FAST AND INFORMATIVE MODEL SELECTION USING LEARNING CURVE CROSS-VALIDATION

A PROJECT REPORT

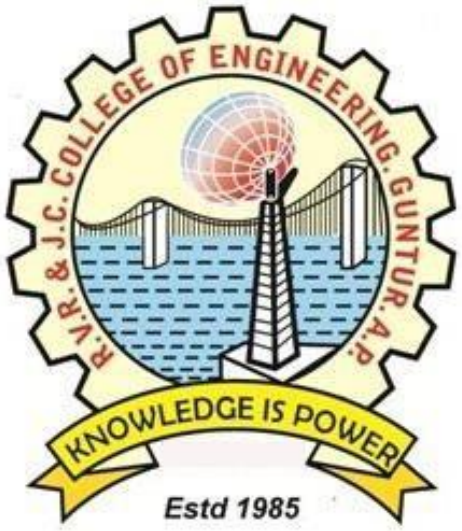
Submitted in the partial fulfillment of requirements to

RVR & JC COLLEGE OF ENGINEERING

For the award of the degree
B.Tech. in CSE

By

Ravinuthala Indeevar (Y20CS154)
Rishitha Gudla (Y20CS157)
Duddu Yaswanth (L21CS198)



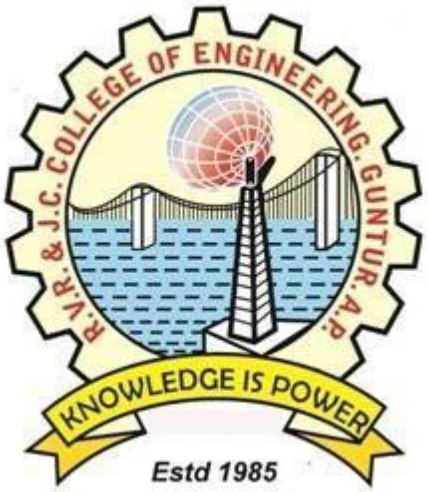
May, 2024

R.V.R. & J.C. COLLEGE OF ENGINEERING
(Autonomous)
(NAAC 'A+' Grade)
(Approved by **AICTE,** Affiliated to Acharya Nagarjuna University)

Chandramoulipuram::Chowdavaram,
GUNTUR – 522 019

R.V. R & J.C. COLLEGE OF ENGINEERING (Autonomous)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING



CERTIFICATE

This is to certify that this project work titled **"FAST AND INFORMATIVE MODEL SELECTION USING LEARNING CURVE CROSS VALIDATION"** is the work done by **Ravinuthala Indeevar (Y20CS154), Rishitha Gudla (Y20CS157), and Duddu Yaswanth (L21CS198)** under my supervision, and submitted in partial fulfillment of the requirements for the award of the degree, B.Tech. in Computer Science & Engineering, during the Academic Year **2023-2024**.

Mr. P. Rama Krishna	**Dr. K. Siva Kumar**	**Dr. M. Sreelatha**
Project Guide	In-charge, Project	Prof. &Head, CSE

ACKNOWLEDGEMENT

The successful completion of any task would be incomplete without proper suggestions, guidance, and environment. A combination of these factors acts as the backbone to our work on **"FAST AND INFORMATIVE MODELSELECTION USING LEARNING CURVE CROSS VALIDATION".**

We are very glad to express our special thanks to **Mr. P. Rama Krishna,** Guide for the project who has inspired us to select this topic and also given a lot of valuable advice in preparing content for this topic.

We are very thankful to **Dr. K. Siva Kumar**, Lecturer in charge of the Project who extended his encouragement and support to carry out this project.

We express our sincere thanks to **Dr. M. Sreelatha,** Head of the Department of Computer Science and Engineering for her encouragement and support to carry out this Project successfully.

We are very much thankful to **Dr. Kolla Srinivas**, Principal of R.V.R &J.C College of Engineering, Guntur for providing this supportive Environment.

Finally, we submit our reserves thanks to the lab staff in the Department of Computer Science and Engineering and to all our friends for their cooperation during preparation.

Ravinuthala Indeevar (Y20CS154)

Rishitha Gudla (Y20CS157)

Duddu Yaswanth (L21CS198)

ABSTRACT

Common cross-validation (CV) methods like k-fold cross-validation or Monte Carlo cross-validation estimate the predictive performance of a learner by repeatedly training it on a large portion of the given data and testing it on the remaining data. These techniques have two major drawbacks. First, they can be unnecessarily slow on large datasets. Second, beyond an estimation of the final performance, they give almost no insights into the learning process of the validated algorithm. Therefore, A new approach for validation which is based on learning curves (LCCV) is introduced. Instead of creating train-test splits with a large portion of training data, LCCV iteratively increases the number of instances used for training. In the context of model selection, it discards models that are unlikely to become competitive. In a series of experiments on 75 datasets, In over 90% of the cases using LCCV led to the same performance as using 5/10-fold CV while substantially reducing the runtime (median runtime reductions of over 50%) The performance using LCCV never deviated from CV by more than 2.5%.LCCV is compared to a racing-based method and successive halving, a multi- armed bandit method. Additionally, it provides important insights, which for example allows assessing the benefits of acquiring more data.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVATIONS

S.no	Abbrevation	Full Form
1	CV	Cross Validation
2	LCCV	Learning Curve Cross Validation
3	WEKA	Waikato Environment for Knowledge Analysis
4	SVM	Support Vector Machine
5	SMBC	Sequential Model-Based Optimization
6	TPE	Tree-structured Parzen Estimators
7	LCE	Learning Curve Extrapolation
8	AutoML	Automated Machine Learning
9	SMAC	Sequential Model-Based Algorithm Configuration
10	AMLB	AutoML Benchmark
11	GP	Gaussian Process
12	CMA-ES	Covariance Matrix Adaptation Evolution Strategy
13	NAML	Naive Automated Machine Learning
14	RP4AML	Runtime Prediction for Automated Machine Learning
16	FSPE	Fast Subsampling Performance Estimates
17	EPS	Efficient Progressive Sampling
18	ERT	Extremely Randomized Trees

1. INTRODUCTION

1.1. Background:

The model selection task is to pick, from a set or sequence of machine learning models,the one that has the best predictive performance. An important sub-task of model selection is to estimate the performance of each candidate model, which is done by so-called validation methods [9], [17], [19], [24]. One problem of these methods is that they cannot easily early discard candidate models.

Additionally, validation procedures provide little additional information to the data owner. Data owners usually need to ask themselves the question of which of the conventional options would improve the quality of their current model: (i) Gather more data points [25], (ii) Gather moreattributes, or (iii) Apply a wider range of models. Common validation techniques only provide information for (iii)

In this paper, these shortcomings are addressed through learning-curve-based cross-validation (LCCV). Instead of evaluating a candidate just for one training set size, it is evaluated, in increasing order [21], at different so-called anchors of training fold sizes, e.g. for 64, 128, 256, 512,... instances [12], [16], [17], [22], [23]. At each anchor, several evaluations are conducted until a stability criterion is met.
The main contribution of LCCV is that it combines empirical learning curves with an convexity assumption on learning curves to early prune candidates only if they are unlikely to be an optimal solution. The discarding decision in LCCV is model-free, which means that it does not rely on an imposed learning curve model like a power law.

The work is based on observation-based learning curves rather than iteration-based curves, which are obtained from learner performance observations across iterations like in neural networks . [1], [4], [7], [13]. First, even though incremental versions have been proposed for many learners, theseare often not realized in widely used implementations like WEKA or scikit-learn and hence not available for data scientists using those libraries. Iteration-based learning curves are hence largely limited to neural networks in practice. Second, empirical evidence supports the convexity assumption of observation-based learning curves whereas for iteration-based learning curves there is even contrary evidence, e.g., double descent.

1.2. Problem Definition:

The problem of model selection is to select, from some set or sequence of learners, theone that creates the best hypotheses on average for some given data. A learner is a function a :D → H , where D = {d | d ⊂ X × Y} is the space of all datasets, each of which is a finite relation between the two spaces.

$$R(h) = \int_{X,Y} loss\big(\, y, h(x)\big)\, dP(x, y) \qquad (1)$$

The focus is on supervised machine learning. For an instance space X and a label space Y, the hypothesis space H = {h | h : X → Y} is the set of all mappings between the instance space and the label space.

The performance of a hypothesis is its out-of-sample error R(h) = Z X ,Y loss(y, h(x)) dP(x, y). (1) Here, loss(y, h(x)) ∈ R is the penalty for predicting h(x) for an instance x ∈ X when thetrue label of that instance is y ∈ Y. Finally, P is a joint probability measure on X × Y from which the available dataset d has been generated.

1.3. Objective:

Given a dataset d and a set A = (a1, a2, ...) of learners, [21] a model selector must select arg min $_{a\in A}$ v(d,a). The work is based on observation-based learning curves rather than iteration-based curves, which are obtained from learner performance observations across iterations likein neural networks.

Interested in sequential model selection, in which A is a stream of candidates (usually generated by an optimizer) [6]. To make a choice, the model selector will keep track of the scorer of the best candidate seen so far [9]. To this end, it will, for each incoming candidate a ∈ A, (i) determine whether ν(d, a) will be better than r , and (ii) update r with ν(d, a) if the first case applies. Such a model selector can use an informed and relaxed validation function.

$$\widetilde{v} : D \times A \times R \rightarrow R \cup \{\perp\} \qquad (2)$$

1.4. Significance of the Work:

Model selection is a critical aspect of the machine learning process, and it offers several advantages that are essential for building effective and generalizable models.

By the help of model selection one can gain the following objectives:

- Reduced Overfitting:

Overfitting occurs when a model performs exceptionally well on the training data but fails to generalize to new data. Model selection helps identify the optimal model complexity that strikes the right balance between bias and variance, reducing the risk of overfitting.

- Different Models Evaluation:

Model selection enables a fair comparison of different models or algorithms. One can evaluate their performance side by side and choose the one that best suits your specificproblem and data.

- Scalability:

Model selection techniques like Cross-Validation can be adapted to handle largedatasets efficiently, allowing you to work with extensive and complex data without sacrificing the model evaluation process.

- Better Interpretability:

In some scenarios, simpler models are preferred because they are easier to interpret andexplain. Model selection can help identify such models while still maintaining satisfactory performance.

- Improved Generalization:

The primary goal of model selection is to identify the model that performs well on unseen data. By rigorously evaluating multiple models and selecting the best one basedon validation metrics, you improve the chances of your model generalizing well to new,unseen data.

2. LITERATURE SURVEY

2.1. Review of the Project:

B. Baker, et al. [1] Neural architecture search (NAS) is the process of automatically finding the best neural architecture for a given task. NAS is computationally expensive, as it requires training a large number of different architectures. The paper proposes a method for accelerating NAS by using performance prediction models. These models are trained on features extracted from neural architectures and their validation performance. Theycan then be used to predict the final performance of a neural architecture based on its partiallytrained weights. The authors show that performance prediction models can be used to significantly accelerate NAS. In particular, they show that their method can achieve a speedupof up to 6x in NAS. The paper's findings suggest that performance prediction models could bea valuable tool for accelerating NAS. However, the paper does have some limitations. For example, the authors only evaluated their method on a small number of datasets and architectures. It would be interesting to see how the method performs on a wider range of datasets and architectures. Overall, the paper "Accelerating Neural Architecture Search using Performance Prediction" is a valuable contribution to the field of NAS. The authors' findings suggest that performance prediction models could be a valuable tool for accelerating NAS, andthey open up new possibilities for the future of NAS research. Here is an example of how Predictive Termination could be used in practice. Suppose that a researcher wants to find the best neural architecture for image classification. The researcher has a limited amount of time and resources to train and evaluate neural architectures. Predictive Termination could be usedto predict the performance of a neural architecture based on its learning curve. The researcher could then use this information to focus on the neural architectures that are most likely to be successful. Predictive Termination is a powerful tool that can be used to accelerate NAS. The authors of the paper have shown that Predictive Termination can be effective in a variety of settings.

J. Bergstra et al. [2] paper discusses the problem of hyperparameter optimization, which is the process of finding the best values for the hyperparameters of a machine learning model. Hyperparameters are thesettings that control the behavior of a machine-learning model, and they can have a significantimpact on the model's performance. The paper proposes a method for hyperparameter optimization called random search. Random search randomly samples a set of hyperparametersand evaluates the model with those hyperparameters. The algorithm then repeats this process, sampling new hyperparameters each time. The best set of hyperparameters is the one that results in the best model performance. The paper shows that random search is a simple and effective method for hyperparameter optimization. In particular, the authors show that randomsearch is often as effective as other methods, such as grid search and Bayesian optimization, but it is much faster. The paper's findings have implications for the future of hyperparameter optimization. By using random search, it is possible to find the best hyperparameters for a machine learning model without spending a lot of time and resources. This could make hyperparameter optimization more accessible to researchers and practitioners, and could lead to the discovery of new and better machine learning models. Some of the key findings of the paper are Random search is a simple and effective method for hyperparameter optimization, Random search is often as effective as other methods, such as grid search and Bayesian optimization, but it is much faster, Random search is a good choice for hyperparameter optimization when the hyperparameter space is large and the computational resources are limited. The paper's findings are promising, and they suggest that random searchcould be a valuable tool for hyperparameter optimization. However, the paper does have somelimitations. For example, the authors only evaluated their method on a small number of machine learning models. It would be interesting to see how the method performs on a wider range of machine learning models. Overall, the paper "Random Search for Hyperparameter Optimization" is a valuable contribution to the field of hyperparameter optimization. The authors' findings suggest that random search could be a valuable tool for hyperparameter optimization, and they open up new possibilities for the future of hyperparameter optimizationresearch.

Y. Bergstra, et al. [3] The paper titled "Algorithms for Hyperparameter Optimization" addresses the critical issue of hyperparameter optimization, which involves tuning the parameters of machine learning algorithms to achieve better performance.The authors propose a novel method called Sequential Model-Based Optimization (SMBO) for hyperparameter optimization. SMBO uses a probabilistic model of the objective function to efficiently search for the optimal hyperparameters in a principled manner. This approach aims to minimize the computational cost and time required for hyperparameter search.The paper introduces the use of Tree-structured Parzen Estimators (TPE) as the probabilistic model for SMBO. TPE is usedto model the objective function, allowing the algorithm to identify promising regions in the hyperparameter space and sample from those regions to update the model iteratively.The authors compare SMBO with traditional hyperparameter optimization techniques, such as random search and grid search, and demonstrate its superior performance in terms of finding better hyperparameter settings with fewer evaluations of the objective function. Additionally, they discuss the advantages of SMBO in handling continuous and discrete hyperparameters efficiently. Grid search exhaustively evaluates hyperparameter combinations, but it becomes infeasible for high-dimensional spaces due to its exponential growth. Random search, on the other hand, samples hyperparameters randomly, but it lacks efficiency. Bayesian optimizationleverages probabilistic models to guide the search process and prioritize promising regions in the hyperparameter space, making it more efficient.

T. Domhan, et al. [4] The paper “Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves” proposes a new method for speeding up the automatic hyperparameter optimization of deep neural networks. The method, called "extrapolation of learning curves," works by predicting the future performance of a neural network based on its current learning curve. This allows the optimizer to focus its search on promising hyperparameter combinations, which can significantly reduce the amount of time ittakes to find a good set of hyperparameters. The paper evaluated the proposed method on a variety of benchmark datasets and tasks. The results showed that the method was able to significantly speed up the optimization process, while still finding good sets of hyperparameters.

For example, on the CIFAR-10 dataset, the method was able to find a modelthat achieved 90% accuracy in 100 trials, while a traditional grid search would have taken over1000 trials. The paper also discussed the limitations of the proposed method. One limitation is that the method can only be used for problems where the learning curves are monotonically increasing.Another limitation is that the method can be sensitive to the noise in the learning curves. Overall, the paper presents a promising new method for speeding up the automatic hyperparameter optimization of deep neural networks. The method is simple to implement andhas been shown to be effective on a variety of benchmark datasets and tasks.

P. Gijsbers, et al. [5] The paper proposes a new benchmark for comparing different Auto-ML frameworks. The benchmark, called AMLB, is designed to be open, extensible, and reproducible. It uses a variety of public datasets and tasks, and it provides a number of metrics for evaluating the performance of Auto-ML frameworks. The paper evaluated AMLB on 9 well-known Auto-ML frameworks. The results showed that therewas a significant variation in the performance of the different frameworks. For example, on theCIFAR-10 dataset, the best framework achieved an accuracy of 94.2%, while the worst framework achieved an accuracy of 82.3%.The paper also discussed the limitations of AMLB. One limitation is that the benchmark is still under development. Another limitation is that the benchmark is not exhaustive. There are many other Auto-ML frameworks that are not includedin the benchmark. Overall, the paper presents a promising new benchmark for comparing different Auto-ML frameworks. The benchmark is open, extensible, and reproducible, and it provides a number of metrics for evaluating the performance of Auto-ML frameworks. Some of the key features of AMLB are It is open-source, which means that anyone can contribute toit. It is extensible, which means that new datasets and tasks can be easily added. It isreproducible, which means that the results of the benchmark can be easily reproduced by others. It provides a number of metrics for evaluating the performance of Auto-ML frameworks, including accuracy, inference time, and framework failures. AMLB is a valuableresource for researchers and practitioners who are interested in comparing different Auto-MLframeworks. The benchmark provides a standardized way to evaluate the performance of Auto-ML frameworks, which makes it easier to compare different frameworks.

K. Jamieson et al. [6] The paper titled "Non-stochastic Best Arm Identification and Hyperparameter Optimization" .The paper introduces the problem of non-stochastic best arm identification (NBI), which is the problem of finding the best arm in a set of arms when the rewards are not drawn from a stochastic distribution. The authors propose a new algorithmfor NBI called Successive Halving, which is based on the idea of dividing the arms into two groups and then recursively halving the groups until the best arm is identified. The paper also discusses the problem of hyperparameter optimization, which is the problem of finding the best hyperparameters for a machine learning model. The authors propose a new algorithm for hyperparameter optimization called Hyperband, which is based on the idea of using Successive Halving to search for the best hyperparameters. The paper evaluates the performance of Successive Halving and Hyperband on a variety of tasks, and shows that they can achieve state-of-the-art results. The paper concludes by discussing the advantages and disadvantages of Successive Halving and Hyperband. The paper "Non-stochastic Best Arm Identification and Hyperparameter Optimization" is a valuable contribution to the field of machine learning. The authors' findings suggest that Successive Halving and Hyperband are powerful tools for a variety of tasks. The paper is well-written and easy to understand. Some of the key findings ofthe paper are Successive Halving is a simple and efficient algorithm for NBI. Hyperband is a powerful algorithm for hyperparameter optimization. Successive Halving and Hyperband can achieve state-of-the-art results on a variety of tasks. The paper's findings suggest that Successive Halving and Hyperband could be valuable tools for a variety of machine learning tasks. However, the paper does have some limitations. For example, the authors only evaluated their methods on a small number of tasks. It would be interesting to see how the methods perform on a wider range of tasks.

A. Klein, et al. [7] The paper proposes a method for predicting learning curvesusing Bayesian neural networks. Learning curves are plots of the performance of a machine learning algorithm as a function of the amount of training data. The proposed method uses Bayesian neural networks to model the uncertainty in the learning curve. The paper evaluates the proposed method on a variety of datasets and shows that it can accurately predict learning

curves. The paper also discusses the advantages and disadvantages of the proposed method. The paper is a valuable contribution to the field of machine learning. The authors' findings suggestthat the proposed method could be a valuable tool for practitioners who are looking to predictthe performance of machine learning algorithms. Some of the key findings of the paper are Bayesian neural networks can be used to predict learning curves, The proposed method can accurately predict learning curves on a variety of datasets. The proposed method is easy to implement and can be used with a variety of machine-learning algorithms. The paper's findingssuggest that the proposed method could be a valuable tool for practitioners who are looking topredict the performance of machine learning algorithms. However, the paper does have some limitations. For example, the authors only evaluated their method on a small number of datasets. It would be interesting to see how the method performs on a wider range ofdatasets. Overall, the paper is a valuable contribution to the field of machine learning. The authors' findings suggest that the proposed method could be a valuable tool for practitioners who are looking to predict the performance of machine learning algorithms.

M. Last, et al. [8] The paper titled "Improving Data Mining Utility with Projective Sampling" challenge the trade-off between data mining accuracy and computational efficiency. Traditional data mining techniques often suffer from increased computational costs when dealing with large datasets, leading to longer processing times and reduced efficiency. Projective sampling aims to overcome this challengeby selectively sampling a subset of data points that are most informative for the data mining task. Instead of using the entire dataset, projective sampling focuses on the most relevant instances and attributes, significantly reducing the data size and computational complexity. The authors propose an iterative algorithm for projective sampling, which involves two main steps:relevance estimation and sampling. The relevance estimation step assesses the importance of each instance and attribute based on their contribution to the overall data mining task. The sampling step selects a representative subset of instances and attributes using a probability-based sampling approach. To evaluate the effectiveness of projective sampling, the authors conduct experiments on various real-world datasets and compare the results with traditional data mining techniques. The experiments demonstrate that projective sampling achievescomparable accuracy to the full dataset while requiring substantially fewer resources and computation time.

The paper highlights that projective sampling offers a practical solution to enhance the utility of data mining tasks, particularly when dealing with large datasets. By focusing on informative subsets, projective sampling enables more efficient and scalable data mining, making it a valuable approach for accelerating data analysis processes in real-world applications.

R. Leite et al. [9] The paper titled "Active Testing Strategy to Predict the Best Classification Algorithm via Sampling and Metalearning". The paper proposesa method for predicting the best classification algorithm for a given task. The method is basedon the idea of active testing, which is a technique for iteratively selecting the most informativedata points to evaluate. The method works by first sampling a small number of data points fromthe dataset. These data points are then used to train a metamodel, which is a model that predictsthe performance of different classification algorithms on unseen data. The metamodel is then used to select the next set of data points to evaluate. This process is repeated until the best classification algorithm is found. The paper evaluates the method on a variety of datasets and shows that it can achieve good accuracy in predicting the best classification algorithm. The paper concludes by discussing the advantages and disadvantages of the method. The paper "Active Testing Strategy to Predict the Best Classification Algorithm via Sampling and Metalearning" is a valuable contribution to the field of machine learning. The authors' findingssuggest that the method could be a valuable tool for selecting classification algorithms for a variety of tasks. Some of the key findings of the paper are (i)Active testing is an effective methodfor predicting the best classification algorithm.(ii)The method can be used to select classification algorithms for a variety of tasks.(iii)The method is relatively efficient and can beused to process large datasets. The paper's findings suggest that active testing could be a valuable tool for selecting classification algorithms for a variety of machine-learning tasks. However, the paper does have some limitations. For example, the authors only evaluated theirmethod on a small number of datasets. It would be interesting to see how the method performson a wider range of datasets.

L. Li, K. G. Jamieson et al. [10] The paper titled "Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization" by L. Li, K. G. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, published in the Journal of Machine Learning Research in 2017, introduces Hyperband, an innovative bandit- based algorithm for hyperparameter optimization in machine learning. Hyperparameter optimization is a crucial task in machine learning, as it involves selecting the optimal set of hyperparameters for a given learning algorithm to achieve the best performance on a specific dataset. Traditional methods, such as random search or grid search, can be computationally expensive and inefficient, especially when dealing with large search spaces. Hyperband addresses this challenge by employing a bandit-based strategy to efficiently explore and exploitthe hyperparameter search space. The key idea behind Hyperband is to use a series of low-cost,resource-constrained configurations (called "brackets") to quickly identify and focus on promising hyperparameter settings.The algorithm proceeds in a multi-armed bandit fashion, where different brackets correspond to different arms. Each bracket executes a fixed number of hyperparameter configurations, randomly sampled from the search space. After each configuration's evaluation, the underperforming configurations are eliminated, and the remaining ones are promoted to the next bracket with more resources, allowing them to be further explored. This process continues iteratively, leading to efficient convergence to the best hyperparameter configuration.To evaluate the effectiveness of Hyperband, the authors conduct experiments on various machine learning tasks, including neural networks and support vector machines, with different datasets. They compare Hyperband's performance against state-of- the-art hyperparameter optimization methods, such as Bayesian optimization and random search. The results demonstrate that Hyperband consistently outperforms other approaches in terms of both efficiency and final model performance, achieving better results with significantly fewer evaluations.

M. Lopez-Ibanez, et al. [11] The paper titled "The irace package: Iterated racing for automatic algorithm configuration". The paper introduces the irace package, which is a software tool for automatic algorithm configuration. irace uses a technique called iterated racing to search for the best configuration of a given algorithm. Iterated racing works by repeatedly running the algorithm with different configurations and selecting the best configuration based on its performance.The paper evaluates irace on a variety of tasks and shows that it can achieve good performance. The paperalso discusses the advantages and disadvantages of irace.The paper "The irace package: Iterated racing for automatic algorithm configuration" is a valuable contribution to the field ofmachine learning. The authors' findings suggest that irace is a powerful tool for automatic algorithm configuration.Some of the key findings of the paper are (i)i-race is a powerful tool for automatic algorithm configuration.(ii)i-race can achieve good performance on a variety of tasks.(iii)i-race is relatively easy to use. The paper's findings suggest that irace could be a valuable tool for a variety of machine learning tasks. However, the paper does have some limitations. For example, the authors only evaluated their method on a small number of tasks.It would be interesting to see how the method performs on a wider range of tasks.

F. Mohr, et al. [12] The paper titled "Towards model selection using learning curve cross-validation".The paper proposes a new method for model selection called learning curve cross-validation (LCCV). LCCV is based on the idea of using learning curves to predict the performance of a model on unseen data.Learning curves are plots of the model's performance as a function of the amount of training data. The paper shows that the shape of the learning curve can be used to predict the model's performance on unseen data.The paper evaluates LCCV on a variety of datasets and shows that it can achieve good accuracy in predicting the performance of a model.The paper concludes by discussing the advantages and disadvantages of LCCV.The paper "Towards model selection using learning curve cross-validation" is a valuable contribution to the field of machine learning. The authors' findings suggest that LCCV could be a valuable toolfor model selection in a variety of tasks.Some of the key findings of the paper are (i) LCCV isa new and effective method for model selection.(ii)LCCV can be used to select models for a variety of tasks.(iii)LCCV is relatively easy to use.

The paper's findings suggest that LCCV could be a valuable tool for a variety of machine learning tasks. However, the paper does have some limitations. For example, the authors onlyevaluated their method on a small number of datasets. It would be interesting to see how the method performs on a wider range of datasets.

F. Mohr et al. [13] Learning curves are a powerful tool for understanding the performance of machine learning algorithms. They show how the performance of an algorithmchanges as the amount of training data increases. This information can be used to make decisions about the algorithm selection where Learning curves can be used to compare the performance of different algorithms on a given dataset. This information can be used to selectthe algorithm that is most likely to perform well on future data. And the Parameter tuning wherethe Learning curves can be used to identify the optimal hyperparameters for an algorithm. Thisinformation can be used to improve the performance of the algorithm on a given dataset. And the Early stopping where the Learning curves can be used to identify when an algorithm has converged. This information can be used to stop training the algorithm early, which can save time and resources.The paper by Mohr and van Rijn provides a comprehensive survey of the literature on learning curves for decision making in supervised machine learning. The paper covers a wide range of topics, including The different types of learning curves: There are two main types of learning curves: error curves and confidence curves. Error curves show how theerror rate of an algorithm changes as the amount of training data increases. Confidence curvesshow how the confidence of an algorithm's predictions changes as the amount of training data increases.The factors that affect learning curves: The performance of a learning curve can be affected by a variety of factors, including the size of the dataset, the complexity of the task, and the noise level in the data.The use of learning curves for decision making: Learning curves can be used for a variety of decision-making tasks, such as algorithm selection, parameter tuning, and early stopping.The paper concludes by discussing the challenges and future directions of research in the area of learning curves for decision making. The authors highlight the need for more research on the use of learning curves for complex tasks, such as natural language processing and medical diagnosis. They also call for more research on the use of learning curves for online learning.

F. Mohr et al. [14] The paper introduces a new approach to automated machine learning (AutoML) called Naive AutoML. Naive AutoML is a simple and effective approach that works by first selecting a set of machine learning algorithms and then optimizing their hyperparameters using a grid search.The paper evaluates Naive AutoML on a variety of datasets and shows that it canachieve comparable performance to state-of-the-art AutoML systems. The paper also discussesthe advantages and disadvantages of Naive AutoML.The paper "Naive Automated Machine Learning - A Late Baseline for AutoML" is a valuable contribution to the field of AutoML. The authors' findings suggest that Naive AutoML could be a valuable tool for practitioners whoare looking for a simple and effective way to automate their machine learning workflows.Someof the key findings of the paper are Naive AutoML is a simple and effective approach to AutoML.Naive AutoML can achieve comparable performance to state-of-the-art AutoML systems.Naive AutoML is easy to use and can be implemented in a variety of programming languages.The paper's findings suggest that Naive AutoML could be a valuable tool for a variety of machine learning tasks. However, the paper does have some limitations. Forexample, the authors only evaluated their method on a small number of datasets. It would be interesting to see how the method performs on a wider range of datasets.

F. Mohr et al. [15] The paper "Naive Automated Machine Learning" introduces a novel approach called "Naive Automated Machine Learning" (NAML) that simplifies and automates the process of configuring machine learning pipelines.Automated Machine Learning (AutoML) has gained attention for its ability to automate the selection and tuning of machine learning models and hyperparameters. However, many existing AutoML approaches are complex and computationally expensive, making them less accessible to non-experts and resource- constrained environments.NAML takes a different approach by proposing a simple andefficient method that is suitable for both novice users and resource-limited settings. The key idea behind NAML is to define a limited space of possible pipelines and their configurations, removing the need for costly optimization algorithms.The paper outlines the NAML framework, which consists of a library of predefined building blocks and a meta-learning approach. The building blocks represent basic operations in a machine learning pipeline, suchas data preprocessing, feature extraction, and model selection. By combining these building blocks, NAML creates a wide variety of pipelines.

The meta-learning component of NAML leverages historical performance data of the predefined pipelines to predict the performance of new pipelines without explicitly training them. This approach significantly reduces the computational burden compared to traditional AutoML methods.To evaluate NAML's effectiveness, the authors conduct experiments on various benchmark datasets and compare it to state-of-the-art AutoML approaches. The results show that NAML achieves competitive performance while requiring substantially less computation time and resources.

F. Mohr, et al. [16] The paper addresses the problem of automating the machine learning process. The authors propose a method called ML-Plan that uses hierarchical planning to automate the process.Hierarchical planning is a type of planning that decomposes a complex task into a series of smaller tasks. The authors use hierarchical planning to decompose the machine learning process into a series of subtasks, such as algorithm selection, hyperparameter optimization, and evaluation.ML-Plan then uses a bandit algorithm to select the best subtask toperform at each step. The bandit algorithm takes into account the cost of performing the subtask, the expected benefit of performing the subtask, and the uncertainty about the benefit of performing the subtask.The authors show that ML-Plan can automate the machine learning process with high accuracy. The authors also show that ML-Plan is more efficient than other methods for automating the machine learning process.The paper ML-Plan: Automated machine learning via Hierarchical Planning is a valuable contribution to the field of machine learning. The authors' findings suggest that ML-Plan could be a valuable tool for practitioners who are looking to automate the machine learning process. Some of the key findings of the paper ML-Plan is a method for automating the machine learning process. ML-Plan uses hierarchical planning to automate the process. ML-Plan can automate the machine learning process with high accuracy. ML-Plan is more efficient than other methods for automating the machine learning process.The paper's findings suggest that ML-Plan could be a valuable tool for practitioners who are looking to automate the machine learning process. However, the paper does have some limitations. For example, the authors only evaluate ML-Plan on a small number of datasets. It would be interesting to see how the method performs on a wider range of datasets.

F. Mohr et al. [17] The paper proposes a method for predicting the runtime of machine learning pipelines in the context of automated machine learning. The method, called "Implicit Runtime Model (IRM)," works by learning a model of the runtime of each individual algorithm that may be used in a pipeline. The IRM then combines these individual models to predict the runtime of the overall pipeline.The paper evaluated the IRM on a variety of machine learning pipelines. The results showed that the IRMwas able to accurately predict the runtime of the pipelines, with an average error of less than 10%. The IRM was also able to significantly improve the performance of an automated machine learning (AutoML) tool, by allowing the tool to focus its search on pipelines that were likely to be within the specified runtime budget. The paper also discussed the limitations of theIRM. One limitation is that the IRM is only as accurate as the individual models that it is trainedon. Another limitation is that the IRM is not able to predict the runtime of pipelines that containnew or unknown algorithms. Here are some of the key features of the IRM It is simple to implement. It is effective on a variety of machine learning pipelines. It can be used to improve the performance of AutoML tools. The IRM is a valuable tool for researchers and practitionerswho are interested in predicting the runtime of machine learning pipelines. The IRM can be used to improve the efficiency of AutoML tools and to ensure that pipelines are completed within the specified runtime budget.

F. Mohr, et al. [18] The paper introduces LCDB 1.0, a new databaseof learning curves for classification tasks. LCDB 1.0 contains learning curves for 20 different classification algorithms on 246 different datasets. The learning curves are collected using a variety of experimental settings, including different training set sizes, different hyperparameter configurations, and different evaluation metrics. The paper evaluates LCDB 1.0 and shows thatit can be used to understand the behavior of different classification algorithms on a variety of datasets. The paper also discusses the potential of LCDB 1.0 for a variety of applications, suchas algorithm selection, hyperparameter tuning, and model explanation. The paper "LCDB 1.0: A First Learning Curves Database for Classification Tasks" is a valuablecontribution to the field of machine learning. The authors' findings suggest that LCDB 1.0 could be a valuable tool for practitioners and researchers who are interested in understanding the behavior of classification algorithms.

Some of the key findings of the paper LCDB 1.0 is avaluable resource for understanding the behavior of classification algorithms. LCDB 1.0 can beused for a variety of applications, such as algorithm selection, hyperparameter tuning, and model explanation.LCDB 1.0 is a valuable tool for practitioners and researchers who are interested in understanding the behavior of classification algorithms. The paper's findings suggest that LCDB 1.0 could be a valuable tool for a variety of machine learning tasks. However, the paper does have some limitations. For example, the authors only evaluated theirmethod on a small number of datasets. It would be interesting to see how the method performson a wider range of datasets.

J. Petrak, et al. [19] The paper addresses the problem of selecting the best classification algorithm for a given dataset. The author proposes a method called Fast Subsampling Performance Estimates(FSPE) that uses subsampling to select the best algorithm. FSPE works by first subsampling thedataset into a number of smaller datasets. The author then trains a different classification algorithm on each of the smaller datasets. The author then uses the performance of the classification algorithms on the smaller datasets to estimate their performance on the entire dataset.The algorithm with the best estimated performance is selected as the best algorithm. The author shows that FSPE can select the best algorithm in a fraction of the time required by other methods for algorithm selection.The paper *Fast Subsampling Performance Estimates for Classification Algorithm Selection* is a valuable contribution to the field of machine learning. The author's findings suggest that FSPE could be a valuable tool for practitioners who are looking to select the best classification algorithm for a given dataset.

FSPE is a method for selecting the best classification algorithm for a given dataset.FSPE uses subsampling to select the best algorithm.FSPE can select the best algorithm in a fraction of the time required by other methods for algorithm selection.The paper's findingssuggest that FSPE could be a valuable tool for practitioners who are looking to select the best classification algorithm for a given dataset. However, the paper does have some limitations. For example, the author only evaluates FSPE on a small number of datasets. Itwould be interesting to see how the method performs on a wider range of datasets.

F. Provost, et al. [20] In the paper "Efficient Progressive Sampling," propose a novel approach for efficient and effective sampling of data in the context of classification tasks.The main goal of the study is to addressthe challenge of dealing with large datasets while ensuring that the selected samples are representative of the entire dataset. Traditional random sampling methods may result in inefficient use of computational resources and suboptimal representation of certain classes or patterns in the data.The paper addresses the problem of selecting a representative sample of a large dataset. The authors propose a method called Efficient Progressive Sampling (EPS) thatuses a divide-and-conquer approach to select the sample.EPS works by first dividing the datasetinto a number of smaller datasets. The authors then select a representative sample from each ofthe smaller datasets. The authors then use the samples from the smaller datasets to construct arepresentative sample of the entire dataset.The authors show that EPS can select a representative sample of a large dataset in a fraction of the time required by other methods forsampling.The paper Efficient progressive sampling is a valuable contribution to the field of machine learning. The authors' findings suggest that EPS could be a valuable tool for practitioners who are looking to select a representative sample of a large dataset.Some of the key findings of the paper are EPS is a method for selecting a representative sample of a large dataset. EPS uses a divide-and-conquer approach to select the sample.EPS can select arepresentative sample of a large dataset in a fraction of the time required by other methods forsampling.

A. Sabharwal et al. [21] The paper addresses the problem of selecting a near-optimal learner from a set of candidate learners in an online setting, where the data is presented one instance at a time. The authors propose a novel algorithm called DAUB (Data Allocation usingUpper Bounds) that allocates data to learners based on their estimated performance.DAUB works by first constructing a set of upper bounds for the performance of each learner. These upper bounds are then used to allocate data to learners in a way that minimizes the regret, whichis the difference between the performance of the best learner and the performance of the learnerthat is actually selected.The paper evaluates DAUB on a variety of datasets and shows that it can achieve good performance. The paper also discusses the advantages and disadvantages of DAUB.The paper "Selecting near-optimal learners via incremental data allocation" is a valuable contribution to the field of machine learning. The authors' findings suggest that DAUB could be a valuable tool for practitioners who are looking to select near-optimal learners in anonline setting.Some of the key findings of the paper are DAUB is a novel algorithm for selecting near-optimal learners in an online setting.DAUB can achieve good performance on avariety of datasets.DAUB is easy to implement and can be used with a variety of machine learning algorithms.The paper's findings suggest that DAUB could be a valuable tool for practitioners who are looking to select near-optimal learners in an online setting. However, the paper does have some limitations. For example, the authors only evaluated their method on a small number of datasets. It would be interesting to see how the method performs on a wider range of datasets.Overall, the paper "Selecting near-optimal learners via incremental data allocation" is a valuable contribution to the field of machine learning. The authors' findings suggest that DAUB could be a valuable tool for practitioners who are looking to select near-optimal learners in an online setting.

J. N. van Rijn, et al. [22] In their paper "Fast Algorithm Selection Using Learning Curves," propose a novel method for efficient algorithm selection based on learning curves.Thepaper addresses the problem of selecting the best machine learning algorithm for a given dataset. The authors propose a method called Fast Algorithm Selection using Learning Curves(Fast-ALS) that uses learning curves to select the best algorithm.Fast-ALS works by first fittinga learning curve to each algorithm on the dataset. The learning curve shows how the performance of the algorithm changes as the amount of training data increases. The authors then use a loss time curve to compare the learning curves of the different algorithms. The loss time curve shows how the regret of an algorithm changes as the amount of training data increases. The regret is the difference between the performance of the algorithm and the best possible performance.The algorithm with the smallest loss time curve is selected as the best algorithm. The authors show that Fast-ALS can select the best algorithm in a fraction of the time required by other methods for algorithm selection.The paper Fast algorithm selection using learning curves is a valuable contribution to the field of machine learning. The authors' findings suggest that Fast-ALS could be a valuable tool for practitioners who are looking to select the best machine learning algorithm for a given dataset.Some of the key findings of the paper are Fast-ALS is a method for selecting the best machine learning algorithm for a given dataset.Fast-ALS uses learning curves to select the best algorithm.Fast-ALS can select the best algorithm in a fraction of the time required by other methods for algorithm selection.The paper's findings suggest that Fast-ALS could be a valuable tool for practitioners who are looking to select the best machine learning algorithm for a given dataset. However, the paper does have some limitations. For example, the authors only evaluate Fast-ALS on a small number of datasets. It would be interesting to see how the method performs on a wider range of datasets.Overall, the paper Fast algorithm selection using learning curves is a valuable contribution to the field of machine learning. The authors' findings suggest that Fast-ALS could be a valuable tool for practitioners who are looking to select the best machine learning algorithm for a given dataset.

T. J. Viering et al. [23] In their paper "The Shape of Learning Curves: A Review," represents a comprehensive review of learning curves in machine learning.Learning curves are essential tools for analyzing the performance of machine learning algorithms and understanding how their accuracy improves with increasing training data. The authors explore various aspects of learning curves, including their shape, behavior, and implications.The paper starts by introducing the concept of learning curves and their importance in machine learning research and practice. It provides a clear definition of learning curves and their common characteristics.Next, the authors discuss the factors that influence the shape of learning curves, such as algorithm complexity, dataset size,and model capacity. They also explore the trade-offs between bias and variance in learning curves and how these impact algorithm performance.The authors identify three main shapes of learning curves:Convex learning curves: These curves show that the performance of themachine learning algorithm improves rapidly at first, and then the improvement slows down.Linear learning curves: These curves show that the performance of the machine learning algorithm improves at a constant rate.Concave learning curves: These curves show that the performance of the machine learning algorithm improves rapidly at first, and then the improvement starts to decline.The review includes an in-depth analysis of different learning curve shapes, including the classic U-shape, sigmoidal shape, and plateau shape, among others. The authors discuss the implications of these shapes and how they relate to algorithm performance and generalization. The authors also identify a number of factors that can affect the shape of learning curves, including The complexity of the machine learning algorithm. Thesize and quality of the training dataset.The noise in the training dataset.The choice of hyperparameters. The authors conclude that the shape of learning curves can be a valuable tool for understanding the behavior of machine learning algorithms. The authors also suggest that the shape of learning curves can be used to select the best machine-learning algorithm for a given task.

D. Wang, et al. [24] The paper introduces AutoDS, a system for automated machine learning that is designed to be human-centered. AutoDS takes into account the preferences and constraints of human users, and it provides users with a high degree of controlover the automated machine learning process.The paper evaluates AutoDS on a variety of datasets and shows that it can achieve good performance. The paper also discusses the advantages and disadvantages of AutoDS. The paper is a valuable contribution to the field of machine learning. The authors' findings suggest that AutoDS could be a valuable tool for practitioners who are looking to automate their machine learning workflows while stillmaintaining human control.Some of the key findings of the paper: AutoDS is a system for automated machine learning that is designed to be human-centered. AutoDS can achieve goodperformance on a variety of datasets.AutoDS provides users with a high degree of control overthe automated machine learning process. The paper's findings suggest that AutoDS could be a valuable tool for practitioners who are looking to automate their machine learning workflows while still maintaining human control. However, the paper does have some limitations. For example, the authors only evaluated their method on a small number of datasets. It would be interesting to see how the method performs on a wider range of datasets.Overall, the paper is avaluable contribution to the field of machine learning. The authors' findings suggest that AutoDS could be a valuable tool for practitioners who are looking to automate their machine learning workflows while still maintaining human control.

G. M. Weiss, et al. [25] In the paper "Maximizing Classifier Utility When There Are Data Acquisition and Modeling Costs," address the problem of optimizing classifier performance while considering both data acquisition and modeling costs.The authors introducethe concept of classifier utility, which combines the classifier's performance with the associatedcosts of data acquisition and modeling. They propose a novel framework for making data acquisition decisions that balance the trade-offs between improving classifier performance andmanaging costs.The paper begins by presenting the motivation behind their research, highlighting the practical challenges of acquiring large amounts of data for model training and the computational costs involved in building complex classifiers.Next, the authors define the concept of utility functions, which quantifies the value of classifier performance relative to the costs incurred. They develop a formal mathematical framework to model utility functions and provide examples of different utility functions tailored to specific real-world scenarios.To optimize the classifier utility, the authors propose a heuristic search algorithm that explores the space of data acquisition and modeling decisions. This algorithm aims to find the optimal balance between increasing data size and model complexity, ultimately maximizing the classifier's performance given the available resources.The paper also discusses various factors that affect utility optimization, such as the accuracy of utility estimation and the trade-offs between data acquisition and modeling costs. The authors conduct experiments using real-world datasets to demonstrate the effectiveness of their approach in maximizing classifier utility.

2.2 Limitations of Existing Techniques:

Model selection is at the heart of many data science approaches [24]. When provided with a new dataset, a data scientist is confronted with the question of which model to apply to this. This problem is typically abstracted as the combined algorithm selection and hyperparameter optimization problem. To properly deploy model selection methods, there are three important components:

1) A configuration space, which specifies the set of algorithms to be considered and, perhaps, their hyperparameter domains

2) A search procedure, which determines the order in which these algorithms are considered
3) An evaluation mechanism, which assesses the quality of a certain algorithm.

The typical methods used as evaluation mechanisms are using classical methods such as a holdout set, cross-validation, leave-one-out cross-validation, and bootstrapping. This can be sped up by applying racing methods and by successive Halving.

Cross-Validation is a technique is used to evaluate the performance of the model by splitting the available data into training and testing subsets.The goal is to estimate how well the model perform on new unseen data.

Bootstrapping is a resampling technique which involves generating multiple samples of original data set by randomly splitting the data set to generate train and test data with replacement.

Racing Methods are the methods that use a probabilistic model. This model is iteratively updated based on the performance of the models on the validation set and the current model isused to select the next set of hyperparameters to be tested.

Successive Halving in one type of racing method which uses principle of elimination. This algorithm is an iterative one which iteratively trains a set of models on fixed parameters and develop the fraction data and then selecting top performance models.

The Following are the common disadvantages of the following methods:

- The methods are Computationally intensive.
- They are highly biased towards certain kind of models
- There is a chance of overfitting of the data
- The best candidates may be early discarded
- They cannot easily early discard candidate models that do not seem to be competitive
- They Evaluate the model on single models.
- They are Restricted to certain kind of Datasets

So, In-order to tackle all these circumstances a new algorithm is introduced. The LCCV Instead of evaluating a candidate just for one training set size, it is evaluated, in increasing order, at different so-called anchors of training fold sizes, e.g. for 64, 128, 256, 512, ... instances [21]. At each anchor, several evaluations are conducted until a stability criterion is met [12], [13], [19]. Other authors also utilized learning curves or sub-sampling to speed-up model selection by discarding unpromising candidates early [1], [7], [11]. The main contribution of LCCV is that it combines empirical learning curves with an convexity assumption on learning curves to early prune candidates onlyif they are unlikely to be an optimal solution [12], [21], [22], [23]. The discarding decision in LCCV is model-free,which means that it does not rely on an imposed learning curve model like a power law.

Instead of a single-point observation, LCCV gives insights about the learning behaviour of a candidate. This is very useful to answer the question of whether more data would help to obtain better results information that is not naturally provided by any of the above validation techniques [25]. Note that this is not the same as retroactively generating the learning curve based on the best performing learner after a cross-validation procedure. There is no guarantee that this learner (or even only any of the k best learners) would also be the best if more data were available; this is precisely something that can be revealed with LCCV since it computes the learning curves of all learners that it considers.

3. SYSTEM ANALYSIS

3.1 Requirements Specification:

Requirements Analysis, also called as requirements engineering is the process of determining user expectations for a new or modified product. These features called requirements must be quantifiable, relevant , and detailed. In Software Engineering, such requirements are often called functional specifications. Requirement analysis is critical to the success or failure of a systems or software project. The requirement should be documented, actionable, measurable, testable, traceable related to identified business needs or opportunities , and defined to a level of detail sufficient for system design.

3.1.1 Functional Requirements:

A Functional Requirement is a description of the service that the software must offer. A function may be input to the system, its behaviour and outputs. It can be any functionality which defines what function a system is likely to perform. Functional requirements are also called as Functional Specifications.

The following are the functional requirements

1. Algorithm Implementation:

- The system must implement the algorithm for generating learning curves.
- It must also implement cross-validation techniques for model evaluation.

2. Model Selection:

- The system should allow for the selection of multiple candidate models for evaluation.
- It should provide functionality to compare the performance of these models using learning curve analysis.

3. Performance Metrics:

- The system must calculate performance metrics such as accuracy, confidence intervals for each model.
- It should provide means to visualize these metrics to aid in model selection.

4. Speed Optimization:

- The system should be designed to execute model selection efficiently, considering the computational cost of generating learning curves and performing cross-validation.

3.1.2 Non-Functional Requirements:

A Non-functional requirement (NFR) is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviours. They are contrasted with functional requirements that define specific behaviour or functions.

1. Performance:

- The system should be capable of handling datasets of varying sizes.
- It should execute model selection and evaluation in a reasonable amount of time, even for large datasets.

2. Scalability:

- The system should scale well with increasing data size and complexity.
- It should be able to handle different types of machine learning models without significant performance degradation.

3. Robustness:

- The system should be resilient to errors and exceptions.
- It should handle invalid inputs gracefully and provide informative error messages to users.

4. Accuracy:

- The system's calculations and visualizations should accurately reflect the performance of the models being evaluated.
- It should avoid introducing biases or inaccuracies that could mislead users in their model selection process.

3.1.3 Software Requirements:

1. Operating System : Windows, Linux
2. Language : Python3
3. Open Source Software: Jupyter Notebook
4. Libraries: The following libraries are needed to be installed in the jupyter notebook.

matplotlib~=3.7.1 setuptools~=67.8.0
naiveautoml~=0.0.16 parameterized~=0.8.1
openml~=0.12.2 scipy~=1.11.1
numpy~=1.24.3 tqdm~=4.65.0
pandas~=1.5.3 pynisher~=1.0.8
scikit-learn~=1.3.0

3.1.4 Hardware Requirements:

- Processor - Intel or higher
- Speed – 2.4GHz
- RAM – 4GB (min)
- Hard Disk - 512 MB(min)

3.2 UML Diagrams for the project work:

UML, or Unified Modeling Language, serves as a contemporary method for modeling and documenting software, widely employed in business process modeling. It utilizes diagrammatic representations of software components to aid in understanding and identifying potential flaws or errors. These components can be interconnected in various ways to form complete diagrams, which are crucial for implementing knowledge in real-life systems.

While initially used primarily in software engineering, UML has expanded its scope to encompass the documentation of business processes and workflows. For instance, activity diagrams, a type of UML diagram, offer a standardized and feature-rich alternative to traditional flowcharts, enhancing readability and efficiency in modeling workflows.

In UML, use cases are identified by examining actors and defining their interactions with the system. Since one use case often cannot address all system needs, a collection of use cases is typically employed to cover all system functionalities. Associations, serving as pathways for communication, are fundamental in UML diagrams. They can link use cases, actors, classes, or interfaces, with unidirectional associations represented by arrows indicating the direction of communication.

The various UML diagrams are:

1. Use Case diagram
2. Activity diagram
3. Sequence diagram
4. Collaboration diagram
5. Object diagram
6. State chart diagram
7. Class diagram
8. Component diagram
9. Deployment diagram

3.2.1 System view diagrams:

Use Case Diagram :

A use case diagram is a graph of actors, a set of use cases enclosed by a system boundary, communication (participation) associations between the actors and users and generalization among use cases. The use case model defines the outside (actors) and inside (use case) of the system's behaviour. Actors are not part of the system. Actors represent anyone or anything that interacts with (input to or receive output from) the system. Use-case diagrams can be used during analysis to capture the system requirements and to understand how the system should work. During the design phase, you can use case diagrams to specify the behaviour of the system as implemented. Use case is a sequence of transactions performed by a system that yields a measurable result of values for a particular actor. The use cases are all the ways the system may be used.

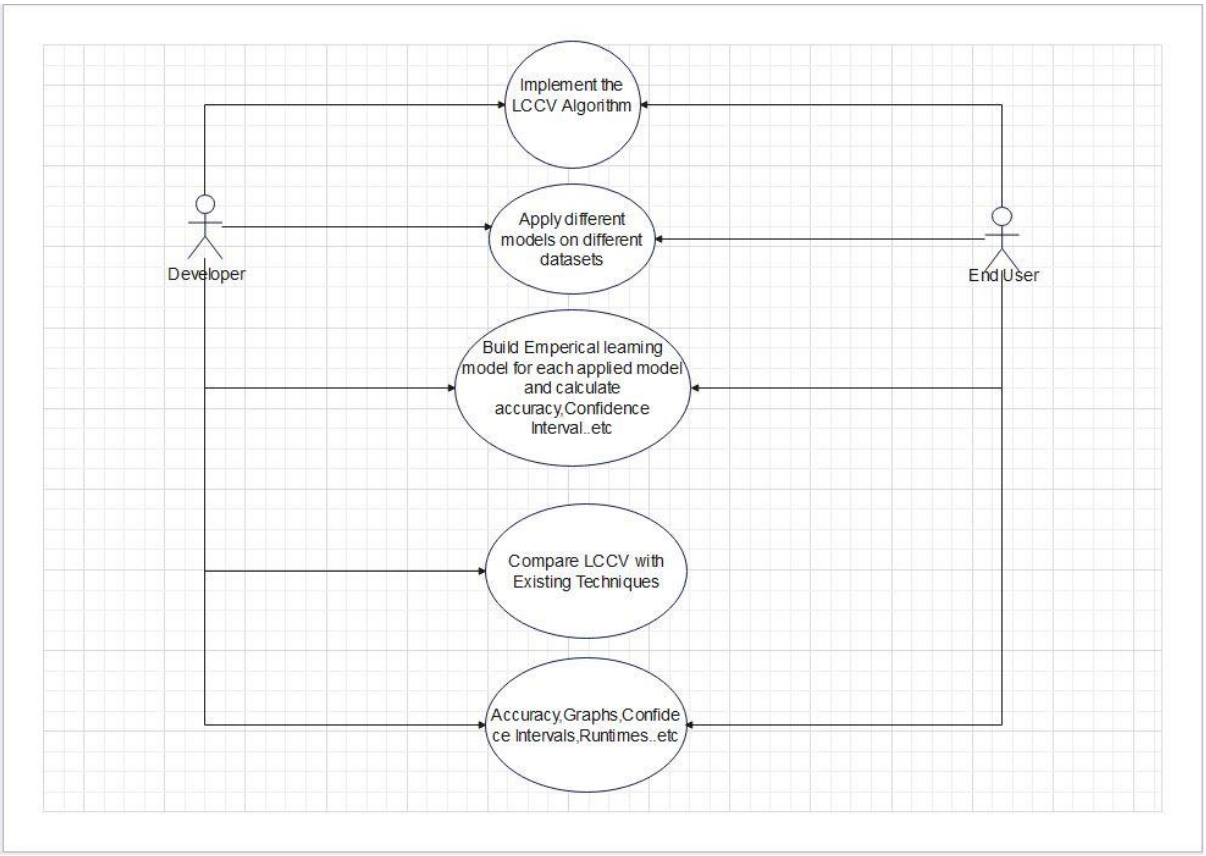


Fig. 3.1 Use Case Diagram

The Use case diagram depicts a developer and end user utilizing the Learning Curve Cross Validation (LCCV) technique to assess various machine learning models. LCCV involves applying different machine learning models to datasets, building secondary models to

measure their performance, and comparing these results to existing techniques. Finally, by evaluating accuracy graphs, confidence intervals, and runtimes, the developer or end user can determine the best model for their specific needs.

Activity Diagram:

An Activity diagram is a variation of a special case of a state machine, in which the states are activities representing the performance of operations and the transitions are triggered by the completion of the operations. The purpose of the Activity diagram is to provide a view of flows and what is going on inside a use case or among several classes. Activity diagrams contain activities, transitions between the activities, decision points, and synchronization bars.

The activity diagram below consist of the following steps:

- Loading the Dataset: The first step involves loading the dataset into the computer's memory. This dataset will be used to train the machine learning model.
- Splitting the Data into Features and Labels: The data is then split into features and labels. Features are the independent variables that the model will learn from, while labels are the dependent variables that the model will try to predict.
- Inputting the Machine Learning Model: The machine learning model is then inputted. This could involve selecting a pre-built model from a library or coding a custom model from scratch.
- Inputting the Threshold: A threshold is then inputted. This threshold is used to determine the performance of the model. For example, the threshold could be the minimum acceptable accuracy for the model.
- Applying LCCV: Next, LCCV, or Learning Curve Cross-Validation, is applied. LCCV is a technique used to evaluate the performance of machine learning models.
- Obtaining the Empirical Learning Model: After applying LCCV, an empirical learning model is obtained. This is a model that represents the performance of the original machine learning model on the dataset.

Obtaining the Accuracies and Confidence Intervals: Finally, the accuracies and confidence intervals are obtained for the empirical learning model. The accuracy is a measure of how well the model performs, while the confidence interval is a measure of how confident one can be in the accuracy.

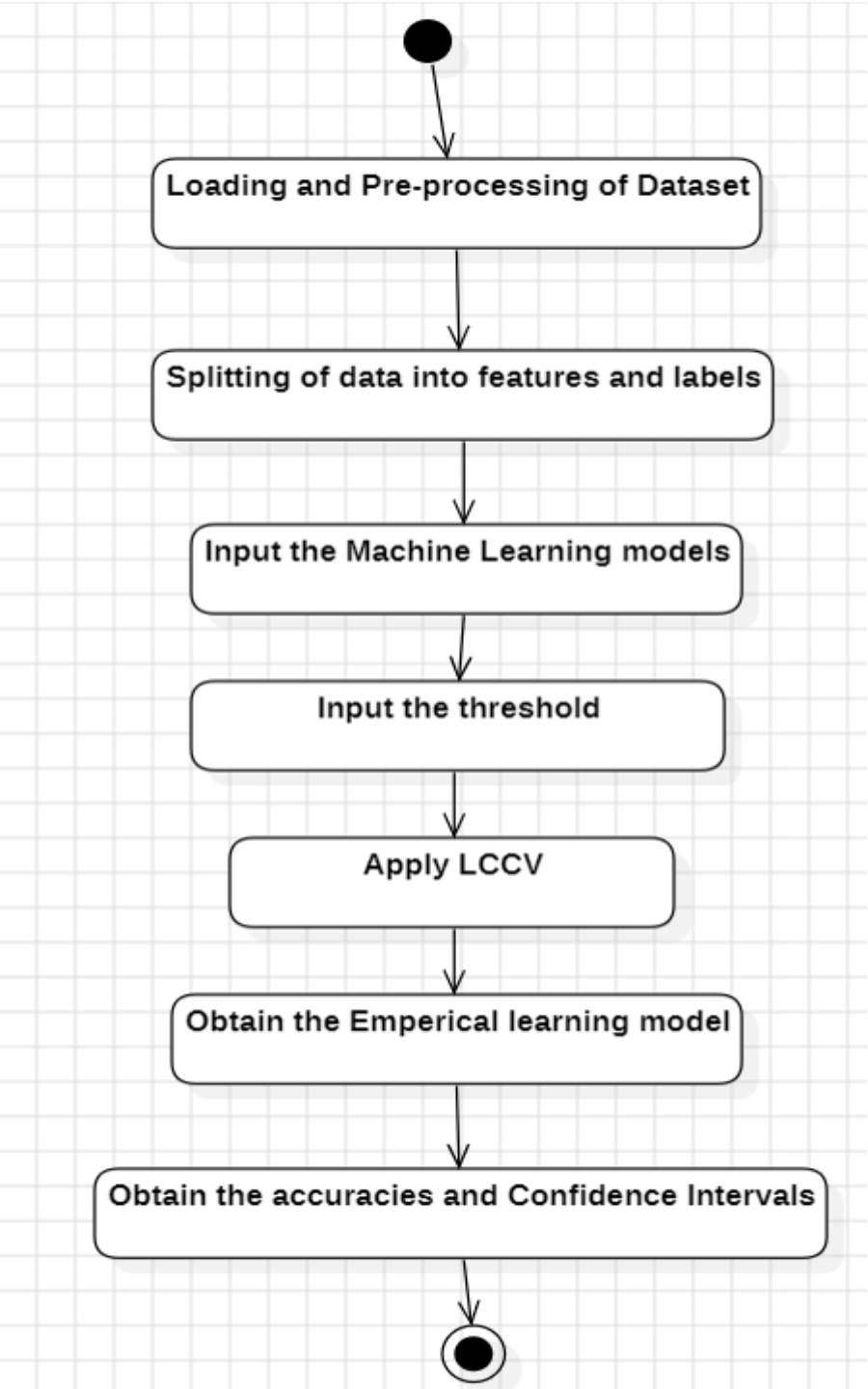


Fig. 3.2 Activity Diagram

Sequence Diagram:

A sequence diagram is an interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams.

The following are the Steps involved in the sequence diagram:

- Input the Dataset: The actor first provides the dataset to the system. This dataset contains the training data that the model will be trained on.
- Preprocess the Dataset (Optional): The system may preprocess the data. This preprocessing step may involve cleaning the data, handling missing values, and scaling the data.
- Split the Data into Features and Labels: The system splits the data into features and labels. Features are the independent variables that the model will learn from, while labels are the dependent variables that the model will try to predict.
- Input the Model and Parameters: The actor selects a machine learning model and provides the parameters for the model. The parameters are the hyperparameters that control the learning process of the model.
- Train the Model: The system trains the model on the training data. This involves the model learning the patterns in the data and fitting a model to those patterns.
- Evaluate the Model: The system evaluates the performance of the model on a separate validation set. This validation set is used to assess how well the model generalizes to unseen data.
- Loop (Optional): If the model performance is not satisfactory, the actor can go back and change the model parameters or preprocess the data differently. This loop can continue until the desired performance is achieved.
- Store the Model: Once the model is trained and evaluated, the system stores the model for future use.

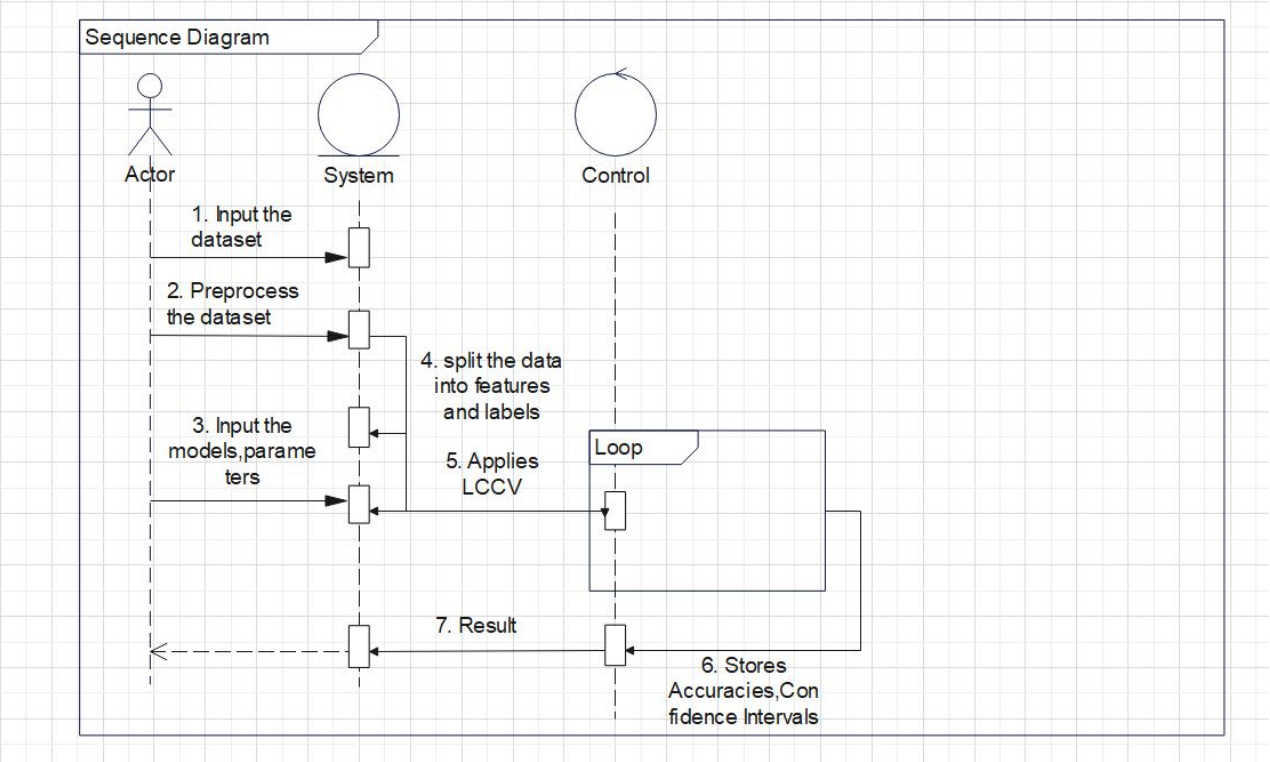


Fig. 3.3 Sequence Diagram

Collaboration Diagram:

A collaboration diagram shows the order of messages that implement an operation or a transaction. Collaboration diagrams show objects, their links, and their messages. They can also contain simple class instances and class utility instances. Each collaboration diagram provides a view of the interactions or structural relationships that occur between objects and object-like entities in the current model. Collaboration diagrams and sequence diagrams are called interaction diagrams.

The collaboration diagram below consist of the following components:

- User: The user initiates the process by inputting the dataset and the different machine learning models they want to consider. They may also set a threshold for acceptable performance.
- System: The system takes the user's input and performs the following actions:

Builds an empirical learning model for each candidate model. This involves using the LCCV

process to evaluate how the model performs on the dataset.

Iteratively evaluates the models on increasing data sizes and discards poorly performing models. This helps to identify the most competitive models.

Stores the accuracies and confidence intervals for each remaining model. These metrics indicate how well the model performed and how confident one can be in those results.

Outputs the results: Once the evaluation is complete, the system outputs the accuracies and confidence intervals for the remaining models. This information helps the user to select the best model for their specific needs.

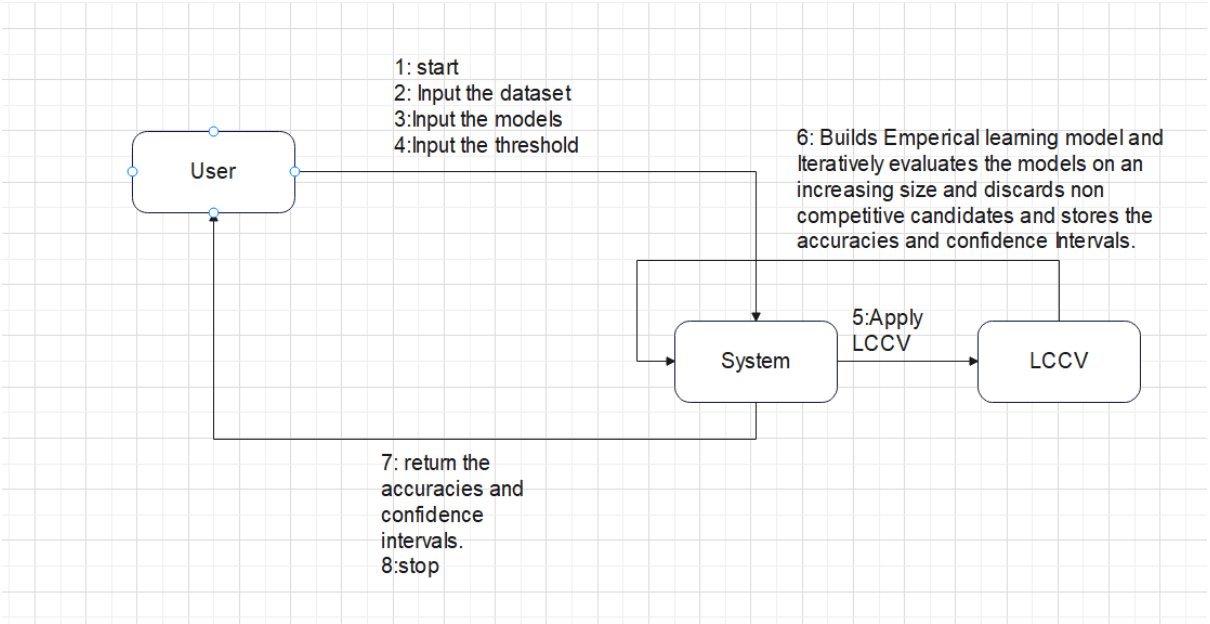


Fig. 3.4 Collaboration Diagram

Class Diagram:

Class diagrams contain icons representing classes, interfaces, and their relationships. You can create one or more class diagrams to represent the classes at the top level of the current model; such class diagrams are themselves contained by the top level of the current model. You can also create one or more class diagrams to represent classes contained by each package in your model; such class diagrams are themselves contained by the package enclosing the classes they represent; the icons representing logical packages and classes in class diagrams. In the UML, classes are represented as compartmentalized rectangles.

1. The top compartment contains the name of the class.
2. The middle compartment contains the structure of the class (attributes).
3. The bottom compartment contains the behaviour of the class (operations).

The below class diagram contains the following classes, interfaces and their relationships:

- User: This class represents the user of the system. The user is responsible for providing the dataset and the parameters for the machine learning model.
- System: This class represents the machine learning system itself. The system is responsible for loading the dataset, pre-processing the data, training the model, and evaluating the model.
- Empirical Learning Model: This class represents a model of the performance of a machine learning model on a dataset. It is likely created using LCCV (Learning Curve Cross Validation).
- Result: This class represents the results of the evaluation of a machine learning model. The results may include the accuracy of the model, the confidence interval of the model, and the timeout of the model.

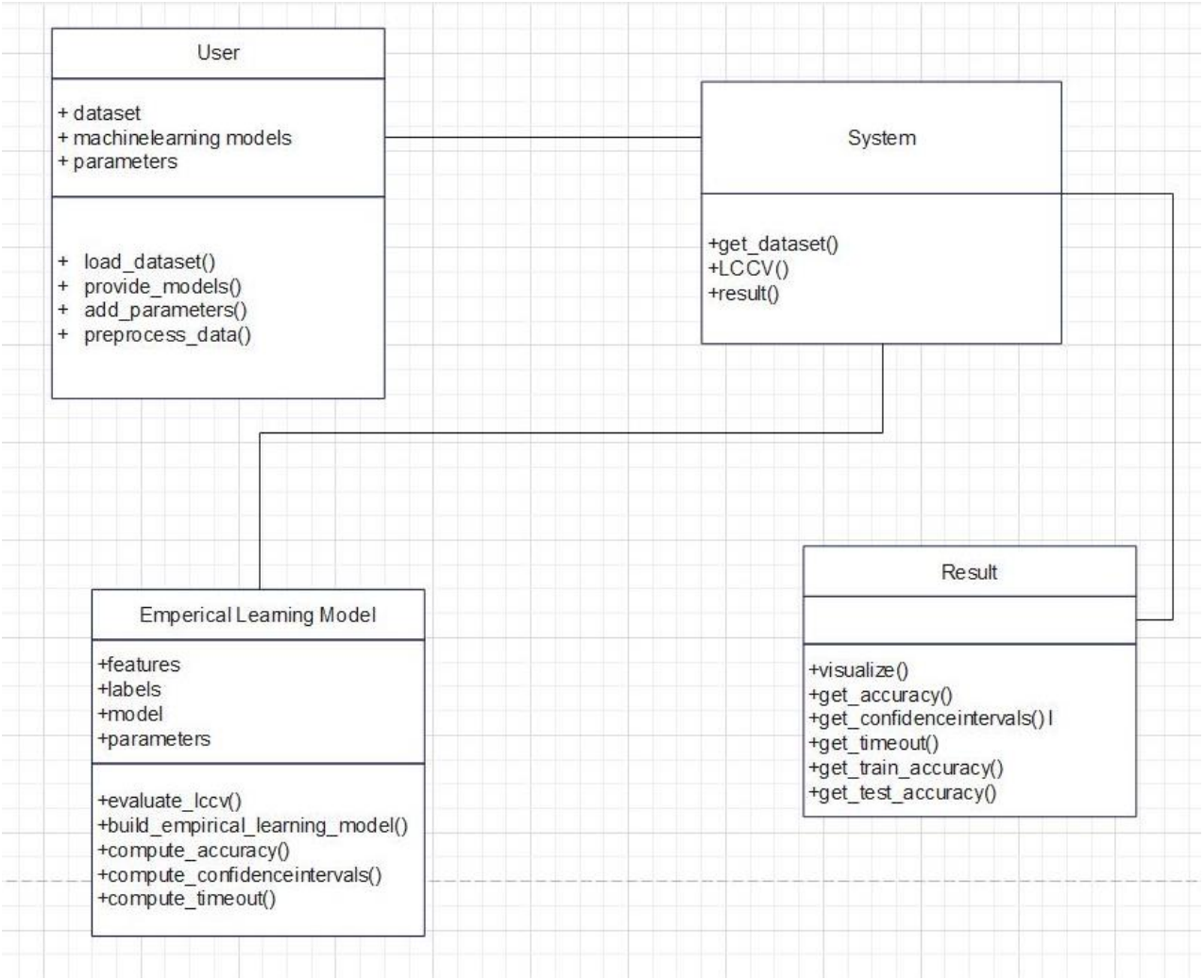


Fig- 3.5 Class Diagram

State Chart Diagram:

Use cases and scenarios provide a way to describe system behaviour; in the form of interaction between objects in the system. Sometimes it is necessary to consider the inside behaviour of an object. A state chart diagram shows the states of a single object, the events or messages that cause a transition from one state to another and the actions that result from a state change. As for the activity diagram, the state chart diagram also contains special symbols for start state and stop state. State chart diagrams cannot be created for every class in the system, it is only for those class objects with significant behaviour.

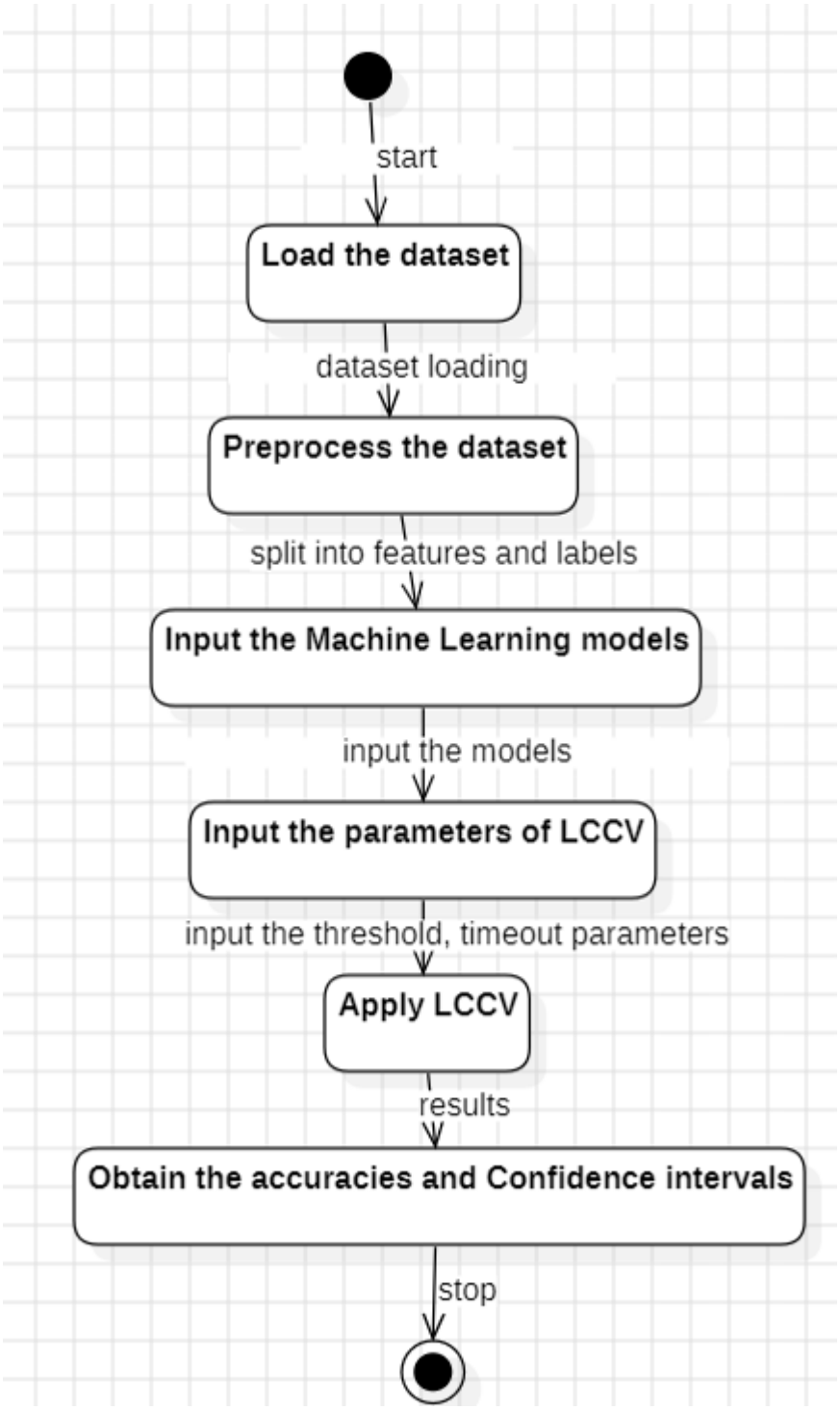


Fig. 3.6 State Diagram

The following are the states and their transitions of the above state chart diagram:

- Start: This is the initial state of the system, where the process begins.
- Load Dataset: The system transitions from the start state to this state, where it loads the dataset for training the machine learning model.

- Preprocess Dataset: After loading the dataset, the system moves to this state where it preprocesses the data. Preprocessing often involves cleaning the data, handling missing values, and scaling the data to ensure a smoother learning process for the model.
- Input the Machine Learning Models: Once the data preprocessing is complete, the system moves to this state where the machine learning models to be evaluated are incorporated.
- Input the Parameters of LCCV: Here, the user specifies the parameters for Learning Curve Cross Validation (LCCV), a technique used to assess the performance of models on a dataset.
- Apply LCCV: The system transitions to this state where it applies LCCV using the specified parameters on the loaded and preprocessed dataset, along with the inputted models. LCCV estimates how a model will perform based on its learning curve over different training data sizes.
- Obtain Results: After applying LCCV, the system moves to this state where the results are obtained. These results likely include metrics such as accuracy and confidence intervals for each evaluated model
- Obtain Accuracies and Confidence Intervals: In this state, the system extracts specific performance metrics, potentially accuracy and confidence intervals, from the obtained results in the prior state.
- Stop: This is the final state of the system, signifying the completion of the machine learning process where the model selection might occur based on the acquired accuracy and confidence intervals

3.2.2 Detailed view diagrams:

Component Diagram:

Component Diagrams show the dependencies between software components in the system. The nature of these dependencies will depend on the language or languages used for the development and may exist at compile-time or at runtime. In a large project there will be manyfiles that make up the system. These files will have dependencies on one another. The nature of these dependencies will depend on the language or languages used for the development and may exist at compile-time, at link-time or at run-time.

There are also dependencies between source code files and the executable files or bytecode files that are derived from them by compilation. Component diagrams are one of the two types of implementation diagrams in UML.

The following are the components of the below component diagram:

- User: This component represents the person who interacts with the system. The user is likely responsible for providing input such as data and selecting options.
- System: This component represents the computer system that performs the machine learning tasks. It likely consists of several sub-components that work together to complete these tasks.

The user and system components are connected with an undirectional arrow, indicating that they communicate with each other. This communication involves the user providing input to the system and the system providing output to the user.

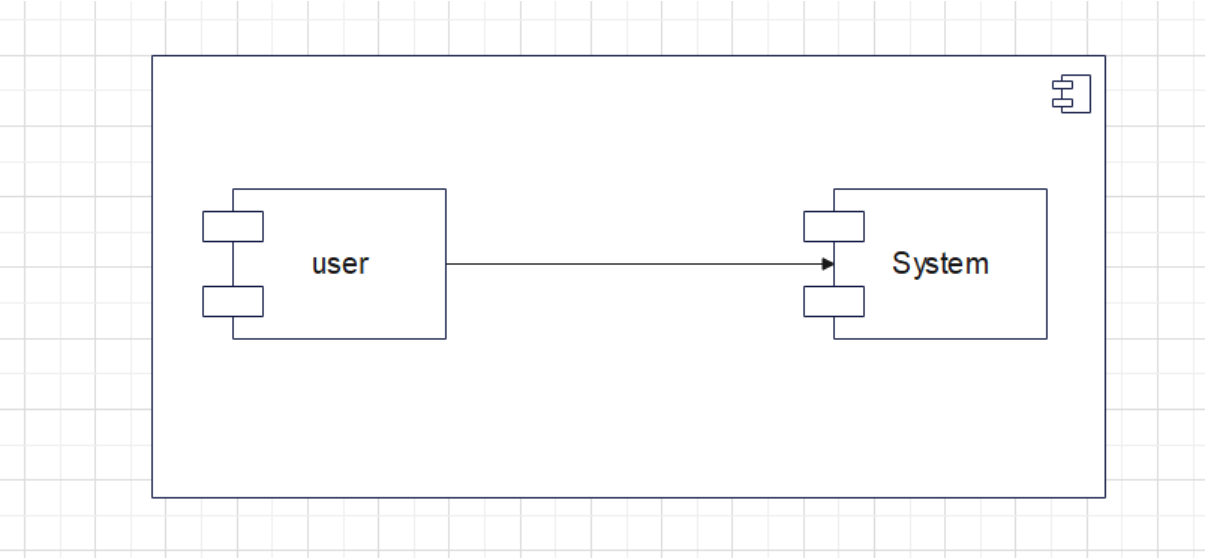


Fig. 3.7 Component Diagram

Deployment Diagram:

The second type of implementation diagram provided by UML is the deployment diagram. Deployment diagrams are used to show the configuration of run-time processing elements and the software components and processes that are located on them. Deployment diagrams are made up of nodes and communication associations. Nodes are typically used to show computers and the communication associations show the network and protocols that are used to communicate between nodes. Nodes can be used to show other processing resources such as people or mechanical resources. Nodes are drawn as 3D views of cubes or rectangular prisms, and the following figure shows a simplest deployment diagram where the nodes are connected by communication associations.

The Deployment Diagram below consist of the following nodes:

- User: This component represents the person who interacts with the system. The user is likely responsible for providing input such as data and selecting parameters.
- System: This component represents the computer system that performs the machine learning tasks.

The user and system components are connected with a solid line, indicating a physical connection, likely a wire.

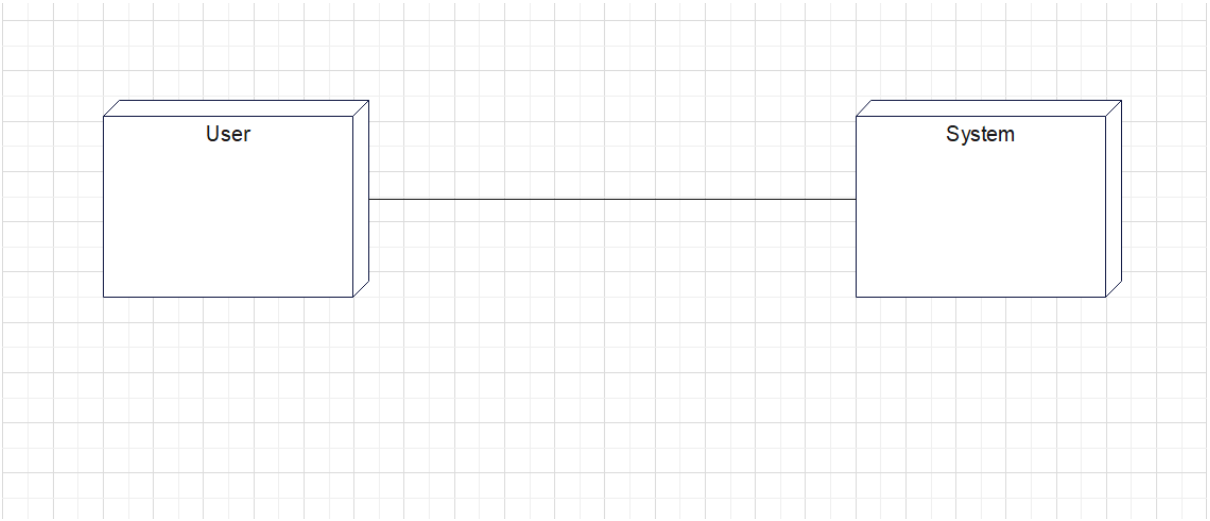


Fig. 3.8 Deployment Diagram

4. SYSTEM DESIGN

4.1 Architecture of Proposed System

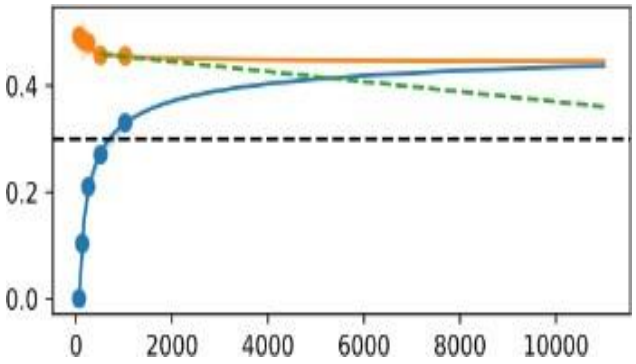


Fig. 4.1 Example of the pruning of LCCV

From the above figure, black dashed line represents threshold value r. the orange dots represents performance observations on validation per anchor, and the orange curve is an MMF-model fitted through these points. The blue dots represents performance observations onthe train set per anchor, and the blue curve is a fitted MMF model. The dashed green line represents the optimistic extrapolation from the last anchors on the validation curve. If this optimistic extrapolation does not improve over the threshold value r, it is unlikely that the candidate will improve, and thus the evaluation of the learner can be stopped early.

Learning-curve-based cross-validation (LCCV) estimates the performance of a learner based on empirical learning curves. LCCV evaluates the learner on ascendingly ordered so-called anchor (points) S = (s_1, .., s_T), which are training set sizes. Several of such evaluations are made on each anchor to ensure a robust performance estimation. The idea is to early recognize whether or not a candidate can be competitive [21].

The Architecture of the LCCV is Divided into 3 main categories:

- How to use the convexity of the learning curve
- Details how to make sure that this process only discards candidates that are sub- optimal with high probability.
- Explains how to use a learning curve model (in this case the Morgan-Mercer-Flodin (MMF)) to skip anchors and directly jump to the final anchor s_T for presumably competitive candidates.

4.2 Workflow of the proposed system:

The following modules are to be implemented:

1. Aborting Based On Validation.
2. Aborting Based on Training.
3. Calculating the Confidence Intervals.
4. Construction of the Curve and Skipping the Intermediate Values.

4.3 Module Description:

4.3.1 Aborting based on Validation Curve:

Convexity is important because it implies that the slope of the learning curve is an increasing function that, for increasing training data sizes, [4] approximates 0 from below.

Learning curve as a function p(s), where p is the true average performance ofthe learner when trained with a train set of size s.

Convexity of p implies that p' (s) ≤p '(s') if s'> s, where p' is the derivative of p.
Of course, as a learning curve is a sequence, it is not differentiable in the strict sense, rather sloppily refer to its slope p'.
The problem is of course that one can compute neither p(s) nor p'(s), because it refers to the true learning curve. That is, p realizes for the hypotheses produced by a learner on different sample sizes. So one can only try to estimate the values of p and p 0 at selected anchors. Given a sequence of anchors s_1, .., s_t for which performance values of a learner were already observed,$p(s_t)$ and $p'(s_t)$ have to be estimated in order to determine whether the learner can be pruned based on the optimistic extrapolation of the learning curve. A canonical estimate of

$p(s_t)$ is the empirical mean $\hat{\mu}^t$ of the observations at s_t. For the slope, one could rely on these estimates anduse convexity to bound the slope with

$$p'(s_t) = p(s_t) - p(s_t - 1) \overset{conv}{\geq} \frac{p(s_t) - p(s_{t-1})}{s_t - s_{t-1}}$$
$$\approx \frac{\hat{\mu}_t - \hat{\mu}_{t-1}}{s_t - s_{t-1}} \quad (3) \cdot$$

However, replacing p' in the above decision rule by this bound is admissible only if the estimates $\hat{\mu}^t$ and $\hat{\mu}^{t-1}$ are accurate (enough). This calls for the usage of confidence intervals toobtain a notion of probabilities [4], [7], [19].

4.3.1 Aborting Based on Training Curve:

Early discarding can also be based on the performance of the model evaluated on the training data. The following assumptions, which are commonsense in machine learning literature.

- The error on the train performances typically mono-tonically increases as the number of observations increases.
- The train performance at any anchor is typically better than the validation performanceat any anchor.

The consequence is that, once the measured train error is already higher than the reference performance r (which is measured based on validation performance), it is unlikely that this learner will improve over r , and the evaluation can therefore be aborted. This criterion can be considered once the observations at each anchor are complete and the best training performance observed at each anchor is considered. If this value is worse than r at any anchor, the candidateis pruned.

4.3.2 Calibrating with Confidence Intervals:

In order to calculate the confidence interval atleast two observations are needed to obtain the standard deviation [13]. The number of samples per anchor is a factor that is under our control [16]. This should be determined carefully as setting this value too low might result in inaccurate or large confidence intervals while setting this value too high might unnecessarily slow down the validation process.

The behaviour of LCCV can here be controlled via four parameters. The first and the second parameters are simply the minimum and the maximum number of evaluations at each anchor, respectively. A minimum of other than 2 can make sense if one either want to allow greediness (1) or to be a bit more conservative about the reliability of small samples (> 2). To enable a dynamic approach between the two extremes, stability criterion ε is considered. As soon as the confidence interval width drops under ε, the observation set is considered to be stable.

The slopes between the anchors are considered and identified whether it is reaches the stability criterion E.

$$p(s_T) \geq \inf C_t - (s_T - s_t)\frac{\sup C_{t-1} - inf C_t}{s_{t-1} - s_t} \qquad (4)$$

Let C_1, C_t be the confidence intervals at the t ascendingly ordered anchors.,Let σ_t be the slopes between the anchors. ,Let u_t be the minimal value that a convex-compatible curve can take at a given anchor t. ,Now at each anchor i.,Starting with $\sigma_t = 0$, one can inductively definean observation set.

$$u_{i-1} = \min\{\, x \in C_{i-1} \mid x \geq u_i - (s_i - s_{i-1})\sigma_i \,\} \text{ and}$$

$$\sigma_{i-1} = \frac{u_i - u_{i-1}}{s_i - s_{i-1}} \qquad (5)$$

If an observation set is not convexity compatible, it needs to be “repaired” before making decisions on pruning based on these results [23]. This repair technique is to simply step back one anchor. Stepping back from anchor s_t to s_{t-1},.

4.3.3 Skipping Intermediate Values:

Once a learner's performance becomes competitive and reaches a stable anchor score with a small confidence interval, it may be reasonable to evaluate the learner on the full dataset.This decision is based on the premise that evaluating the learner on the full dataset will not cause the loss of relevant candidates.

To estimate the performance of the learner on the full dataset, the Morgan-Mercer-Flodin(MMF) model is proposed.

By fitting the MMF model using nonlinear regression with observations from at least fouranchor points, the model parameters can be determined.

The learner's performance on the full dataset is satisfactory, and the evaluation process can beimmediately skipped to the final anchor point.

5. IMPLEMENTATION

5.1 Algorithm:

	Algorithm 1: LCCV : Learning Curve Cross Validation
1.	$(s_1,\ldots,s_T) \leftarrow$initialize anchor points from min_exp and data size
2.	$(O_1,\ldots,O_T) \leftarrow (\emptyset,\ldots,\emptyset)$ // sets of performance observations
3.	$(C_1,\ldots,C_T) \leftarrow$ initialize confidence intervals as [0,1] each
4.	$t \leftarrow 1$
5.	**while** $t<=T \wedge (\sup C_T - \inf C_T > \varepsilon) \wedge \lvert O_T \rvert < n$ **do**
6.	repair_convexity ← false
7.	/* gather samples at current anchor point s_t */
8.	**while** $\sup C_T - \inf C_T > \varepsilon \wedge \lvert O_T \rvert < n \wedge \sim$ repair_convexity **do**
9.	run classifier on a new train-validation split for anchor s_t
10.	training points, and add performance observations to O_t
11.	update confidence interval C_t based on O_t
12.	**if** $t > 1$ **then** $\sigma_{t-1} = (\sup C_{t-1} - \inf C_t) / (s_{t-1} - s_t)$
13.	**if** $t > 2 \wedge \sigma_{t-1} < \sigma_{t-2} \wedge \lvert O_{t-1} \rvert < n$ **then**
14.	repair_convexity ←true
15.	/* Decide how to proceed from this anchor */
16.	**if** average train error > r **then**
17.	/* Not applicable for tree-based learners */
18.	**return** $\perp$
19.	**else if** repair_convexity **then**
20.	$t \leftarrow t - 1$
21.	**else if** projected bound for s_t is > r **then**
22.	**return** $\perp$
23.	**else if** $r=1 \vee (t \geq 4 \wedge$ MMF_ESTIMATE$(s_t \leq r)$ **then**
24.	$t \leftarrow T$
25.	**else**
26.	$t \leftarrow t + 1$
27.	**return** $<$ mean(C_T), $(C_1,\ldots C_T)>$

Table 5.1 Algorithm

The LCCV algorithm is sketched in Alg. 1(Fig 5.1.1) and puts all of the above steps together. It initiates variables for the various anchor points (l. 1), the performance observations per anchor (l. 2, one set per anchor) and the various confidence intervals. Additionally, t represents the index of the anchor that is being evaluated. The algorithm iterates over anchors in a fixed schedule S (in ascending order) [17]. At anchor s (lines 7-12), the learner is validated by drawing folds of training size s, training the learner on them, and computing its predictive performanceon data not in those folds. These validations are repeated until a stopping criterion is met . In the pseudo-code, n represents the maximum number of evaluations per anchor. Each anchor also has a minimum number of evaluations, but this is left out for readability. Then it is checkedwhether the observations are currently compatible with a convex learning curve model. If this is not the case, the algorithm steps back one anchor and gathers more observations to get a better picture and "repair" the currently non-convex view (l. 15); details are described in Sec. 4.2. Otherwise, a decision is taken with respect to the four possible interpretations of the currentlearning curve. First, if the train performance at this anchor is already worse than the reference score r , one can safely assume that the validation error will not improve over this, and the configuration can be pruned. In fact, this can be done before the convexity is repaired (l. 13). An exception to this are the tree-based learners, as explained in Section 4.1. Second, if it can be inferred that the validation performance at s_T will not be competitive with the best observedperformance r , LCCV returns $\perp$ (l. 17, cf. Sec. 4.1). Third, if extrapolation of the learning curve gives rise to the belief that the learner is competitive, then LCCV directly jumps to validation on the full dataset of the candidate (l. 19, cf. Sec. 4.3); the condition $t \geq 4$ is requiredbecause so many points are necessary to fit the MMF model. In any other case, just keep evaluating at the next anchor (l. 21). In the end, the estimate for s_T is returned together with the confidence intervals for all anchors.

For the given model the LCCV builds the Empirical Learning model [12], [13], [16], [19]. The Empirical Learning Model class serves as a framework for evaluating machine learning models empirically, allowing for flexible configuration of evaluation parameters and scoring metrics.

If an explicit evaluator function is not provided, the constructor partitions the input data into training and testing sets. This step ensures that the evaluation process proceeds smoothly, even without an externally defined evaluator. Additionally, it performs basic data validation

checks, such as ensuring that the input dataset contains a positive number of instances.

At its core, the class constructor initializes various attributes crucial for empirical learning, such as the learner itself (learner), the input data (X and y), the number of target instances (n_target), a seed for randomization (seed), settings for cross-validation folds an evaluator function (evaluator), scoring metrics (base_scoring and additional_scorings), and options for error handling (raise_errors). This initialization phase sets up the groundwork for subsequent evaluation tasks.

The evaluate method is responsible for conducting the actual model evaluation. It first prepares the training and testing data based on the specified configuration, including handling fixed train-test folds or dynamically partitioning the data. Notably, it incorporates a mechanism for random seed management to ensure reproducibility across evaluation runs.

During the evaluation process, the method measures various performance metrics for the trained model, including both training and testing scores for each specified scoring metric. It employs timeout mechanisms to prevent evaluation tasks from exceeding predefined time limits, safeguarding against potential resource overuse or stalled evaluations.

5.2 Datasets used:

The meta-features of the datasets are summarized in below figure and detailed in Table

1. Of these datasets, 5 have at least 100k instances, 18 have at least 50k instances, 31 have at least 10k instances, and 42 have at least 5000 instances. Even datasets with a small number ofinstances are not necessarily easy to perform algorithm selection on, since they can have manyattributes, which are even more crucial for runtime . Examples are gina prior, amazon commerce reviews, micro-mass, Biosphere, Internet-Advertisement, and others. The 13 datasets on which a memory limit of 60GB instead of 20GB was applied were: 1475, 4135, 4541, 23512, 23517, 40996, 41147, 41162, 41165, 41167, 41169, 42732, 42734.

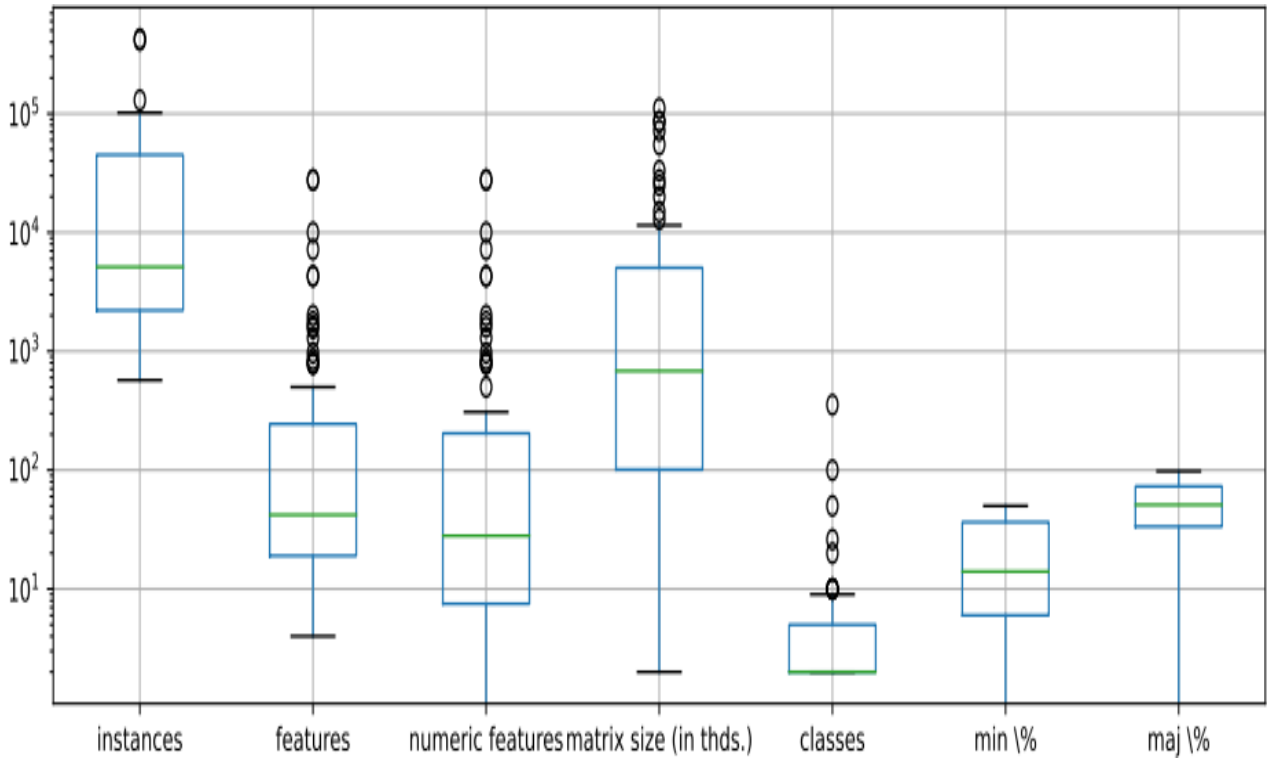


Fig. 5.1 Meta features of Data Set

```
In [3]: from scipy.io import arff
        import pandas as pd

        data = arff.lo('C:\\Users\\Downloads\\eucalyptus.mat')
        df = pd.DataFrame(data[0])

        df
```

Out[3]:

	Abbrev	Rep	Locality	Map_Ref	Latitude	Altitude	Rainfall	Frosts	Year	Sp	PMCno	DBH	Ht	Surv	Vig	Ins_res	Stem_F
0	b'Cra'	1.0	b'Central_Hawkes_Bay'	b'N135_382/137'	b'39__38'	100.0	850.0	-2.0	1980.0	b'co'	1520.0	18.45	9.96	40.0	4.0	3.0	
1	b'Cra'	1.0	b'Central_Hawkes_Bay'	b'N135_382/137'	b'39__38'	100.0	850.0	-2.0	1980.0	b'fr'	1487.0	13.15	9.65	90.0	4.5	4.0	
2	b'Cra'	1.0	b'Central_Hawkes_Bay'	b'N135_382/137'	b'39__38'	100.0	850.0	-2.0	1980.0	b'ma'	1362.0	10.32	6.50	50.0	2.3	2.5	
3	b'Cra'	1.0	b'Central_Hawkes_Bay'	b'N135_382/137'	b'39__38'	100.0	850.0	-2.0	1980.0	b'nd'	1596.0	14.80	9.48	70.0	3.7	3.0	
4	b'Cra'	1.0	b'Central_Hawkes_Bay'	b'N135_382/137'	b'39__38'	100.0	850.0	-2.0	1980.0	b'ni'	2088.0	14.50	10.78	90.0	4.0	2.7	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
731	b'WSh'	1.0	b'Southern_Hawkes_Bay'	b'N151_922/226'	b'40__36'	100.0	1250.0	-2.0	1983.0	b'fa'	2548.0	41.63	12.64	28.0	4.2	3.2	
732	b'WSh'	1.0	b'Southern_Hawkes_Bay'	b'N151_922/226'	b'40__36'	100.0	1250.0	-2.0	1983.0	b'fr'	2552.0	33.35	10.61	33.0	4.5	4.0	
733	b'WSh'	1.0	b'Southern_Hawkes_Bay'	b'N151_922/226'	b'40__36'	100.0	1250.0	-2.0	1983.0	b'ni'	2568.0	28.21	9.47	94.0	4.6	3.0	
734	b'WSh'	1.0	b'Southern_Hawkes_Bay'	b'N151_922/226'	b'40__36'	100.0	1250.0	-2.0	1983.0	b'ob'	1522.0	27.36	11.49	67.0	4.7	3.3	
735	b'WSh'	1.0	b'Southern_Hawkes_Bay'	b'N151_922/226'	b'40__36'	100.0	1250.0	-2.0	1983.0	b're'	NaN	22.34	8.84	25.0	3.8	3.5	

736 rows × 20 columns

Fig. 5.2 Sample Data Set

The Data Sets are of format ARFF (Attribute-Relation File Format) is a file format used to represent datasets in machine learning and data mining. The ARFF file format has two main sections: the header and the data.

The header contains meta-information about the dataset, such as the name of the relation (dataset), a list of attributes, and any additional comments. The data section comes after the header and contains the actual records of the dataset.

Each line in the data section corresponds to one data instance (example) of the dataset.ARFF files are easy to read and write, making them a popular choice for sharing datasets across different machine learning platforms and tools.

openmlid	name	instances	features	numeric features	matrix size (in thds.)	classes	min %	maj %
3	kr-vs-kp	3196	36	0	115	2	47	52
6	letter	20000	16	16	320	26	3	4
11	balance-scale	625	4	4	2	3	7	46
12	mfeat-factors	2000	216	216	432	10	10	10
23	cmc	1473	9	2	13	3	22	42
30	page-blocks	5473	10	10	54	5	0	89
31	credit-g	1000	20	7	20	2	30	70
54	vehicle	846	18	18	15	4	23	25
60	waveform-5000	5000	40	40	200	3	33	33
181	yeast	1484	8	8	11	10	0	31
188	eucalyptus	736	19	14	13	5	14	29
1042	gina_prior	3468	784	784	2718	2	49	50
1049	pc4	1458	37	37	53	2	12	87
1067	kc1	2109	21	21	44	2	15	84
1111	KDDCup09_appetency	50000	230	192	11500	2	1	98
1457	amazon-commerce-revi	1500	10000	10000	15000	50	2	2
1461	bank-marketing	45211	16	7	723	2	11	88
1464	blood-transfusion-se	748	4	4	2	2	23	76
1468	cnae-9	1080	856	856	924	9	11	11
1475	first-order-theorem-	6118	51	51	312	6	7	41
1477	gas-drift-different-	13910	129	129	1794	6	11	21
1479	hill-valley	1212	100	100	121	2	50	50
1485	madelon	2600	500	500	1300	2	50	50
1486	nomao	34465	118	89	4066	2	28	71
1487	ozone-level-8hr	2534	72	72	182	2	6	93
1489	phoneme	5404	5	5	27	2	29	70
1494	qsar-biodeg	1055	41	41	43	2	33	66
1515	micro-mass	571	1300	1300	742	20	1	10
1590	adult	48842	14	6	683	2	23	76
1591	connect-4	67557	126	127	8512	3	9	65
4134	Bioresponse	3751	1776	1776	6661	2	45	54
4135	Amazon_employee_acce	32769	9	0	294	2	5	94
4534	PhishingWebsites	11055	30	0	331	2	44	55
4538	GesturePhaseSegmenta	9873	32	32	315	5	10	29
4541	Diabetes130US	101766	49	13	4986	3	11	53
23512	higgs	98050	28	28	2745	2	47	52
23517	numerai28.6	96320	21	21	2022	2	49	50
40498	wine-quality-white	4898	11	11	53	7	0	44
40668	connect-4	67557	42	0	2837	3	9	65
40670	dna	3186	180	0	573	3	24	51
40685	shuttle	58000	9	9	522	7	0	78
40701	churn	5000	20	16	100	2	14	85
40900	Satellite	5100	36	36	183	2	1	98
40975	car	1728	6	0	10	4	3	70
40978	Internet-Advertiseme	3279	1558	3	5108	2	13	86
40981	Australian	690	14	6	9	2	44	55
40982	steel-plates-fault	1941	27	27	52	7	2	34
40983	wilt	4839	5	5	24	2	5	94
40984	segment	2310	19	19	43	7	14	14
40996	Fashion-MNIST	70000	784	784	54880	10	10	10
41027	jungle_chess_2pcs_ra	44819	6	6	268	3	9	51
41138	APSFailure	76000	170	170	12920	2	1	98
41142	christine	5418	1636	1599	8863	2	50	50
41143	jasmine	2984	144	8	429	2	50	50
41144	madeline	3140	259	259	813	2	49	50
41145	philippine	5832	308	308	1796	2	50	50
41146	sylvine	5124	20	20	102	2	50	50
41147	albert	425240	78	26	33168	2	50	50
41150	MiniBooNE	130064	50	50	6503	2	28	71
41156	ada	4147	48	48	199	2	24	75
41158	gina	3153	970	970	3058	2	49	50
41159	guillermo	20000	4296	4296	85920	2	40	59
41161	riccardo	20000	4296	4296	85920	2	25	75
41162	kick	72983	32	14	2335	2	12	87
41163	dilbert	10000	2000	2000	20000	5	19	20
41164	fabert	8237	800	800	6589	7	6	23
41165	robert	10000	7200	7200	72000	10	9	10
41166	volkert	58310	180	180	10495	10	2	21
41167	dionis	416188	60	60	24971	355	0	0
41168	jannis	83733	54	54	4521	4	2	46
41169	helena	65196	27	27	1760	100	0	6
42733	Click_prediction_sma	39948	11	5	439	2	16	83
42734	okcupid-stem	50789	19	2	964	3	9	71
42809	kits	1000	27648	27648	27648	2	48	52
42810	PCam	4000	27648	27648	110592	2	49	50

Table-5.2 Table Of Datasets

5.3 Metrics Calculated:

In the context of machine learning model evaluation, "metrics" refer to quantitative measures used to assess the performance of the model. These metrics provide insights into how well the model is performing in terms of its predictive capabilities, generalization ability, and other relevant criteria.

The Empirical Learning Model class calculates several metrics to assess the performance of machine learning models. These metrics include both training and testing scores for each specified scoring metric. Let's break down the key metrics calculated during the evaluation process:

1. Training Scores (score_train_{scoring}):
- These metrics evaluate the performance of the model on the training data. They provide insights into how well the model fits the training data and its ability to capture patterns within the training set.
2. Testing Scores (score_test_{scoring}):
- These metrics measure the performance of the model on unseen testing data. They indicate how well the model generalizes to new, unseen instances and provide an estimate of its predictive accuracy in real-world scenarios.
3. Scoring Time for Training (scoretime_train_{scoring}):
- This metric represents the time taken to compute the training scores for each specified scoring metric. It measures the computational overhead associated with evaluating the model's performance on the training data.
4. Scoring Time for Testing (scoretime_test_{scoring}):
- Similar to scoring time for training, this metric denotes the time taken to compute the testing scores for each specified scoring metric. It captures the computational effort required to assess the model's performance on the testing data.

These metrics are computed for each specified scoring metric, including both the base scoring metric (base_scoring) and any additional scoring metrics provided. The class allows for flexibility in defining custom scoring functions, enabling users to evaluate models based on a wide range of performance criteria.

The LCCV function primarily focuses on estimating the performance of a machine learning model using learning curves and iterative evaluation. While it doesn't directly calculate traditional metrics like accuracy, precision, recall, or F1-score, it does provide estimates of the model's performance under different conditions.

Here's how it assesses model performance:

1. Target Performance Estimate:

- The function returns an estimate of the model's performance, typically represented as the mean performance at the maximum anchor point or the predicted performance at a specified target anchor point. This estimate serves as the primary metric indicating the model's expected performance under given conditions.

2. Confidence Intervals:

- It calculates confidence intervals for the estimated performance at each anchor point. These intervals represent the uncertainty associated with the performance estimates and provide insights into the reliability of the estimates.

3. Learning Curve Trends:

- The function analyzes trends in the learning curve, such as convexity violations or performance improvements, to optimize the evaluation process. It identifies instances where the learning curve deviates from expected convex behavior, indicating potential issues with the model's performance or the evaluation process.

4. Slope Ranges:

- It computes slope ranges for segments of the learning curve, which are used to detect convexity violations and guide the evaluation process. By analyzing changes in slope, the function identifies points where the learning curve may need adjustments or further evaluations.

5. Performance Estimates at Anchor Points:

- The function provides performance estimates at specific anchor points defined by the schedule parameter. These estimates indicate how well the model performs at different sample sizes and guide the decision-making process during evaluation.

6. Runtime Information:

- It logs runtime information for each evaluation step, including the time taken to compute performance estimates and the overall runtime of the evaluation process. This information helps assess the efficiency of the evaluation method and identify potential bottlenecks.

5.4 Methods Compared:

Three sequential model validation techniques are compared.

- CV (Cross-Validation).
- Wilcoxon Signed Rank Test.
- Successive Halving
- LCCV (Learning Curve Cross Validation).

1. CV(Cross-validation):

Cross-validation is a technique used to evaluate the performance of a model by splitting the available data into training and testing subsets. The goal is to estimate how well the model will perform on new, unseen data. Some of the techniques of CV are: K-fold Cross Validation / Vanilla Cross Validation and Monte Carlo Cross Validation

a. K-Fold Cross-Validation: K-fold cross-validation is a technique used in machine learning to evaluate the performance of a model and estimate its generalization error. It involves splitting the dataset into K subsets, or folds, where K is typically set to 5 or 10.

b. Monte Carlo cross-validation: Monte Carlo cross-validation is a technique used in machine learning to evaluate the performance of a model using random sampling. It involves randomly selecting a subset of the data for training and validation multiple times and averaging the results to estimate the model's generalization error.

2. Wilcoxon Signed Rank Test:

The Wilcoxon Signed Rank Test assesses differences between paired samples without assuming a normal distribution. It's commonly used in machine learning to compare the performance of algorithms or models. By comparing performance metrics from paired experiments, it determines whether one method consistently outperforms another, providing insights into the significance of observed differences in performance [11].

3. Successive Halving:

Successive Halving in one type of validation method which uses principle of elimination. This algorithm is an iterative one which iteratively trains a set of models on fixed parameters and develop the fraction data and then selecting top performance models [6].

At each iteration, the fraction of data used to train the models is increased, and the number of models evaluated is reduced. This allows for more efficient use of computational resources,

as the number of models evaluated decreases exponentially with each iteration.

4. LCCV (Proposed Method):

Learning-curve-based cross-validation (LCCV) is a new approach for validation based on learning curves (LCCV). Instead of evaluating a candidate just for one training set size, it is evaluated, in increasing order, at different so-called anchors of training fold sizes, e.g. for 64, 128, 256, 512, ... instances. At each anchor, several evaluations are conducted until a stability criterion is met. The LCCV idea is early recognize whether or not a candidate can be competitive. It makes the evaluation mechanism faster while at the same time not compromising the performance of the algorithm selection procedure.

6. TESTING

6.1 Introduction Of Testing:

Software testing is defined as an activity to check whether the actual results match the expected results and to ensure that the software system is Defect-free. It involves the execution of a software component or system component to evaluate one or more properties of interest. It is required for evaluating the system. This phase is the critical phase of software quality assurance and presents the ultimate view of coding.

6.1.1 Importance of Testing:

The importance of software testing is imperative. A lot of times this process is skipped, therefore, the product and business might suffer. To understand the importance of testing, here are some key points to explain.

- Software Testing saves money
- Provides Security
- Improves Product Quality
- Customer satisfaction

Testing is done in different ways The main idea behind the testing is to reduce the errors and do it with minimum time and effort.

6.2 Objective Of Testing:

Testing is a fault detection technique that tries to create failure and erroneous states in a planned way. This allows the developer to detect failures in the system before it is released to the customer.

Here the functionality of the LCCV is going to be tested using the following tests:

The main objective of these tests is to ensure the correctness, reliability, and robustness of the Learning Curve-based Cross-Validation (LCCV) method implementation. Specifically, the tests aim to:

- Verify Data Partitioning: Ensure that the function responsible for partitioning data into training and testing sets (_partition_train_test_data()) works correctly, produces partitions of the expected sizes, and maintains reproducibility across different random seeds.
- Validate Core Functionality: Confirm that the main function lccv() executes without errors, behaves syntactically well, and produces the desired results when applied to a simple classification problem using the Iris dataset. This includes checking if the function produces valid outcomes for different anchor sizes and correctly logs information using a

DataFrame.

- Test Timeout Handling: Validate whether the LCCV method respects the timeout constraints provided during its execution, ensuring that it terminates within the specified time limit, even under various configurations involving different preprocessors, learners, and datasets.
- Evaluate Real Case Scenario: Examine the behavior of the LCCV method in a practical scenario involving real datasets and different classifiers. This test assesses if the LCCV method adheres to expected stopping criteria and produces meaningful results when applied to more complex classification tasks.

6.3 Test Code:

```
import logging
import numpy as np
import sklearn.datasets
import sklearn.tree
import lccv
import unittest
import pandas as pd
import time
import sklearn.pipeline
import sklearn.base
import itertools as it
from parameterized import parameterized
import openml
from sklearn.dummy import DummyClassifier



def get_dataset(openmlid):
    ds = openml.datasets.get_dataset(openmlid, download_data=True)
    df = ds.get_data()[0].dropna()
    y = df[ds.default_target_attribute].values
    X = pd.get_dummies(df[[c for c in df.columns if c !=
ds.default_target_attribute]]).values.astype(float)
```

```
    return X, y


class TestLccv(unittest.TestCase):
    preprocessors = [None]#, sklearn.preprocessing.RobustScaler,
sklearn.kernel_approximation.RBFSampler]

    learners = [sklearn.tree.DecisionTreeClassifier]#sklearn.svm.LinearSVC,
sklearn.tree.ExtraTreeClassifier, sklearn.linear_model.LogisticRegression,
sklearn.linear_model.PassiveAggressiveClassifier, sklearn.linear_model.Perceptron,
sklearn.linear_model.RidgeClassifier, sklearn.linear_model.SGDClassifier,
sklearn.neural_network.MLPClassifier,
sklearn.discriminant_analysis.LinearDiscriminantAnalysis,
sklearn.discriminant_analysis.QuadraticDiscriminantAnalysis,
sklearn.naive_bayes.BernoulliNB, sklearn.naive_bayes.MultinomialNB,
sklearn.neighbors.KNeighborsClassifier, sklearn.ensemble.ExtraTreesClassifier,
sklearn.ensemble.RandomForestClassifier, sklearn.ensemble.GradientBoostingClassifier,
sklearn.ensemble.GradientBoostingClassifier,
sklearn.ensemble.HistGradientBoostingClassifier]

    @staticmethod
    def setUpClass():
        print("Setting up the test suite...")

        # setup logger for this test suite
        print("Setting up logging...")
        lccv_logger = logging.getLogger("lccv")
        lccv_logger.setLevel(logging.INFO)
        elm_logger = logging.getLogger("elm")
        elm_logger.setLevel(logging.WARN)

    def setUp(self):
        print("Setting up the test case...")
        self.lccv_logger = logging.getLogger("lccv")
```

```
#Test 01:
    def test_partition_train_test_data(self):

        """
            tests whether the partitions created with the LCCV partition function are propert and reproducible.
        :return:
        """

        print("Start Test on Partitioning")
        features, labels = sklearn.datasets.load_iris(return_X_y=True)
        for seed in [0, 1, 2, 3, 4, 5]:
            print(f"Run test for seed {seed}")
            n_te = 32
            n_tr = 150 - n_te
            f_tr, l_tr, f_te, l_te = lccv._partition_train_test_data(
                features, labels, n_te, seed)

            # check correct sizes
            assert f_tr.shape == (n_tr, 4)
            assert l_tr.shape == (n_tr, )
            assert f_te.shape == (n_te, 4)
            assert l_te.shape == (n_te, )

            # check reproducibility for same seed
            f_tr2, l_tr2, f_te2, l_te2 = lccv._partition_train_test_data(
                features, labels, n_te, seed)
            np.testing.assert_array_equal(f_te, f_te2)
            np.testing.assert_array_equal(l_te, l_te2)
            np.testing.assert_array_equal(f_tr, f_tr2)
            np.testing.assert_array_equal(l_tr, l_tr2)
            print(f"Finished test for seed {seed}")
```

```
# Test 02:
    def test_lccv_normal_function(self):
        '''
            Just test whether the function
                * runs through successfully,
                * syntactically behaves well, and
                * produces (syntactically) the desired results.
        '''
        features, labels = sklearn.datasets.load_iris(return_X_y=True)
        learner = sklearn.tree.DecisionTreeClassifier(random_state=42)
        print(f"Starting test of LCCV on {learner.__class__.__name__}")
        _, _, res, elm = lccv.lccv(
            learner,
            features,
            labels,
            r=0.0,
            base=2,
            min_exp=4,
            enforce_all_anchor_evaluations=True,
            logger=self.lccv_logger
        )
        assert set(res.keys()) == {16, 32, 64, 128, 135}
        for key, val in res.items():
            print(f"Key: {key}, Val: {val}")
            assert not np.isnan(val['conf'][0])
            assert not np.isnan(val['conf'][1])
        assert isinstance(elm.df, pd.DataFrame)
        print(f"Finished test of LCCV on {learner.__class__.__name__}")


#Test 03
    @parameterized.expand(list(it.product(preprocessors, learners, [(61, 0.0), (1485, 0.2)])))
    def test_lccv_respects_timeouts(self, preprocessor, learner, dataset):
        """
        This checks whether LCCV respects the timeout
```

```
        """
        X, y = get_dataset(dataset[0])
        r = dataset[1]
        print(f"Start Test LCCV when running with r={r} on dataset {dataset[0]} with preprocessor {preprocessor} and learner {learner}")

        # configure pipeline
        steps = []
        if preprocessor is not None:
            pp = preprocessor()
            if "copy" in pp.get_params().keys():
                pp = preprocessor(copy=False)
            steps.append(("pp", pp))
        learner_inst = learner()
        if "warm_start" in learner_inst.get_params().keys(): # active warm starting if available, because this can cause problems.
            learner_inst = learner(warm_start=True)
        steps.append(("predictor", learner_inst))
        pl = sklearn.pipeline.Pipeline(steps)

        timeout = 3

        # do tests
        try:

            # run 80lccv
            print("Running 80LCCV")
            start = time.time()
            score_80lccv = lccv.lccv(sklearn.base.clone(pl), X, y, r=r, target_anchor=.8, max_evaluations=5, timeout=timeout)[0]
            end = time.time()
            runtime_80lccv = end - start
            print(f"80LCCV Runtime: {runtime_80lccv}s")
            self.assertTrue(runtime_80lccv <= timeout, msg=f"Permitted runtime exceeded.
```

```
Permitted was {timeout}s but true runtime was {runtime_80lccv}")
        except ValueError:
            print("Skipping case in which training is not possible!")


#Test 04
    def test_lccv_pruning_in_real_case(self):
        X, y = get_dataset(737)
        learners = [
            sklearn.tree.DecisionTreeClassifier(random_state=42),
            sklearn.dummy.DummyClassifier()
        ]

        r = -np.inf
        for i, learner in enumerate(learners):
            print(f"Testing {learner.__class__.__name__}")
            score, _, res, _ = lccv.lccv(
                learner,
                X,
                y,
                r=r,
                base=2,
                min_exp=4,
                logger=self.lccv_logger,
                use_train_curve=False
            )
            if score >= r:
                r = np.max([score, r])

            if i == 1:  # majority classifier should stop early
                print("Majority Classifier:")
                self.assertTrue(
                    max(res.keys()) <= 1024,
                    f"Biggest anchor evaluated by majority classifier should be 1024 but is
{max(res.keys())}"
```

```
            )
            for key, val in res.items():
                print(f"Key: {key}, Val: {val}")
                self.assertFalse(np.isnan(val['conf'][0]))
                self.assertFalse(np.isnan(val['conf'][1]))

        if i == 1:  # XT-Forest should jump early
            print("XT-Forest Classifier:")
            self.assertTrue(
                max(res.keys()) <= 1024,
                f"Biggest anchor evaluated by majority classifier should be 1024 but is
{max(res.keys())}"
            )
            for key, val in res.items():
                print(f"Key: {key}, Val: {val}")
                self.assertFalse(np.isnan(val['conf'][0]))
                self.assertFalse(np.isnan(val['conf'][1]))

#Test 05
def test_lccv_pruning_in_theoretic_cases(self):
    features, labels = sklearn.datasets.load_digits(return_X_y=True)
    learner = sklearn.tree.DecisionTreeClassifier(random_state=42)
    logger = None  # replace None with your logger object
    print(f"Starting test of LCCV on {learner.__class__.__name__}")

    r = 2.0  # This threshold is not reachable (in terms of accuracy), so there should be
pruning

    _, _, res, _ = lccv.lccv(
        learner,
        features,
        labels,
        r=r,
        base=2,
```

```
            min_exp=4,
            enforce_all_anchor_evaluations=True,
            logger=logger,
            use_train_curve=False
        )
        assert set(res.keys()) == {16, 32, 64, 128, 256}
        for key, val in res.items():
            print(f"Key: {key}, Val: {val}")
            assert not np.isnan(val['conf'][0])
            assert not np.isnan(val['conf'][1])
        print(f"Finished test of LCCV on {learner.__class__.__name__}")

if __name__ == '__main__':
    unittest.main()
```

6.4 TEST CASES:

Test Case ID	01
Test Scenario	Testing partition train-test data.
Test Case	Verify proper and reproducible partitions.
Test Steps	Load iris dataset, iterate over different seeds, partition data.
Test Data	Iris dataset (features and labels).
Expected Result	Proper and reproducible partitions.
Actual Result	Proper and reproducible partitions.
Status (Pass/Fail)	Pass

Table 6.1 Table for Test Case 01

Test Case ID	02
Test Scenario	Testing LCCV normal functionality.
Test Case	Verify LCCV method functionality.
Test Steps	Load iris dataset, apply LCCV method with Decision Tree Classifier.
Test Data	Iris dataset (features and labels).
Expected Result	Successful LCCV execution with valid results.
Actual Result	Successful LCCV execution with valid results.
Status (Pass/Fail)	Pass

Table 6.2 Table for Test Case 02

Test Case ID	03
Test Scenario	Testing LCCV with respect to timeouts
Test Case	Ensure LCCV method respects specified timeouts
Test Steps	1. Get dataset using OpenML ID. 2. Set timeout for each execution. 3. Run LCCV method with timeout specified. 4. Check if the actual runtime does not exceed the specified timeout.
Test Data	OpenML dataset (features and labels)
Expected Result	LCCV method respects specified timeouts
Actual Result	LCCV method respects specified timeouts
Status (Pass/Fail)	Pass

Table 6.3 Table for Test Case 03

Test Case ID	04
Test Scenario	Testing lccv pruning in real case
Test Case	Evaluate behavior of LCCV method in real-case scenario
Test Steps	1. Get dataset using OpenML ID. 2. Test with different classifiers: DecisionTreeClassifier and DummyClassifier 3. Evaluate if the LCCV method behaves as expected in terms of stopping criteria and maximum anchor evaluated.
Test Data	OpenML dataset (features and labels)
Expected Result	Expected behavior of LCCV method in real-case scenario
Actual Result	Expected behavior of LCCV method in real-case scenario
Status (Pass/Fail)	Pass

Table 6.4 Table for Test Case 04

Test Case ID	05
Test Scenario	Testing lccv pruning in theoretical case
Test Case	Evaluate behavior of LCCV method in real-case scenario
Test Steps	1. Get dataset using OpenML ID. 2. Test with different classifiers: DecisionTreeClassifier 3. Evaluate if the LCCV method behaves as expected in terms of stopping criteria and maximum anchor evaluated.
Test Data	Load_digits dataset
Expected Result	Expected behavior of LCCV method in theoretic-case scenario
Status (Pass/Fail)	Pass

Table 6.5 Table for Test Case 05

6.5 TEST RESULT:

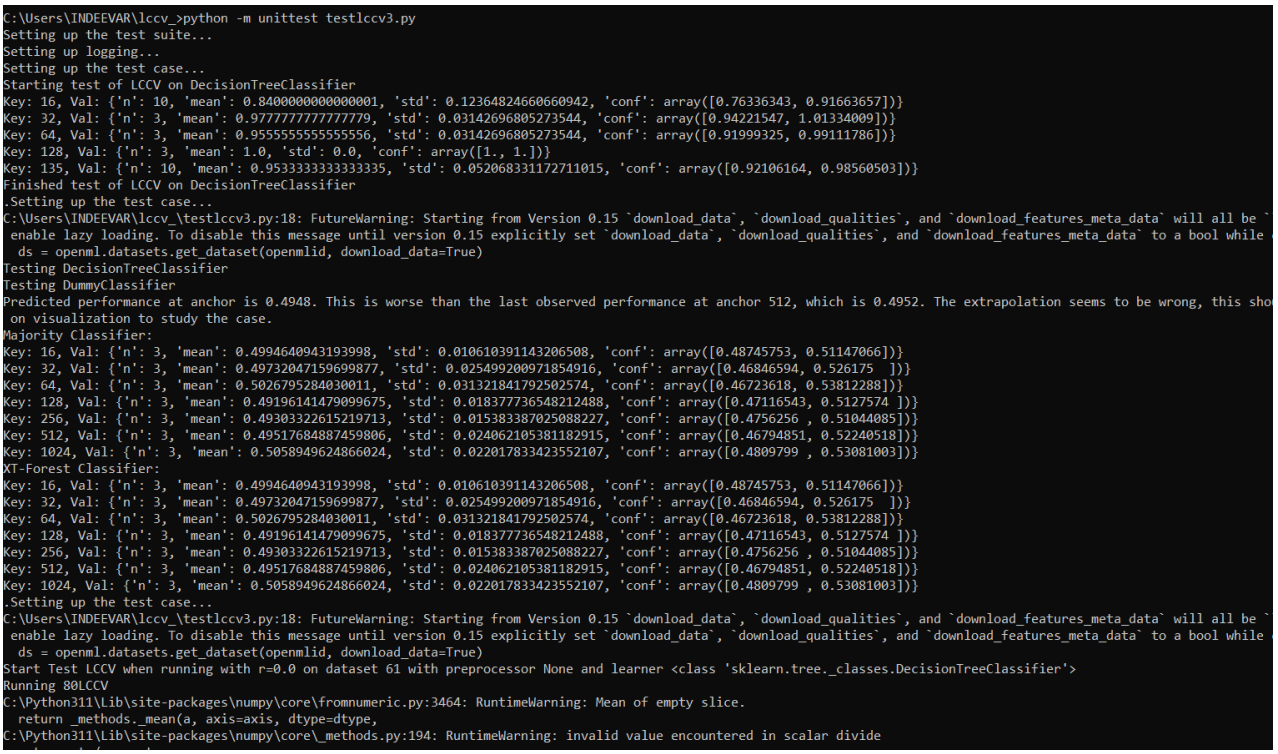


Fig. 6.1 Test Results-1

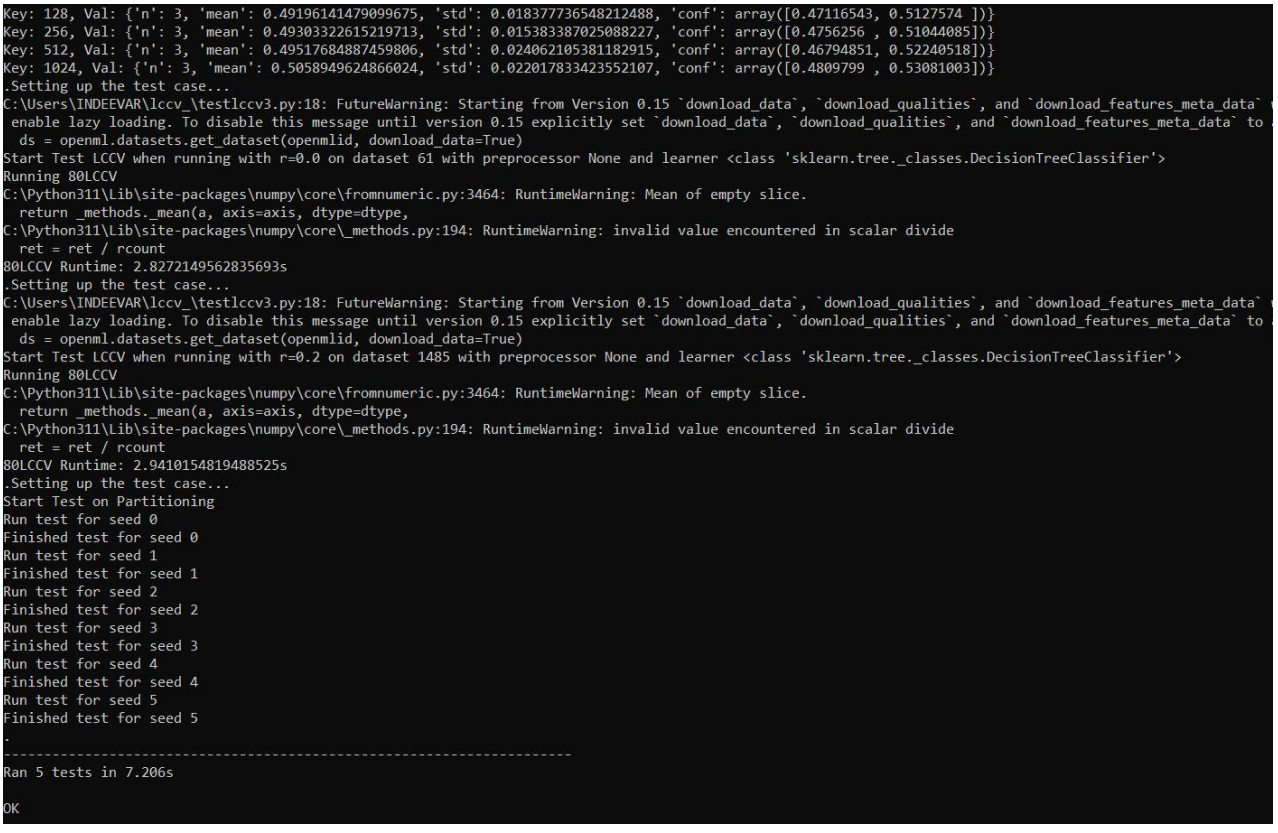


Fig. 6.2 Test Results-2

7. RESULT ANALYSIS

7.1 Result Evaluation:

The Result Evaluation mainly addresses the following:

- Comparison of Runtime Performance:

The Runtime performance of a model selection process employing Learning Curve Cross Validation (LCCV) is compared to 5-fold Cross-Validation(5-CV) and 10-fold Cross-Validation(10-CV)

- Utilizing Empirical Learning Curves:

The empirical learning curves collected are examined whether they can be used to make accurate recommendations regarding more data would improve the overall result.

7.1.1 LCCV For Model Selection:

The evaluation involves an AutoML tool that performs a random search on 256 classification pipelines from the scikit-learn library using various validation techniques to assess candidate performance. The pipelines consist of classification algorithms, with optionalpreprocessing algorithms [17].

The three sequential model validation techniques compared are as follows:

- CV (Cross-Validation): The candidate assessment is done using the same technique (CV) for validation.
- Wilcoxon: A racing-inspired variant of CV, where candidates are tested on different data samples, and those that can be statistically shown to not beat the current baseline are discardedusing a Wilcoxon signed rank test (p-value 0.05).
- LCCV (Learning-Curve Cross-Validation): Candidates are assessed using a customized CV technique called LCCV, with variations 80-LCCV and 90-LCCV based on the percentageα of available data used for training.

Additionally, the evaluation includes a technique called Successive Halving (SH) that performs its own model selection process. SH halves the candidate population in each round while increasing the evaluation budget for surviving candidates.

The runtime of each method on a dataset is the average total runtime of the random search using that validation method. To account for variation, the experiments are run over 10 seeds, generating different classifier portfolios to be validated by the techniques. The average overall runtime is computed. The chosen learner's performance is assessed using an exhaustive MCCV (Monte Carlo Cross-Validation). This involves forming 100 bipartitions of the data and using

90% for training and 10% for testing in each repetition. The average error rate over these partitions is used as the validation score.

The experiments are conducted on 75 datasets, offering a diverse range of instances and attributes, and avoiding label leakage in the preprocessing phase [5].

A timeout of 30 minutes per validation is set to interrupt the validation process and obtain partial results if needed. The focus is on comparing the effectiveness of the validation techniques in model selection [16], [17].

7.1.2 Parametrization Of LCCV:

The specific parameters for the LCCV technique are set used in the experiments. These parameter choices were made to strike a balance between confident extrapolation and performance [22]. Here are the key parameter settings:

- Minimum Number of Samples per Anchor: Set to 3, which is considered a fair trade-off for confident extrapolation.
- Maximum Number of Samples per Anchor: Aligned with the CV number of samples, i.e., 5 and 10 for the respective cases.
- Minimum Exponent ρ: Set to 6, ensuring training over at least 64 instances to be in line with successive halving and previous experiments.
- Maximum Width Parameter ε for Confidence Intervals: For the last anchor (maximum exponent): $\varepsilon_{max} = 0.001$. For intermediate anchors: $\varepsilon = 0.1$

The choice of ε max and ε allows some uncertainty at intermediate anchors, mitigated by the conservative extrapolation methodology, while tolerating less uncertainty at the final anchor.

7.1.3 Result:

By Running the above models on 13 large datasets the following factors are considered for result evaluation.

- Runtime Calculations.
- Performance Gaps

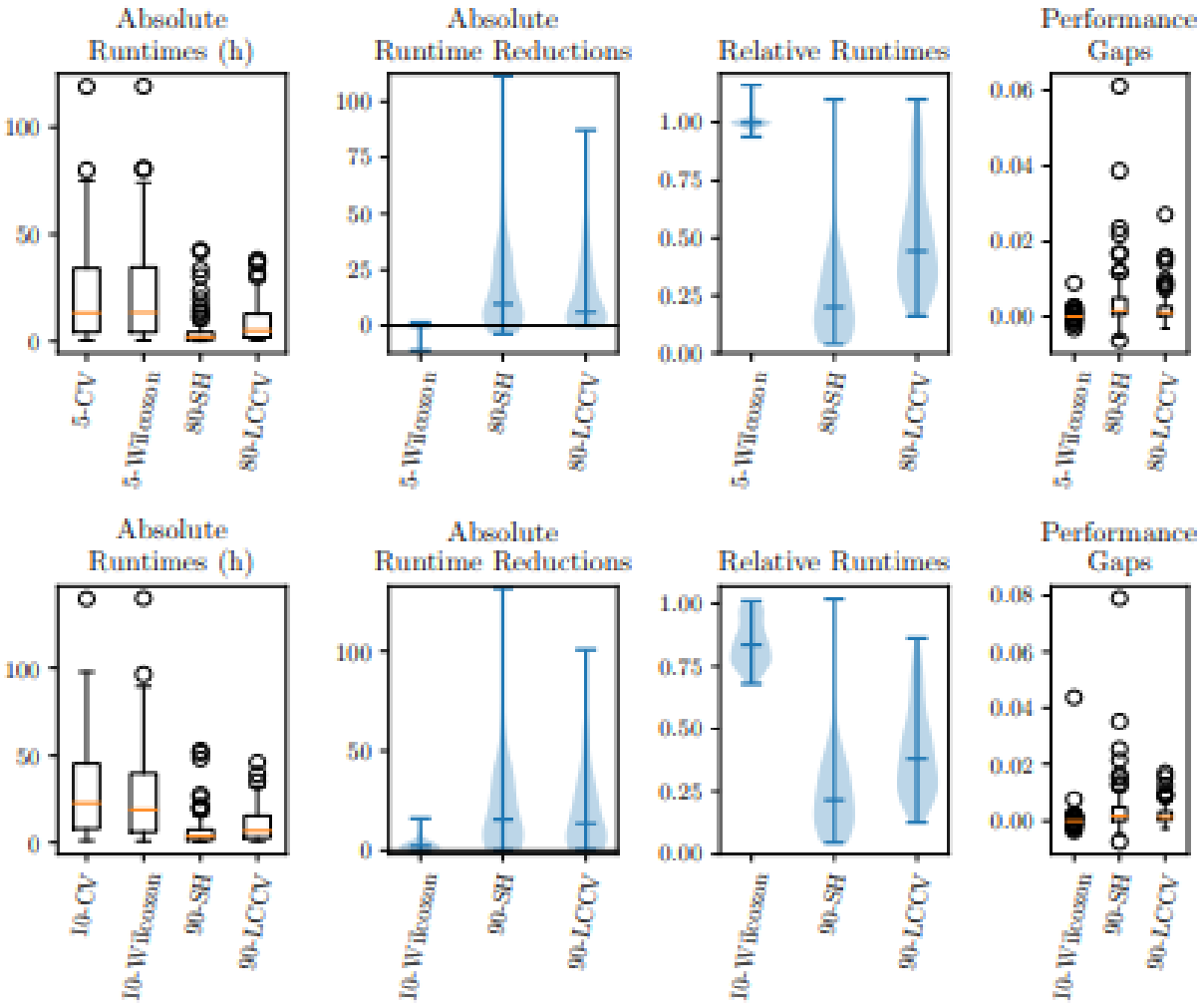


Fig. 7.1 Comparison between LCCV ,vanilla cross validation a racing implementation and successivehalving on 75 datasets

The results of the output show that LCCV is a fast and informative model selection method. Itcan achieve similar performance to CV while substantially reducing the runtime. LCCV is particularly strong on datasets with high training runtimes. However, LCCV does not always achieve the optimal performance.

Here are some of the key findings:

- LCCV can achieve a median runtime reduction of 7 to 20 hours compared to CV.
- LCCV is particularly strong on datasets with high training runtimes.
- LCCV does not always achieve the optimal performance, but the difference in performance is usually small.

The output also provides some insights into how LCCV works. LCCV iteratively increases the number of instances used for training. In the context of model selection, it discards models that are very unlikely to become competitive. This allows LCCV to quickly identify the most promising models and focus on training them.

The output concludes that LCCV is a promising model selection method. It is fast, informative, and can achieve similar performance to CV on most datasets. However, LCCV does not always achieve the optimal performance, so it is important to consider the specific application when choosing a model selection method.

Here are some of the limitations of the output:

- The output was conducted on a limited set of datasets.
- The output did not consider all possible model selection methods.

Despite these limitations, the output provides valuable insights into LCCV and its potential benefits. The output also suggests that LCCV is a promising method for model selection in a variety of applications.

7.1.4. LCCV for Data Acquisition Recommendations:

The claim that LCCV gives more in-sights than CV in the data collection process is justified. The produced empirical learning curves are used to derive a power law model with which the question of whether more instances will lead to sufficiently better results or not is answered. Of course, in practice, one is not interested in any marginal improvement at any cost but will require a reasonable ratio between additional instances and improvement.

A binary decision approach is proposed to address the question: whether to double the available instances or not. A threshold (β) is introduced that expresses the minimum performance improvement required to make accepting additional data advantageous. The study focuses on the inverse power law (IPL) to model learning curves, and they fit the IPL function to the empirical curves produced by LCCV [15].

The Model seek to answer two research questions:

i. Regression Problem: Can the Model predict the performance improvement [21] (in terms ofregression) of individual learners and the portfolio as a whole when doubling the available data?

ii. Binary Classification Problem: Can the Model predict whether a certain performance improvement β can be achieved (binary classification) for individual learners and the portfolioas a whole when doubling the available data, for various values of β [25].

By answering these questions, one aim to provide insights into how well the IPL model can beused to predict the benefits of increasing the dataset size for machine learning models and whether it is beneficial to do so for both individual learners and a portfolio of learners [12] , [13]. This study helps understand the potential return on investment when increasing data, considering the trade-off between improved performance and additional costs.

In this evaluation, the performance of a recommender is aimed to assess for predicting whether doubling the number of instances in a dataset will yield an improvement of at least β for a portfolio of classifiers. To achieve this, use a dataset d and divide it into two subsets: d_{full} and d_{half}.

The sizes of these subsets are chosen such that using 90% of the data for training corresponds to the highest power of 2 possible for the train set of the dataset.Using only the data from d_{half}, All classifiers in the portfolio are validated using LCCV without cancellation and skipping, ensuring full evaluations at all anchors. This process is computationally efficient and provides information about the performance of the algorithms on the validation data.

And to make predictions about whether doubling the dataset size will yield an improvement ofat least β, the authors build an Inverse Power Law (IPL) model for each classifier in the portfolio using non-linear regression. This model, called the IPL-regressor, is then used to predict the performance of each algorithm on the full dataset d_{full}, denoted as $\hat{P}^{i}_{d\,full.}$

To determine whether the performance improvement is sufficient (i.e., $P^{*}_{d\,half} - P^{*}_{dfull} \geq \beta$), the authors use the IPL-classifier with threshold β. They compute $\hat{P}^{*}_{dfull}$ as the minimum of $\hat{P}^{i}\,d_{full}$ across all classifiers and return "yes" if $\hat{P}^{*}_{dfull}$ is at most $P^{*}_{dhalf} - \beta$ and "no" otherwise.

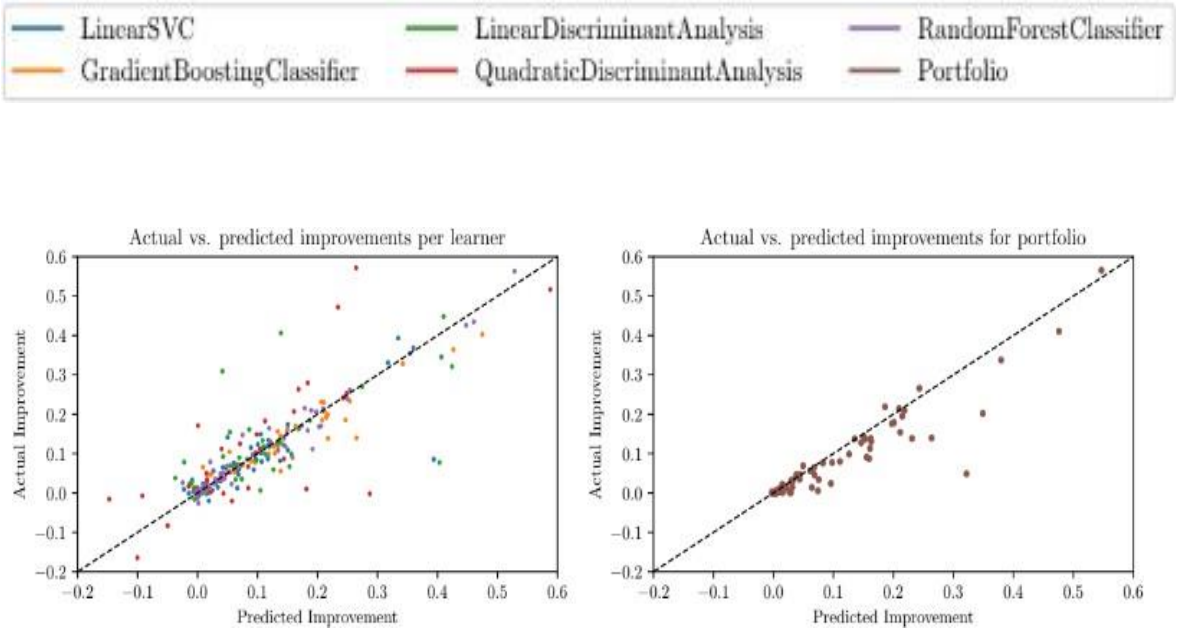


Fig. 7.2 Predicted vs. actual improvement when doubling the data

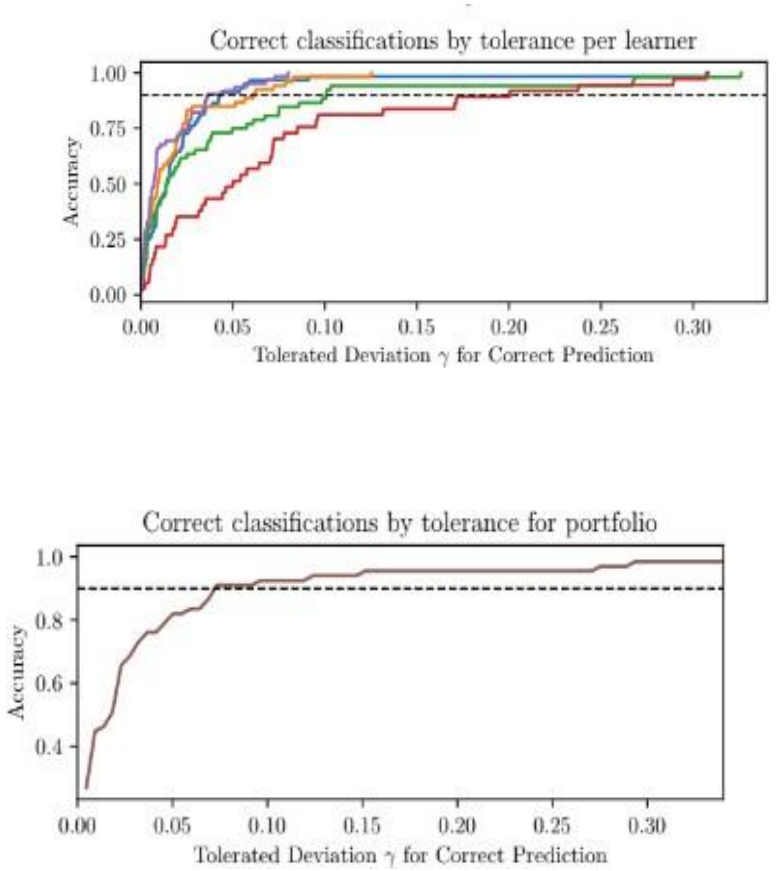


Fig. 7.3 Cumulative empirical distribution of the gap between these numbers

The results of the evaluation are presented in Fig. 7.2 and Fig. 7.3. Fig. 7.2 shows the predicted and actual improvements for individual classifiers and the portfolio when doubling the dataset size. The top row displays the correlation betweenpredicted and actual improvements for each classifier, and the bottom row presents the aggregated view of the data for different tolerance parameters (γ). The graphs indicate that thepredicted improvements are generally correlated with the actual improvements for mostclassifiers.

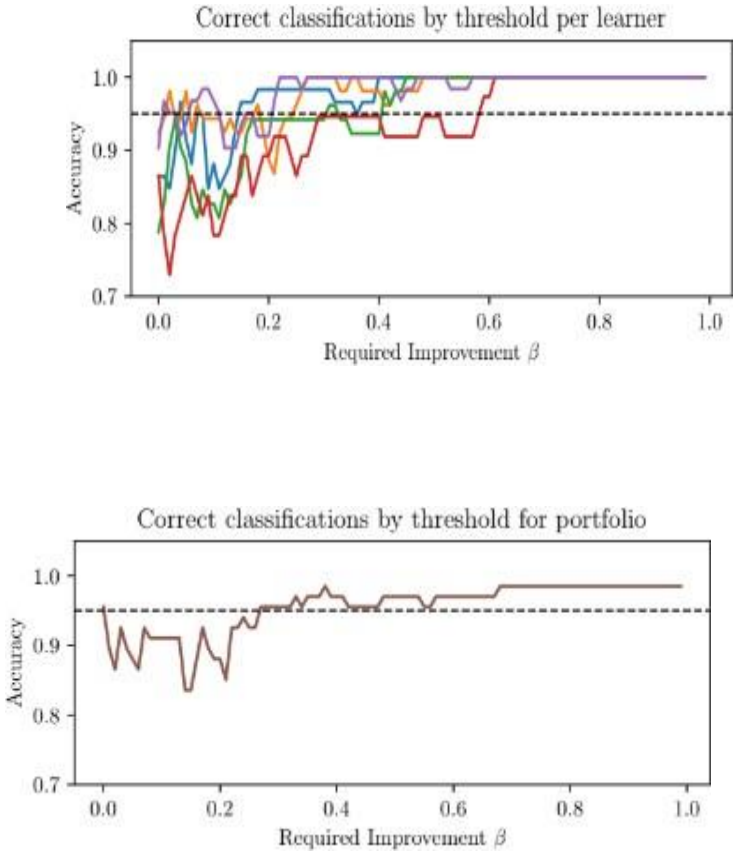


Fig. 7.4 Accuracy of the IPL classifier

Fig. 7.3 addresses the second research question, focusing on the accuracy of the IPL-classifier when predicting whether doubling the dataset size will lead to an improvement of at least β for individual classifiers and the portfolio. The results demonstrate that the IPL-classifier performs well in recommending the correct decision in accepting the cost for additional data points or not. The accuracy of the classifier is above 80% for all values of β, and for most datasets, it iseven above 90%.

Overall, the empirical learning curves generated with LCCV provide valuable insights into the learning behavior of the classifiers. Although the predictions of potential improvements may not always be highly precise, they are still sufficient for making correct decisions regarding acquiring additional data points when the required improvement to achieve a positive net utility is defined. This approach can be a useful resource for decision-making in data acquisition scenarios.

8. CONCLUSION AND FUTURE WORK

8.1 Conclusion:

The LCCV, a technique to validate learning algorithms based on learning curves. In contrast to other evaluation methods that leverage learning curves, LCCV is designed to avoid pruning candidates that can potentially still outperform the current best candidate.

Based on a convexity assumption, which turns out to hold almost always in practice, candidates are pruned once they are unlikely to improve upon the current best candidate. This makes LCCV faster than vanilla cross-validation methods while usually delivering equally good results. It had_re empirically shown that LCCV outperforms both 5- CV and 10-CV in many cases in terms of the runtime of a model selection algorithm based on random search that employs these methods for validation.

The median runtime reduction of LCCV compared to vanilla cross-validation is more than 50%, and for some datasets, a reduction of 85% is achieved, which corresponds to absolute reductions of 20h (median) and 100h (max) when compared to CV. And further compared LCCV to a Wilcoxon-based racing method as well as successive halving and found that all methods have their logical place on the Pareto front of runtime and performance. The greedy successive halving algorithm is usually even faster than LCCV, but LCCV is less prone to miss the optimal solution and can be used in sequential model selection, a strict superset of the situations to which successive halving can be applied in a straightforward way. Moreover, the learning curves can be used to give additional information to the data owner. LCCV showed that it gives good confidence recommendations (> 80% accuracy) of whether the acquisition of new data will enable a given desired reduction of the error rate. Future work can focus on several refinements of the technique.

8.2 Future work:

First, it would be highly desirable to detect non convexities to disable pruning in those cases. Second, it would be interesting to adopt a more dynamic and curve dependent manner of choosing anchors instead of fixing these as constants before. Finally, it would be interesting tointegrate LCCV in various AutoML toolboxes, such as Naive AutoML , ML-Plan , Auto-sklearn , and AutoWeka . By utilizing iteration-based learning curves, it could even be integrated into deep learning toolboxes.

9. REFERENCES

[1] B. Baker, O. Gupta, R. Raskar, and N. Naik. Accelerating neural architecture search using performance prediction. In 6th International Conference on Learning Representations – Workshop Track Proceedings, ICLR'18, 2018.

[2] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. Journal of Machine Learning Research, 13(Feb):281– 305, 2012.

[3] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kegl. Algorithms for ´ hyperparameter optimization. In Advances in Neural Information Processing Systems 24, NIPS'11, pages 2546–2554. Curran Associates, Inc., 2011.

[4] T. Domhan, J. T. Springenberg, and F. Hutter. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In Proceedings of the 24th International Joint Conference on Artificial Intelligence, IJCAI'15, 2015.

[5] P. Gijsbers, M. L. Bueno, S. Coors, E. LeDell, S. Poirier, J. Thomas, B. Bischl, and J. Vanschoren. Amlb: an automl benchmark. arXiv preprint arXiv:2207.12560, 2022

[6] K. Jamieson and A. Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. In Artificial Intelligence and Statistics, AISTATS'16, pages 240–248, 2016.

[7] A. Klein, S. Falkner, J. T. Springenberg, and F. Hutter. Learning curve prediction with Bayesian neural networks. In International Conference on Learning Representations, ICLR'17, 2017.

[8] M. Last. Improving data mining utility with projective sampling. In Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 - July 1, 2009, pages 487–496. ACM, 2009.

[9] R. Leite and P. Brazdil. Active testing strategy to predict the best classification algorithm via sampling and metalearning. In Proceedings of the 19th European Conference on Artificial Intelligence, ECAI'10, pages 309–314, 2010.

[10] L. Li, K. G. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. Journal of Machine Learning Research, 18: 185:1–185:52, 2017.

[11] M. Lopez-Ib ´ a´nez, J. Dubois-Lacoste, L. P. C ˜ aceres, M. Birattari, ´and T. Stutzle. The irace package: Iterated racing for automatic ¨ algorithm configuration. Operations Research Perspectives, 3:43–58,2016.

[12] F. Mohr and J. N. van Rijn. Towards model selection using learning curve cross-validation. In 8th ICML Workshop on Automated Machine Learning (AutoML), 2021.

[13] F. Mohr and J. N. van Rijn. Learning curves for decision making in supervised machine learning–a survey. arXiv preprint arXiv:2201.12150, 2022.

[14] F. Mohr and M. Wever. Naive automated machine learning – A late baseline for automl. CoRR, abs/2103.10496, 2021.

[15] F. Mohr and M. Wever. Naive automated machine learning.Machine Learning, pages 1–40, 2022.

[16] F. Mohr, M. Wever, and E. Hullermeier. ML-Plan: Automated ¨machine learning via hierarchical planning. Machine Learning, 107 (8):1495–1515, 2018.

[17] F. Mohr, M. Wever, A. Tornede, and E. Hullermeier. Predicting ¨ machine learning pipeline runtimes in the context of automated machine learning. IEEE Transactions on Pattern Analysis and Machine Intelligence, 43(9):3055–3066, 2021.

[18] F. Mohr, T. J. Viering, M. Loog, and J. N. van Rijn. LCDB 1.0: A First Learning Curves Database for Classification Tasks. In Machine Learning and Knowledge Discovery in Databases. Research Track - European Conference, ECML PKDD 2022. Springer, 2022.

[19] J. Petrak. Fast Subsampling Performance Estimates for Classification Algorithm Selection. In Proceedings of ECML-00: Workshop on Meta-Learning, 2000.

[20] F. Provost, D. Jensen, and T. Oates. Efficient progressive sampling. In Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'99, page 23–32. ACM,1999.

[21] A. Sabharwal, H. Samulowitz, and G. Tesauro. Selecting nearoptimal learners via incremental data allocation. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 30, 2016.

[22] J. N. van Rijn, S. M. Abdulrahman, P. Brazdil, and J. Vanschoren. Fast algorithm selectionusing learning curves. In International symposium on intelligent data analysis, pages 298–309.Springer,2015.

[23] T. J. Viering and M. Loog. The shape of learning curves: a review. CoRR, abs/2103.10948,2021.

[24] D. Wang, J. Andres, J. Weisz, E. Oduor, and C. Dugan. Autods: Towards human-centeredautomation of data science. arXiv preprint arXiv:2101.05273, 2021.

[25] G. M. Weiss and Y. Tian. Maximizing classifier utility when there are data acquisition and modeling costs. Data Mining and Knowledge Discovery, 17(2):253–282, 2008