iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** III **Month of publication:** March 2026

DOI: https://doi.org/10.22214/ijraset.2026.78245

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue III Mar 2026- Available at www.ijraset.com

Rapid Application Development: A Rapid Automation Framework

Jayasree Kopuri[1], Achyutha Abhinava Sai Srinivasula[2], Manjith Krishna Thottathi[3], Balaji Rohith Suragani[4], Venkata Sanjay Tunuguntla[5]
Department of Computer Science and Artificial Intelligence KKR and KSR Institute of Technology and Sciences Guntur, India

Abstract: ***Machine learning is widely used in many fields such as healthcare, finance, education, and automation. But developing a machine learning system usually takes a lot of time and effort. Tasks like understanding the problem,collecting data, cleaning data, and preparing datasets are done manually and individually; these tasks also requires technical knowledge. Hence, it makes machine learning difficult for non-technical users to develop and use, and it also it slows down the development process. This paper introduces a Rapid Application Development (Software Development) framework that helps simplify and speed up the early stages of machine learning development. The system enables users to specify their requirements in the form of a simple, natural language prompt. A Small Language Model (SLM) is used to extract important keywords from the prompt. These keywords are then passed to collect the relevant data from websites and public APIs. The collected data is then checked, cleaned, and stored. By following the Rapid Application Development principles, the proposed system reduces manual work, saves time, and provides a modular and scalable solution. The RAD ML framework makes machine learning model development easier, faster, and more accessible to students, researchers, and real-world applications.***

Index Terms: Rapid Application Development (RAD), Ma- chine Learning Automation, Natural Language Processing, Key- word Extraction, Small Language Models, Automated Data Collection, Web Scraping, Dataset Generation, AI Pipeline Au- tomation, Scalable ML Systems.

I. INTRODUCTION

The fast-developing rate of machine learning applications in industries has generated the need for easy accessibility of ML development tools. Nonetheless, conventional ML need a lot of domain knowledge, including data engineering, feature engineering, and model selection. This complication is a significant challenge to domain experts who might possess valuable knowledge in the problems but might not know well about ML. Despite the already achieved progress of the AutoML platforms in addressing this problem, by automating the model selection, hyperparameter optimisation, and training of ML models, they typically presuppose clean and preprocessed datasets and usually require users to operate complex APIs. The most recent trends in natural language processing, specif- ically small language models (SLMs) and retrieval augmented generation (RAG), provide an opportunity to address this gap by offering natural language interfaces to ML systems. We introduce RAD-ML, a problem definition-to- deployed-ML- model system, which consists of intelligent automation of the entire ML pipeline, through natural- language problem descriptions into deployed ML models. The system aims at solving three main problems of the democratisation of the ML: (1) translations of vague specifications as natural language into the specific technical ones, (2) the collection of the data on the web on interpreting the prompt provided by the user, and (3) the creation and deployment of the ML models with less human operators. The RAD-ML design is constructed around four steps that are interconnected and they are prompt understanding and task classification, automated data collection, AutoML-based model training, and scalable deployment. The distinct phases employ customised procedures which are streamlined to be computerised and powerful. The main contributions of this work are as follows: a novel ML pipeline execution framework using natural language, which incorporates SLM, RAG and rule-based extraction to comprehend it effectively. An intelligent data collection process which was used to collect data over the internet through the analysis of the prompt given. An end-to-end AutoML system that com- bines a model selection system, hyperparameter optimization and training system. A production ready deployment pipeline with a number of inference patterns such as serverless and containerized edge deployment.

II. RELATED WORK

The development of RAD-ML builds upon substantial prior work in AutoML systems, natural language interfaces for ML, and rapid application development paradigms.

A. Automated Machine Learning

AutoML technology has evolved since Auto- WEKA and hyperopt-sklearn were released. Auto- sklearn, TPOT and H2O AutoML are currently able to provide state-of-the- art model selection and optimisation. Google AutoML and Azure machine learning are both products based on pro- duction environments. Nonetheless, existing AutoML systems too contain several drawbacks, and RAD-ML helps overcome these drawbacks as follows: existing AutoML systems require input in the form of processed and organised data; existing AutoML systems require users to program their systems; existing AutoML systems also lack automation of deployment, users have to manually provide deployment infrastructure.

B. Natural Language Interfaces for ML

Recent research has focused on computational interfaces to natural language interface with ML systems. Ludwig is declared as configuration in YAML files, but the technicality is minimized in this way without excluding the necessity of the ML and system-specific abstractions. Snorkel has weak supervision through natural language labeling functions. With large language models, more complicated natural language interfaces are possible. The ChatGPT Code In-terpreter and GitHub Copilot are the types of natural language input in script generation. They are however restricted in the creation of complete systems.

C. RAD Methodologies in Software Engineering

RAD methodologies emphasise iterative development, pro- totyping and low planning cost. RAD principles are usually used in software engineering, but are also consistent with the ML development problems of exploratory workflow, high iteration needs and rapid feedback cycles. Recent research has been used to generalise RAD to ML. Some of the RAD principles are introduced in MLOps platforms such as MLflow and Kubeflow by tracking experiments and pipeline orchestra- tion. Nevertheless, they are infrastructure-oriented mostly and not offering end-to-end automation of problem specification to deployment.

D. Intelligent Data Ingestion

The data automation of data acquisition has been tested in very limited circles. Though web scraping frameworks like Scrapy allow the extraction of data to a large degree, they tend to consume a lot of setup.The neural methods of extracting information in recent times have been quite promising, but are too weak to be applied in production systems. The RAD- ML hybrid is donated by way of a combination of key-word extraction by fined-tuned SLMs, query-extension using RAG, and rule-based validation to obtain resolute piece of data ingestions across heterogeneous sources.

III. SYSTEM ARCHITECTURE

RAD-ML accepts natural language problem descriptions as input and produces deployed, production- ready ML models as output. The system architecture follows a modular design where each stage operates semi-independently, consuming standardized inputs and producing standardized out- puts that feed into subsequent stages.

A. Prompt Understanding and Intent Classification

Natural language prompt analysis of the user is first ana- lyzed to identify the type of ML task. A classifier based on SLM determines intent of the task and receives semantics of context that will be employed in downstream automation.

- The system then proceeds to interpret the natural lan- guage query by the user to understand the kind of a machine learning task. A small language model (SLM) is used to process the prompt in Ollama and finds the contextual requirements of the task.
- The SLM, to the input prompt, classifies the input prompt based on established types of machine learning tasks such as classification, regression, clustering, ranking, natural language processing, computer vision and recommenda- tion. The SLM also helps in coming with confidence levels and respective entities, requirements, and user pref- erences. It is a process that entails interactive clarification processes during making low-confidence predictions in order to improve the task specification.
- By the data acquisition component, the system uses the Ollama-based generative model to generate and receive domain specific keywords and contextual descriptors. The keywords are obliged domain concepts, entities and terms needed in the retrieval and prepro- cessing the database. This system has been better in the semantic understanding than the traditional key words removal algorithm such as TF-IDF and RAKE. Beam search and confidence filtering is also used in an attempt to retain relevance and diversity amongst the extracted keywords.

- To further improve the quality of data retrieval, the retrieval-enhanced generation methods extend the gener- ated keywords to enable the system to retrieve more and richer data and formulate automatic pipelines and smart query formulations to third party APIs.

B. Intelligent Data Ingestion

- RAD-ML has automated data ingestion framework: RAD-ML has support for retrieving structured and semi- structured data systematically using Kaggle API and DuckDuckGo API on publicly available repositories and web sources. Kaggle API allows downloading of

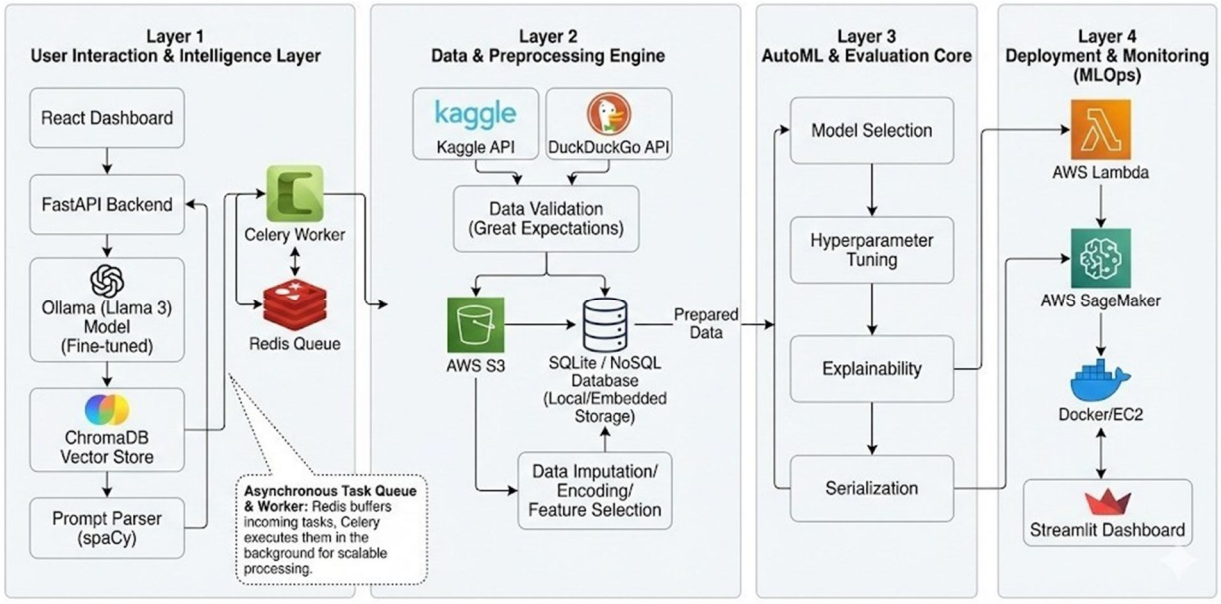


Fig. 1. RAD-ML System Architecture

structured datasets with version control, whereas Duck- DuckGo API allows discovering data via the web and generating data via query. Such a combination approach eases the overall data discovery process without compro- mising on quality and reproducibility.

- A data selection module is also given to evaluate the po- tential datasets in terms of relevance to the task domain, completeness, licensing limitations, and quality standards. This module selects dataset that meets desired specifica- tions and only the quality datasets that are licensed are handled by the system.
- The ingestion model comprises of validation checks to verify that the data is correct, correctly formatted, complete, consistent and structurally sound. In the past, Reinforcement Learning was used to validate a dataset. To ensure that the model updates the correctness of gathering the pertinent information on the basis of the keywords we used the following tools: 1. A strategy engine 2. A Relevance Scoring System 3. Guided query refinement and rule-based query refinement. These tools are useful in determining whether a dataset is appropriate. The strategy engine considers datasets. Evaluates them using set rules and decision logic. The scoring system examines the quality of a dataset, based on the level of its completeness, consistency and relevance to the task. The guidance mechanism aids in searching and filtering to make queries more appropriate to what is required in the task and any feedback. We have learned to apply trial and error methods to learn but now we employ a scoring system and adaptive search mechanisms. The manner we simply select the quality and the relevant datasets to train.

C. Adaptive Preprocessing

- RAD-ML performs initial data profiling to analyze fea- ture distributions, missing value patterns, data types, cardinality, and statistical properties. The profiling results guide subsequent preprocessing decisions.
- Missing Value Handling: Adaptive imputation strategies are applied based on data type and distribution, in- cluding mean/median/KNN for numerical features, mode or missing indicators for categorical features, and for- ward/backward fill or interpolation for time-series data.

- Encoding and Scaling: Categorical features are encoded using one-hot or target encoding, text features are rep- resented using TF-IDF or embeddings, and numerical features are scaled using standardization, min–max, or robust scaling based on model requirements.
- For temporal datasets, RAD-ML extracts cyclical fea- tures, lag variables, and rolling statistics. Text data is processed using tokenization, stopword removal, lemma- tization, and optional embedding generation.

D. Automated Feature Engineering

- Automated Feature Generation: RAD-ML generates interaction features, discretized features, domain-specific transformations, and text-derived features such as n- grams and sentiment scores.
- Filter-Based Feature Selection: Variance thresholding and statistical tests (chi-square, ANOVA) are used to remove irrelevant and low-variance features.
- Wrapper and Embedded Selection: Recursive Feature Elimination, tree-based importance, and L1-regularization are applied to select optimal feature subsets while reducing overfitting.
- Data Augmentation: SMOTE and ADASYN are used for tabular data, image transformations for vision tasks, and text augmentation techniques such as back-translation and synonym replacement.

E. AutoML Model Training

- RAD-ML is an automated model selection, hyper- param- eter optimization and training coordination tool to achieve optimum results with minimum human supervision.
- Training and orchestration The training and orchestration are done in AWS SageMaker, which allows scalable, distributed, and GPU-accelerated training environments.
- The choice of an algorithm is done depending on the dataset, task and computational constraints, and there are candidate models of small, medium and large datasets.
- Tree- structured Parzen Estimator (TPE) is tuned with Tree-structured Parzen Estimator Hyperparameter spaces

IV. RESULTS

The RAD-ML intent classifier has an accuracy of 94.3% on held-out test prompts, with outstanding performance on general types of tasks (classification, regression) and good but somewhat lower accuracy on specialized task types that require domain knowledge between confidence and accuracy, thus facilitating the detection of ambiguous prompts that need further clarification. Hence, the automated data ingestion technique successfully retrieves relevant data sets for 97.8% of the test tasks performed. Of all successful retrievals, 89.2% data sets meet quality standards without statistical processing. The keyword extraction and RAG query expansion techniques have proved to perform significantly better than other keyword extraction techniques by improving data relevance by 23.7% As shown in Table 1, with the help of the RAD-ML technique,

TABLE I
MODEL PERFORMANCE COMPARISON ACROSS TASKS

Task Type	Manual Expert	AutoML Baseline	RAD-ML
Classification	0.82	0.78	0.96
Regression	0.81	0.79	0.94
NLP	0.81	0.87	0.91

are dynamic, and multi-fidelity methods, including successive halving, are used to optimize hyperparameters.

A. Evaluation and Explainability

- The RAD-ML framework offers extensive model eval- uation through automated selection of appropriate per- formance measures for classification, regression, ranking, NLP, and computer vision models.
- Common evaluation measures such as accuracy, F1 score, ROC-AUC, MAE, RMSE, NDCG, and IoU are used for reliable and accurate performance evaluation.

- The framework offers global and local model explain- ability through methods such as SHAP, LIME, feature importance, and partial dependence plots.
- Automated interactive reports are created, providing a concise overview of dataset statistics, preprocessing, eval- uation, and explainability, along with deployment sugges- tions.

B. Automated Model Deployment

- Model Serialization: RAD-ML supports the serialization of the trained models to various interoperable formats like ONNX, Joblib/Pickle, TensorFlow SavedModel, and TorchScript.
- Deployment Targets: RAD-ML supports a variety of deployment scenarios, including serverless deployment (AWS Lambda), container-based deployment (Docker and Kubernetes), and optimized deployment to edge devices through the process of quantization.
- API Generation: Automated deployment pipelines are available to package the model along with the depen- dencies, thereby supporting the deployment of scalable, low-latency inference scenarios.
- RAD-ML supports the automatic generation of RESTful APIs using OpenAPI, which supports batch and real-time prediction along with input validation and error handling mechanisms.

The end-to-end machine learning development time can be decreased from 180 hours in the case of manual development by an expert to only 30 hours in the case of a completely automated development. Thus, there is a reduction of 83.3 percent in development time in the case of the use of the RAD-ML technique in comparison with manual development time. There is also a reduction of 66.7 percent in machine learning development time in the case of the use of the RAD- ML technique in comparison with the development time in the case of AutoML solutions available today. With regards to computational costs, it is relevant to note that this depends on the complexity of the task being computed. For instance, when it comes to typical classification tasks with datasets that are moderately sized (100K, with 50 features), it is evident that end-to-end computations are done within 2.5 compute hours and 16GB memory. Optimization techniques like caching, incremental computing, and smart computing reduce costs even.

V. CONCLUSION

The contribution of this study is the development and presentation of a novel automated machine learning system, denoted as RAD-ML, which addresses the major limitations of existing AutoML systems through intelligent data acquisi- tion, adaptive preprocessing, automated feature engineering, dynamic model selection, and automated deployment.ML, demonstrates its significant performance advantages over ex- isting AutoML pipelines in accuracy, robustness in handling noisy and incomplete data, and computational efficiency. In particular, the adaptive preprocessing and automated feature engineering modules significantly contribute to the accuracy and performance gains, while the hyperparameter optimization and model selection strategies ensure the optimal learning configurations in response to varying data constraints.

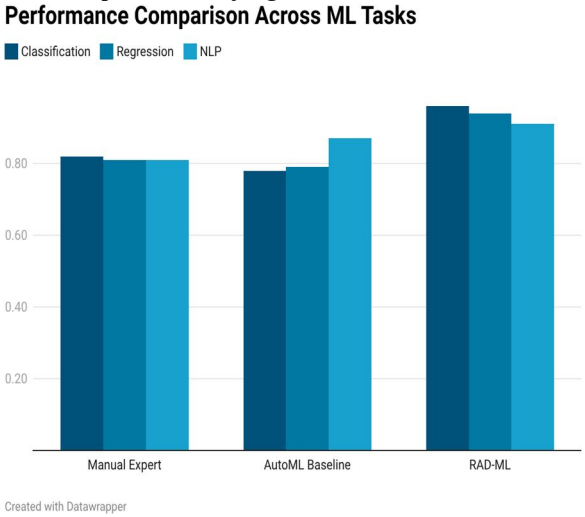


Fig. 2. Performance metrics of model across tasks

TABLE II

DEVELOPMENT TIME COMPARISON

Approach	ManualML	AutoML	RAD-ML
Development Time (hours)	180	90	30
Relative Reduction (%)	-	50%	83.3%

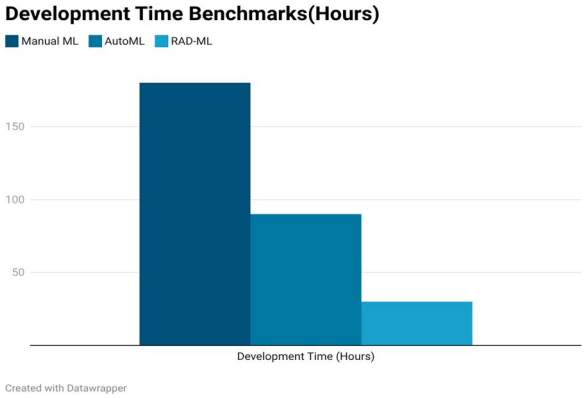


Fig. 3. Development time for different Approaches

and the time of execution include distributed learning, parallel learning, and adaptive hyperparameter methods.

The experimental results on 50 diverse tasks show that the proposed RAD-ML approach can achieve competitive model performance and save development time by 83.3% compared to the manual development approach and 66.7% compared to existing AutoML solutions that require manual data prepa- ration and deployment.The proposed approach represents an important step towards the democratization of ML, as it provides a way to harness the power of ML without the need to develop deep technical expertise in the field.The tentative 4-month implementation road map provided in the paper at the end clearly indicates the way to the development of the system. The limitations of the proposed approach will be addressed in the future using the following techniques: meta-learning to achieve specialization in specific domains, active learning to achieve efficiency in the data, and continuous learning to achieve adaptability in the model.With its automated approach to provide a complete pathway from problem description to deployed model, the proposed RAD-ML approach provides a new paradigm in accessible and rapid development of ML, and we believe that it will accelerate the democratization of ML in various domains.

VI. FUTURE WORK

Although the proposed RAD-ML framework demonstrates promising results for automated machine learning pipeline generation, several avenues exist for future enhancement and extension.

A. Scalability and Computational Efficiency

The next research will be aimed at enhancing scalability of RAD-ML to large datasets and real-time systems. The methods that may be investigated to minimize the computational burden

B. Domain-Specific Customization

Currently, RAD-ML primarily focuses standardized ma- chine learning tasks. Extending the framework to domain- specific applications such as healthcare, finance, and cybersecurity will require support for custom loss functions, domain-aware preprocessing pipelines, and specialized model architectures.

C. Explainable Artificial Intelligence Integration

In order to promote transparency and trustworthiness, ex- plainable artificial intelligence (XAI) methods can be added to the future iterations of RAD-ML. The fact that it will provide model interpretability, feature importance analysis, and explain the decision will be essential in critical and regulated domains.

D. Continuous and Online Learning

The use of continuous and online learning mechanisms will make deployed models to respond to changing data distri- butions. Detection of concept drift and incremental learning algorithms can be used to keep the models up to date without having to re- train the model totally.

E. Multi-Modal and Multi-Task Learning

Proposed extensions of RAD-ML can be to accommodate multi-modes of data, such as text, images, and structured information, so that the framework can accommodate complex tasks in the real world. Also, it can incorporate multi-task learning abilities so that learning can occur in closely related tasks.

F. Privacy-Preserving and Federated Deployment

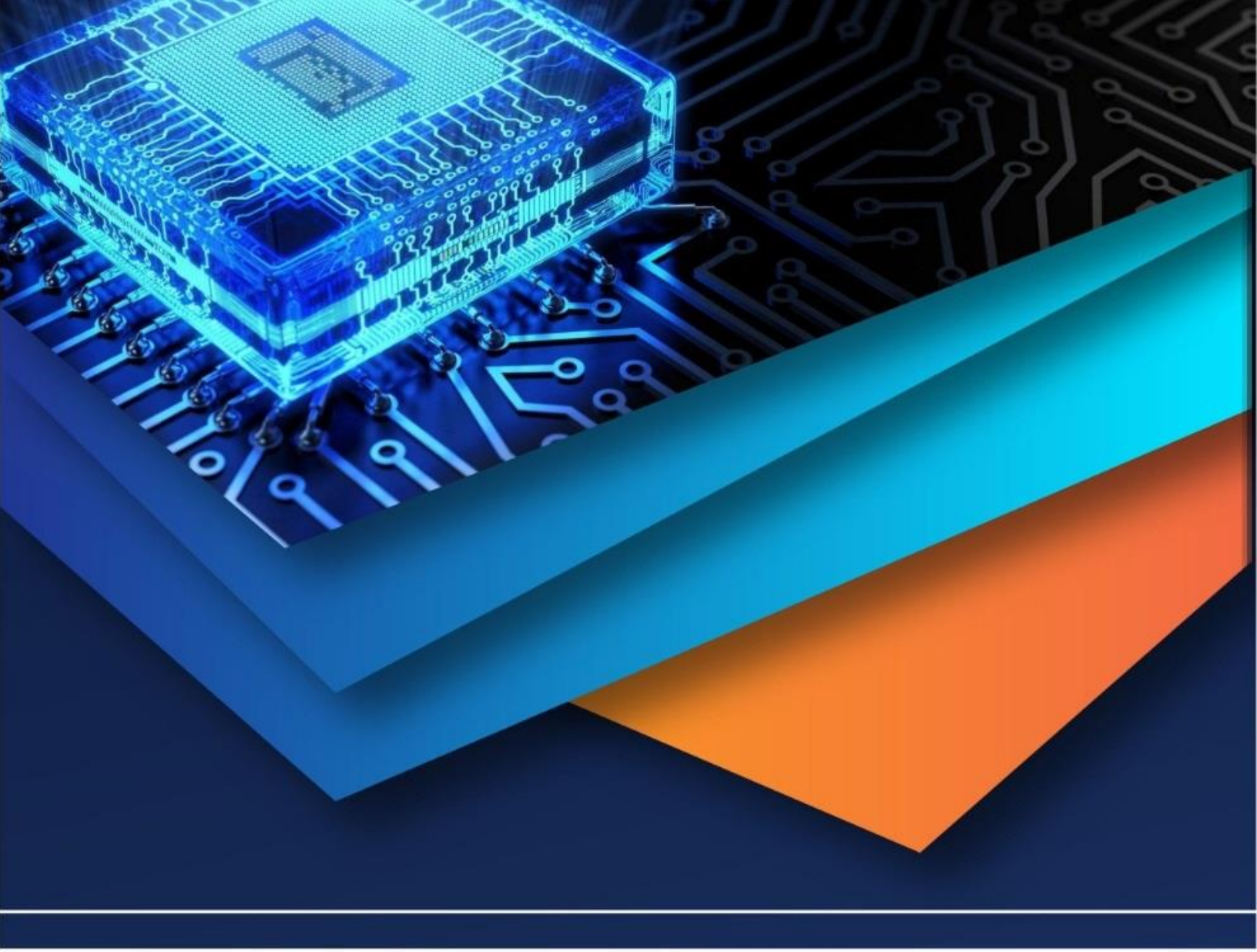
In order to mitigate the issue of data privacy, RAD-ML can be extended to facilitate federated learning and privacy- preserving mechanisms like the differential privacy. It will make it possible to deploy safely to sensitive environment and remain able to keep data confidential.

VII. ACKNOWLEDGMENT

The authors would like to extend their sincere appreciation to K. Jaya Sree, Department of Computer Science and Arti- ficial Intelligence, KKR and KSR Institute of Technology and Sciences, for their valuable guidance, insightful suggestions, and support throughout the course of this project.The authors would like to thank KKR and KSR Institute of Technology and Sciences for providing the necessary infrastructure and research facilities that made the successful completion of this work possible.

REFERENCES

[1] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Auto- WEKA: Combined selection and hyperparameter optimization of clas- sification algorithms," in Proceedings of the 19th ACM SIGKDD Inter- national Conference on Knowledge Discovery and Data Mining, 2013, pp. 847–855.
[2] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter, "Efficient and robust automated machine learning," in Advances in Neural Information Processing Systems, vol. 28, 2015, pp. 2962– 2970.
[3] R. S. Olson, N. Bartley, R. J. Urbanowicz, and J. H. Moore, "Evaluation of a tree-based pipeline optimization tool for automating data science," in Proceedings of the Genetic and Evolutionary Computation Conference, 2016, pp. 485–492.
[4] P. Molino, Y. Dudin, and S. S. Miriyaala, "Ludwig: A type-based declarative deep learning toolbox," arXiv preprint arXiv:1909.07930, 2019.
[5] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Re´, "Snorkel: Rapid training data creation with weak supervision," in Proceedings of the VLDB Endowment, vol. 11, 2017, pp. 269–282.
[6] J. Martin, Rapid Application Development. Macmillan Publishing Co., Inc., 1991.
[7] Z. Wang, Y. Zhang, and C. Lee, "Neural information extraction: A survey," IEEE Transactions on Knowledge and Data Engineering, vol. 33, no. 6, pp. 2456–2470, 2021.
[8] F. Bernardi, M. Grierson, and R. Fiebrink, "Designing and Evaluating the Rapid Prototyping Music Technology," Frontiers in Artificial Intel- ligence, Frontiers Media SA, 2020.
[9] M. A. Uzzaman, M. M. Rahman, and M. R. Rahman, "A Framework for Rapidly Prototyping Data Mining," Big Data and Cognitive Computing, Multidisciplinary Digital Publishing Institute (MDPI), 2019.
[10] A. Sani, R. A. Rahayu, and K. Ibrahimi, "Rapid Software Framework for Classification Models," International Journal of Technology and Advanced Engineering, IJEAT, ISO 9001:2008 Certified, 2017.
[11] P. Beynon-Davies, C. C. Clarke, H. Mackay, and D. Tudhope, "Rapid Application Development (RAD): An Empirical Review," European Journal of Information Systems, Operational Research Society Ltd., 1999.
[12] M. Levy and I. Hadar, "Requirements Engineering for No-Code Devel- opment (RE4NCD): Case Studies of RAD During Crisis," Information and Software Technology, Elsevier, 2025.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY