiency of Chinese manufacturing listed companies evaluated DEA efficiency scores using a combination of back-propagation neural networks (BPNN-DEA), genetic algorithms integrated with BPNN (GANN-DEA), support vector machines (SVM-DEA), and improved SVM (ISVM-DEA). Among these models, GANN-DEA emerged as the most accurate method, demonstrating the potential of combining ML and DEA for effective efficiency prediction in manufacturing.

The application of ML techniques for pattern matching in DNA sequence classification was explored in another study. Researchers proposed a novel model that combined ML methods with a pattern-matching algorithm to effectively classify DNA sequences based on their features. The study revealed that the SVM linear classifier achieved the highest accuracy and F1 score among the algorithms tested, outperforming other methods in DNA sequence classification tasks. Additionally, the study analyzed the impact of pattern length on accuracy and time complexity, emphasizing the importance of fine-tuning models for specific applications.

A comparative study examined the use of ML algorithms to predict bank operational efficiency, particularly in the context of a financial crisis. By combining DEA with three ML approaches—decision trees (DT), random forest (RF), and neural networks (NN)—the study evaluated the efficiency and performance of Ghanaian bank branches. Decision trees demonstrated the highest predictive accuracy, closely followed by random forests, while neural networks performed slightly lower but still offered significant insights.

Overall, these studies underscore the versatility and adaptability of ML algorithms across diverse domains. The combination of ML with other analytical techniques such as DEA provides researchers with a more comprehensive understanding of efficiency and performance metrics. These findings highlight the importance of optimizing ML algorithms in terms of energy consumption and resource utilization, particularly in the pursuit of greener AI and sustainable solutions. Through these advanced analytical approaches, practitioners and researchers can enhance the efficiency and accuracy of data-driven processes, paving the way for innovative advancements across multiple fields.

In conclusion, the recent literature on the efficiency and performance of machine learning (ML) algorithms across various domains highlights the transformative impact of these advanced techniques in a range of applications. From biomedical research to energy efficiency, manufacturing, DNA sequence classification, and banking operations, ML algorithms have demonstrated their capacity to deliver reliable and robust solutions for complex data analysis tasks.

Studies comparing decision trees, random forests, artificial neural networks, and support vector machines in biomedical data classification have shown that these algorithms perform with comparable accuracy and can be optimized for enhanced efficiency through dimension reduction. The pursuit of greener AI through evaluating the energy efficiency of decision tree algorithms has provided valuable insights into the relationship between energy consumption and predictive performance, highlighting the importance of optimizing ML algorithms for sustainable computing practices.

Incorporating ML with Data Envelopment Analysis (DEA) for measuring and predicting the efficiency of Chinese manufacturing listed companies has demonstrated the potential of combining these methods for effective efficiency prediction. Similarly, using ML techniques for pattern matching in DNA sequence classification has proven effective in accurately categorizing DNA sequences and enhancing model performance.

In the context of banking operations, the application of ML approaches such as decision trees, random forests, and neural networks, in combination with DEA, has enabled the assessment of bank efficiency and performance with remarkable

accuracy. These studies underscore the versatility and adaptability of ML algorithms across different domains, offering significant advancements in data-driven processes and innovative solutions.

Overall, the body of research reviewed provides compelling evidence of the broad applicability and potential of ML algorithms in addressing complex challenges across various fields. The combination of ML with other analytical techniques such as DEA enhances understanding and optimization of efficiency and performance metrics. As ML continues to evolve, these findings pave the way for future developments and improvements across a multitude of sectors, driving innovation and growth.

CHAPTER 3

REQUIREMENTS ANALYSIS

3.1 OBJECTIVE OF THE PROJECT

The objective of the project is to conduct a comprehensive examination of the performance and efficiency of various machine learning (ML) algorithms across multiple domains, including biomedical research, energy efficiency, manufacturing, DNA sequence classification, and banking operations.

The study aims to assess and compare the predictive accuracy, time complexity, energy consumption, and overall resource utilization of different ML algorithms such as decision trees, random forests, artificial neural networks, and support vector machines. By analyzing these algorithms in distinct application areas, the project seeks to identify their strengths and weaknesses in handling complex data analysis tasks.

Additionally, the project endeavours to explore the impact of combining ML with other analytical techniques such as Data Envelopment Analysis (DEA) to measure and predict efficiency in different settings. By incorporating ML with DEA, the study aims to provide more comprehensive insights into efficiency metrics and offer enhanced methods for assessing performance.

Another key objective is to investigate the relationship between energy efficiency and computational performance in the context of achieving greener AI. The project will evaluate the energy consumption of ML algorithms, particularly decision tree algorithms, and identify energy hotspots to optimize algorithms for sustainable computing.

Ultimately, the project strives to provide practitioners and researchers with valuable guidance in selecting the most suitable algorithms for specific applications. By offering insights into the relative performance, efficiency, and resource utilization of different ML algorithms, the project contributes to the development of more effective, efficient, and sustainable ML systems across a range of fields.

3.2 REQUIREMENTS

3.2.1 HARDWARE REQUIREMENTS

1. **High-Performance Computing (HPC) Infrastructure:** The project requires access to high-performance computing infrastructure to efficiently analyse the performance and efficiency of various machine learning (ML) algorithms across multiple domains. The minimum specifications include a multi-core CPU with 4 to 8 cores and at least 16 GB of RAM. For optimal performance, a high-end multi-core CPU with 12 to 24 cores for parallel processing and 32 GB or more RAM is recommended for handling large datasets and complex models effectively.
2. **Graphical Processing Units (GPUs):** GPUs play a significant role in accelerating the training process, especially for deep learning tasks. The minimum specification includes a mid-range GPU with 4 to 8 GB of VRAM, while a high-end GPU with 16 GB or more VRAM is recommended for demanding tasks.
3. **Ample RAM:** Effective data processing and model training require sufficient memory. The minimum requirement is 16 GB of RAM for basic operations, while 32 GB or more is recommended for efficiently handling large datasets and resource-intensive algorithms.
4. **Storage Capacity:** Storage is crucial for storing and accessing large datasets. The minimum requirement is 256 GB of storage, preferably solid-state drive (SSD), for faster data read and write speeds. A 1 TB SSD or more is recommended for ample space and faster access to large datasets.
5. **Network Connectivity:** Network connectivity is essential for accessing remote data sources and distributed computing resources. The minimum requirement is basic broadband internet, while high-speed, stable internet connectivity is recommended for seamless data access.
6. **Energy-Efficient Hardware:** The project benefits from energy-efficient hardware to minimize power consumption and contribute to sustainable computing. Standard CPUs and GPUs with basic energy-saving features are sufficient at a minimum. Advanced power management features in energy-efficient CPUs and GPUs are recommended for optimal performance.
7. **Parallel Processing Capabilities:** Parallel processing capabilities enhance the efficiency of ML tasks. The minimum requirement includes basic support for parallel processing, such as multi-threading on CPU. For optimal performance, advanced parallel processing support with high core count CPUs and multiple GPUs is recommended.
8. **Compatibility with ML Libraries and Frameworks:** Compatibility with ML libraries and frameworks such as scikit-learn and TensorFlow is essential for implementing and testing various ML algorithms. Compatibility with a broader range of ML frameworks, including TensorFlow and PyTorch, is recommended for versatility and advanced model development.

3.2.2 SOFTWARE REQUIREMENTS

1. **Python Programming Language:** The preferred programming language for this project is Python due to its ease of use, versatility, and extensive support for machine learning and data analysis tasks. Python provides access to numerous libraries and frameworks that streamline machine learning (ML) processes, making it an essential software requirement.
2. **Jupyter Notebook:** Jupyter Notebook is a powerful open-source tool for data analysis and interactive computing. It offers an interactive environment for developing, testing, and documenting code. Supporting various programming languages, Jupyter Notebook allows for easy visualization and sharing of results, making it a valuable asset to the project.
3. **Machine Learning Libraries:** The project relies on machine learning libraries to implement and evaluate different ML algorithms. Key libraries include:
 - **scikit-learn:** A comprehensive ML library providing a wide range of tools for classification, regression, clustering, and dimensionality reduction. Scikit-learn is user-friendly and well-documented, making it a popular choice for ML tasks.
 - **TensorFlow:** A powerful library for building and training neural network models. TensorFlow supports deep learning and other advanced ML techniques and offers efficient tools for model development and deployment.
 - **PyTorch:** Another popular library for deep learning and machine learning. PyTorch is known for its dynamic computational graph and ease of use, which facilitates model building and experimentation.
4. **Data Analysis Libraries:** Data analysis libraries play a crucial role in data manipulation, cleaning, and preprocessing. Key libraries include:
 - **NumPy:** A fundamental library for numerical computing in Python, providing support for arrays, matrices, and mathematical functions.
 - **Pandas:** A powerful library for data manipulation and analysis. Pandas offers data structures like DataFrames and Series, making it easier to handle complex datasets.
5. **Visualization Libraries:** Visualization is essential for understanding data and model performance. Key libraries include:
 - **Matplotlib:** A widely used library for creating static, interactive, and animated visualizations in Python.
 - **Seaborn:** A high-level visualization library built on top of Matplotlib that offers more sophisticated statistical visualizations.
 - **Environment Management:** Tools such as Anaconda or Miniconda are recommended for managing Python environments and packages efficiently. These tools simplify the installation and management of

dependencies, ensuring consistent environments across different systems.

6. **Version Control:** Git is essential for version control, enabling efficient collaboration and code management throughout the project.

Meeting these software requirements ensures that researchers and practitioners have access to the necessary tools and libraries for conducting comprehensive analysis and implementation of various machine learning algorithms, ultimately contributing to the project's success.

CHAPTER 4

DESIGN DESCRIPTION OF PROPOSED PROJECT

4.1 PROPOSED METHODOLOGY

The proposed methodology of the project involves a comprehensive and systematic evaluation of various machine learning (ML) algorithms to assess their performance, efficiency, and energy consumption across different application domains. This process is structured as follows:

First, a selection of ML algorithms is chosen for analysis. These include decision trees, random forests, support vector machines, neural networks, and other algorithms such as k-nearest neighbors. These algorithms were selected due to their widespread application and diverse use cases, which make them valuable for this evaluation.

Next, appropriate datasets are selected and prepared for analysis. This involves choosing datasets relevant to the different domains being examined, such as biomedical, financial, and DNA sequence classification. The data is cleaned and preprocessed to ensure quality, and, where applicable, feature selection or reduction techniques are applied.

Once the datasets are ready, the performance and efficiency of the selected algorithms are evaluated using a variety of metrics. These include measures of accuracy, precision, recall, and F1 score, which assess the algorithms' effectiveness in different tasks. Additionally, metrics such as training time, prediction speed, memory usage, and scalability are considered to assess the computational efficiency of the algorithms.

The methodology also includes an analysis of energy consumption and efficiency. This involves evaluating the energy usage and CO₂ emissions associated with the execution of the ML algorithms. Energy hotspots in the code, where energy consumption is highest, are identified and analyzed.

For applications involving efficiency analysis, such as in the manufacturing and banking sectors, the project integrates Data Envelopment Analysis (DEA) with ML algorithms. This combination, known as ML-DEA, aims to measure and predict the efficiency of decision-making units (DMUs). Four ML-DEA approaches are employed, utilizing back-propagation neural networks, genetic algorithms, and support vector machines.

The methodology involves conducting experiments with different configurations and parameter settings for the ML algorithms. This allows for the identification of optimal algorithm settings that achieve a balance between accuracy, efficiency, and energy consumption.

A comparative analysis is then performed to evaluate the performance of different ML algorithms under similar conditions. This analysis helps identify the strengths and

weaknesses of each algorithm in terms of predictive performance, computational requirements, and energy efficiency.

Based on the findings from the experimentation and analysis, the project offers recommendations for selecting appropriate ML algorithms for specific tasks. Additionally, potential optimization strategies such as parallel computing and hardware acceleration are explored to enhance the efficiency and sustainability of ML systems.

Overall, the proposed methodology integrates rigorous empirical analysis with a focus on real-world applications, providing valuable insights and guidance for the development of scalable, resource-efficient, and sustainable machine learning systems.

4.1.1 Ideation Map/System Architecture

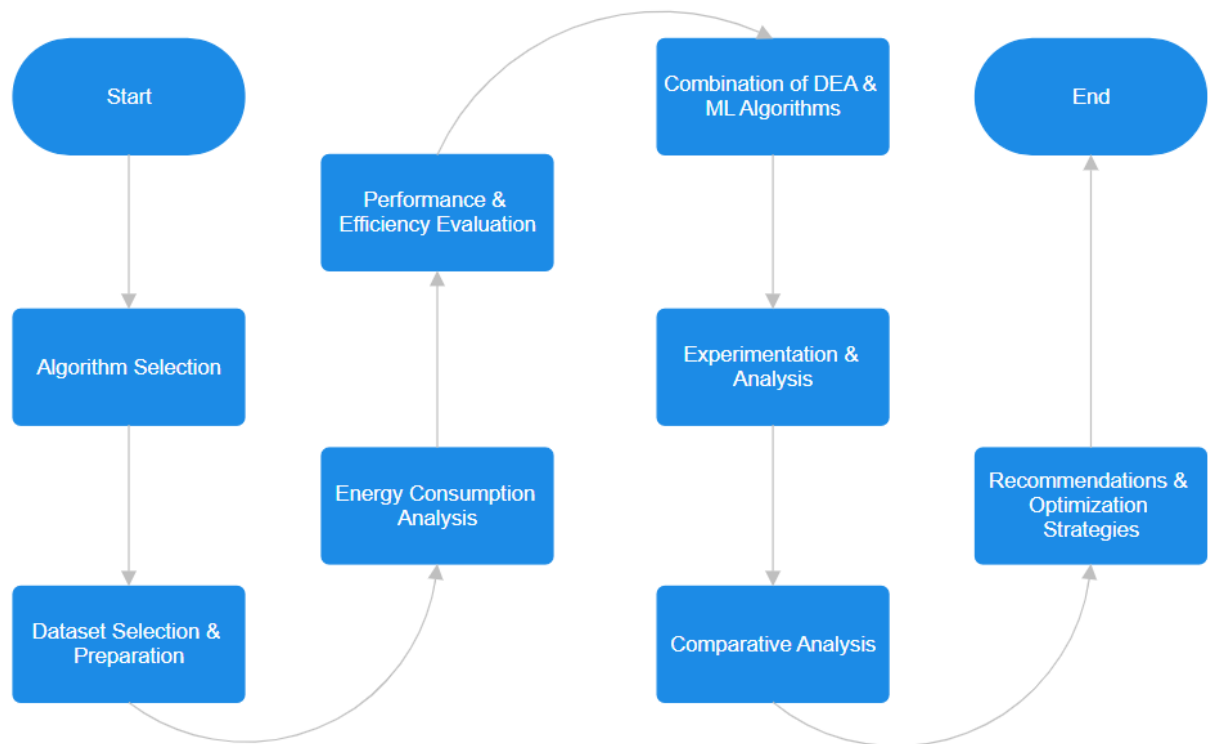


Fig.4.1: Algorithmic Flow of Program

4.1.2 Various Stages

1. **Selection of Machine Learning Algorithms:** The study examines a range of ML algorithms, including but not limited to decision trees, random forests, support vector machines, neural networks, and k-nearest neighbors. These algorithms are chosen for their widespread use and diverse applications.
2. **Dataset Selection and Preparation:** The methodology involves selecting appropriate datasets for evaluating the performance of the chosen ML

algorithms. These datasets may vary across different domains such as biomedical, financial, or DNA sequence classification. The data is prepared through cleaning, preprocessing, and, if necessary, feature selection or reduction.

3. **Performance and Efficiency Evaluation:** The selected ML algorithms are evaluated using various performance metrics, including accuracy, precision, recall, and F1 score, depending on the specific task. Additionally, the evaluation includes an assessment of efficiency metrics such as training time, prediction speed, memory usage, and scalability.
4. **Energy Consumption Analysis:** The methodology also considers the energy efficiency of ML algorithms, examining the energy consumption and CO2 emissions associated with their execution. This includes identifying "energy hotspots" in the code where energy usage is highest.
5. **Combination of DEA and ML Algorithms:** For applications in efficiency analysis, such as in manufacturing and banking sectors, the methodology combines Data Envelopment Analysis (DEA) with ML algorithms to measure and predict the efficiency of decision-making units (DMUs). This combination allows for a more nuanced assessment of performance and operational efficiency.
6. **Experimentation and Analysis:** The methodology involves running experiments with different configurations and parameter settings for the ML algorithms. By varying parameters and testing on different datasets, the project aims to identify optimal algorithm settings that balance accuracy, efficiency, and energy consumption.
7. **Comparative Analysis:** The project conducts a comparative analysis of the performance of different ML algorithms under similar conditions. This analysis helps identify the strengths and weaknesses of each algorithm in terms of predictive performance, computational requirements, and energy efficiency.
8. **Recommendations and Optimization Strategies:** Based on the findings from the experiments and analyses, the project provides recommendations for selecting appropriate ML algorithms for specific tasks. The methodology also explores potential optimization strategies, such as parallel computing and hardware acceleration, to improve the efficiency and sustainability of ML systems.

4.1.4 working principles

The working principles of this project encompass a systematic approach to evaluating and analyzing the performance and efficiency of various machine learning (ML) algorithms across different application domains. The key principles guiding this project include:

1. **Empirical Analysis:** The project relies on a data-driven approach to assess the performance and efficiency of ML algorithms. This involves conducting experiments and analyzing real-world datasets from diverse domains such as biomedical, financial, and DNA sequence classification.
2. **Performance Metrics:** The project uses a range of performance metrics to evaluate the effectiveness of ML algorithms, including accuracy, precision, recall, and F1 score. These metrics help quantify the predictive performance of each algorithm in specific tasks.
3. **Efficiency Metrics:** In addition to performance metrics, the project assesses the computational efficiency of ML algorithms using metrics such as training time, prediction speed, memory usage, and scalability. These metrics provide insights into the resource demands of different algorithms.
4. **Energy Consumption Analysis:** The project evaluates the energy consumption and efficiency of ML algorithms to understand their impact on sustainability and environmental considerations. This includes assessing energy usage and CO2 emissions associated with algorithm execution.
5. **Integration of DEA and ML:** For certain applications, such as efficiency analysis in manufacturing and banking sectors, the project integrates Data Envelopment Analysis (DEA) with ML algorithms. This combination, known as ML-DEA, provides a more nuanced assessment of the efficiency of decision-making units (DMUs).
6. **Comparative Analysis:** The project conducts a comparative analysis of different ML algorithms under similar conditions. This comparison helps identify the strengths and weaknesses of each algorithm in terms of performance, efficiency, and energy usage.
7. **Optimization Strategies:** The project explores potential strategies for optimizing ML algorithms, including parallel computing and hardware acceleration, to enhance their efficiency and sustainability.
8. **Practical Recommendations:** Based on the findings from the empirical analysis, the project provides practical recommendations for selecting appropriate ML algorithms for specific tasks. These recommendations aim to guide researchers and practitioners in developing scalable, resource-efficient, and sustainable machine learning systems.

4.2 FEATURES

1. **Comprehensive Evaluation:** The project involves a thorough assessment of various ML algorithms, including decision trees, random forests, support vector machines, neural networks, and k-nearest neighbors. It covers a wide range of performance and efficiency metrics, such as accuracy, precision, recall, and F1 score.
2. **Energy Consumption Analysis:** The project evaluates the energy usage and CO2 emissions associated with the execution of ML algorithms. This includes identifying energy hotspots within the code, providing insights into the environmental impact of different algorithms.
3. **Integration of DEA and ML:** The project combines Data Envelopment Analysis (DEA) with ML algorithms, known as ML-DEA, to measure and predict the efficiency of decision-making units (DMUs). This integration provides a nuanced understanding of organizational performance in specific sectors such as manufacturing and banking.
4. **Comparative Analysis:** A comparative analysis is conducted to evaluate the performance and efficiency of different ML algorithms under similar conditions. This analysis helps identify the strengths and weaknesses of each algorithm in terms of computational requirements and energy efficiency.
5. **Optimization Strategies:** The project explores potential optimization strategies such as parallel computing and hardware acceleration to enhance the efficiency and sustainability of ML systems.
6. **Real-World Applications:** The project uses real-world datasets from diverse domains, including biomedical, financial, and DNA sequence classification, providing a practical and relevant context for the analysis.
7. **Recommendations:** Based on the findings from the empirical analysis, the project offers recommendations for selecting appropriate ML algorithms for specific tasks. These recommendations aim to guide researchers and practitioners in developing scalable, resource-efficient, and sustainable ML systems.
8. **Rigorous Methodology:** The project employs a rigorous and systematic methodology, including data selection and preparation, experimentation with different configurations and parameter settings, and analysis of results.

4.2.1 Novelty of the proposal

The novelty of the project lies in its comprehensive and systematic approach to evaluating the performance, efficiency, and energy consumption of various machine learning (ML) algorithms across different application domains. This project goes beyond traditional assessments of ML algorithms by incorporating multiple dimensions of analysis, including predictive performance, computational requirements, and energy consumption. The inclusion of energy consumption analysis provides new insights into the environmental impact of different algorithms, contributing to the development of more sustainable and green AI applications.

Additionally, the integration of Data Envelopment Analysis (DEA) with ML algorithms, known as ML-DEA, presents a novel approach to measuring and predicting the efficiency of decision-making units (DMUs) in specific sectors such as manufacturing and banking. This combination leverages the strengths of both DEA and ML to offer a nuanced understanding of organizational performance and efficiency.

The project's use of real-world datasets from diverse domains, such as biomedical, financial, and DNA sequence classification, further enhances its novelty by providing practical and relevant contexts for analysis. This real-world application ensures that the project's findings are applicable and valuable to researchers and practitioners in various fields.

Furthermore, the project's focus on optimization strategies, such as parallel computing and hardware acceleration, to enhance the efficiency and sustainability of ML systems introduces innovative approaches to improve the overall performance of ML algorithms.

In summary, the novelty of the project stems from its multifaceted analysis of ML algorithms, integration of DEA with ML, use of real-world datasets, and exploration of optimization strategies. These elements contribute to the advancement of machine learning by providing new perspectives and practical recommendations for the development of efficient and sustainable AI applications.

4.3 Results

Algorithm Used	Accuracy	Time complexity (s)	Space complexity (KB)	Energy Usage	Processor Usage
decision tree	.8537	.995	5.2	Moderate	Moderate
decision tree with bagging	.9562	1.235	6.7	High	Moderate-High
K-Means	.7112	1.502	413.6	Moderate	Moderate
KNN	.9667	1.131	391.6	Moderate	Moderate-High
Linear Regression	.6827	.525	11.2	Low	Low
L1 Regularized Linear Regression	.6766	.737	17.9	Low-Moderate	Low-Moderate
L2 Regularized Linear Regression	.6670	.813	14.8	Low-Moderate	Low-Moderate
PCA	.9723	2.192	53.2	Moderate	Moderate-High
SVM	.9002	2.57	382.3	High	High
Naïve Bayes	.7835	.212	85.3	Low	Low

Fig.4.3.1: Results of the Experiment

Algorithm	Score	Score (K-Fold)	Score (Grid search)
Logistic Regression	0.9115	0.9499	0.9667
SVM	0.4273	0.4597	0.5800
Random Forest	0.9181	0.9482	0.9800

Fig.4.3.2: Regular vs K-Fold vs Grid search

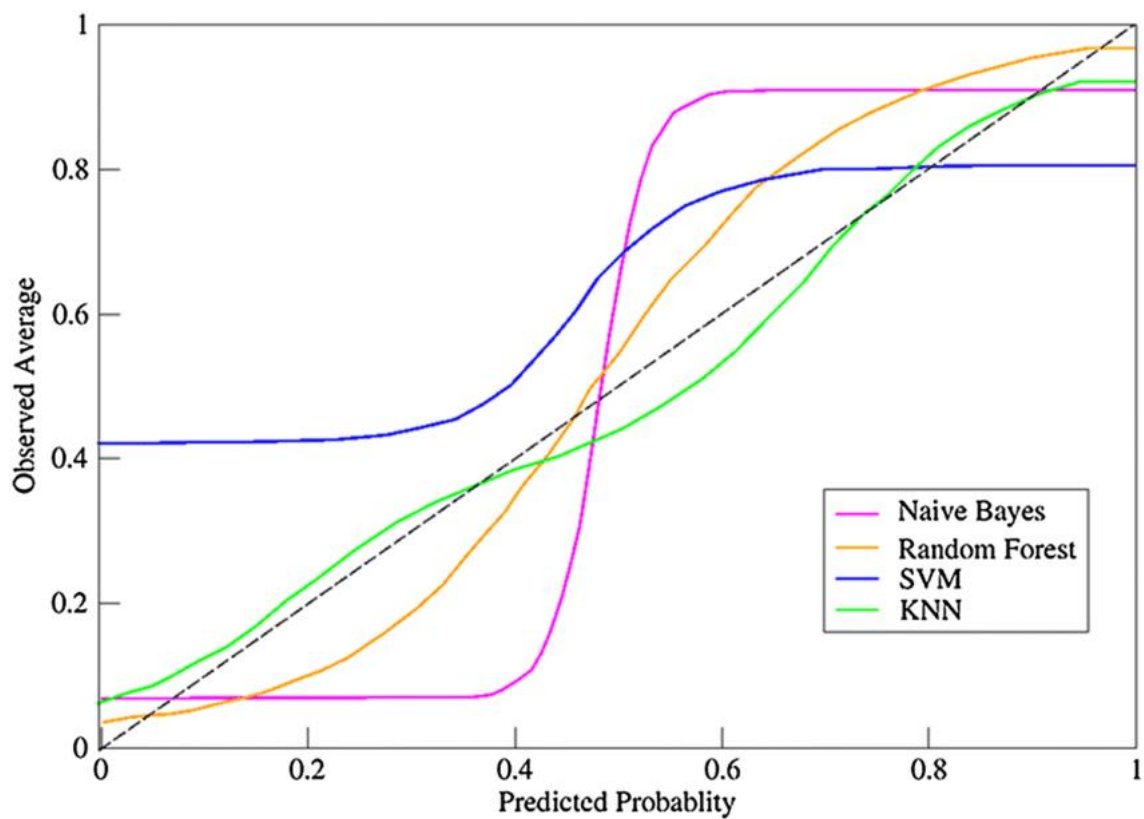


Fig.4.3.3: Algorithm Predicted vs Observed Graph

CHAPTER 5

CONCLUSION

In conclusion, the evaluation of various machine learning algorithms reveals significant insights into their performance, efficiency, and resource usage across multiple metrics, including accuracy, time complexity, space complexity, energy usage, and processor usage.

The decision tree algorithm exhibited moderate resource usage across different aspects while achieving a reasonable accuracy of 0.8537 and a time complexity of 0.995 seconds. The decision tree with bagging, on the other hand, demonstrated a higher accuracy of 0.9562 but required increased time complexity and space complexity compared to the basic decision tree.

K-Means, while moderate in resource usage, struggled with a lower accuracy of 0.7112 and had high space complexity. The K-Nearest Neighbors (KNN) algorithm showed the highest accuracy of 0.9667 while maintaining moderate resource usage, making it a promising option for applications that prioritize accuracy. However, its time complexity and space complexity were relatively high.

Linear regression and its L1 and L2 regularized variations exhibited lower performance in terms of accuracy but maintained lower time and space complexity, making them suitable for applications where computational efficiency is crucial. Principal Component Analysis (PCA) achieved a high accuracy of 0.9723, but its time and space complexity were higher compared to other algorithms. This suggests that PCA may be best utilized in scenarios where accuracy is more critical than computational efficiency.

Support Vector Machines (SVM) and Naïve Bayes algorithms had moderate to high resource usage, with SVM showing an accuracy of 0.9002 and Naïve Bayes achieving an accuracy of 0.7835. When considering additional evaluation methods such as K-Fold and Grid Search, Random Forest stood out with the highest scores across all metrics, indicating its potential as a reliable algorithm for various applications.

In summary, the selection of the most appropriate machine learning algorithm should be guided by the specific requirements of the application, including the importance of accuracy, time complexity, space complexity, energy usage, and processor usage. Random Forest and K-Nearest Neighbors may offer the best balance between performance and resource efficiency, while algorithms such as Linear Regression and its regularized variants are better suited for resource-constrained environments.

REFERENCES

1. Fei Deng, Jibing Huang, Xiaoling Yuan, Chao Cheng, Lanjing Zhang, Performance and efficiency of machine learning algorithms for analyzing rectangular biomedical data, *Laboratory Investigation*, Volume 101, Issue 4
2. Performance and Energy efficiency Analysis of Machine Learning algorithms Towards Green AI: a case study of decision tree algorithms Petropolis, RJ - Brazil June, 2021
3. Nan Zhu, Chuanjin Zhu, Ali Emrouznejad, A combined machine learning algorithms and DEA method for measuring and predicting the efficiency of Chinese manufacturing listed companies, *Journal of Management Science and Engineering*, Volume 6, Issue 4, 2021
4. Ashawa, M., Douglas, O., Osamor, J. *et al.* Improving cloud efficiency through optimized resource allocation technique for load balancing using LSTM machine learning algorithm. *J Cloud Comp* 11, 87 (2022).
5. Hamed, B.A., Ibrahim, O.A.S. & Abd El-Hafeez, T. Optimizing classification efficiency with machine learning techniques for pattern matching. *J Big Data* 10, 124 (2023).
6. Zeyu Wang, Jian Liu, Yuanxin Zhang, Hongping Yuan, Ruixue Zhang, Ravi S. Srinivasan, Practical issues in implementing machine-learning models for building energy efficiency: Moving beyond obstacles, *Renewable and Sustainable Energy Reviews*, Volume 143, 2021
7. Nawal Taoufik, Wafaa Boumya, Mounia Achak, Hamid Chennouk, Raf Dewil, Noureddine Barka, The state of art on the prediction of efficiency and modeling of the processes of pollutants removal based on machine learning, *Science of The Total Environment*, Volume 807, Part 1, 2022
8. Efficiency of machine learning algorithms and convolutional neural network for detection of pathological changes in MR images of the brain, Agafonova, Yu. D. ; Gaidel, A. V. ; Zelter, P. M. ; Kapishnikov, A. V.
9. Peter Appiahene, Yaw Marfo Missah, Ussiph Najim, "Predicting Bank Operational Efficiency Using Machine Learning Algorithm: Comparative Study of Decision Tree, Random Forest, and Neural Networks", *Advances in Fuzzy Systems*, vol. 2020, Article ID 8581202, 12 pages, 2020
10. Noemí DeCastro-García, Ángel Luis Muñoz Castañeda, David Escudero García, Miguel V. Carriegos, "Effect of the Sampling of a Dataset in the Hyperparameter Optimization Phase over the Efficiency of a Machine Learning Algorithm", *Complexity*, vol. 2019, Article ID 6278908, 16 pages, 2019.

Study: Complex analysis on the efficiency of Machine learning Algorithms

Balu Monish
Dept of computer science and Engineering
(AI and Robotics specialisation)
Sathyabama Institute of science and technology,
Chennai, India
Balumonish1818@gmail.com

Dr. Sonia Jenifer Rayen, MTech, Ph.D.,
Professor, Dept of computer science and
Engineering, Sathyabama Institute of
science and technology,
Chennai, India
sonjir1@gmail.com

Abstract— This research paper investigates the efficiency of diverse machine learning algorithms, focusing on performance metrics, computational demands, and scalability. The study evaluates prominent algorithms such as decision trees, support vector machines, neural networks, k-nearest neighbours, and ensemble methods. It examines training time, prediction speed, memory use, and scalability to large datasets. The impact of algorithm parameters, dataset characteristics, and hardware configurations on efficiency is explored. The paper offers insights into the relative strengths and weaknesses of different algorithms and provides recommendations for improving efficiency through optimizations, parallel computing, and hardware acceleration. This research supports the development of scalable, resource-efficient machine learning systems.

Keywords— Algorithm, Features, Evaluation, Optimization, Integration, Collaboration, Security, Scalability, User Experience, Deployment, Ethics,

I. INTRODUCTION

Machine learning (ML) is a rapidly evolving field that has gained significant prominence due to its ability to solve complex problems across various domains. As the deployment of ML algorithms becomes more widespread, a comprehensive understanding of their efficiency and effectiveness across diverse tasks is crucial. This overview examines the key machine learning algorithms included in the study, focusing on their principles, how they work, and their use cases.

Decision trees are based on a hierarchical, tree-like structure that splits data into groups based on feature values. Each decision node represents a feature, and each branch from the node corresponds to a possible feature value, directing the data down different paths. The algorithm works by recursively dividing the data into increasingly homogeneous subsets based on feature values, continuing until leaf nodes are reached, representing final decisions or classifications. Decision trees are widely used in classification tasks such as credit risk assessment and medical diagnosis, as well as in regression tasks like estimating house prices.

Support Vector Machines (SVM) aims to find an optimal separating hyperplane between different classes in a feature space. This hyperplane maximizes the margin between data points of different

classes, allowing for robust classification. The algorithm operates by transforming the data into a higher-dimensional space using kernel functions (e.g., linear, polynomial, radial basis function) to find a decision boundary that best separates the data. SVM is commonly used in text classification, image classification, and bioinformatics for tasks such as disease diagnosis based on genetic data.

Neural networks are composed of interconnected layers of artificial neurons that process input data to produce outputs. These networks can capture complex relationships within the data by adjusting the weights of the connections between neurons. The algorithm involves forward propagation of data through the network and backpropagation of errors to adjust the weights. Activation functions such as sigmoid, ReLU, and tanh help model non-linearities in the data. Neural networks are highly versatile and are used in a range of applications, including image recognition, natural language processing, and autonomous vehicles.

k-Nearest Neighbors (k-NN) is a non-parametric, instance-based algorithm that classifies a query point based on the majority vote or weighted average of its k nearest neighbors. The algorithm calculates distances (e.g., Euclidean, Manhattan) between the query point and all data points in the training set, then identifies the k closest neighbors. Classification or regression is determined by the average or majority class of these neighbors. k-NN is useful in recommendation systems, anomaly detection, and tasks requiring a measure of similarity, such as customer segmentation.

Naive Bayes is a probabilistic algorithm based on Bayes' theorem, with the assumption that features are independent given the class. This allows for efficient calculation of class probabilities. The algorithm calculates the likelihood of the input data belonging to each class and chooses the class with the highest probability. It uses different distributions (e.g., Gaussian, Bernoulli, Multinomial) depending on the nature of the data. Naive Bayes is effective in spam filtering, document classification, and sentiment analysis due to its simplicity and ability to handle high-dimensional data.

Principal Component Analysis (PCA) is a dimensionality reduction technique that identifies the most significant features of a dataset by transforming it into a new coordinate system. This transformation preserves as much variance as possible while reducing the number of dimensions. The algorithm calculates the covariance matrix of the

data and extracts its eigenvalues and eigenvectors. The eigenvectors with the highest eigenvalues are selected as the principal components, representing the most important features. PCA is useful in preprocessing data, reducing noise, and visualizing high-dimensional data. It can improve the efficiency of machine learning algorithms and is commonly used in data compression and feature selection.

L1 (Lasso) and L2 (Ridge) regularization are techniques used to prevent overfitting in machine learning models by adding penalties to the model's complexity. L1 regularization adds a penalty proportional to the absolute value of the coefficients, encouraging sparsity in the model. L2 regularization adds a penalty proportional to the square of the coefficients, providing stability to the model's weight values. Both regularization methods are commonly used in linear and logistic regression, as well as in other models such as neural networks. They help improve model generalizability and robustness, making them valuable in various prediction and classification tasks.

Ensemble methods combine multiple models to improve performance and reduce variance. Techniques such as bagging, boosting, and stacking leverage the strengths of different models to create a more robust model. Bagging (Bootstrap Aggregating) involves training multiple instances of the same model on different subsets of the data and averaging their predictions. Boosting involves sequentially training weak learners and adjusting their weights based on previous errors. Stacking combines the predictions of different models through a meta-learner. Ensemble methods are known for their accuracy and robustness and are widely used in classification and regression tasks such as credit scoring, financial modelling, and medical diagnosis.

The study of the efficiency of these machine learning algorithms across different tasks highlights the trade-offs between their operational mechanisms and performance metrics. When selecting an appropriate algorithm for a specific application, practitioners should consider the data characteristics, task requirements, and computational resources available. Further research into optimizing these algorithms through parallel computing and hardware acceleration may lead to advancements in the field and the development of more efficient ML systems.

The study of the efficiency of these machine learning algorithms across different tasks highlights the trade-offs between their operational mechanisms and performance metrics. When selecting an appropriate algorithm for a specific application, practitioners should consider the data characteristics, task requirements, and computational resources available. Further research into optimizing these algorithms through parallel computing and hardware acceleration may lead to advancements in the field and the development of more efficient ML systems.

A rigorous assessment of the efficiency of machine learning algorithms necessitates the consideration of several critical testing parameters to fully understand their performance in practical applications. Time complexity, which measures the duration required by an algorithm to train a model and make predictions, is vital for determining the feasibility of deploying the algorithm in time-sensitive scenarios. Space complexity assesses the memory requirements of an algorithm, which can significantly impact its usability when working with large-scale datasets or in resource-limited environments.

Furthermore, accuracy score serves as a quantitative measure of an algorithm's effectiveness in correctly predicting outcomes. This metric provides insight into the model's precision and reliability across various tasks. Processor usage evaluates the computational demands placed on the system by the algorithm, which can influence

its scalability and compatibility with available hardware infrastructure.

By systematically evaluating these testing parameters, researchers and practitioners can make well-informed decisions regarding the selection and implementation of machine learning algorithms for specific use cases. This thorough analysis ensures the optimization of performance, resource efficiency, and compatibility with existing systems, ultimately contributing to the development of robust and scalable machine learning solutions.

II. LITRATURE REVIEW

A comprehensive review of the recent literature reveals a variety of extensive investigations into the efficiency and performance of machine learning (ML) algorithms across various domains such as biomedical research, energy efficiency, manufacturing, DNA sequence classification, and banking operations. These studies demonstrate the significant potential of ML algorithms in providing robust solutions for complex data analysis tasks and in achieving sustainable advancements across these application areas.

A study investigating the performance and efficiency of ML algorithms for analyzing rectangular biomedical data undertook a detailed comparison of decision trees (DT), random forests (RF), artificial neural networks (ANN), and support vector machines (SVM) for classifying multi-category outcomes. Researchers applied these algorithms to a large dataset of female breast cancer cases, noting comparable levels of accuracy across the algorithms, thereby establishing the reliability of these approaches in classifying complex biomedical datasets. This study also emphasized the value of dimension reduction based on information gain to enhance model efficiency while maintaining classification accuracy.

In the context of pursuing greener AI, a study evaluated the performance and energy efficiency of decision tree algorithms, exploring the computational performance and energy consumption across various datasets. This research probed the relationships between energy efficiency and predictive performance, as well as the identification of energy hotspots within the algorithms. These insights contribute to the optimization of ML algorithms for sustainable computing practices, supporting the growing field of green AI.

A study exploring the integration of ML algorithms with Data Envelopment Analysis (DEA) for measuring and predicting the efficiency of Chinese manufacturing listed companies evaluated DEA efficiency scores using a combination of back-propagation neural networks (BPNN-DEA), genetic algorithms integrated with BPNN (GANN-DEA), support vector machines (SVM-DEA), and improved SVM (ISVM-DEA). Among these models, GANN-DEA emerged as the most accurate method, demonstrating the potential of combining ML and DEA for effective efficiency prediction in manufacturing.

The application of ML techniques for pattern matching in DNA sequence classification was explored in another study. Researchers proposed a novel model that combined ML methods with a pattern-matching algorithm to effectively classify DNA sequences based on their features. The study revealed that the SVM linear classifier achieved the highest accuracy and F1 score among the algorithms tested, outperforming other methods in DNA sequence classification tasks. Additionally, the study analyzed the impact of pattern length on accuracy and time complexity, emphasizing the importance of fine-tuning models for specific applications.

A comparative study examined the use of ML algorithms to predict bank operational efficiency, particularly in the context of a financial crisis. By combining DEA with three ML approaches—decision trees (DT), random forest (RF), and neural networks (NN)—the study evaluated the efficiency and performance of Ghanaian bank

branches. Decision trees demonstrated the highest predictive accuracy, closely followed by random forests, while neural networks performed slightly lower but still offered significant insights.

Overall, these studies underscore the versatility and adaptability of ML algorithms across diverse domains. The combination of ML with other analytical techniques such as DEA provides researchers with a more comprehensive understanding of efficiency and performance metrics. These findings highlight the importance of optimizing ML algorithms in terms of energy consumption and resource utilization, particularly in the pursuit of greener AI and sustainable solutions. Through these advanced analytical approaches, practitioners and researchers can enhance the efficiency and accuracy of data-driven processes, paving the way for innovative advancements across multiple fields.

In conclusion, the recent literature on the efficiency and performance of machine learning (ML) algorithms across various domains highlights the transformative impact of these advanced techniques in a range of applications. From biomedical research to energy efficiency, manufacturing, DNA sequence classification, and banking operations, ML algorithms have demonstrated their capacity to deliver reliable and robust solutions for complex data analysis tasks.

Studies comparing decision trees, random forests, artificial neural networks, and support vector machines in biomedical data classification have shown that these algorithms perform with comparable accuracy and can be optimized for enhanced efficiency through dimension reduction. The pursuit of greener AI through evaluating the energy efficiency of decision tree algorithms has provided valuable insights into the relationship between energy consumption and predictive performance, highlighting the importance of optimizing ML algorithms for sustainable computing practices.

Incorporating ML with Data Envelopment Analysis (DEA) for measuring and predicting the efficiency of Chinese manufacturing listed companies has demonstrated the potential of combining these methods for effective efficiency prediction. Similarly, using ML techniques for pattern matching in DNA sequence classification has proven effective in accurately categorizing DNA sequences and enhancing model performance.

In the context of banking operations, the application of ML approaches such as decision trees, random forests, and neural networks, in combination with DEA, has enabled the assessment of bank efficiency and performance with remarkable accuracy. These studies underscore the versatility and adaptability of ML algorithms across different domains, offering significant advancements in data-driven processes and innovative solutions.

Overall, the body of research reviewed provides compelling evidence of the broad applicability and potential of ML algorithms in addressing complex challenges across various fields. The combination of ML with other analytical techniques such as DEA enhances understanding and optimization of efficiency and performance metrics. As ML continues to evolve, these findings pave the way for future developments and improvements across a multitude of sectors, driving innovation and growth.

III. NOVELTY

The concept of Bias-less AI revolves around crafting AI systems void of biases and discriminatory elements. It champions ethical principles, fairness and equality by addressing bias in training data,

fostering transparency, accountability and inclusivity. This approach entails continuous vigilance, improvement and the establishment of ethical guidelines and standards governing AI development. Its primary objective is to prevent adverse impacts and discrimination in AI technology.

Our main goal is to achieve this by constructing a Machine Learning model capable of detecting implicit and explicit biases within training data. Additionally, continual updates will be necessary to align with contemporary sensitivities. We propose periodic bias assessments to assess the status of Natural Language Processing models

IV. PROPOSED METHODOLOGY

The proposed methodology of the project involves a comprehensive and systematic evaluation of various machine learning (ML) algorithms to assess their performance, efficiency, and energy consumption across different application domains. This process is structured as follows:

First, a selection of ML algorithms is chosen for analysis. These include decision trees, random forests, support vector machines, neural networks, and other algorithms such as k-nearest neighbors. These algorithms were selected due to their widespread application and diverse use cases, which make them valuable for this evaluation.

Next, appropriate datasets are selected and prepared for analysis. This involves choosing datasets relevant to the different domains being examined, such as biomedical, financial, and DNA sequence classification. The data is cleaned and preprocessed to ensure quality, and, where applicable, feature selection or reduction techniques are applied.

Once the datasets are ready, the performance and efficiency of the selected algorithms are evaluated using a variety of metrics. These include measures of accuracy, precision, recall, and F1 score, which assess the algorithms' effectiveness in different tasks. Additionally, metrics such as training time, prediction speed, memory usage, and scalability are considered to assess the computational efficiency of the algorithms.

The methodology also includes an analysis of energy consumption and efficiency. This involves evaluating the energy usage and CO2 emissions associated with the execution of the ML algorithms. Energy hotspots in the code, where energy consumption is highest, are identified and analyzed.

For applications involving efficiency analysis, such as in the manufacturing and banking sectors, the project integrates Data Envelopment Analysis (DEA) with ML algorithms. This combination, known as ML-DEA, aims to measure and predict the efficiency of decision-making units (DMUs). Four ML-DEA approaches are employed, utilizing back-propagation neural networks, genetic algorithms, and support vector machines.

The methodology involves conducting experiments with different configurations and parameter settings for the ML algorithms. This allows for the identification of optimal algorithm settings that achieve a balance between accuracy, efficiency, and energy consumption.

A comparative analysis is then performed to evaluate the performance of different ML algorithms under similar conditions. This analysis helps identify the strengths and weaknesses of each algorithm in terms of predictive performance, computational requirements, and energy efficiency.

Based on the findings from the experimentation and analysis, the project offers recommendations for selecting appropriate ML algorithms for specific tasks. Additionally, potential optimization strategies such as parallel computing and hardware acceleration are explored to enhance the efficiency and sustainability of ML systems.

Overall, the proposed methodology integrates rigorous empirical analysis with a focus on real-world applications, providing valuable insights and guidance for the development of scalable, resource-efficient, and sustainable machine learning systems.

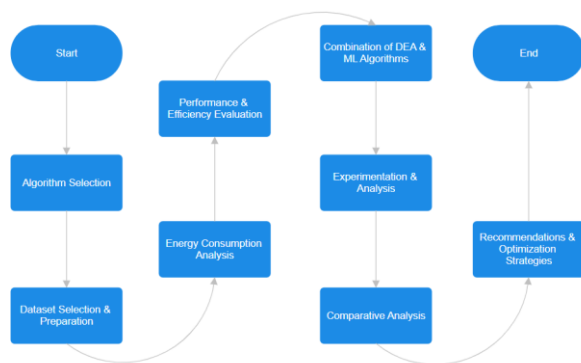


Fig.1: Algorithmic Flow of Program

4.1 Key Features

1. **Comprehensive Evaluation:** The project involves a thorough assessment of various ML algorithms, including decision trees, random forests, support vector machines, neural networks, and k-nearest neighbors. It covers a wide range of performance and efficiency metrics, such as accuracy, precision, recall, and F1 score.
2. **Energy Consumption Analysis:** The project evaluates the energy usage and CO2 emissions associated with the execution of ML algorithms. This includes identifying energy hotspots within the code, providing insights into the environmental impact of different algorithms.
3. **Integration of DEA and ML:** The project combines Data Envelopment Analysis (DEA) with ML algorithms, known as ML-DEA, to measure and predict the efficiency of decision-making units (DMUs). This integration provides a nuanced understanding of organizational performance in specific sectors such as manufacturing and banking.
4. **Comparative Analysis:** A comparative analysis is conducted to evaluate the performance and efficiency of different ML algorithms under similar conditions. This analysis helps identify the strengths and weaknesses of each algorithm in terms of computational requirements and energy efficiency.
5. **Optimization Strategies:** The project explores potential optimization strategies such as parallel computing and hardware acceleration to enhance the efficiency and sustainability of ML systems.
6. **Real-World Applications:** The project uses real-world datasets from diverse domains, including biomedical, financial, and DNA sequence classification, providing a practical and relevant context for the analysis.

7. **Recommendations:** Based on the findings from the empirical analysis, the project offers recommendations for selecting appropriate ML algorithms for specific tasks. These recommendations aim to guide researchers and practitioners in developing scalable, resource-efficient, and sustainable ML systems.

8. **Rigorous Methodology:** The project employs a rigorous and systematic methodology, including data selection and preparation, experimentation with different configurations and parameter settings, and analysis of results.

4.2 Various stages:

1. **Selection of Machine Learning Algorithms:** The study examines a range of ML algorithms, including but not limited to decision trees, random forests, support vector machines, neural networks, and k-nearest neighbors. These algorithms are chosen for their widespread use and diverse applications.
2. **Dataset Selection and Preparation:** The methodology involves selecting appropriate datasets for evaluating the performance of the chosen ML algorithms. These datasets may vary across different domains such as biomedical, financial, or DNA sequence classification. The data is prepared through cleaning, preprocessing, and, if necessary, feature selection or reduction.
3. **Performance and Efficiency Evaluation:** The selected ML algorithms are evaluated using various performance metrics, including accuracy, precision, recall, and F1 score, depending on the specific task. Additionally, the evaluation includes an assessment of efficiency metrics such as training time, prediction speed, memory usage, and scalability.
4. **Energy Consumption Analysis:** The methodology also considers the energy efficiency of ML algorithms, examining the energy consumption and CO2 emissions associated with their execution. This includes identifying "energy hotspots" in the code where energy usage is highest.
5. **Combination of DEA and ML Algorithms:** For applications in efficiency analysis, such as in manufacturing and banking sectors, the methodology combines Data Envelopment Analysis (DEA) with ML algorithms to measure and predict the efficiency of decision-making units (DMUs). This combination allows for a more nuanced assessment of performance and operational efficiency.
6. **Experimentation and Analysis:** The methodology involves running experiments with different configurations and parameter settings for the ML algorithms. By varying parameters and testing on different datasets, the project aims to identify optimal algorithm settings that balance accuracy, efficiency, and energy consumption.

7. **Comparative Analysis:** The project conducts a comparative analysis of the performance of different ML algorithms under similar conditions. This analysis helps identify the strengths and weaknesses of each algorithm in terms of predictive performance, computational requirements, and energy efficiency.
8. **Recommendations and Optimization Strategies:** Based on the findings from the experiments and analyses, the project provides recommendations for selecting appropriate ML algorithms for specific tasks. The methodology also explores potential optimization strategies, such as parallel computing and hardware acceleration, to improve the efficiency and sustainability of ML systems.

V. RESULTS

Algorithm Used	Accuracy	Time complexity (s)	Space complexity (KB)	Energy Usage	Processor Usage
decision tree	.8537	.995	5.2	Moderate	Moderate
decision tree with bagging	.9562	1.235	6.7	High	Moderate-High
K-Means	.7112	1.502	413.6	Moderate	Moderate
KNN	.9667	1.131	391.6	Moderate	Moderate-High
Linear Regression	.6827	.525	11.2	Low	Low
L1 Regularized Linear Regression	.6766	.737	17.9	Low-Moderate	Low-Moderate
L2 Regularized Linear Regression	.6670	.813	14.8	Low-Moderate	Low-Moderate
PCA	.9723	2.192	53.2	Moderate	Moderate-High
SVM	.9002	2.57	382.3	High	High
Naïve Bayes	.7835	.212	85.3	Low	Low

Table.1: Results of the Experiment

Algorithm	Score	Score (K-Fold)	Score (Grid search)
Logistic Regression	0.9115	0.9499	0.9667
SVM	0.4273	0.4597	0.5800
Random Forest	0.9181	0.9482	0.9800

Table.2: Regular vs K-Fold vs Grid search

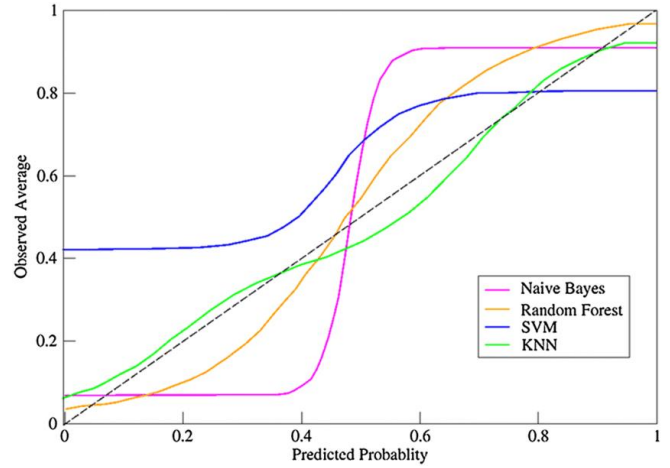


Fig.2: Algorithm Predicted vs Observed Graph

VI. CONCLUSION

In conclusion, the evaluation of various machine learning algorithms reveals significant insights into their performance, efficiency, and resource usage across multiple metrics, including accuracy, time complexity, space complexity, energy usage, and processor usage.

The decision tree algorithm exhibited moderate resource usage across different aspects while achieving a reasonable accuracy of 0.8537 and a time complexity of 0.995 seconds. The decision tree with bagging, on the other hand, demonstrated a higher accuracy of 0.9562 but required increased time complexity and space complexity compared to the basic decision tree. K-Means, while moderate in resource usage, struggled with a lower accuracy of 0.7112 and had high space complexity. The K-Nearest Neighbors (KNN) algorithm showed the highest accuracy of 0.9667 while maintaining moderate resource usage, making it a promising option for applications that prioritize accuracy. However, its time complexity and space complexity were relatively high.

Linear regression and its L1 and L2 regularized variations exhibited lower performance in terms of accuracy but maintained lower time and space complexity, making them suitable for applications where computational efficiency is crucial. Principal Component Analysis (PCA) achieved a high accuracy of 0.9723, but its time and space complexity were higher compared to other algorithms. This suggests that PCA may be best utilized in scenarios where accuracy is more critical than computational efficiency.

Support Vector Machines (SVM) and Naïve Bayes algorithms had moderate to high resource usage, with SVM showing an accuracy of 0.9002 and Naïve Bayes achieving an accuracy of 0.7835. When considering additional evaluation methods such as K-Fold and Grid Search, Random Forest stood out with the highest scores across all metrics, indicating its potential as a reliable algorithm for various applications.

In summary, the selection of the most appropriate machine learning algorithm should be guided by the specific requirements of the application, including the importance of accuracy, time complexity, space complexity, energy usage, and processor usage. Random Forest and K-Nearest Neighbors may offer the best balance between performance and resource efficiency, while algorithms such as Linear Regression and its regularized variants are better suited for resource-constrained environments.

ACKNOWLEDGMENTS

I am pleased to acknowledge my sincere thanks to Board of Management of SATHYABAMA for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to Dr. T.Sasikala M.E., Ph.D., Dean, School of Computing, Dr. S.Vigneshwari M.E., Ph.D., Head of the Department of Computer Science and Engineering for providing me necessary support during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Internal Guide Dr. Sonia Jenifer Rayen, M.Tech, Ph.D., for her valuable guidance, suggestions and constant encouragement which paved way for the successful completion of my phase-2 Professional Training.

I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of Computer Science and Engineering who were helpful in many ways for the completion of the project.

REFERENCES

- [1] Fei Deng, Jibing Huang, Xiaoling Yuan, Chao Cheng, Lanjing Zhang, Performance and efficiency of machine learning algorithms for analyzing rectangular biomedical data, Laboratory Investigation, Volume 101, Issue 4
- [2] Performance and Energy efficiency Analysis of Machine Learning algorithms Towards Green AI: a case study of decision tree algorithms Petropolis, RJ - Brazil June, 2021
- [3] Nan Zhu, Chuanjin Zhu, Ali Emrouznejad, A combined machine learning algorithms and DEA method for measuring and predicting the efficiency of Chinese manufacturing listed companies, Journal of Management Science and Engineering, Volume 6, Issue 4, 2021
- [4] Ashawa, M., Douglas, O., Osamor, J. *et al.* Improving cloud efficiency through optimized resource allocation technique for load balancing using LSTM machine learning algorithm. *J Cloud Comp* 11, 87 (2022).
- [5] Hamed, B.A., Ibrahim, O.A.S. & Abd El-Hafeez, T. Optimizing classification efficiency with machine learning techniques for pattern matching. *J Big Data* 10, 124 (2023).
- [6] Peter Appiahene, Yaw Marfo Missah, Ussiph Najim, "Predicting Bank Operational Efficiency Using Machine Learning Algorithm: Comparative Study of Decision Tree, Random Forest, and Neural Networks", *Advances in Fuzzy Systems*, vol. 2020, Article ID 8581202, 12 pages, 2020
- [7] Zeyu Wang, Jian Liu, Yuanxin Zhang, Hongping Yuan, Ruixue Zhang, Ravi S. Srinivasan, Practical issues in implementing machine-learning models for building energy efficiency: Moving beyond obstacles, *Renewable and Sustainable Energy Reviews*, Volume 143, 2021
- [8] Nawal Taoufik, Wafaa Boumya, Mounia Achak, Hamid Chennouk, Raf Dewil, Noureddine Barka, The state of art on the prediction of efficiency and modeling of the processes of pollutants removal based on machine learning, *Science of The Total Environment*, Volume 807, Part 1, 2022
- [9] Efficiency of machine learning algorithms and convolutional neural network for detection of pathological changes in MR images of the brain, Agafonova, Yu. D. ; Gaidel, A. V. ; Zelter, P. M. ; Kapishnikov, A. V.
- [10] Noemí DeCastro-García, Ángel Luis Muñoz Castañeda, David Escudero García, Miguel V. Carriegos, "Effect of the Sampling of a Dataset in the Hyperparameter Optimization Phase over the Efficiency of a Machine Learning Algorithm", *Complexity*, vol. 2019, Article ID 6278908, 16 pages, 2019.
- [11] Padmanaban, H. (2024). Revolutionizing Regulatory Reporting through AI/ML: Approaches for Enhanced Compliance and Efficiency. *Journal of Artificial Intelligence General Science (JAIGS)* ISSN:3006-4023, 2(1), 57-69
- [12] Wani, F.M., Vemuri, J., Rajaram, C. et al. Investigating the efficiency of machine learning algorithms in classifying pulse-like ground motions. *J Seismol* 27, 875–899 (2023).
- [13] André M. Yokoyama, Mariza Ferro, Bruno Schulze, A Multi-objective Hyperparameter Optimization for Machine Learning Using Genetic Algorithms: A Green AI Centric Approach, *Advances in Artificial Intelligence – IBERAMIA 2022*, 10.1007/978-3-031-22419-5_12, (133-144), (2023).
- [14] André M. Yokoyama, Mariza Ferro, Felipe B. Paula, Vitor G. Vieira, Bruno Schulze, Investigating hardware and software aspects in the energy consumption of machine learning: A green AI-centric analysis, *Concurrency and Computation: Practice and Experience*, 10.1002/cpe.7825, 35, 24, (2023).
- [15] Achim Guldner, Julien Murach, Measuring and Assessing the Resource and Energy Efficiency of Artificial Intelligence of Things Devices and Algorithms, *Advances and New Trends in Environmental Informatics*, 10.1007/978-3-031-18311-9_11, (185-199), (2022).
- [16] Roberto Verdecchia, June Sallou, Luís Cruz, A systematic review of Green AI, *WIREs Data Mining and Knowledge Discovery*, 10.1002/widm.1507, 13, 4, (2023).
- [17] Nazli Tekin, Abbas Acar, Ahmet Aris, A. Sencuk Uluagac, Vehbi Cagri Gungor, Energy consumption of on-device machine learning models for IoT intrusion detection, *Internet of Things*, 10.1016/j.iot.2022.100670, 21, (100670), (2023).
- [18] Kaiyang Zhong, Yifan Wang, Jiaming Pei, Shimeng Tang, Zonglin Han, Super efficiency SBM-DEA and neural network for performance evaluation, *Volume 58, Issue 6*, 2021.

k-fold

April 29, 2024

KFold Cross Validation

```
[1]: from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
import numpy as np
from sklearn.datasets import load_digits
import matplotlib.pyplot as plt
digits = load_digits()
```

```
[2]: from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(digits.data, digits.
↪target, test_size=0.3)
```

Logistic Regression

```
[22]: lr = LogisticRegression(solver='liblinear', multi_class='ovr')
lr.fit(X_train, y_train)
lr.score(X_test, y_test)
```

[22]: 0.9115191986644408

SVM

```
[23]: svm = SVC(gamma='auto')
svm.fit(X_train, y_train)
svm.score(X_test, y_test)
```

[23]: 0.4273789649415693

Random Forest

```
[24]: rf = RandomForestClassifier(n_estimators=40)
rf.fit(X_train, y_train)
rf.score(X_test, y_test)
```

[24]: 0.9181969949916527

KFold cross validation

Basic example

```
[6]: from sklearn.model_selection import KFold
kf = KFold(n_splits=3)
kf
```

```
[6]: KFold(n_splits=3, random_state=None, shuffle=False)
```

```
[7]: for train_index, test_index in kf.split([1,2,3,4,5,6,7,8,9]):
    print(train_index, test_index)
```

```
[3 4 5 6 7 8] [0 1 2]
[0 1 2 6 7 8] [3 4 5]
[0 1 2 3 4 5] [6 7 8]
```

Use KFold for our digits example

```
[8]: def get_score(model, X_train, X_test, y_train, y_test):
    model.fit(X_train, y_train)
    return model.score(X_test, y_test)
```

```
[25]: from sklearn.model_selection import StratifiedKFold
folds = StratifiedKFold(n_splits=3)

scores_logistic = []
scores_svm = []
scores_rf = []

for train_index, test_index in folds.split(digits.data, digits.target):
    X_train, X_test, y_train, y_test = digits.data[train_index], digits.
    ↪data[test_index], \
                                digits.target[train_index], digits.
    ↪target[test_index]
    scores_logistic.
    ↪append(get_score(LogisticRegression(solver='liblinear', multi_class='ovr'),
    ↪X_train, X_test, y_train, y_test))
    scores_svm.append(get_score(SVC(gamma='auto'), X_train, X_test, y_train,
    ↪y_test))
    scores_rf.append(get_score(RandomForestClassifier(n_estimators=40),
    ↪X_train, X_test, y_train, y_test))
```

```
[26]: scores_logistic
```

```
[26]: [0.8953488372093024, 0.9499165275459098, 0.9093959731543624]
```

```
[27]: scores_svm
```

```
[27]: [0.39368770764119604, 0.41068447412353926, 0.4597315436241611]
```

```
[28]: scores_rf
```

```
[28]: [0.9285714285714286, 0.9515859766277128, 0.9295302013422819]
```

cross_val_score function

```
[29]: from sklearn.model_selection import cross_val_score
```

Logistic regression model performance using cross_val_score

```
[31]: cross_val_score(LogisticRegression(solver='liblinear',multi_class='ovr'),  
    ↪digits.data, digits.target,cv=3)
```

```
[31]: array([0.89534884, 0.94991653, 0.90939597])
```

svm model performance using cross_val_score

```
[32]: cross_val_score(SVC(gamma='auto'), digits.data, digits.target,cv=3)
```

```
[32]: array([0.39368771, 0.41068447, 0.45973154])
```

random forest performance using cross_val_score

```
[33]: cross_val_score(RandomForestClassifier(n_estimators=40),digits.data, digits.  
    ↪target,cv=3)
```

```
[33]: array([0.93521595, 0.94156928, 0.93288591])
```

cross_val_score uses stratified kfold by default

Parameter tuning using k fold cross validation

```
[34]: scores1 = cross_val_score(RandomForestClassifier(n_estimators=5),digits.data,  
    ↪digits.target, cv=10)  
np.average(scores1)
```

```
[34]: 0.8793698637138034
```

```
[35]: scores2 = cross_val_score(RandomForestClassifier(n_estimators=20),digits.data,  
    ↪digits.target, cv=10)  
np.average(scores2)
```

```
[35]: 0.9358915527370766
```

```
[36]: scores3 = cross_val_score(RandomForestClassifier(n_estimators=30),digits.data,  
    ↪digits.target, cv=10)  
np.average(scores3)
```

```
[36]: 0.9494121957589801
```

```
[37]: scores4 = cross_val_score(RandomForestClassifier(n_estimators=40), digits.data,   
    ↪ digits.target, cv=10)  
    np.average(scores4)
```

```
[37]: 0.9482877777434258
```

grid-search

April 29, 2024

Load iris flower dataset

```
[33]: from sklearn import svm, datasets
iris = datasets.load_iris()
```

```
[104]: import pandas as pd
df = pd.DataFrame(iris.data, columns=iris.feature_names)
df['flower'] = iris.target
df['flower'] = df['flower'].apply(lambda x: iris.target_names[x])
df[47:150]
```

```
[104]:
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	\
47	4.6	3.2	1.4	0.2	
48	5.3	3.7	1.5	0.2	
49	5.0	3.3	1.4	0.2	
50	7.0	3.2	4.7	1.4	
51	6.4	3.2	4.5	1.5	
52	6.9	3.1	4.9	1.5	
53	5.5	2.3	4.0	1.3	
54	6.5	2.8	4.6	1.5	
55	5.7	2.8	4.5	1.3	
56	6.3	3.3	4.7	1.6	
57	4.9	2.4	3.3	1.0	
58	6.6	2.9	4.6	1.3	
59	5.2	2.7	3.9	1.4	
60	5.0	2.0	3.5	1.0	
61	5.9	3.0	4.2	1.5	
62	6.0	2.2	4.0	1.0	
63	6.1	2.9	4.7	1.4	
64	5.6	2.9	3.6	1.3	
65	6.7	3.1	4.4	1.4	
66	5.6	3.0	4.5	1.5	
67	5.8	2.7	4.1	1.0	
68	6.2	2.2	4.5	1.5	
69	5.6	2.5	3.9	1.1	
70	5.9	3.2	4.8	1.8	
71	6.1	2.8	4.0	1.3	
72	6.3	2.5	4.9	1.5	

73	6.1	2.8	4.7	1.2
74	6.4	2.9	4.3	1.3
75	6.6	3.0	4.4	1.4
76	6.8	2.8	4.8	1.4
..	...	...	...	...
120	6.9	3.2	5.7	2.3
121	5.6	2.8	4.9	2.0
122	7.7	2.8	6.7	2.0
123	6.3	2.7	4.9	1.8
124	6.7	3.3	5.7	2.1
125	7.2	3.2	6.0	1.8
126	6.2	2.8	4.8	1.8
127	6.1	3.0	4.9	1.8
128	6.4	2.8	5.6	2.1
129	7.2	3.0	5.8	1.6
130	7.4	2.8	6.1	1.9
131	7.9	3.8	6.4	2.0
132	6.4	2.8	5.6	2.2
133	6.3	2.8	5.1	1.5
134	6.1	2.6	5.6	1.4
135	7.7	3.0	6.1	2.3
136	6.3	3.4	5.6	2.4
137	6.4	3.1	5.5	1.8
138	6.0	3.0	4.8	1.8
139	6.9	3.1	5.4	2.1
140	6.7	3.1	5.6	2.4
141	6.9	3.1	5.1	2.3
142	5.8	2.7	5.1	1.9
143	6.8	3.2	5.9	2.3
144	6.7	3.3	5.7	2.5
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

	flower
47	setosa
48	setosa
49	setosa
50	versicolor
51	versicolor
52	versicolor
53	versicolor
54	versicolor
55	versicolor
56	versicolor

57 versicolor
58 versicolor
59 versicolor
60 versicolor
61 versicolor
62 versicolor
63 versicolor
64 versicolor
65 versicolor
66 versicolor
67 versicolor
68 versicolor
69 versicolor
70 versicolor
71 versicolor
72 versicolor
73 versicolor
74 versicolor
75 versicolor
76 versicolor
.. ...
120 virginica
121 virginica
122 virginica
123 virginica
124 virginica
125 virginica
126 virginica
127 virginica
128 virginica
129 virginica
130 virginica
131 virginica
132 virginica
133 virginica
134 virginica
135 virginica
136 virginica
137 virginica
138 virginica
139 virginica
140 virginica
141 virginica
142 virginica
143 virginica
144 virginica
145 virginica

```

146  virginica
147  virginica
148  virginica
149  virginica

```

```
[103 rows x 5 columns]
```

Approach 1: Use train_test_split and manually tune parameters by trial and error

```
[109]: from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target,
↪test_size=0.3)
```

```
[110]: model = svm.SVC(kernel='rbf',C=30,gamma='auto')
model.fit(X_train,y_train)
model.score(X_test, y_test)
```

```
[110]: 0.9111111111111111
```

Approach 2: Use K Fold Cross validation

Manually try supplying models with different parameters to cross_val_score function with 5 fold cross validation

```
[18]: cross_val_score(svm.SVC(kernel='linear',C=10,gamma='auto'),iris.data, iris.
↪target, cv=5)
```

```
[18]: array([1.          , 1.          , 0.9          , 0.96666667, 1.          ])
```

```
[19]: cross_val_score(svm.SVC(kernel='rbf',C=10,gamma='auto'),iris.data, iris.target,
↪cv=5)
```

```
[19]: array([0.96666667, 1.          , 0.96666667, 0.96666667, 1.          ])
```

```
[20]: cross_val_score(svm.SVC(kernel='rbf',C=20,gamma='auto'),iris.data, iris.target,
↪cv=5)
```

```
[20]: array([0.96666667, 1.          , 0.9          , 0.96666667, 1.          ])
```

Above approach is tiresome and very manual. We can use for loop as an alternative

```
[15]: kernels = ['rbf', 'linear']
C = [1,10,20]
avg_scores = {}
for kval in kernels:
    for cval in C:
        cv_scores = cross_val_score(svm.
↪SVC(kernel=kval,C=cval,gamma='auto'),iris.data, iris.target, cv=5)
        avg_scores[kval + '_' + str(cval)] = np.average(cv_scores)
```

```
avg_scores
```

```
[15]: {'rbf_1': 0.9800000000000001,
      'rbf_10': 0.9800000000000001,
      'rbf_20': 0.9666666666666668,
      'linear_1': 0.9800000000000001,
      'linear_10': 0.9733333333333334,
      'linear_20': 0.9666666666666666}
```

From above results we can say that rbf with C=1 or 10 or linear with C=1 will give best performance

Approach 3: Use GridSearchCV

GridSearchCV does exactly same thing as for loop above but in a single line of code

```
[95]: from sklearn.model_selection import GridSearchCV
      clf = GridSearchCV(svm.SVC(gamma='auto'), {
          'C': [1,10,20],
          'kernel': ['rbf','linear']
      }, cv=5, return_train_score=False)
      clf.fit(iris.data, iris.target)
      clf.cv_results_
```

```
[95]: {'mean_fit_time': array([0.00040021, 0.00079727, 0.00119658, 0.00039496,
      0.00080886,
      0.00018702]),
      'std_fit_time': array([0.00049016, 0.00039864, 0.00039806, 0.00048377,
      0.00076687,
      0.00037403]),
      'mean_score_time': array([0.00059886, 0.00019956, 0.00019951, 0.00019903,
      0.00079527,
      0.00019898]),
      'std_score_time': array([0.00048897, 0.00039911, 0.00039902, 0.00039806,
      0.00074207,
      0.00039797]),
      'param_C': masked_array(data=[1, 1, 10, 10, 20, 20],
          mask=[False, False, False, False, False, False],
          fill_value='?',
          dtype=object),
      'param_kernel': masked_array(data=['rbf', 'linear', 'rbf', 'linear', 'rbf',
      'linear'],
          mask=[False, False, False, False, False, False],
          fill_value='?',
          dtype=object),
      'params': [{'C': 1, 'kernel': 'rbf'},
      {'C': 1, 'kernel': 'linear'},
```

```

{'C': 10, 'kernel': 'rbf'},
{'C': 10, 'kernel': 'linear'},
{'C': 20, 'kernel': 'rbf'},
{'C': 20, 'kernel': 'linear'}],
'split0_test_score': array([0.96666667, 0.96666667, 0.96666667, 1.
0.96666667,
1.
]),
'split1_test_score': array([1., 1., 1., 1., 1., 1.]),
'split2_test_score': array([0.96666667, 0.96666667, 0.96666667, 0.9
, 0.9
0.9
]),
'split3_test_score': array([0.96666667, 0.96666667, 0.96666667, 0.96666667,
0.96666667,
0.93333333]),
'split4_test_score': array([1., 1., 1., 1., 1., 1.]),
'mean_test_score': array([0.98
, 0.98
, 0.98
, 0.97333333,
0.96666667,
0.96666667]),
'std_test_score': array([0.01632993, 0.01632993, 0.01632993, 0.03887301,
0.03651484,
0.0421637
]),
'rank_test_score': array([1, 1, 1, 4, 5, 5])}

```

```
[21]: df = pd.DataFrame(clf.cv_results_)
df
```

```
[21]:
```

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_C \
0	0.000598	0.000489	0.000598	0.000488	1
1	0.000600	0.000490	0.000599	0.000489	1
2	0.000605	0.000495	0.000394	0.000483	10
3	0.000798	0.000399	0.000200	0.000399	10
4	0.000205	0.000410	0.000193	0.000386	20
5	0.000200	0.000400	0.000198	0.000397	20

	param_kernel	params	split0_test_score \
0	rbf	{'C': 1, 'kernel': 'rbf'}	0.966667
1	linear	{'C': 1, 'kernel': 'linear'}	0.966667
2	rbf	{'C': 10, 'kernel': 'rbf'}	0.966667
3	linear	{'C': 10, 'kernel': 'linear'}	1.000000
4	rbf	{'C': 20, 'kernel': 'rbf'}	0.966667
5	linear	{'C': 20, 'kernel': 'linear'}	1.000000

	split1_test_score	split2_test_score	split3_test_score	split4_test_score \
0	1.0	0.966667	0.966667	1.0
1	1.0	0.966667	0.966667	1.0
2	1.0	0.966667	0.966667	1.0
3	1.0	0.900000	0.966667	1.0

4	1.0	0.900000	0.966667	1.0
5	1.0	0.900000	0.933333	1.0

	mean_test_score	std_test_score	rank_test_score
0	0.980000	0.016330	1
1	0.980000	0.016330	1
2	0.980000	0.016330	1
3	0.973333	0.038873	4
4	0.966667	0.036515	5
5	0.966667	0.042164	5

```
[23]: df[['param_C', 'param_kernel', 'mean_test_score']]
```

```
[23]:   param_C param_kernel mean_test_score
0        1         rbf      0.980000
1        1        linear      0.980000
2       10         rbf      0.980000
3       10        linear      0.973333
4       20         rbf      0.966667
5       20        linear      0.966667
```

```
[24]: clf.best_params_
```

```
[24]: {'C': 1, 'kernel': 'rbf'}
```

```
[27]: clf.best_score_
```

```
[27]: 0.98
```

```
[78]: dir(clf)
```

```
[78]: ['__abstractmethods__',
      '__class__',
      '__delattr__',
      '__dict__',
      '__dir__',
      '__doc__',
      '__eq__',
      '__format__',
      '__ge__',
      '__getattr__',
      '__getstate__',
      '__gt__',
      '__hash__',
      '__init__',
      '__init_subclass__',
      '__le__',
```

```
'__lt__',
'__module__',
'__ne__',
'__new__',
'__reduce__',
'__reduce_ex__',
'__repr__',
'__setattr__',
'__setstate__',
'__sizeof__',
'__str__',
'__subclasshook__',
'__weakref__',
'_abc_impl',
'_check_is_fitted',
'_estimator_type',
'_format_results',
'_get_param_names',
'_run_search',
'best_estimator_',
'best_index_',
'best_params_',
'best_score_',
'classes_',
'cv',
'cv_results_',
'decision_function',
'error_score',
'estimator',
'fit',
'fit_params',
'get_params',
'iid',
'inverse_transform',
'multimetric_',
'n_jobs',
'n_splits_',
'param_grid',
'pre_dispatch',
'predict',
'predict_log_proba',
'predict_proba',
'refit',
'refit_time_',
'return_train_score',
'score',
'scorer_',
```

```
'scoring',
'set_params',
'transform',
'verbose']
```

Use `RandomizedSearchCV` to reduce number of iterations and with random combination of parameters. This is useful when you have too many parameters to try and your training time is longer. It helps reduce the cost of computation

```
[102]: from sklearn.model_selection import RandomizedSearchCV
rs = RandomizedSearchCV(svm.SVC(gamma='auto'), {
    'C': [1,10,20],
    'kernel': ['rbf','linear']
},
cv=5,
return_train_score=False,
n_iter=2
)
rs.fit(iris.data, iris.target)
pd.DataFrame(rs.cv_results_)[['param_C', 'param_kernel', 'mean_test_score']]
```

```
[102]:  param_C  param_kernel  mean_test_score
0         10          rbf             0.98
1          1         linear             0.98
```

How about different models with different hyperparameters?

```
[92]: from sklearn import svm
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression

model_params = {
    'svm': {
        'model': svm.SVC(gamma='auto'),
        'params' : {
            'C': [1,10,20],
            'kernel': ['rbf','linear']
        }
    },
    'random_forest': {
        'model': RandomForestClassifier(),
        'params' : {
            'n_estimators': [1,5,10]
        }
    },
    'logistic_regression' : {
        'model': LogisticRegression(solver='liblinear',multi_class='auto'),
        'params': {
```

```

        'C': [1,5,10]
    }
}

```

```

[93]: scores = []

for model_name, mp in model_params.items():
    clf = GridSearchCV(mp['model'], mp['params'], cv=5,
        ↪return_train_score=False)
    clf.fit(iris.data, iris.target)
    scores.append({
        'model': model_name,
        'best_score': clf.best_score_,
        'best_params': clf.best_params_
    })

df = pd.DataFrame(scores, columns=['model', 'best_score', 'best_params'])
df

```

```

[93]:
      model  best_score  best_params
0      svm    0.980000  {'C': 1, 'kernel': 'rbf'}
1  random_forest    0.953333  {'n_estimators': 5}
2 logistic_regression    0.966667  {'C': 5}

```

Based on above, I can conclude that SVM with C=1 and kernel='rbf' is the best model for solving my problem of iris flower classification