iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** V **Month of publication:** May 2025

DOI: https://doi.org/10.22214/ijraset.2025.71009

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 13 Issue V May 2025- Available at www.ijraset.com

Enhancing Machine Learning Systems with AUTOML: A TPOT Approach

Dr. V. Kavitha[1], Dr. R. G. Suresh Kumar[2], Ms. Divyapriya P[3], Ms. Haripriya C[4], Ms. Lavanya J[5]
[1]*Associate Professor, RGCET, Puducherry*
[2]*Professor, RGCET, Puducherry*
[3, 4, 5]*B.Tech (CSE), RGCET, Puducherry*

Abstract: ***In the modern era of data-driven decision-making, Automated Machine Learning (AutoML) has emerged as a transformative approach to streamline and optimize the process of building machine learning models. Traditional model development often requires deep domain knowledge, significant manual effort, and time-consuming trial-and-error processes for selecting algorithms, tuning hyperparameters, and designing data preprocessing steps. AutoML addresses these challenges by automating these tasks, making advanced machine learning accessible to both experts and non-experts. In the proposed system, AutoML is integrated using the TPOT (Tree-based Pipeline Optimization Tool) framework. TPOT is a genetic programming-based AutoML library in Python that automatically explores and optimizes machine learning pipelines for regression and classification problems. It evaluates numerous combinations of data preprocessing techniques, feature selection methods, and algorithms to identify the best-performing model pipeline. This automation significantly enhances the system's efficiency by reducing development time and eliminating manual intervention. By employing TPOT, the proposed system benefits from an intelligent search strategy that adapts to the specific characteristics of the dataset. It ensures that the most appropriate models are selected and fine-tuned to achieve high accuracy and generalization performance. The pipeline not only generates models with minimal human effort but also enhances scalability and adaptability to various domains and data types. Furthermore, integrating TPOT into the system allows for continuous learning and improvement as new data becomes available. This makes the system robust, dynamic, and well-suited for real-world applications where data evolves over time. The use of TPOT demonstrates the value of AutoML in reducing complexity while maintaining high performance, making it an essential component in the development of intelligent, automated, and scalable machine learning systems. Ultimately, the adoption of AutoML through TPOT reinforces the goal of building efficient and reliable models without requiring deep machine learning expertise.***

Keywords: AutoML, Data, Preprocessing, Machine Learning, Hyperparameters, Feature selection, Report generation, Data Visualization.

I. INTRODUCTION

The presented paper navigates the intricate relationship between Machine Learning (ML) and AutoML, emphasizing their symbiotic collaboration. ML, relying on vast datasets, employs algorithms to discern patterns, make predictions, and facilitate decision-making. AutoML complements this by automating essential yet tedious tasks, enhancing the efficiency and accessibility within the ML pipeline.

In the pursuit of refining data preprocessing methodologies, the survey proposes a comprehensive exploration of research papers and contributions. This extensive study encompasses a spectrum of papers that delve into innovative solutions bridging the gap between current challenges and the future promises of data preprocessing. The objective is to extract crucial insights by merging the intricacies of data preprocessing with the transformative potential of AutoML, aiming to advance data-driven decision-making in the evolving ML landscape. Highlighted within this survey are various research papers, from "DataAssist" to "REIN," each contributing vital principles to guide this quest. These papers shed light on the complex terrain where challenges such as imbalanced data, hyperparameter optimization nuances, and the need for advanced feature engineering converge, necessitating holistic solutions to bridge the divide between data and model.

As the frontiers of machine learning continue to expand, the principles extracted from these research papers act as guiding beacons, urging the crafting of automated solutions capable of addressing multifaceted challenges. Rooted in this extensive literature survey, the forthcoming architectural overview promises not only innovation but a transformation of the status quo. It aims to provide all-encompassing, end-to-end solutions for data preprocessing and the automation of pivotal tasks.

The survey's principles revolve around problem identification and the search for innovative solutions, bridging the gap between present challenges and the promising future of data preprocessing. Each research paper in the survey addresses a specific facet of data preprocessing, collectively contributing to a comprehensive understanding of this critical domain.

II. RELATED WORKS

Runtime Prediction of Machine Learning Algorithms in Automated Systems Parijat Dube; Theodoros Salonidis [1] DataAssist sounds like a promising addition to the landscape of automated machine learning (AutoML) tools. By focusing on data preparation and cleaning, it addresses a crucial aspect of the machine learning workflow that is often overlooked by existing tools. The key features of DataAssist seem to align well with the needs of data scientists and analysts, particularly in industries where data quality is paramount, such as economics, business, and forecasting. By providing functionalities for exploratory data analysis, visualization, anomaly detection, and data preprocessing, DataAssist streamlines the process of preparing data for modeling, potentially saving significant time and effort for practitioners. Moreover, the ability to export cleaned and preprocessed datasets for integration with other AutoML tools or user-specified models enhances its versatility and interoperability within existing workflows. This flexibility is crucial for accommodating different preferences and requirements in data analysis pipelines. Overall, DataAssist appears to fill a significant gap in the existing landscape AutoML tools by prioritizing data-centric tasks and offering comprehensive support for data preparation and cleaning. Its potential to save over 50% of the time typically spent on these tasks underscores its value proposition for practitioners across various domains.

Suraj Juddoo investigates data repair steps for EHR Big Data.[2] This paper addresses a significant challenge with relation to big data systems, particularly focusing on Electronic Health Records (EHR). The emphasis on optimizing data quality methodologies aligns well with the growing importance of leveraging high-quality data for meaningful insights, especially in sensitive domains like healthcare. The recognition of the data repair stage as a critical component of the data quality life cycle involves crucial, as addressing dirty data is often a complex and resource-intensive task. The acknowledgment an ignorance of how well-performing data restoration tools and algorithms work in the context of big data is an important observation, highlighting the need for specialized solutions in this domain. The systematic examination of data repair techniques and tools, then an experiment-based evaluation, is a robust methodology for gaining insights into their effectiveness. The comparison with a prototype built from previous study results adds a practical dimension to the evaluation. The finding that For Big Data, no algorithm or tool was found to be exceptionally sufficient emphasizes the challenges in this domain. However, identifying some algorithms and tools as marginally better than others provides valuable insights for potential improvements. The recommendations for enhancing data repair algorithms and tools for Big Data represent a valuable contribution to the field, guiding future research and development efforts.

Assessing the performance of AutoML algorithms using a set of simulated classification tasks Henrique Pedro Ribeiroa and Patryk Orzechowski [3] This paper explores the growing landscape of The popularity of machine learning automated (AutoML) programs can be attributed to their great performance and versatility in solving a wide range of issues. The challenge lies in choosing the most suitable AutoML algorithm for a given problem amid the increasing options available. To address this, the study examines the output of four well-known AutoML algorithms using their Diverse and generative ML benchmarking (DIGEN): Auto-Sklearn, H2O AutoML, Auto-Sklearn 2, . Synthetic datasets called DIGEN are used to demonstrate the advantages and disadvantages of popular machine learning algorithms. The outcomes demonstrate how successfully AutoML detects pipelines across datasets. While the majority of AutoML algorithms demonstrated similar performance, subtle differences emerged based on specific datasets and evaluation metrics, providing valuable insights into their comparative effectiveness.

A Whole-System Benchmarking Structure for Data Cleaning Techniques in Machine Learning Pipelines: REIN Christian Hammacher, Harald Schoening, and Mohamed Abdelaal [4] The paper emphasizes the crucial role of machine learning (ML) in daily life and emphasizes how important high-quality data is throughout the ML application lifecycle. It acknowledges the common discrepancies present in real-world tabular data, include inconsistencies, duplication, outliers, missing values, and pattern violations, which often arise during data collection, transfer, storage, or integration. Despite numerous data cleaning methods addressing these issues, the paper points out a gap in considering downstream ML model requirements. To bridge this gap, the work introduces a comprehensive benchmark named REIN1, aiming must carefully evaluate how various ML models are affected by data cleaning techniques. The benchmark addresses key research questions, exploring the necessity and efficacy of data cleaning in ML pipelines. The evaluation involves 38 error detection and repair methods, ranging from simple to advanced. To provide comprehensive insights, the benchmark employs a broad range of machine learning models that were trained on 14 publicly- accessible datasets that span multiple domains and include both synthetic and realistic error characteristics.

AutoCure: Machine Learning Pipeline Automatic Tabular Data Curation Method Ahmad Schoening, Harald Schoening, and Rashmi Koparde [5] The paper introduces Data curation pipeline AutoCure is innovative and requires no setting designed to address the persistent challenge of data preparation in machine learning applications across domains like autonomous driving, healthcare, and finance. The need for expert knowledge and considerable time investment in navigating the extensive search space for suitable data curation and transformation tools is a recognized hurdle in model development. AutoCure stands out by synthetically enhancing the clean data fraction by combining a data augmentation module with an inventive adjustable ensemble-based error detection technique. Notably, its configuration-free nature streamlines the implementation process, making it accessible for integration using free and open-source resources such as Auto-sklearn, H2O, therefore advancing the general democratization of machine learning.

Reciprocal neural networks for bidirectional mistake detection in databases Holzer and Stockinger, Kurt[6] This paper presents an innovative architecture leveraging bidirectional recurrent neural networks for the purpose of error detection in databases. Through experiments conducted on six distinct datasets, The outcomes demonstrate how well this strategy performs in comparison to cutting-edge mistake detection technologies. Specifically, the average F1-scores across all datasets demonstrate the effectiveness of the proposed architecture. Notably, the system exhibits a lower standard deviation, indicating greater robustness compared to existing methods. An additional advantage is the system's ability to achieve high F1-scores without the need for supplementary data augmentation techniques. This signifies the potential of the introduced bidirectional recurrent neural network architecture as a robust and efficient solution for error detection in diverse database scenarios.

III. PROPOSED METHODOLOGY

The proposed system employs Automated Machine Learning (AutoML) to streamline the development of predictive models using the TPOT (Tree-based Pipeline Optimization Tool) framework. TPOT, based on genetic programming, automates the process of creating optimized machine learning pipelines for both regression and classification tasks. This eliminates the need for manual selection of algorithms, feature engineering, preprocessing techniques, and hyperparameter tuning.The system accepts input data and utilizes TPOT to automatically explore and evolve numerous combinations of data transformations, model algorithms, and configurations. It evaluates each pipeline based on performance metrics such as accuracy or R^2 score, ensuring the best model is selected with minimal human intervention. By integrating TPOT, the system significantly reduces development time and enhances scalability across different datasets and application domains. It supports continuous learning by allowing retraining as new data becomes available, making it dynamic and adaptable to changing environments. This automated approach not only improves efficiency and model quality but also makes machine learning accessible to users without deep expertise. Ultimately, the proposed system demonstrates how AutoML can transform traditional model development into a fully automated, intelligent, and efficient process suitable for real-world deployment and continuous optimization.

A. Data Input Module

The Data Input Module is the foundational stage of the system, responsible for accepting user-uploaded datasets. This module supports common formats such as CSV and Excel, ensuring flexibility in usage. Once a file is uploaded, the module verifies the file structure, checks for missing or malformed entries, and identifies the type of machine learning task—regression or classification—based on the target column. Users may be prompted to specify which column serves as the output (label), while all other columns are treated as input features. This step ensures the system receives a clear and structured dataset, ready for further processing. Additionally, the module checks for consistent data types, invalid values, and high-cardinality categorical features that could influence model performance. Through these early validations, the Data Input Module minimizes the risk of errors in the downstream pipeline and creates a reliable foundation for the preprocessing and modelling stages.

B. Data Pre-processing Module

The Data Pre-processing Module handles the critical task of preparing raw data for modelling. It performs automated operations such as handling missing values through imputation (mean, median, or mode), encoding categorical variables (label or one-hot encoding), feature scaling (normalization or standardization), and detecting outliers. This module either applies default transformations or allows TPOT to handle them as part of the pipeline optimization. The goal is to transform raw, inconsistent data into a clean, uniform dataset that improves machine learning performance. It also includes optional steps like removing irrelevant features, balancing datasets (for classification), and performing feature selection using statistical techniques. By standardizing the data, this module ensures that machine learning algorithms perform consistently and reduces the chances of data leakage. It forms a

crucial link between the user-uploaded dataset and the AutoML engine.

C. AutoML Engine (TPOT Module)

This module is the core intelligence of the system. TPOT, a Tree- based Pipeline Optimization Tool, uses genetic programming to automatically generate, optimize, and evolve machine learning pipelines. It intelligently explores a wide range of combinations involving preprocessing methods, feature selectors, and estimators (e.g., Decision Tree, Random Forest, Logistic Regression, etc.). Over several iterations called generations, TPOT uses evolutionary strategies such as selection, mutation, and crossover to refine these pipelines. It automates hyperparameter tuning and evaluates models using cross-validation to reduce overfitting. Unlike traditional manual model selection, TPOT's approach is adaptive, data-driven, and ensures that optimal pipelines are generated with minimal human effort. The result is a high-performing model that is well- suited to the dataset, whether for classification or regression. Additionally, TPOT can export the entire pipeline as clean Python code, allowing developers to further modify, analyze, or deploy the model independently. This level of automation and flexibility is what makes TPOT a powerful component in the system.

D. Evaluation Module

The Evaluation Module is responsible for assessing the performance of all generated machine learning pipelines. Once TPOT produces candidate pipelines, this module applies cross- validation to ensure the robustness of each model. It evaluates performance using task-appropriate metrics—such as accuracy, precision, recall, F1-score for classification, and R^2 or RMSE for regression. The module ensures that the selected pipeline is not only high-performing on training data but also generalizes well to unseen data. It ranks pipelines based on these scores and identifies the best- performing one. This best pipeline is then marked as ready for deployment or export. The Evaluation Module also provides performance summaries, confusion matrices (for classification), and error distributions (for regression), which can be visualized through the user interface. These insights help users understand model reliability and identify areas for improvement. By including evaluation as a separate stage, the system ensures transparency, accountability, and the selection of genuinely effective models.

E. User Interface Module (Using Streamlit)

This module provides a clean and interactive front end for users to interact with the AutoML system, built using Streamlit. The interface allows users—whether technical or non-technical—to upload datasets, choose task types (classification or regression), and visualize results without writing code. It guides users through every step, from uploading the dataset to viewing the final evaluation metrics and downloading the best pipeline. Streamlit makes it easy to render charts, performance summaries, and confusion matrices. The UI also supports options for retraining models, selecting target columns, or adjusting basic preprocessing settings. This improves user accessibility and engagement, making the AutoML process more intuitive and less intimidating. With real-time feedback and dynamic content rendering, the Streamlit interface acts as a bridge between complex backend automation and user-friendly decision- making.

F. File Upload Module

This module is tightly integrated with the UI and is responsible for managing dataset uploads. It validates file formats (CSV/XLSX), reads the contents using Pandas, and stores it temporarily for further processing. Upon upload, the file is previewed to the user for verification. The module ensures secure and clean data intake by checking for file size limits, unsupported characters, or encoding issues. It also detects column headers, automatically parsing the data into features and targets based on user input or heuristics. The uploaded file is then passed to the preprocessing module, forming the first operational step in the AutoML pipeline. In short, the File Upload Module acts as the interface between the user's raw dataset and the machine learning engine.

G. Processed with ML Engine Module

After the data is cleaned and validated, it is passed into this module where it is processed through the AutoML engine. This module is where the real action happens—it orchestrates data flow from preprocessing to TPOT's optimization logic. Internally, it handles pipeline configuration, initiates the genetic search algorithm, monitors progress across generations, and logs model performance for comparison. It uses TPOT to explore numerous pipeline variations, ensuring exhaustive yet efficient model discovery. Once processing is complete, the best pipeline is retained while others are discarded. This module also facilitates re-processing in case of dataset updates or user-triggered re-training. The results are stored and forwarded to the evaluation and UI modules.

H. Suggest Best Model Module

This module finalizes the system's goal: to recommend the most effective model pipeline based on comprehensive evaluation. Using results from the Evaluation Module, it selects the top-performing pipeline and presents it to the user through the interface. It provides a summary of the chosen pipeline's structure along with performance metrics. The selected pipeline can be exported as a .py file or saved as a serialized model (e.g., .pkl) for deployment. In future system updates, this module can also suggest alternative models with slightly lower scores but faster inference or better interpretability—supporting explainable AI (XAI). The Suggest Best Model Module ensures that users are guided toward the most reliable and efficient solution without needing to interpret raw model comparisons manually.

Architecture Diagram

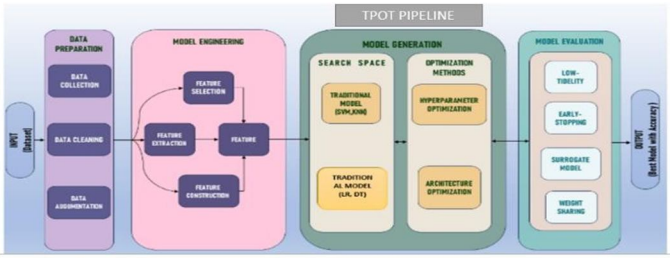


Fig 1: System Architecture

An architecture diagram makes the structure's visual representation available and components of a system or application. It typically includes various elements such as modules, databases, servers, and their interactions. The diagram serves as a high-level overview, illustrating how different parts of the system are connected and work together to achieve the intended functionalities. This visual representation aids in understanding the overall design, dependencies, and flow of data or processes within the architecture. It is a valuable tool for communication among stakeholders, allowing developers, architects, and other team members to have a shared understanding of the system's structure, helping in decision- making, troubleshooting, and system documentation.

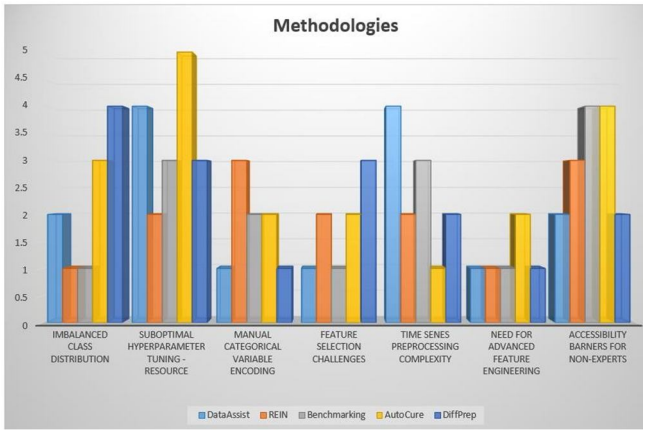


Fig 2: Methodologies

The research paper aims to explore the intricate details of the AutoML system, providing an in-depth analysis of its capabilities, experimental results, and its potential to revolutionize the field of machine learning. With a particular focus on addressing the shortcomings in existing data preprocessing methodologies, the system is positioned as a promising approach to enhancing datasets and subsequently improving findings across diverse domains. The paper likely delves into the system's innovative features, experimental validations, and how it contributes to overcoming challenges in data preprocessing, ultimately paving the way for more effective and efficient machine learning applications. The emphasis on improving datasets suggests a commitment to elevating the overall quality of input data, It is essential to machine learning models' ability to succeed.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue V May 2025- Available at www.ijraset.com

IV. RESULT AND DISCUSSION

AutoML (Automated Machine Learning) is a broad term that encompasses a variety of techniques and methodologies to automate The procedure for creating ML models. The specific formulas and methods used in AutoML systems can indeed vary based on the tasks being automated.

Table 1: Individual Values from the Dataset.

These are some outputs and overview of the given dataset. The given bar graphs are the analysed values of the given dataset which consits of no. of variables, Missing cells, Duplicate rows, Total Size in Memory and Missing Values.

A. Hyperparameter Tunning

Hyperparameter tuning is an essential component in machine learning model optimization, involving the adjustment of external settings that influence the learning process and model performance. These configurations, referred to as hyperparameters, are pre-established before training begins and cannot be learned from the learning set. The goal of hyperparameter tuning is to find the most effective configuration that maximizes a predefined performance metric. Grid search and random search are two commonly employed techniques for this purpose.

$$Best\ Hyperparameter\ Value = \arg\ max_{Hyperparameter\ \ Values}\ Model\ \ Performance\ \ Metric$$

Grid search examines a preset set of combinations of hyperparameters methodically, exploring the entire search space. In contrast, random search randomly samples configurations, offering a more stochastic approach. To ensure that the model performs as well as possible on data that hasn't been seen before, both approaches try to find a compromise between generalization and model complexity. A key component of fine-tuning models is hyperparameter tuning for specific tasks, ultimately enhancing their predictive capabilities and robustness.

B. Feature Engineering

Indeed, feature engineering is a critical aspect of the machine learning model development process, encompassing various techniques to enhance the representation of data and improve model performance. This process provide the model with more relevant and informative input.

$$New\ Feature = Feature^2$$

Creating new features may involve combining or synthesizing existing features to capture higher-order relationships or patterns in the data. Transformation of features can include normalizing or scaling numerical features, handling missing values, or encoding categorical variables. Additionally, finding and keeping the most essential features is the goal of feature selection while discarding less important ones, reducing dimensionality and potentially mitigating overfitting.

C. Ensemble Methods

AutoML frequently leverages ensemble methods as a powerful strategy to boost overall model performance. Ensemble methods involve combining predictions from multiple individual models, often of diverse architectures or trained with different subsets of data. The goal is to exploit the complementary strengths of various models, mitigating individual weaknesses and improving overall predictive accuracy. Common ensemble techniques include bagging, boosting, and stacking. Bagging, such as in Random Forests, aggregates predictions from several decision trees that were trained using arbitrary portions of the data improved robustness and decreased overfitting.

$$Ensemble\ Prediction = \frac{1}{n} - \sum_{i=1}^{n} Model_i\ (Input\ Data)$$

Boosting, exemplified by algorithms like AdaBoost or Gradient Boosting, sequentially trains models, with each subsequent model focusing on correcting the errors of its predecessor, leading to increased accuracy. Stacking combines predictions from different models using a meta-model, learning to weigh individual model outputs optimally. Ensemble methods are effective in handling complex relationships within data, increasing model stability, and generalizing well to unseen instances, making them a valuable tool in the AutoML toolkit for achieving superior predictive performance.

D. Model Selection

In AutoML, selecting the best-performing model is pivotal and relies heavily on evaluating various performance metrics. Common metrics include area under the Receiver Operating Characteristic (ROC) curve, accuracy, and F1-score. Accuracy measures the proportion of correctly predicted instances, offering a straightforward assessment of overall correctness. F1-score strikes a balance between recall and precision, making it ideal for applications where the costs of false positives and false negatives differ. The area under the ROC curve quantifies the trade-off between the true positive rate and false positive rate, indicating a model's ability to distinguish between classes. The choice of metric depends on the dataset's nature; accuracy is suitable for balanced datasets, while F1-score is preferable for imbalanced classes. With PyCaret, model creation becomes streamlined, empowering users to leverage a diverse array of algorithms and leverage a diverse array of algorithms and performance evaluation techniques to select the most suitable model for their specific application.

$Best\ Mod$ = arg max_{Models} $Model\ Performance\ Metric$ AutoML systems often perform a systematic search over hyperparameter configurations, and The model that performs the best
according to the selected metric is the one that gets deployed. These metrics guide the AutoML process, ensuring the chosen model aligns with the specific objectives and requirements of the given machine learning task

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
gbr	Gradient Boosting Regressor	0.7224	1.0575	1.028	0.0924	0.4842	0.1577	0.177
lightgbm	Light Gradient Boosting Machine	0.7035	1.0753	1.0364	0.0773	0.4813	0.155	0.144
rf	Random Forest Regressor	0.6797	1.0811	1.0392	0.0712	0.4773	0.1527	0.409
lar	Least Angle Regression	0.7809	1.1385	1.0664	0.0237	0.4977	0.1736	0.015
lr	Linear Regression	0.7825	1.1396	1.0669	0.0228	0.4982	0.1739	0.772
ridge	Ridge Regression	0.7826	1.1396	1.0669	0.0228	0.4982	0.1739	0.01
br	Bayesian Ridge	0.7848	1.1402	1.0672	0.0223	0.4984	0.1747	0.012
et	Extra Trees Regressor	0.6681	1.1443	1.0691	0.0169	0.4867	0.1503	0.208
en	Elastic Net	0.7975	1.15	1.0718	0.0139	0.5002	0.1793	0.01
lasso	Lasso Regression	0.8003	1.1524	1.0729	0.0119	0.5008	0.18	0.012
llar	Lasso Least Angle Regression	0.8003	1.1524	1.0729	0.0119	0.5008	0.18	0.012
ada	AdaBoost Regressor	0.8867	1.1618	1.0775	0.0027	0.4851	0.231	0.023
omp	Orthogonal Matching Pursuit	0.8072	1.1656	1.079	0.0006	0.5029	0.1812	0.01
dummy	Dummy Regressor	0.8107	1.1683	1.0803	-0.0016	0.5033	0.1827	0.01
knn	K Neighbors Regressor	0.7091	1.2439	1.1143	-0.0656	0.5012	0.1544	0.017
huber	Huber Regressor	0.7665	1.4728	1.2128	-0.2629	0.538	0.1499	0.039
dt	Decision Tree Regressor	0.6865	1.9758	1.405	-0.6979	0.6495	0.1604	0.016
par	Passive Aggressive Regressor	1.3615	4.9985	1.8137	-3.1395	0.6204	0.3573	0.013

Fig 3: Precision for Every Algorithm

Fig 4: Best Model Analysis

V. CONCLUSION

In conclusion, the integration of AutoML through TPOT in the proposed system significantly simplifies and accelerates the machine learning workflow. By automating key tasks such as data preprocessing, feature selection, algorithm selection, and hyperparameter tuning, TPOT enables users—regardless of their ML expertise—to build high-performing models efficiently. Its genetic programming-based optimization ensures that the best possible pipeline is selected for any given dataset, making the system adaptable and scalable across different domains. This automation not only reduces development time but also enhances model accuracy and generalization. Furthermore, TPOT's ability to continuously improve with new data makes the system dynamic and suitable for real-world, evolving data environments. Overall, the use of TPOT reinforces the core objective of the proposed system: to democratize machine learning by making it accessible, reliable, and efficient for both technical and non-technical users. It establishes a robust foundation for intelligent decision-making powered by automated, data-driven insights.

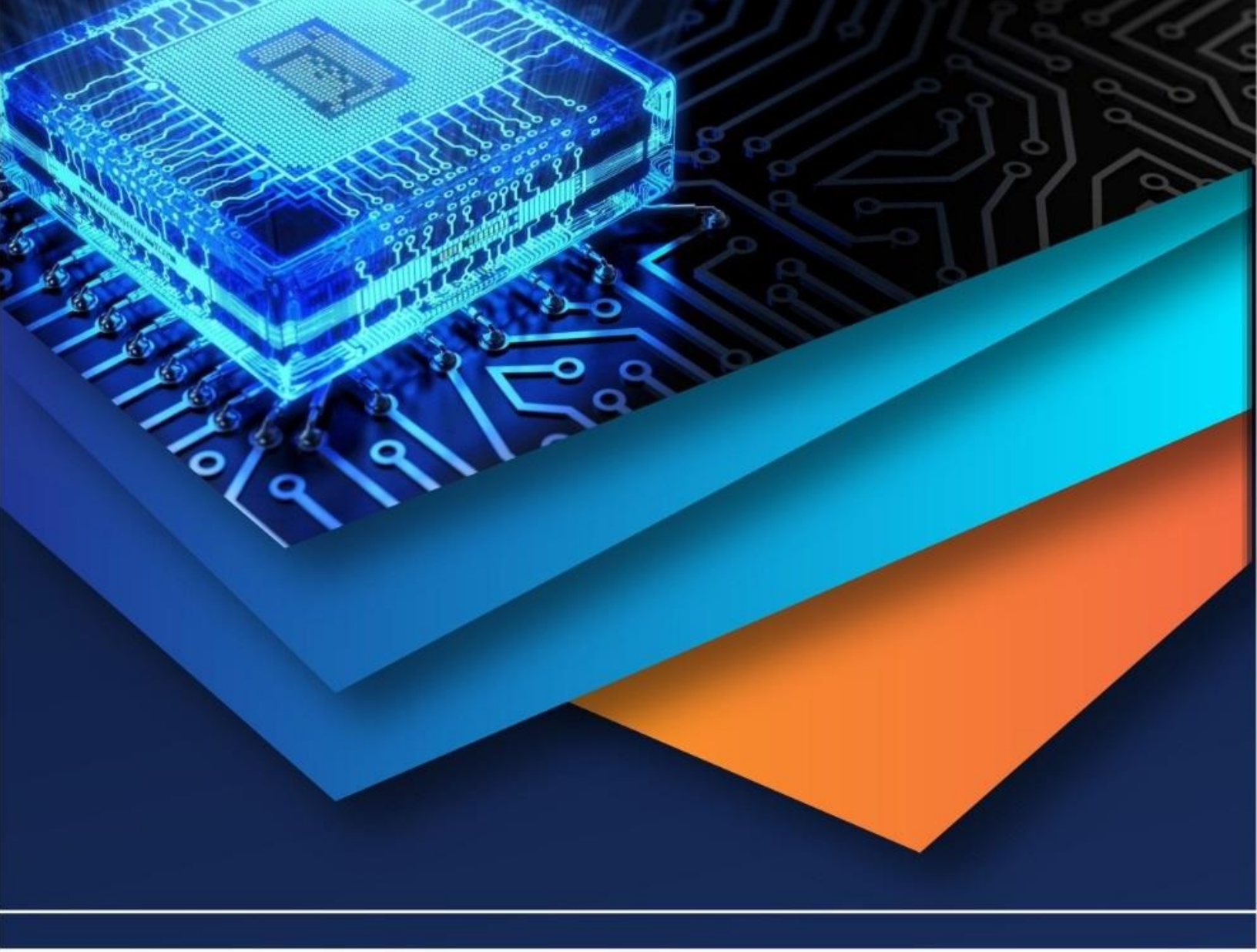
VI. FUTURE WORK

In future developments, addressing the intricacies of data preprocessing is committed to several key areas of focus. Firstly, there is a commitment to enhancing automation capabilities, aiming to make the entire machine learning pipeline more accessible and efficient. This involves further streamlining processes, reducing manual intervention, and improving the overall user experience. Additionally, a focus on advanced model explainability is prioritized, aiming to enhance or refine methods that allow users to gain deeper insights into the decisions and predictions made by machine learning models. This is crucial for fostering trust and understanding in the application of these models.

REFERENCES

[1] K. Goyle, Q. Xie, & V. Goyle, "DataAssist: A Machine Learning Approach to Data Cleaning and Preparation," eprint arXiv:2307.07119, 2023.
[2] S. Juddoo, "Investigating Data Repair steps for EHR Big Data," in International Conference on Next Generation Computing Applications, 2022.
[3] P. Ribeiro, P. Orzechowski, J. B. Wagenaar, & J. H. Moore, "Benchmarking AutoML algorithms on a collection of synthetic classification problems," eprint arXiv:2212.02704, 2022.
[4] M. Abdelaal, C. Hammacher, & H. Schoening, "REIN: A Comprehensive Benchmark Framework for Data Cleaning Methods in ML Pipelines," eprint arXiv:2302.04702, 2023.
[5] F. Neutatz, B. Chen, Y. Alkhatib, J. Ye, & Z. Abedjan, "Data Cleaning and AutoML: Would an Optimizer Choose to Clean?" Eprint Springer s13222-022-00413-2, 2022.
[6] M. Abdelaal, R. Koparde, & H. Schoening, "AutoCure: Automated Tabular Data Curation Technique for ML Pipelines," eprint arXiv:2304.13636, 2023.
[7] S. Holzer & K. Stockinger, "Detecting errors in databases with bidirectional recurrent neural networks," OpenProceedings ZHAW, 2022.
[8] P. Li, Z. Chen, X. Chu, & K. Rong, "DiffPrep: Differentiable Data Preprocessing Pipeline Search for Learning over Tabular Data," eprint arXiv:2308.10915, 2023.
[9] M. Singh, J. Cambronero, S. Gulwani, V. Le, C. Negreanu, & G. Verbruggen, "DataVinci: Learning Syntactic and Semantic String Repairs," eprint arXiv:2308.10922, 2023.
[10] S. Guha, F. A. Khan, J. Stoyanovich, & S. Schelter, "Automated Data Cleaning Can Hurt Fairness in Machine Learning-based Decision Making," in IEEE 39th International Conference on Data Engineering, 2023.
[11] R. Wang, Y. Li, & J. Wang, "Sudowoodo: Contrastive Self-supervised Learning for Multi-purpose Data Integration and Preparation," eprint arXiv:2207.04122, 2022.
[12] B. Hilprecht, C. Hammacher, E. Reis, M. Abdelaal, & C. Binnig, "DiffML: End-to-end Differentiable ML Pipelines," eprint arXiv:2207.01269, 2022.
[13] V. Restat, M. Klettke, & U. Störl, "Towards a Holistic Data Preparation Tool," in EDBT/ICDT Workshops, 2022.
[14] M. Nashaat, A. Ghosh, J. Miller, & S. Quader, "TabReformer: Unsupervised Representation Learning for Erroneous Data Detection," eprint https://doi.org/10.1145/3447541, 2021.
[15] F. Calefato, L. Quaranta, F. Lanubile, & M. Kalinowski, "Assessing the Use of AutoML for Data-Driven Software Engineering," eprint arXiv:2307.10774, 2023.
[16] H. Stühler, M. A. Zöller, D. Klau, A. Beiderwellen- Bedrikow, & C. Tutschku, "Benchmarking Automated Machine Learning Methods for Price Forecasting Applications," eprint arXiv:2304.14735, 2023
[17] M. Feurer, A. Klein, J. Eggensperger, Katharina Springenberg, M. Blum, F. Hutter, Efficient and robust automated machine learning, in: Advances in Neural Information Processing Systems 28 (2015), 2015, pp. 2962–2970.
[18] E. LeDell, S. Poirier, H2O AutoML: Scalable automatic machine learning, 7th ICML Workshop on Automated Machine Learning (AutoML) (July 2020).
[19] P. Gijsbers, E. LeDell, S. Poirier, J. Thomas, B. Bischl, J. Vanschoren, An open source automl benchmark, 2019, 6th ICML Workshop on Automated Machine Learning, AutoML@ICML2019 ; Conference date: 14-06-2019 Through 14-06-2019.
[20] P. Gijsbers, M. L. P. Bueno, S. Coors, E. LeDell, S. Poirier, J. Thomas, B. Bischl, J. Vanschoren, Amlb: an automl benchmark (2022). doi:10.48550/ARXIV.2207.12560.
[21] I. Guyon, L. Sun-Hosoya, M. Boull´e, H. J. Escalante, S. Escalera, Z. Liu, D. Jajetic, B. Ray, M. Saeed, M. Sebag, A. Statnikov, W.-W. Tu, E. Viegas, Analysis of the AutoML Challenge Series 2015–2018, Springer International Publishing, Cham, 2019, pp. 177– 219.doi:10.1007/978-3-030-05318-5_10.
[22] Erickson N, Mueller J, Shirkov A, Zhang H, Larroy P, Li M, Smola AJ (2020) Autogluon-tabular: Robust and accurate automl for structured data. CoRR, abs/2003.06505
[23] K. Van der Blom, A. Serban, H. Hoos, and J. Visser, "AutoML Adoption in ML Software,8th ICML Workshop on Automated Machine Learning, 2021
[24] T. T. Le, W. Fu, J. H. Moore, Scaling tree-based automated machine learning to biomedical big data with a feature set selector, Bioinformatics 36 (1) (2020) 250– 256.
[25] M. Nashaat, A. Ghosh, J. Miller, and S. Quader,"TabReformer:Unsupervised Representation Learning for Erroneous Data Detection,"in ACM Transactions on benchmark, 2019, 6th ICML Workshop on Automated Machine Learning, AutoML@ICML2019 ; Conference date: 14-06-2019 Through 14-06-2019.

.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY