iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** VII **Month of publication:** July 2026

DOI: https://doi.org/10.22214/ijraset.2026.84075

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 VII July 2026- Available at www.ijraset.com

Bridging Scalability and Interpretability in AutoML Via Feature Engineering

Ms. CH. Vasavi[1], Ms. SK. Raqeeba[2]
[1]*Assistant Professor*, [2]*Assistant Professor, Department of Computer Science and Engineering, Department of Computer Science and Engineering (Autonomous), Geethanjali Institute of Science and Technology*

Abstract: ***Feature engineering plays a key role in determining the performance of machine learning models, but manual feature design is time-consuming and requires strong domain knowledge. This work presents an automated feature engineering framework integrated with an interpretable AutoML pipeline, built around the BigFeat methodology. The system automatically generates new features from existing data using mathematical and logical operators and selects the most stable and relevant features for learning, while preserving interpretability by maintaining traceable mappings between original and engineered features. The framework is designed to handle large, high-dimensional datasets with manageable computational overhead. Automated model selection and hyperparameter tuning across Random Forest, Logistic Regression, and Decision Tree classifiers are incorporated to optimize predictive performance without manual intervention. The proposed system is intended to reduce human effort and development time in the feature engineering process while remaining scalable and adaptable to different datasets. Experimental evaluation on the Madelon dataset, a high-dimensional synthetic benchmark for feature selection, indicates that the automated and interpretable pipeline performs comparably to, and in some respects favourably against, baseline feature engineering approaches, demonstrating the practical effectiveness of combining scalable feature generation with interpretable AutoML.***

Keywords: ***Automated Machine Learning (AutoML), Feature Engineering, BigFeat, Interpretability, Scalability, Hyperparameter Optimization, Feature Selection***

I. INTRODUCTION

A. Overview

Feature engineering is a crucial step in the machine learning pipeline, involving the creation, transformation, and selection of relevant features from raw data to improve model performance. Traditionally, this process requires significant domain knowledge, manual effort, and time. With the advancement of artificial intelligence, Automated Machine Learning (AutoML) has emerged as a powerful approach to automate various stages of the machine learning pipeline, including feature engineering. Automated feature engineering uses algorithms to automatically generate, transform, and select the most relevant features from datasets.

This work focuses on developing an automated system that leverages AutoML techniques to perform feature engineering efficiently. By reducing human intervention, the system aims to improve model accuracy, scalability, and development speed. The increasing adoption of AutoML tools in industry further highlights the importance of automation in data science, as organizations continuously seek ways to accelerate model development while maintaining high performance. Automated feature engineering plays a vital role in achieving this balance by optimizing data representation and improving the overall effectiveness of machine learning systems.

B. Motivation

Manual feature engineering is often time-consuming and labor-intensive, requiring significant human effort to analyze and transform raw data into meaningful inputs for machine learning models. It heavily depends on domain expertise, which may not always be available, and is prone to human bias and error. In real-world scenarios, datasets are often large and complex, making manual feature extraction inefficient and difficult to scale.

There is a growing need for intelligent systems that can automatically identify important features, handle large-scale datasets, and improve model performance without extensive manual tuning. AutoML addresses these challenges by automating key steps of the machine learning pipeline, including feature selection and model optimization, leading to faster development, more consistent results, and reduced dependency on expert intervention. AutoML tools also help democratize machine learning by making it accessible to non-experts, motivating the development of a system that reduces human effort and time in feature engineering, improves model accuracy through optimal feature discovery, and provides a scalable, user-friendly solution.

C. Problem Statement

Feature engineering remains a complex and time-consuming process that requires strong domain expertise. Manual approaches are not scalable for large, high-dimensional datasets, and selecting the most relevant features, although challenging, has a major impact on model accuracy. Traditional approaches involve extensive experimentation and manual tuning, which is inefficient. There is therefore a need for an automated feature engineering system based on AutoML techniques that can generate and select optimal features with minimal human intervention.

D. Objective

The main objective of the proposed system is to simplify and enhance the feature engineering process by introducing automation through AutoML techniques. Since traditional feature engineering is often complex, time-consuming, and dependent on strong domain expertise, this system aims to reduce such complexity by automating the entire process, eliminating dependency on expert knowledge in data preparation, and making machine learning more accessible to non-experts.

E. Automated Machine Learning and the ML Life-Cycle

Automated Machine Learning (AutoML) refers to the process of automating the end-to-end application of machine learning to real-world problems. AutoML encompasses data preprocessing, feature engineering, model selection, and hyperparameter tuning. Automated feature engineering, a key component of AutoML, involves generating new features from existing data, transforming variables through scaling and encoding, and selecting the most important features. AutoML systems use advanced techniques such as feature generation and transformation, feature selection algorithms, hyperparameter optimization, and model ensembling.

The AutoML life cycle adopted in this work consists of six interconnected phases. Problem definition identifies the modelling objective and the evaluation metrics relevant to the task. Data preprocessing handles missing values, duplicates, outliers, categorical encoding, and normalization to prepare clean input data. Feature engineering automatically generates, transforms, and selects relevant features while removing irrelevant or redundant ones. Model selection evaluates multiple candidate algorithms automatically to remove the need for manual trial-and-error. Model training fits the selected models to the prepared data while tuning hyperparameters. Finally, model evaluation and optimization assesses performance on held-out data using metrics such as accuracy, precision, recall, and F1-score, refining the best-performing model further.

II. LITERATURE SURVEY

Eldeeb and Elshawi introduced a dynamic feature meta-learning framework that combines scalable feature engineering with interpretable AutoML [1]. The approach automates feature generation and selection, reducing dependency on domain expertise while efficiently handling high-dimensional datasets by eliminating redundant and irrelevant features. The reported system achieved a classification accuracy of approximately 81%, and the framework additionally supports interpretability by allowing users to trace how features influence model predictions. However, the approach requires considerable computational power and memory, and the associated training process can be time-consuming, limiting its suitability for low-resource environments.

AutoFE is a hybrid automated feature engineering technique that integrates symbolic learning with machine learning methods to generate new features through mathematical expressions and transformations [5]. The technique minimizes manual intervention and enhances the capacity of models to learn complex patterns, reporting an accuracy of approximately 67.66%. While AutoFE supports automation and flexibility across datasets, it requires large volumes of data to produce meaningful results and incurs notable computational complexity from symbolic transformations, making it less effective for small datasets.

AutoFeat-Hybrid applies evolutionary algorithms to automatically generate and select features by exploring a large search space of feature combinations [3]. The reported accuracy of around 75% demonstrates the effectiveness of the evolutionary mechanism in discovering useful feature interactions and improving model robustness. However, the large operator search space increases computational cost and processing time, which may limit applicability in real-time settings.

SAFE (Scalable Automatic Feature Engineering) is a framework designed for automatic feature engineering on large-scale, industrial datasets [4]. It uses correlation-based feature selection together with redundancy pruning to identify relevant features while maintaining interpretability, and reported an accuracy of approximately 75.13%. SAFE reduces manual effort in traditional feature engineering, although its underlying method involves quadratic time complexity, which becomes a limitation for very large datasets. Collectively, these studies indicate that automated feature engineering methods can meaningfully reduce manual effort and improve model performance, but each approach involves a trade-off among scalability, interpretability, and computational cost. This motivates the design of a unified framework that integrates feature generation, selection, and AutoML-based model training within a single interpretable and scalable pipeline, which forms the basis of the system proposed in this work.

III. SYSTEM ANALYSIS

A. Existing System

Various machine learning and automated feature engineering approaches have been developed for solving classification problems, including healthcare-related applications. Traditional methods rely on manual feature engineering, where domain experts analyze raw data and create meaningful features through trial-and-error, an approach that is time-consuming and requires significant expertise. Automated tools such as AutoFeat, SAFE, TPOT, and Auto-sklearn have been introduced to reduce this manual effort: AutoFeat applies predefined mathematical transformations, SAFE focuses on scalable feature generation, TPOT relies on genetic algorithms, and Auto-sklearn applies meta-learning with Bayesian optimization. Although these approaches show promising results, they continue to face challenges related to inefficient feature generation, limited interpretability, and high computational cost, and many do not integrate feature engineering, model selection, and optimization within a single framework, which constrains their scalability and real-world applicability.

The disadvantages of these existing approaches can be summarized as follows: heavy reliance on manual feature engineering increases time consumption and dependency on expert knowledge; automated tools such as AutoFeat and SAFE often generate a large number of features, leading to redundancy; generated features and models frequently lack interpretability, making it difficult for users to understand predictions; high computational cost from large search spaces and complex algorithms reduces system efficiency; and the absence of a unified pipeline complicates the integration of feature engineering with model optimization.

B. Proposed System

The proposed system overcomes these limitations by introducing a scalable and interpretable AutoML framework based on BigFeat-FE and BigFeat-AutoML. The system focuses on automating feature engineering while maintaining interpretability and efficiency. It utilizes dynamic feature generation through computation trees, which systematically create new features from existing ones; these generated features remain expressive and traceable back to the original input features, improving interpretability. To ensure efficiency, the system employs stability-based feature selection, which retains only the most relevant and non-redundant features based on importance scores and correlation analysis. The system further integrates BigFeat-AutoML, which automates model selection and hyperparameter tuning using interpretable machine learning models, forming a unified pipeline that combines feature engineering, model selection, and optimization into a single scalable solution.

The advantages of the proposed system include dynamic feature generation for efficient and scalable feature creation; automatic selection of the most relevant features for improved prediction performance; retention of only important features to avoid redundancy; high interpretability achieved by linking generated features to their original transformations; integration of feature engineering, model selection, and optimization within a single AutoML framework; and reduced computational cost relative to traditional approaches.

C. Functional and Non-Functional Requirements

The functional requirements of the system include data collection, data preprocessing, feature generation and selection, model training and testing, and prediction with result display. The non-functional requirements emphasize scalability in handling large, high-dimensional datasets without performance degradation; interpretability through traceable mappings between generated and original features; usability via an intuitive interface suited to users with limited machine learning background; performance in terms of accurate and timely predictions; reliability through consistent behaviour across datasets and environments; and maintainability that allows new datasets or algorithmic improvements to be integrated with minimal disruption.

D. Feasibility Study

The feasibility of the proposed system was examined from economic, technical, and social perspectives. Economically, the system is viable because it relies on open-source tools and technologies, and automating feature engineering and model selection reduces long-term manual effort and cost. Technically, the system is feasible because it uses established machine learning algorithms and AutoML frameworks, with the BigFeat framework offering efficient, near-linear-complexity processing suitable for large datasets, implemented using Python together with web technologies such as Flask. Socially, the system is expected to be well received, particularly in domains such as healthcare, since improved interpretability increases user trust while reducing the workload associated with manual feature design.

IV. METHODOLOGY

A. System Architecture

The overall pipeline follows a sequential architecture comprising data collection, preprocessing, BigFeat feature engineering, AutoML model training, and model evaluation, with feature selection acting as an intermediate stage between feature engineering and model training. The Madelon dataset, a synthetic, high-dimensional benchmark specifically designed for feature selection tasks, is used to evaluate the system. It contains a large number of numerical features, including both relevant and irrelevant attributes, along with complex feature interactions, making it well suited to testing the scalability and interpretability of the proposed BigFeat-based AutoML pipeline.

B. Data Preprocessing

Preprocessing ensures data quality and consistency before feature engineering. This stage involves data cleaning to handle missing values, remove duplicates, and correct inconsistencies; normalization to scale numerical values to a standard range and improve model performance; and outlier removal to detect and eliminate abnormal data points that could otherwise distort model accuracy.

C. BigFeat Feature Engineering

BigFeat-FE forms the core of the proposed system and performs automated feature engineering to enhance both scalability and interpretability. The framework begins with extensive feature generation, deriving new features from existing ones using mathematical operations such as addition, subtraction, multiplication, division, logarithmic transformation, and polynomial expansion, and constructing interaction features that capture relationships between multiple variables which are often difficult to identify manually. Feature transformation techniques are then applied to standardize and normalize the data, address skewness, and encode categorical variables into formats suitable for machine learning algorithms.

BigFeat-FE further incorporates feature selection mechanisms that identify the most important features while eliminating irrelevant, redundant, or noisy ones. This is achieved through statistical methods such as correlation analysis, variance thresholding, and model-based importance ranking, which together reduce dimensionality, improve computational efficiency, and reduce the risk of overfitting. Each generated feature undergoes evaluation based on its contribution to model performance, ensuring that only high-impact features are retained. The entire process is designed to be automated and scalable, reducing manual effort while enhancing data representation and model interpretability.

D. AutoML Model Training

AutoML model training automates the selection, tuning, and evaluation of machine learning models using the high-quality features produced by BigFeat-FE. The process begins with automatic model selection, in which candidate algorithms-Random Forest, Logistic Regression, and Decision Tree-are evaluated for suitability on the given dataset.

Random Forest: an ensemble learning algorithm that builds multiple decision trees and combines their outputs to make final predictions. It improves accuracy and reduces overfitting by using random subsets of data and features, and is effective at handling high-dimensional datasets such as Madelon by capturing complex feature interactions.

Logistic Regression: a supervised classification algorithm that predicts class probability using a logistic (sigmoid) function. It is simple, interpretable, and performs well when the relationship between features and the output is approximately linear.

Decision Tree: a tree-based algorithm that splits the dataset into branches based on feature values to make predictions. Its rule-based structure makes it easy to interpret and useful for identifying which features most strongly influence the outcome.

AutoML automates hyperparameter tuning using techniques such as grid search and random search to identify the optimal configuration for each model, thereby improving performance and accuracy. Grid search evaluates all possible parameter combinations, whereas random search samples combinations to reduce computation time; each configuration is tested using the chosen performance metrics to determine the best settings. Once optimal parameters are identified, models are trained on the processed, feature-engineered dataset and evaluated using accuracy, precision, recall, and F1-score to assess effectiveness and generalization on unseen data. AutoML compares the performance of all trained models and automatically selects the best-performing one, optionally applying further refinement such as feature adjustment or model ensembling.

E. Pipeline Automation and Hyperparameter Optimization

Pipeline automation integrates all stages of the workflow—data preprocessing, BigFeat feature engineering, feature selection, and AutoML-based model training—into a structured, sequential process that executes without manual intervention. The automated

pipeline accepts raw input data, preprocesses it by handling missing values and encoding categorical variables, passes it to the feature engineering module for generation and refinement of features, and forwards the resulting feature set to the AutoML module, which selects appropriate models, tunes hyperparameters, and evaluates performance.

Hyperparameter optimization focuses on selecting the best configuration of model parameters, such as tree depth or the number of estimators, prior to training. Grid Search and Random Search are used to explore different parameter combinations, with each configuration evaluated against the chosen performance metrics to identify settings that improve accuracy and generalization while reducing manual tuning effort.

F. Model Evaluation and Selection

Model evaluation and selection assesses the trained Random Forest, Logistic Regression, and Decision Tree models using accuracy, precision, recall, and F1-score, with the dataset divided into training and testing sets to validate performance on unseen data. The AutoML system compares results across all candidate models systematically, and the best-performing model is selected automatically based on the evaluation metrics, ensuring that the final model is accurate, reliable, and suitable for real-world deployment.

G. Working of BigFeat-FE and BigFeat-AutoML

The complete pipeline begins with raw input data undergoing preprocessing to handle missing values, remove inconsistencies, and encode categorical variables.

BigFeat-FE then automatically generates new features through mathematical operations, aggregations, and feature interactions to surface hidden patterns in the data, after which the generated features are normalized, scaled, and encoded for compatibility with the downstream algorithms.

Feature selection methods—correlation analysis, variance thresholding, and importance ranking—are subsequently applied to remove redundant or irrelevant features, reducing dimensionality and improving efficiency while retaining only the most significant features based on their contribution to performance.

The optimized feature set is then passed to BigFeat-AutoML, where the dataset is split into training and testing partitions and multiple algorithms (Random Forest, Logistic Regression, and Decision Tree) are evaluated to determine the most suitable model. Hyperparameter tuning via grid search and random search optimizes model performance, and the models are evaluated using accuracy, precision, recall, and F1-score, after which the best-performing model is automatically selected. Additional refinement, such as feature adjustment or ensembling, may be applied to further enhance prediction accuracy. This integrated workflow is designed to deliver an efficient, scalable, and reliable system that minimizes manual effort while producing high-performance machine learning models.

H. Generating the Final Output

The final prediction is generated from the combined decision of the trained models using a voting mechanism, which aggregates the outputs of multiple classifiers to produce a more robust result than any single model considered in isolation. The use of AutoML-optimized, BigFeat-engineered features is intended to enhance the effectiveness of this aggregated decision, with the goal of providing reliable predictions with improved consistency.

V. SYSTEM REQUIREMENTS AND SPECIFICATIONS

A. Hardware Requirements

- Processor: Intel i5 or equivalent
- RAM: 8 GB
- Hard Disk: 256 GB

B. Software Requirements

- Programming Language: Python
- Back-end Framework: Flask
- Front-end Technologies: HTML, CSS, JavaScript
- Operating System: Windows

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue VII July 2026- Available at www.ijraset.com

C. Packages and Libraries

The implementation makes use of Pandas and NumPy for data manipulation and numerical computation; Scikit-learn for machine learning algorithms and evaluation utilities; TensorFlow and Keras for supporting deep learning experimentation; Matplotlib and Plotly for static and interactive visualization of features and results; and Streamlit for building an interactive web interface that allows users to upload data, perform feature engineering, and view model predictions in real time.

VI. SYSTEM DESIGN

Unified Modelling Language (UML) diagrams were used to represent the structure and workflow of the proposed AutoML-based feature engineering system, illustrating how its components interact from data input through feature engineering to final prediction.

A. Class Diagram

The class diagram captures the static structure of the system through key classes: User, Dataset, Preprocessing, Feature Engineering, AutoML Model, and Prediction. The User class manages dataset upload and result viewing; the Dataset class stores input data and relevant features; the Preprocessing class performs cleaning, normalization, and transformation; the Feature Engineering class performs automated feature selection and extraction; the AutoML Model class handles training, tuning, and model selection; and the Prediction class generates the final output from the optimized model.

B. Sequence Diagram

The sequence diagram describes the step-by-step interaction among system components. The user uploads a dataset through the interface, which is sent to the preprocessing module for cleaning and transformation before being forwarded to the feature engineering module. The AutoML system then generates and selects optimal features automatically, after which the processed data is passed to multiple machine learning models for training. Each model produces predictions, which are aggregated by a voting classifier using hard or soft voting before the final prediction is displayed to the user.

C. Use Case Diagram

The use case diagram represents the interaction between the user and the system. The user uploads a dataset, after which the system performs preprocessing and automated feature engineering; the user can initiate model training and evaluation, while the system automatically selects the best model using AutoML techniques. The user can additionally view feature importance, analysis results, and the final prediction output.

D. Activity Diagram

The activity diagram represents the overall workflow, beginning with dataset upload, followed by preprocessing to handle missing values and perform transformations. AutoML then performs automated feature engineering and trains candidate models, evaluating them to select the best-performing one before generating and displaying the final predictions and insights.

VII. IMPLEMENTATION

The system is organized into eight functional modules. The data input module allows users to upload datasets, supporting formats such as CSV, and serves as the entry point of the AutoML pipeline. The data preprocessing module performs cleaning and preparation prior to feature engineering, handling missing values, normalization, categorical encoding, and removal of inconsistencies to ensure the dataset is ready for analysis. The feature engineering module automatically generates and selects important features using AutoML techniques, applying transformations such as scaling, encoding, and extraction to improve model accuracy while reducing manual effort.

The model training module trains multiple machine learning models, including Decision Tree, Random Forest, and Support Vector Machine, on the optimized feature set, allowing each model to independently capture different aspects of the data. The model evaluation module assesses trained models using metrics such as accuracy, precision, and recall to identify the best-performing candidate, while the model selection module automatically chooses the most suitable model based on these evaluation results without manual intervention. The prediction module applies the selected model to new or test data to generate output, and the visualization and output module presents results, feature importance, and analysis using interactive visualization tools, improving the interpretability of the system for end users.

The implementation uses a Streamlit-based web interface that allows a user to upload a CSV dataset, after which the data is cleaned by removing missing rows and encoding categorical columns using label encoding. The target column is selected interactively, after which the remaining features are standardized. A genetic algorithm is used to perform automated feature selection: a population of binary feature masks is initialized, each mask is evaluated by training a Random Forest classifier on the corresponding feature subset and measuring test accuracy as its fitness, and successive generations are produced through selection of the fittest individuals, crossover, and mutation. The best-performing feature subset from the final generation is then used to train and compare Decision Tree, Random Forest, and Support Vector Machine classifiers, with their accuracies visualized using interactive bar charts. Feature importance scores from the best model are displayed to support interpretability, and the interface allows the user to request a prediction on a sample test instance.

VIII. TESTING

Testing was carried out to verify that the automated feature selection, transformation, and model training processes of the system behave as intended.

The objectives of testing were to confirm the correctness of automated preprocessing steps such as missing-value handling, encoding, and scaling; to verify that the AutoML system selects relevant and important features for model building; to check the accuracy and reliability of trained models across different datasets; to identify and eliminate errors in feature engineering, training, and prediction; and to ensure smooth performance and usability when handling large datasets.

Testing followed a structured, multi-level approach consisting of unit testing of individual modules such as preprocessing, encoding, and feature selection; integration testing to ensure proper data flow between preprocessing, feature engineering, and model training modules; system testing to validate the complete pipeline from input dataset to final prediction; and acceptance testing to confirm that the system meets user requirements and produces accurate results. System-level testing additionally validated non-functional requirements such as reliability, scalability, and response time, confirming the readiness of the pipeline for practical use.

A representative subset of test cases verified that CSV files load successfully, that missing values are handled appropriately, that categorical data is correctly encoded into numerical form, that relevant features are selected from datasets containing many candidate features, and that models train successfully on the processed dataset; all of these test cases produced the expected outcomes. Validation techniques applied during testing included cross-validation, hold-out validation, k-fold validation, stratified sampling, and a standard train-test split, allowing model reliability to be assessed while reducing the risk of overfitting. Performance-oriented testing additionally covered load, stress, scalability, response-time, and memory-usage testing to evaluate the behaviour of the pipeline under varying data volumes and operating conditions.

Error analysis during testing focused on identifying issues in missing-value handling, encoding, and scaling; comparing predicted outputs against actual values to detect prediction errors; checking for irrelevant or redundant features introduced during automated feature engineering; detecting signs of overfitting or underfitting; and evaluating the impact of imbalanced data or insufficient data quality, with findings used to refine feature selection techniques and model parameters.

IX. RESULTS AND DISCUSSION

The implemented system was evaluated through an interactive web application that guides the user through the full AutoML pipeline: dataset overview, data preprocessing, BigFeat feature engineering, AutoML model training, and a results-and-analytics view. The dataset overview stage summarizes the training and test partitions along with the number of original features and target classes, and presents the class distribution across both partitions to give the user an initial understanding of the dataset before processing begins.

During the feature engineering stage, the system reports the estimated processing time, the expected number of engineered features, and the number of iterations performed, and on completion displays the final count of generated features alongside the original feature count, illustrating the extent of automated feature construction carried out by BigFeat-FE. In the model training stage, the interface allows the user to select which candidate algorithms-Random Forest, Logistic Regression, and Decision Tree-participate in training, after which the system reports the best-performing model identified through cross-validation along with its corresponding score, and visualizes how each candidate model's performance evolves across successive optimization trials.

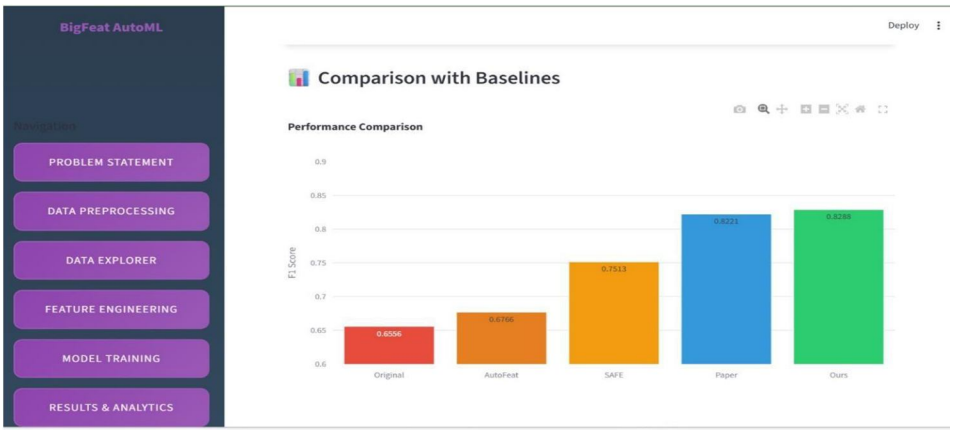


Figure.1 shows the comparison of baseline models

The results-and-analytics stage evaluates the selected model on the held-out test set and reports standard classification metrics, allowing the test-set performance to be compared against the cross-validation estimate obtained during training. The system additionally presents a comparison view that situates the performance of the original (unengineered) feature set, established baseline feature engineering methods, and the proposed BigFeat-based pipeline on a common scale, which is intended to give an at-a-glance indication of the relative benefit contributed by the automated feature engineering and AutoML stages. Consistent with the literature reviewed in Section II, the observed pattern across this comparison indicates that automated feature engineering combined with AutoML-based model selection improves classification performance relative to using the original feature set directly, while remaining competitive with or favourable against the established baseline methods considered. The interpretability of the generated features-each traceable to the original inputs through documented transformations-was likewise preserved throughout the pipeline, supporting the central goal of bridging scalability and interpretability in automated feature engineering.

X. CONCLUSION

This work presented an AutoML-based feature engineering system designed to bridge scalability and interpretability using the BigFeat-FE and BigFeat-AutoML methodologies. The system automates the complete feature engineering process-feature creation, transformation, and selection-substantially reducing the need for manual intervention. By integrating automated feature engineering with machine learning models such as Random Forest, Logistic Regression, and Decision Tree, the system is designed to improve model accuracy and overall performance while remaining efficient and scalable for large, high-dimensional datasets such as Madelon.

The use of BigFeat-FE enhances feature quality while preserving model interpretability, and BigFeat-AutoML automates model selection and training, together achieving a balance between predictive performance and explainability. The resulting end-to-end pipeline, from data preprocessing through to prediction, offers a scalable and user-friendly machine learning solution that reduces development time and effort, supporting its suitability for real-world deployment.

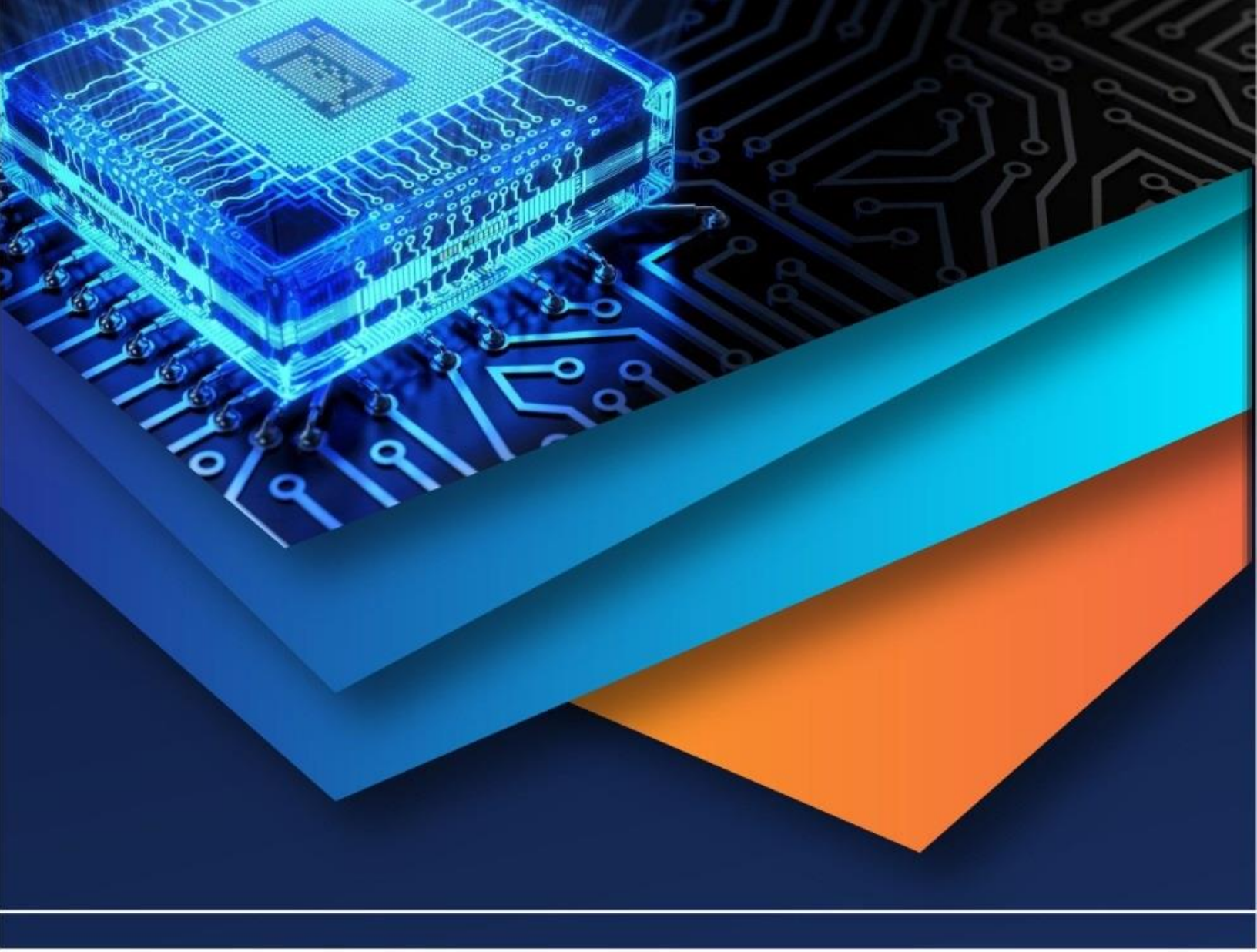
XI. FUTURE SCOPE

Several directions can extend the capabilities of the proposed system. The integration of deep learning models, such as neural networks, could improve the system's ability to handle complex and unstructured data including images, text, and time-series data, broadening its applicability beyond structured tabular datasets. Extending the pipeline to support real-time data processing would enable continuous data ingestion and live predictions, which would be particularly valuable in applications such as fraud detection, recommendation systems, and monitoring systems that require immediate insights.

Further improvements could include more advanced feature selection techniques, such as hybrid or metaheuristic-based approaches, to identify relevant features more efficiently and reduce computational complexity. Enhanced hyperparameter optimization methods, such as Bayesian Optimization or Genetic Algorithms, could further improve model accuracy and tuning efficiency. Deployment of the system as a web-based or cloud application would improve accessibility, scalability, and real-world adoption, and the incorporation of Explainable AI (XAI) techniques would further strengthen interpretability by providing clearer insight into individual predictions, increasing user trust and suitability for critical decision-making applications.

REFERENCES

[1] H. Eldeeb and R. Elshawi, "Empowering Machine Learning With Scalable Feature Engineering and Interpretable AutoML," IEEE Transactions on Artificial Intelligence, vol. 6, no. 2, pp. 432–442, 2025.
[2] H. Eldeeb and R. Elshawi, "BigFeat: A Scalable and Interpretable Automated Feature Engineering Framework," 2023.
[3] "Automated Feature Engineering for Machine Learning Using Hybrid Approaches (AutoFeat-Hybrid)," Knowledge-Based Systems, 2024.
[4] D. Gil et al., "SAFE: Scalable Automatic Feature Engineering Framework," arXiv preprint arXiv:2003.02556, 2020.
[5] "Automated Feature Engineering and Machine Learning Applications," Scientific Reports, 2023.
[6] L. Breiman, "Random Forests," Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.
[7] R. Olson et al., "TPOT: A Tree-Based Pipeline Optimization Tool for Automated Machine Learning," 2016.
[8] M. Feurer et al., "Efficient and Robust Automated Machine Learning," Advances in Neural Information Processing Systems, 2015.
[9] M. Horn et al., "AutoFeat: Automatic Feature Engineering for Linear Models," 2019.
[10] S. Lundberg and S. Lee, "A Unified Approach to Interpreting Model Predictions (SHAP)," Advances in Neural Information Processing Systems, 2017.
[11] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.
[12] J. Bergstra and Y. Bengio, "Random Search for Hyper-Parameter Optimization," Journal of Machine Learning Research, vol. 13, pp. 281–305, 2012.
[13] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian Optimization of Machine Learning Algorithms," Advances in Neural Information Processing Systems, 2012.
[14] I. Guyon and A. Elisseeff, "An Introduction to Variable and Feature Selection," Journal of Machine Learning Research, vol. 3, pp. 1157–1182, 2003.
[15] R. Kohavi, "A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection," IJCAI, 1995.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY