WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

AUTOMATED MACHINE LEARNING SYSTEM USING ENHANCED GENETIC ALGORITHM

Dr. J MADHUSUDANAN[1],Gokul R[2],Niranjan S[3], Parameshwaran N[4],Pranavesh S[5]

[1] Professor – Department of Artificial Intelligence and Data Science Sri Manakula Vinayagar Engineering College Puducherry, India
madhu**@smvec.ac.in**
[2] Bachelor of Technology – Artificial Intelligence and Data Science Sri Manakula Vinayagar Engineering College Puducherry, India
gokulstarbolt@gmail.com
[3] Bachelor of Technology – Artificial Intelligence and Data Science Sri Manakula Vinayagar Engineering College Puducherry, India
niranjan011003**@gmail.com**
[4] Bachelor of Technology – Artificial Intelligence and Data Science Sri Manakula Vinayagar Engineering College Puducherry, India
n.parameshwarancdm@gmail.com
[5] Bachelor of Technology – Artificial Intelligence and Data Science Sri Manakula Vinayagar Engineering College Puducherry, India
pranavesh25@gmail.com

ABSTRACT :

Machine Learning is one of the subfields of Artificial Intelligence that has taken almost every domain by storm. It has consistently gained favor among large groups of people, typically researchers or students looking to advance in their respective domains. The primary limitation one may face in Machine Learning is the knowledge to implement an efficient model for their specific use case. This is where AutoML comes in handy. The main objective of AutoML is to build ready-to-deploy machine learning models for any given datasets. All one needs is the dataset required for their use case, and our AutoML model handles the rest. Initially, the dataset is preprocessed through numerous automated cleansing functions without compromising the accuracy of the dataset. After preprocessing, samples from random parts of the dataset are evaluated upon various functions, and with the help of PandasAI library, which is pretrained with hundreds of defined prompts to select the most suitable variables for training. Upon choosing the training set our versatile AutoML AI network, works upon numerous iterations to find the best possible model for that dataset. Followed by training the dataset over the chosen model on a cloud-based system. The final trained model is made accessible to the user by providing APIs to feed input and retrieve output in the form of JSON. This project would greatly benefit small scales websites and businesses and provide much more efficient access to build scalable machine learning models.

Introduction

In the realm of data science, machine learning (ML) has revolutionized the way organizations process, analyze, and draw insights from data. However, effective implementation of ML often requires significant technical expertise, substantial time investment, and a deep understanding of each stage in the ML pipeline. For a typical machine learning project, these stages encompass data preprocessing, feature selection, model selection, and rigorous evaluation. Each stage introduces complexities that necessitate specialized knowledge—especially in handling data irregularities, choosing appropriate algorithms, and configuring hyperparameters. Moreover, to maintain ML models relevance and performance, constant monitoring and retraining are required.

Despite the rise in demand for ML applications, many businesses, research institutions, and individuals face challenges due to a shortage of expertise in this domain. Automated Machine Learning (AutoML) has emerged as a groundbreaking solution that mitigates these challenges by providing automated pipelines that simplify the development of ML models. AutoML platforms guide users through essential tasks without requiring a deep technical background, facilitating faster deployment of ML solutions. Recent years have seen the introduction of several AutoML tools, each varying in scope, flexibility, and accessibility. These platforms enable users with limited programming knowledge to leverage powerful ML algorithms and techniques efficiently. As the popularity of AutoML grows, so does the demand for user-friendly, no-code solutions that democratize ML and allow broader access across fields. This project builds upon the principles of AutoML, aiming to create an accessible, no-code machine learning platform that accommodates users' needs, regardless of their technical expertise.

Motivation

The motivation behind this AutoML platform originates from the challenges non-experts face in building effective machine learning models. Traditional ML development involves numerous complex tasks: selecting the appropriate algorithms, preparing and cleaning data, engineering relevant features, tuning hyperparameters, and ensuring the model generalizes well to new data. These tasks are daunting for users without a technical background. For

businesses, this complexity often translates into a significant time and financial investment, as they need to employ ML engineers and data scientists to tackle the intricate processes of model development and deployment. Even among experts, many time-consuming processes, like feature engineering and model selection, could be automated, allowing them to focus on higher-level tasks and refining model insights rather than on repetitive operations.

A key driving factor behind AutoML's rise is the increasing demand for data-driven insights across industries. From healthcare to finance and beyond, fields have come to rely on ML's ability to provide predictive insights. A user-friendly, no-code AutoML platform enables smaller businesses, independent researchers, and data enthusiasts to harness the power of ML. With accessibility in mind, the AutoML platform presented in this paper is designed to provide all users—whether seasoned data scientists or non-experts—with a streamlined, efficient pipeline that covers the ML workflow from data preprocessing to model evaluation. This system thus addresses a critical need for accessible and cost-effective solutions in ML, promoting a greater democratization of data science.

Objective

The objective of this project is to create a fully automated machine learning (AutoML) platform that simplifies and automates the ML pipeline, delivering an end-to-end solution for users with varying levels of expertise. Specifically, the platform aims to handle key stages in the machine learning pipeline: data preprocessing, feature selection, model selection, and evaluation. By automating these steps, the platform seeks to provide a user-friendly interface that allows individuals and organizations to build and deploy ML models quickly and effectively. The core functionalities of the platform include automated data cleaning and preprocessing, feature selection through a genetic algorithm, a model selection component driven by a decision tree, and an evaluation module that ensures model accuracy aligns with realistic benchmarks.

Designed as a no-code solution, this platform utilizes Python for backend processes and Streamlit for an accessible frontend. It empowers users to upload their datasets, specify target variables, and, without any coding, obtain a trained model ready for deployment. The system also aims to optimize user time by incorporating multithreading in the model selection process, enabling concurrent evaluation of multiple models. By predicting suitable models based on data characteristics, the decision tree model reduces guesswork, offering an efficient and data-driven approach to model selection. In summary, this AutoML project aspires to create an intuitive, accessible, and accurate machine learning pipeline for users, regardless of their technical backgrounds, thus advancing the reach and impact of machine learning across diverse fields and applications.

Problem Statement

Machine learning (ML) has become an integral tool across industries, transforming raw data into actionable insights. However, deploying a successful machine learning model often requires extensive knowledge and experience. The ML pipeline encompasses various complex stages, from data preprocessing and feature engineering to model selection and evaluation, making it inaccessible for those without substantial technical expertise. This project addresses the need for a simplified, automated, and user-friendly machine learning (AutoML) platform that can accommodate users at all experience levels. The AutoML system presented here aims to provide a no-code, end-to-end machine learning solution that enables users to seamlessly go from raw data to an optimized model ready for deployment.

Complexity of the Machine Learning Pipeline

One of the primary challenges in the ML workflow is the complexity associated with various tasks that require specific skills. These tasks are vital to creating an effective model, and they include:

- **Data Preprocessing**: Raw data is often filled with inconsistencies, missing values, or outliers, which must be addressed to avoid misleading model results. Data cleaning, normalization, and encoding are essential steps in ensuring the data is suitable for training.
- **Feature Selection**: Identifying the most relevant features within a dataset is crucial for improving model accuracy and efficiency. This is a task that typically requires expert intuition and domain knowledge, making it difficult for beginners.
- **Model Selection**: Each machine learning task, such as classification, regression, or clustering, requires a different set of algorithms. Selecting the optimal model architecture that aligns with the dataset's properties is often time-consuming and demands a deep understanding of algorithms and their hyperparameters.
- **Model Evaluation**: Evaluating a model's performance is essential to ensure it generalizes well to unseen data. Techniques such as cross-validation and metrics like accuracy, precision, and recall must be carefully chosen and interpreted to assess the model's effectiveness accurately.

These stages demand domain expertise, high computational resources, and significant time. The expertise and time requirements make ML inaccessible to individuals and organizations without dedicated ML teams. Even experienced professionals often spend a considerable amount of time on repetitive tasks. Hence, there is a strong need for automation in the ML pipeline to make model building more efficient and accessible.

Current Limitations in Existing AutoML Solutions

While AutoML has gained traction, existing solutions often fall short in flexibility, usability, and transparency. Some AutoML platforms rely heavily on proprietary algorithms or specific ML libraries, which can limit customization and integration with other tools. Others may cater predominantly to classification or regression tasks without sufficient flexibility for users working with unique or complex datasets. The high-level automation often provided by commercial AutoML solutions may also prevent users from understanding the underlying decision-making process in model selection and feature engineering, reducing transparency and user trust.

Moreover, many AutoML tools are computationally intensive, requiring powerful hardware and sometimes even cloud-based services to function optimally. For small businesses, independent researchers, and enthusiasts, the high cost of hardware or cloud resources is prohibitive. Additionally, existing solutions may not be suitable for users who require highly specialized configurations or insights into each stage of the ML pipeline. This gap presents an opportunity for an AutoML solution that balances accessibility, transparency, efficiency, and performance, allowing users from diverse backgrounds to build high-quality ML models.

Goals of the Proposed AutoML System

To address the above challenges, the objective of this AutoML project is to create a platform that allows users with minimal technical knowledge to navigate the entire ML pipeline—from data preprocessing and feature selection to model selection and evaluation—without coding. This platform is designed to address several key requirements:

- **End-to-End Automation**: Each stage of the ML pipeline will be automated, enabling users to upload a dataset, specify a target variable, and receive a trained model ready for deployment. This approach will allow users to bypass intricate details, creating a seamless workflow.
- **User-Friendly Interface**: Using Streamlit as the frontend framework, the platform will offer a clean and intuitive user interface. Users can upload datasets, track progress, and view model recommendations without engaging in coding, making the experience accessible to users with little to no technical background.
- **Feature Selection using Genetic Algorithms**: Feature selection will be handled through a genetic algorithm, which evaluates combinations of features to identify those that provide the most predictive power. This approach will allow the system to autonomously determine the most significant features without requiring user intervention, improving model accuracy and reducing computation.
- **Model Selection through a Decision Tree**: The platform will leverage a pre-trained decision tree model to recommend the most suitable machine learning algorithms based on the dataset's characteristics. The decision tree model is trained on a synthetic dataset crafted under human supervision, enabling it to make accurate recommendations across various types of ML tasks.
- **Multi-Threaded Model Training**: Once models are selected, they will be trained concurrently using multithreading to reduce the overall computation time. This feature not only improves efficiency but also allows the platform to provide faster results, which is especially beneficial for larger datasets.
- **Performance Evaluation and Calibration**: Model evaluation will include mechanisms for adjusting performance metrics to ensure they fall within realistic ranges, preventing overfitting or underfitting. By making adjustments to scores that appear artificially high or low, the platform will aim to produce models that perform well on unseen data.
- **Transparency and Explainability**: Unlike many black-box AutoML solutions, this platform provides users with insights into each stage of the ML pipeline. This transparency fosters greater trust and helps users understand the model development process, even if they are not directly involved in coding or algorithmic decisions.

Proposed System

The proposed AutoML system aims to provide an end-to-end, user-friendly machine learning platform that automates each stage of the ML pipeline. Designed with a no-code interface, this AutoML platform empowers users with minimal technical knowledge to upload data, select a target variable, and obtain an optimized model. The system leverages Streamlit for a clean and intuitive frontend experience, a decision tree model for automated model selection, and genetic algorithms for feature selection. This section will describe the system architecture, data sources, and key functional components to outline the platform's capabilities and design.

System Architecture

The system architecture of the proposed AutoML platform is organized into four main layers:

- **User Interface Layer**: Built with Streamlit, this layer serves as the frontend, allowing users to upload datasets, specify target variables, and interact with the system. Streamlit provides an accessible interface that guides users through the ML workflow, displaying model recommendations, progress, and performance metrics.
- **Data Processing Layer**: This layer is responsible for handling data preprocessing and feature selection. Automated preprocessing functions clean the data by managing missing values, encoding categorical variables, and normalizing numerical features. The genetic algorithm within this layer selects relevant features by evaluating various combinations, ensuring that only impactful features are retained to optimize model accuracy.
- **Model Selection and Training Layer**: This layer contains the pre-trained decision tree model, which recommends suitable ML algorithms based on the dataset characteristics. Once models are recommended, they are trained concurrently using multithreading, optimizing efficiency by reducing overall computation time. Each model is then evaluated, and performance metrics are adjusted to ensure reliability.
- **Output Layer**: After training and evaluation, this layer generates the final model with an accuracy report. Users can download the trained model in a.pickle format, ready for deployment, along with information on the model's performance.
- This layered approach enables the platform to execute tasks in a structured and efficient manner, enhancing its scalability and usability.

Data Source

To ensure the system's adaptability to a wide range of datasets, the AutoML platform can accommodate structured data in various formats, including CSV, Excel, and SQL databases. For training the internal decision tree model used in model selection, a synthetic dataset was generated under human supervision. This dataset represents a variety of possible data distributions, target variable types (classification or regression), and complexity levels. It allows the decision tree model to make informed recommendations, even with datasets it has never encountered.

The synthetic dataset includes features representing essential properties of typical datasets, such as the number of outliers, dataset size, class distribution, and balance, which guide the decision tree's recommendations for optimal model types. This synthetic data ensures that the decision tree model remains generalized and versatile, performing well across different dataset configurations and structures.

For real-time applications, users provide their own datasets, which undergo automated preprocessing and feature selection before moving to model training. This adaptability allows users from diverse domains—such as healthcare, finance, and retail—to use the platform, regardless of the specific nature of their datasets.

Key Functional Components

The AutoML system includes several key functional components that enable it to execute the ML pipeline efficiently and accurately. These components handle everything from data preprocessing to model evaluation, ensuring that each stage is optimized for a seamless user experience.

Automated Data Preprocessing

Data preprocessing is crucial to ensuring data consistency and quality before model training. The system's preprocessing module handles missing values by employing mean, median, or mode imputation, depending on the data type and distribution. Categorical variables are automatically encoded using techniques like one-hot encoding for nominal data or ordinal encoding for ordered data. Numerical columns undergo normalization to scale the data uniformly, allowing algorithms to process it more effectively.

This automation ensures that users do not need to perform manual data cleaning or transformations, a process that can otherwise be labor-intensive and prone to errors.

Feature Selection using Genetic Algorithm

Feature selection plays a vital role in enhancing model performance by eliminating irrelevant or redundant features. The platform employs a genetic algorithm for feature selection, which optimizes feature combinations based on a fitness function that leverages information gain. The genetic algorithm evaluates feature combinations as binary genomes, selecting or discarding features based on their contribution to prediction accuracy. This approach refines the dataset, improving efficiency and reducing training time while enhancing model accuracy.

Automated Model Selection with Decision Tree Model

The system uses a pre-trained decision tree model to recommend suitable ML algorithms based on specific dataset characteristics. The decision tree model was trained using a synthetic dataset with features such as dataset size, presence of outliers, class distribution, and target type (classification or regression). These features are used as input to the decision tree, which outputs a recommendation in the range of 0-7. Each output corresponds to a particular set of models that align with the dataset's characteristics, ensuring that the recommended algorithms are well-suited to the problem.

This decision tree-based recommendation process saves users the effort of manually selecting models or tuning hyperparameters, which would require expertise in ML algorithms and data analysis.

Concurrent Model Training with Multithreading

To accelerate the training process, the system uses multithreading, allowing multiple models to be trained simultaneously. Each recommended model is initialized and trained in a separate thread, reducing computational load and saving time, particularly for larger datasets. This concurrent training approach also ensures that the platform is responsive, providing timely feedback to users on model performance.

Performance Evaluation and Calibration

After training, the models undergo a performance evaluation process, where metrics such as accuracy, precision, recall, and F1-score are computed to assess their effectiveness. The system includes a calibration mechanism that adjusts scores to realistic ranges, preventing any extreme values from misleading users. For instance, if a model's accuracy is exceptionally high, the system may adjust the metric within an expected range to avoid overfitting risks. This calibration builds user confidence in the model's robustness and reliability.

Model Export and Deployment

Upon completing training and evaluation, the system saves the best-performing model in .pickle format, which users can download for deployment in their applications. This ready-to-use format simplifies model deployment, allowing users to integrate their models into production systems seamlessly.

Methodology

The AutoML system methodology is developed to enable automated processing, model selection, and optimization across various stages of the machine learning pipeline. The system automates preprocessing, feature selection, model recommendation, concurrent training, and performance evaluation. Each component is meticulously crafted to support users through a seamless ML workflow.

Workflow Overview

The AutoML platform can be broken down into six main stages:

- User Data Input and Preprocessing
- Feature Selection with Genetic Algorithm
- Model Selection via Decision Tree
- Concurrent Model Training with Multithreading
- Model Evaluation and Calibration
- Model Export and Deployment

This section details each stage and the methods applied within.

User Data Input and Preprocessing

Users initiate the process by uploading their dataset through the Streamlit interface, specifying the target variable for prediction. This dataset undergoes preprocessing, an essential step to clean, encode, and standardize data. Preprocessing ensures that each feature is compatible with the model training requirements.

Handling Missing Values

Missing values are handled based on data type:
Numerical values: Replaced by mean or median imputation.

Categorical values: Replaced by mode or most frequent values.

Mathematically, if Xi is a missing numerical feature, it is replaced by:

$$X_i = \frac{\sum_{j=1}^{n} X_j}{n} \quad \text{or} \quad \text{median}(X)$$

Encoding Categorical Variables
Categorical data is encoded to be numerically interpretable by models. For example:
One-hot encoding for nominal variables.

Ordinal encoding for ordered variables.

Normalization
Normalization scales numerical data to a common range (e.g., [0,1]), improving model performance, especially for distance-based algorithms. The min-max scaling formula is:

$$X' = \frac{X - X_{\min}}{X_{\max} - X_{\min}}$$

After preprocessing, the dataset is ready for feature selection.
Feature Selection using Genetic Algorithm
Feature selection is crucial for improving model efficiency and interpretability. The system employs a genetic algorithm (GA) to select the most relevant features, balancing between accuracy and computational cost.
Genetic Algorithm for Feature Selection
A GA works by simulating natural selection, iterating over generations to optimize a solution. Each feature subset (chromosome) is represented as a

binary vector, where 1 indicates feature inclusion and 0 indicates exclusion.
Initialization: Randomly generate initial chromosomes.

Fitness Evaluation: Calculate fitness scores for each chromosome.

Selection: Choose chromosomes for mating based on fitness.

Crossover and Mutation: Generate offspring by combining genes and mutating.

New Generation: Repeat steps until convergence.

Fitness Function using Information Gain
The fitness function evaluates each chromosome based on information gain (IG), which measures how much a feature contributes to predicting the target. For a target Y and a feature X, information gain is:

$$IG(Y, X) = H(Y) - H(Y|X)$$

where H(Y) is the entropy of Y, defined as:

$$H(Y) = -\sum_{i=1}^{k} p(y_i) \log_2 p(y_i)$$

The GA selects features maximizing information gain, leading to optimized subsets of predictive features.

Model Selection via Decision Tree

A decision tree model is pre-trained on a synthetic dataset to recommend suitable ML models based on dataset characteristics. This synthetic dataset includes metadata such as class distribution, size, and presence of outliers, helping the decision tree make recommendations tailored to dataset properties.
Decision Criteria
The decision tree considers several criteria:
Target Type: Classification or regression.

Class Distribution Balance: Whether class labels are balanced.

Presence of Outliers: Degree of outlier influence.

Dataset Size Category: Small, medium, or large.

Each criterion is a node in the tree, guiding the decision based on conditions like binary splits or threshold comparisons. For example, if the target type is classification and the class distribution is balanced, models like logistic regression or random forest classifier may be recommended.

Model Recommendation Process

The system assigns a value (0-7) based on the decision tree's recommendations, mapping to specific model sets, such as:

$$\text{Result} = f(\text{dataset characteristics})$$

where f is the decision tree function, outputting values representing models like linear regression, random forest, or XGBoost.

Concurrent Model Training with Multithreading

Once models are selected, they are trained concurrently using multithreading to enhance efficiency, particularly for large datasets or multiple models.
Multithreading for Efficiency
Each model is trained in a separate thread:
A thread pool is created with a predefined number of threads.

Each thread handles a model training process.

Threads run concurrently, reducing overall training time.

In Python, threading implementation allows asynchronous training. If T is the training time for each model and N is the number of threads, the effective time T′ becomes approximately:

$$T' \approx \frac{T}{N}$$

Model Initialization and Hyperparameter Tuning

Each model is initialized with default or lightly tuned hyperparameters:
RandomForest: n-estimators and max depth.

XGBoost: learning rate and number of rounds.

These models are trained and validated on the test dataset, with the training process handled asynchronously.

Model Evaluation and Calibration

Once trained, the models undergo a comprehensive evaluation process to assess accuracy, robustness, and reliability.
Performance Metrics
The system calculates performance metrics based on model type:
Classification: Accuracy, precision, recall, F1-score.

Regression: Mean squared error (MSE), mean absolute error (MAE), R-squared.

For classification:

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Predictions}}$$

For regression:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Calibration of Results

To prevent overfitting biases, model performance scores are calibrated:
Accuracy is capped realistically if it exceeds predefined thresholds.

Extremely high accuracy (near 100%) are adjusted to prevent overconfidence, especially for unbalanced datasets.

For instance, if a model achieves perfect accuracy on the training set, it is recalibrated within a 98–99% range to reflect real-world feasibility.

Model Export and Deployment

Upon evaluation, the best-performing model is saved in a pickle format for easy deployment. Users can download the model and deploy it directly in production environments.

Export in Pickle Format

Serialization using pickle allows the model object to be saved and reloaded with minimal overhead. The pickle format ensures compatibility across Python environments:

Save model as: pickle.dump(model, filename)

This file contains the model's learned parameters, enabling it to predict without re-training.

User Download and Deployment

The user downloads the pickle file and integrates it within their application environment. This allows the model to be used in production systems with real-time data for predictive analytics.