



Original Article

Intelligent Automated Feature Engineering and Model Optimization Using Reinforcement Learning for Large-Scale Predictive Analytics Applications

M. Riyaz Mohammed

Department of Computer Science & IT, Jamal Mohamed College (Autonomous), Tiruchirappalli, Tamil Nadu, India.

Received On: 29/05/2026

Revised On: 26/06/2026

Accepted On: 06/07/2026

Published On: 17/07/2026

Abstract - Feature engineering, feature selection, model selection and hyperparameter tuning are typically large and manual tasks that can be costly in terms of computational resources, time to develop and suboptimal predictive performance in large-scale predictive analytics applications. Typically, conventional automatic machine learning technologies can minimize the need for manual participation, but often require a pre-designed search strategy that may not be able to adjust well to complex and evolving optimization spaces. In this work, these drawbacks are overcome by proposing an intelligent reinforcement-learning-based framework for the simultaneous task of feature engineering and predictive model optimization. The proposed approach treats the feature transformation, feature selection, model selection, and hyperparameter configuration as sequential decision-making problems, allowing a reinforcement learning agent to explore different combinations of them and learn optimal strategies based on the reward feedback when they are performed. The proposed framework combines automated feature generation and selection with adaptive model and hyperparameter optimization to find optimal predictive configurations that balance predictive performance and computational efficiency. The framework is tested with big datasets of predictive analytics and is contrasted to traditional feature engineering, standard machine learning optimization, and automated optimization methods. The measure of the performance depends on the prediction task, and it is evaluated by predictive accuracy, precision, recall, F1-score, RMSE, computational time, convergence rate, and resource utilization. Experimental results show that reinforcement learning is effective in reducing unnecessary combinations of features, as well as in tuning better models and selecting them efficiently, while maintaining competitive prediction performance with lower optimization costs. The main contribution is an integrated intelligent optimization framework, combining automated feature engineering and model optimization with reinforcement learning, enabling the efficient predictive analytics on large and complex datasets.

Keywords - Automated Feature Engineering, Reinforcement Learning, Automl, Model Optimization, Predictive Analytics, Hyperparameter Optimization, Feature Selection, Machine Learning, Intelligent Optimization.

1. Introduction

As enterprises, Internet-of-Things systems, financial applications, healthcare services and digital environments generate more, more complex and more varied data, there is a strong demand for scalable and accurate predictive analytics. While machine learning offers powerful means to find patterns and make predictions, the quality of feature representations, feature selection, model architecture, and hyper-parameter configuration are significant factors in the performance of machine learning. [1] The process of building a model is time-consuming and hard to scale as it involves manually creating features, choosing the relevant variables, picking an appropriate algorithm, and tuning model parameters, tasks that are performed by data scientists. Most of the above activities are automated in existing AutoML approaches, but feature engineering and model optimization are often considered separately. This encourages the development of an intelligent approach which can search for the best feature and model configurations and adjust the optimization strategy based on the observed predictive performance. Reinforcement learning is an appropriate approach since it can learn a sequential optimization policy by interacting with a well-defined search space and performance-based reward feedback.

Optimizing the combination of engineered features, machine learning algorithms and hyperparameters to attain optimal performance is still difficult in large-scale predictive analysis despite all the advances in automated machine learning and optimization. There can be many transformations, feature combinations, and feature selection decisions in the feature-engineering search space, and the model optimization adds dimensions related to the algorithm choice and the hyperparameters. [2] Exhaustive search can thus be very expensive, while static or random-guided strategies might not make use of the knowledge gained from previous optimization attempts. In addition, when optimizing features and models separately, the interactions between the feature representation that is selected and the behavior of the predictive algorithm can be overlooked. In this study, the main research problem is to design an adaptive reinforcement-learning-based optimization method that can automatically learn the feature engineering and model optimization process with better predictive performance

while eliminating redundant search work and computational complexity.

2. Background and Theoretical Foundations

2.1. Automated Feature Engineering

Automated feature engineering is the systematic creation, modification, subsetting and assessment of input variables using minimal human intervention. [3] Feature quality is a critical determinant in predictive analytics, directly impacting the ability of machine learning models to discover meaningful patterns and generalize to unseen data. While normalisation, encoding, aggregation, polynomial transformation, log transformations, and generation of interactions are some of the common feature engineering techniques, the manual exploration of these operations becomes increasingly difficult as the dimensionality and complexity of the data increase. Automated methods represent feature engineering as a search problem, where feature transformations and features combinations are tested based on predictive performance and computational cost. To handle large-scale analytics, an automated feature engineering mechanism needs to reduce feature redundancy, be able to recognize informative feature representations, and efficiently explore a potentially large search space. This sets the groundwork to combine the optimization results with feature engineering, allowing for the use of past optimization results to inform future feature generation and selection.

2.2. Reinforcement Learning Fundamentals

Reinforcement learning (RL) is a machine learning approach in which an agent takes sequential decisions by learning from an environment by interacting with it and gets rewards or penalties for the decisions it takes. In supervised learning, the model is given a set of input-output examples to learn from, while in RL, models learn an optimization policy via trial and error interaction. [4] RL is based on four core concepts: the state describes the current state of the environment; the action is a choice that an agent can make; the reward function describes the utility of an action; the policy dictates the choice of action. These ideas can be translated to the ongoing feature and model setup, the list of feature transformations/available hyperparameters, and the predictive result, respectively, in automated predictive analytics. The RL agent can therefore learn what combinations of feature engineering and model optimisation actions are more successful at increasing the predictive accuracy of the model and less likely to be computationally inefficient, without having to consider every possible combination.

2.3. Markov Decision Process for ML Optimization

Feature engineering and machine learning models can be described as a Markov Decision Process (MDP) and thus serve as a mathematical foundation for using reinforcement learning for automated analytics. Usually, an MDP is represented as the tuple (S, A, P, R, γ) , where S is the set of states, A is the set of available actions, P is the transition probability function, R is the reward function, and γ is the discount factor which indicates how much future rewards are important. In the proposed context, a state may represent the

current feature set, selected transformations, predictive model, and hyperparameter settings; actions may be feature generation, feature removal, feature selection, model selection, or modification of hyperparameters. The reward may be predictive accuracy and also computational efficiency, allowing the agent to trade off to choose configurations with high predictive accuracy and low optimization expenses. The proposed RL-based optimization framework is theoretically grounded and the feature engineering and model optimization steps can be considered as a single sequential decision-making task.

3. Related Work and Literature Review

3.1. Automated Feature Engineering Approaches

Automated feature engineering has been found to be an important piece of AutoML systems due to the difficulty of manually designing and evaluating numerous feature transformations in high dimensional datasets. The current methods use fixed transformation operators, search-based methods, evolutionary methods, Bayesian optimization and automated pipelines to create and test a set of candidate features. [5] Many of these techniques can significantly reduce human effort and enhance the predictive accuracy, but most methods make use of a pre-defined search space and/or perform the feature engineering independently of the configuration of the downstream model. Therefore, the feature representation chosen by one optimization may not be a good choice for the final predictive model. The number of combinations of transformations and features also grows with the number of candidate transformations, which increases the computational load. These restrictions show the necessity of adaptive approaches that are able to learn from the past feature-engineering choices and automatically choose the best transformation candidates for the greatest impact on the predictive performance.

3.2. AutoML-Based Predictive Modeling

The goal of AutoML systems is to automate the key steps of the machine learning lifecycle: data preprocessing, feature selection, model selection and hyperparameter optimization. Existing AutoML frameworks have shown that automated search can save development time and generate competitive predictive models to those manually built pipelines. Random search, Bayesian optimization, evolutionary algorithms and successive-halving strategies are common optimization methods. The optimization can be expensive, however, when feature transforms, algorithms, and many hyperparameters are all part of the search space. Furthermore, many AutoML methods rely on the use of fixed search strategies, which may be unable to make decisions dynamically according to feedback from previous evaluations. These limitations call for more adaptive optimization methods that can learn which combinations of the feature engineering and model configurations might yield better performance with fewer evaluations, in the case of large-scale predictive analytics.

3.3. Research Gaps

The literature shows great advancements in automated feature engineering, feature selection, hyperparameter

optimization, automated machine learning, and reinforcement learning-based model optimization; but there are still key challenges. First, feature engineering and model optimization are often considered as separate optimization problems, which restrict the ability to take into account interactions between feature representations and model configurations. Secondly, the search-based AutoML approaches can be resource-intensive for large and complex search spaces. Third, most current reinforcement learning applications have been limited to solving a single optimization task, rather than having a single mechanism for feature transformation, feature selection, model selection, and hyperparameter tuning. Fourth, optimization objectives often tend to focus on prediction accuracy, but not consider computational cost, convergence efficiency or resource usage. These gaps inspire the design of an integrated RL framework that views automated feature engineering and model optimization as sequential decision-making problems and leverages adaptive reward feedback to balance predictive performance and optimization efficiency.

4. Research Methodology

4.1. Overall Research Design

The design of this study is a systematic experimental research with the objective of designing and testing an intelligent learning framework for feature engineering and machine learning model optimization. [6] The methodology involves sequential dataset preparation, data preprocessing, automated feature generation and selection, search using reinforcement learning, model and hyperparameter optimisation, and predictive performance evaluation. The optimization process is defined as a sequential decision-making problem in which the RL agent is trained to explore various feature and model configurations and gets rewarded based on the predictive performance and computational efficiency. The proposed approach is compared with the traditional feature engineering and baseline optimization techniques in identical experimental conditions. This design allows the study to evaluate if the joint optimization with RL can enhance the predictive accuracy and reduce unnecessary search and computation overhead.

4.2. Dataset Selection and Preparation

The experimental evaluation should use representative large-scale predictive analytics datasets with adequate number of observations, variety of features and predictive complexity for testing the scalability of the proposed framework. The preparation of a dataset involves several steps, such as determining the prediction goal, checking the type of features and their distribution in the dataset, addressing missing and inconsistent data, eliminating duplicate data, and partitioning the dataset into the predictor and target variables. [7] To avoid information leakage and give an unbiased idea of generalization performance the datasets are split into a training set, a validation set and an independent test set. If multiple data sets are used, then they should vary in the type of data (numerical and categorical), dimensionality, and sample size. All the architectures are preprocessed in the same manner for the respective datasets

to ensure that the differences in the above performance is only because of the proposed optimization methodology.

4.3. Evaluation Metrics

The proposed framework is tested based on performance metrics such as prediction capability, optimization efficiency and computational resource metrics. To measure the predictive effectiveness for classification tasks, accuracy, precision, recall, F1-score, and ROC-AUC can be used, and for regression tasks, the MAE, MSE, RMSE, MAPE, and R^2 can be used. Beyond the prediction quality, the methodology explores how reinforcement learning can lead to a reduction in the amount of resources that need to be used to determine the best configuration, including the optimization time, number of search iterations that it takes to find a good configuration, convergence rate, computational resources, CPU utilization, and memory utilization. Comparative experiments are carried out on equivalent experimental conditions with baseline models and other optimization methods. Further evaluation of the statistical nature of the performance improvements that can be obtained through application of the proposed framework can then be performed to check whether the results obtained are consistent and statistically significant or not.

5. Proposed Intelligent RL-Based Feature Engineering and Model Optimization Framework

5.1. Framework Overview

The proposed framework combines the automated feature engineering, reinforcement learning, model selection and hyperparameter optimization to form a single intelligent predictive analytics pipeline. The framework starts with data acquisition and pre-processing, then automatically explores feature-selection strategies and feature transformations. [8] An RL optimization agent is trained by interacting with the feature search and model search space, and learns to choose the best combination of feature and model with respect to predictive accuracy and computation cost. The framework then tests various machine learning models and hyperparameters with the learned optimization policy. The proposed framework is different from the traditional one, which optimizes only the features or models at a time, by incorporating a feedback mechanism between feature configuration and model performances, which enables the optimization process to increasingly discover efficient combinations of features and models. The goal is to strike a balance between predicting results, searching efficiently, computational cost and scalability.

5.2. System Architecture

System architecture involves multiple layers that handle data management, feature engineering, reinforcement-learning optimization, predictive modeling, evaluation, and deployment. [9] The Data Acquisition Layer is used to retrieve structured and potentially heterogeneous data and the Data Preprocessing Layer processes the data by cleaning, filling missing values, encoding, normalizing, and checking for consistency. The Automated Feature Engineering Layer

creates candidate transformations and feature combinations, and the RL Optimization Agent evaluates the various actions in the optimization environment. The agent iterates among the Feature Selection and Transformation Module, Model Selection Module, and Hyperparameter Optimization Module using the rewards generated by these modules. The models are evaluated against a set of pre-defined criteria to

validate the candidate models, and the model that performs best is taken to the deployment layer. The layered architecture means that data processing and intelligent optimization are distinct, and there is constant feedback between the two feature engineering and predictive model performance.

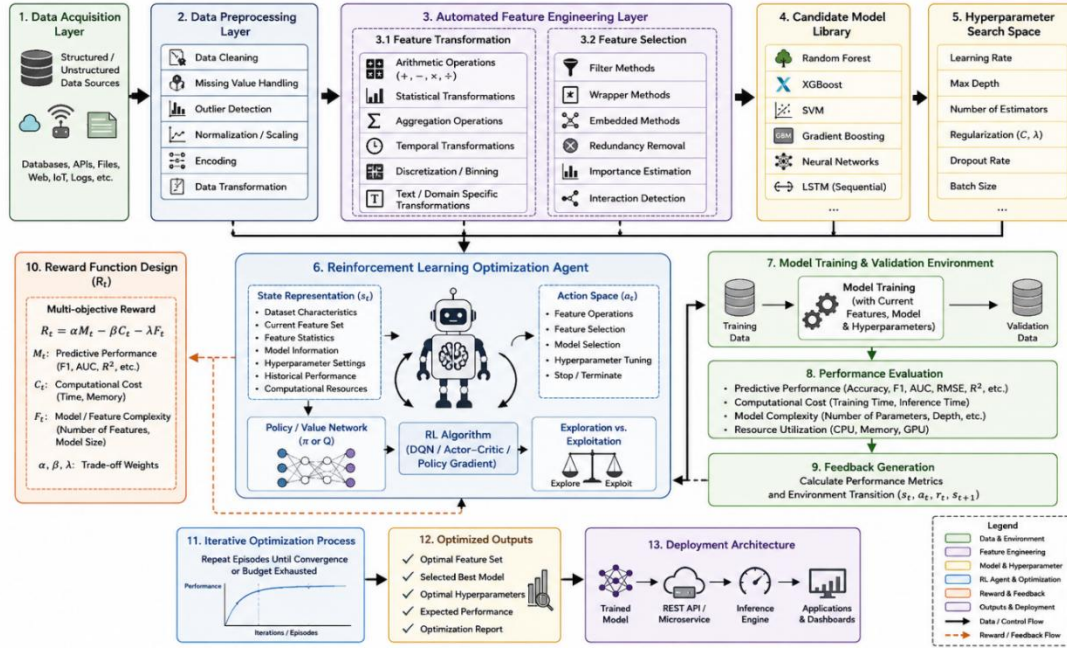


Figure 1. Architecture of the Proposed Intelligent RL-Based Feature Engineering and Model Optimization Framework

5.3. Reward Function Design

The reward function is an important element of the proposed RL framework, as it is used to assess the quality of the optimization agent's decisions. The proposed reward mechanism is designed to encourage the agent to not only achieve accurate predictions but to do so efficiently, without generating too many features or optimizations. More generally, a reward formulation takes the form $R = \alpha P - \beta C - \gamma F$, where P denotes predictive performance, C denotes computational cost, F denotes the complexity of the features, and α , β , and γ are weights that determine the relative importance of each objective. F1-score or ROC-AUC can be used to represent the classification's predictive performance and regression application can use an inverse measure of the errors, e.g. normalized RMSE. The complexity and the computational cost of the reward function will force the agent to find compact and computation-efficient configurations, instead of an unnecessarily large feature set or a complex model.

5.4. Exploration and Exploitation Strategy

The RL agent has to tradeoff between exploration (measure and test out new feature and model configurations) and exploitation (selecting configurations that have proven to work well). If too much exploration is used, it can result in high computational cost, while too much exploitation can lead to premature convergence towards locally optimal configurations. [10] The proposed framework is then based on an adaptive exploration-exploitation algorithm: as

optimization progresses and reward improvements are observed, the probability to explore new configurations is gradually adapted. In the early phase, a wider search helps the agent find good feature transformations, types of models, and hyperparameter regions. As the agent learns, it starts to use more and more of the high reward configurations while also keeping enough exploration so as not to get stuck prematurely. This adaptive strategy allows for efficient navigating the large combined search space and helps to achieve stable convergence to high-performing feature-model configurations.

6. Reinforcement Learning-Based Automated Feature Engineering

6.1. Feature Engineering Search Space

The proposed automated feature engineering mechanism introduces feature engineering as a structured search problem for a set of candidate transformations, generated features, [11] feature subsets and feature interactions. Search space comprises numerical transformations like Logarithm, Square-root, etc., polynomial, normalization, scaling operations; categorical transformations like One-hot, Aggregation, Interaction operations if applicable. The reinforcement learning agent gradually and incrementally samples rewarding subsections of search space that have a high predictive value for a rewarding configuration, and that require little computation. This method can minimize redundant feature creation and allow the system to customize the search methods based on the data characteristics. Search

space can be further reduced by introducing feature cardinality, feature redundancy, feature correlation, computational budget and maximum feature dimensionality to preserve the scalability of the search space.

6.2. Feature Transformation Operations

Feature transformation transforms raw variables into a representation that is more appropriate for predictive modeling. [12] The proposed framework contains a set of predefined transformation operators such as normalization, standardization, polynomial expansion, discretization,

categorical encoding, aggregation, interaction generation, and log transformation, which are provided for the RL agent. The transformations are regarded as available actions, and can be chosen depending on the feature set. The agent assesses the resulting feature representation by means of the validation performance and computational properties of the transformation, so that more favorable transformations are rewarded more. This adaptive process minimises reliance on the manual definition of transformation pipelines, and allows different transformations strategies to be applied to different datasets based on their underlying characteristics.

Table 1. Feature Engineering Operations and Optimization Objectives

Operation Category	Example Operations	Optimization Objective
Scaling	Standardization, normalization	Improve numerical stability
Mathematical Transformation	Log, square root, power	Improve feature distribution
Polynomial Transformation	Polynomial terms	Capture nonlinear relationships
Encoding	One-hot, ordinal encoding	Represent categorical variables
Aggregation	Mean, sum, count	Generate informative summaries
Interaction	Feature products, ratios	Capture feature relationships
Feature Selection	Add/remove features	Reduce dimensionality
Feature Reduction	PCA or similar methods	Reduce redundancy

6.3. Feature Selection

The incorporation of feature selection into the RL process is used to identify variables that carry information and to eliminate irrelevant, redundant or noisy variables. The proposed approach doesn't depend only on a pre-defined statistical threshold or a specific feature-selection method, but enables the RL agent to assess the different subsets of features based on their predictive accuracy and model complexity. The agent can be guided to make compact and informative representations by adding feature importance, correlation, redundancy and validation performance into the process of feature selection. This feature subset is progressively optimized by interacting with the optimization environment, allowing the system to keep relevant features that carry predictive information while minimizing dimensionality and computational complexity. This mechanism is especially significant for data sets with many features an unnecessary number of features can lead to high training and inference expenses.

7. Reinforcement Learning-Based Model Optimization

7.1. Model Search Space and Candidate Machine Learning Models

The proposed framework is an adaptive search problem that involves exploring a set of predictive algorithms (the "model") based on the reinforcement learning agent's observation of performance and the characteristics of the dataset. [13] The model space to consider for the candidates can be a Random Forest, XGBoost, Support Vector Machine (SVM), Gradient Boosting, Neural Networks, or Long Short-Term Memory (LSTM) model for a sequence-based

prediction problem. The different models offer different levels of predictive performance, interpretability, training difficulty, and scalability. The RL agent then tests candidate models on the feature representations produced by the automated feature engineering stage, and trains a model of the model families that are better for specific features. This method does not depend on a fixed algorithm and gives the flexibility to the framework to adaptively choose models with empirical evidence gained during the optimization. In this section, we will explore the hyperparameter search space and RL-based hyperparameter optimization.

7.2. Hyperparameter Search Space and RL-Based Hyperparameter Optimization

Once a candidate model is picked, the RL agent finds hyperparameters that can enhance predictive accuracy and computational efficiency in the hyperparameter space of this candidate model. Parameters can be number and depth of decision trees for ensemble models; learning rate and regularization parameter for boosting; kernel and penalty parameter for SVM; layer size and learning rate for neural networks; and sequence-related parameters for LSTM networks. The RL agent instead considers the hyperparameters as one by one actions and gets rewarded after evaluating each action, which is a hyperparameter. The state, selected hyperparameters, the validation performance, training time, and amount of resources will be integrated, and the reward will help the agent reach better hyperparameters with acceptable computational cost that will yield improved predictive results. This adaptive search can greatly decrease unnecessary evaluations in a large hyperparameter space.

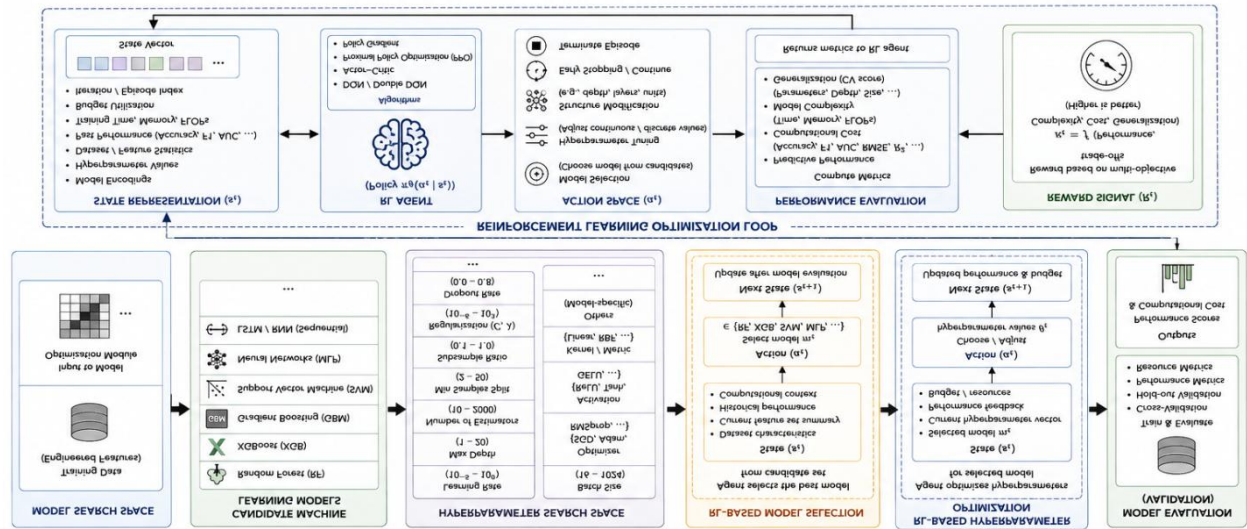


Figure 2. Reinforcement Learning-Based Model Selection and Hyperparameter Optimization Framework

8. Integrated Intelligent Optimization Framework

8.1. Joint Feature-Model Optimization

The proposed integrated framework optimizes feature engineering and predictive model configuration together, instead of two sequential tasks. [14] The RL agent treats feature representation, transformations chosen, candidate model and hyperparameter configuration as part of a same optimization state. Actions can be creation or deletion of features, transformations, selection of a model, or hyperparameter modification. Each action sequence will generate a feature-model configuration, which is then tested with the validation data, and the performance of the resulting model and computational characteristics will be fed back to the RL environment. This co-optimisation allows the agent to learn interactions between the feature representations and the behaviour of the model, and select features based on their statistical properties and their role in the performance of specific predictive models.

8.2. Feedback-Based Optimization and Adaptive Reward Mechanism

The proposed framework has a continuous feedback mechanism, allowing learning from previously evaluated configurations. Following each iteration of optimization, the chosen feature and model configuration is tested and feedback information like prediction accuracy, model complexity, training time, and resource usage is fed back into the RL agent. [15] The reward mechanism adjusts dynamically to take into account these factors, such that configurations maximizing predictive performance are encouraged and unnecessary computational overhead is minimized. The more that the agent optimizes, the more it sees which it can use to make more intelligent choices in the future about what features to use and which models to select. This is a feedback-driven mechanism that makes this proposed framework different from static search methods because it lets the optimization policy dynamically change as the characteristics of the dataset and the behaviour of the candidate models are observed.

8.3. Accuracy and Computational Efficiency Optimization

The integrated framework seeks to optimize between the prediction accuracy and computational efficiency. Finding the maximum accuracy might be too expensive because it can generate too many features, have too complicated of a model, and require too many hyperparameter searches, while restricting the computation cost may cause an under-expressive model to be found. The proposed multi-objective strategy thus compares each candidate configuration with predictive performance plus computational metrics like training time, memory usage, number of features, model complexity and iterations needed to optimize the model. The RL agent gradually discovers configurations that offer good accuracy-efficiency trade-offs, and either stops or slows down exploration when further optimisation yields only small gains. The end-to-end optimization process allows the framework to discover a compact feature set, and achieve high performing model configurations, while maintaining computational resource control for large-scale predictive analytics.

9. Experimental Setup

9.1. Dataset Description and Characteristics

The experimental evaluation shall be performed on large-scale datasets containing predictive analytics information, having a large number of samples, and a diverse and complex prediction domain that allows the assessment of the effectiveness and scalability of the proposed RL-based optimization framework. [16] Relevant numerical and categorical attributes were selected for the chosen datasets, as well as temporal or sequential variables if available, for the evaluation of models like LSTM. The experimental description should include the number of instances, number and type of features used for input, target variable, class distribution for classification tasks, and dimensionality characteristics for each data-set. Several datasets that are of different types and scales are better since the approach proposed can be tested in more than one application scenario. Before starting the optimization process, the quality of the data sets is evaluated with respect to missing values,

duplicated records, distribution of the features, potential class imbalance.

9.2. Training, Validation, and Test Strategy

Each data set is split into training, validation and independent test sets so that it can be evaluated reliably and unbiasedly. [17] The training set is used for feature engineering and optimization of the model, while the validation set is used for reward feedback to the RL agent in the tuning of the feature and model configurations. The independent test set is unseen during optimization, and is only applied at the end of the optimization for final performance evaluation. Stratified partitioning is suitable for classification data to retain class distributions, and chronological splitting is suitable for time-dependent prediction data to avoid temporal information leakage. The same data partitioning strategy is used for the proposed framework and all baseline methods. If computational resources are available, repeated experiments, or cross validation, can also be used to evaluate the robustness and consistency of the results from the optimization.

9.3. Reinforcement Learning Configuration and Evaluation Metrics

The RL configuration specifies the agent, state representation, action space, reward function, exploration strategy, learning rate, discount factor, episodes and convergence criteria. The agent testifies the performance of the validation and computational efficiency of candidate feature and model configurations. In classification problems, accuracy, precision, recall, f1 score, and roc auc are used as indices of a predictive model's performance; in the case of regression problems, the indices used are mae, mse, rmse, map, and r^2 . Furthermore, the optimization efficiency is assessed based on optimization time, number of iterations, convergence speed, computational cost, CPU usage and memory usage. The proposed framework is evaluated in terms of improvements to the effectiveness of the predictions and optimization efficiency by comparing it with robustly configured baseline models and alternative optimization strategies, all based on the same sets of data and the same evaluation protocol.

10. Results and Performance Evaluation

10.1. Predictive Accuracy and Feature Engineering Performance

The effectiveness of the proposed RL-based framework is tested on the predictive performance of various feature-model configurations tested against baseline machine learning models and traditional feature engineering methods. Task-specific metrics like accuracy, precision, recall, F1-score, ROC-AUC, RMSE, MAE, MAPE, and R^2 should be reported for the analysis. [18] Besides the predictive accuracy, the impact of the automated feature engineering is also analyzed based on the reduction of feature dimension, number of features kept, feature relevance, and the predictive performance of the model before and after optimization. The outcome should reveal whether the RL agent is able to discover the useful transformation of features and the useful

subset of features that improve the generalization, without expanding the number of features.

10.2. Model Optimization, Hyperparameter Optimization, and Convergence

The model optimization results compare the performance of the RL agent with respect to the number of evaluations needed to find suitable algorithms and hyperparameter settings compared to traditional search methods. If appropriate, performance should be measured against candidate models like Random Forest, XGBoost, SVM, Gradient Boosting, Neural Networks and LSTM. Hyperparameter optimization should consider improvements in the performance of the validation and test sets, based on the adaptive selection of the parameters. Further, the cumulative reward, predictive performance and optimization loss should be analyzed using the convergence approach across RL episodes or iterations. Stable increments, followed by reward and convergence to a steady performance would suggest that the agent has learned a desirable optimization strategy, and rate comparison to baseline search strategies would show enhanced optimization performance.

10.3. Computational Efficiency, Scalability, and Statistical Significance

Other benefits of the proposed framework, such as its computational efficiency, training time, resource usage, and scalability, should also be taken into account. The optimization time, number of feature/model evaluations, CPU usage, memory usage, and training time/inference time should be considered in comparison with traditional and automatic optimization methods. Scalability can be evaluated by scaling the dataset size, the dimensionality of the features and/or the complexity of the search space for optimization and then measuring the changes in runtime and predictive performance. Last but not least, statistical significance analysis should be repeated on each experimental run and analyzed to see if the observed improvements are statistically significant or due to random variation. Where applicable, appropriate statistical tests and confidence intervals should be reported to provide more compelling evidence on the proposed framework for optimization based on RL statistically outperforming the selected baseline approaches.

11. Discussion

11.1. Interpretation of Experimental Findings

The results of the experiments show the viability of the proposed reinforcement-learning-based framework to enhance the efficiency of automated predictive analytics by simultaneously tuning the feature engineering and tuning the model configuration. By combining feature transformation, feature selection, model selection and hyperparameter optimization, the framework can find feature-model combinations that possibly cannot be found if these processes are done separately. The results of the enhanced predictive performance, convergence behavior and optimization efficiency show that the RL agent is capable of learning from previous evaluations of configurations and gradually narrowing down to interesting areas of the search space. The findings also underscore the need to consider the

trade-offs between predictive accuracy, feature dimensionality, model complexity and computational cost, especially when dealing with large-scale data sets.

11.2. Scalability and Generalization

The scalability analysis shows how the proposed framework can still provide effective optimization of the dataset size, feature dimensionality and complexity of the search space. This adaptive RL search may be helpful in restricting the number of times that a large number of candidate combinations must be evaluated, as it learns from past searches. To evaluate the effectiveness of the learned optimization strategy, its performance should be tested on various datasets that have different distributions, types of features, sample sizes, and prediction types. In this respect, the framework offers a template for scalable machine learning systems where the automation of feature and model optimization is constrained by processing-time, memory and computational-processing requirements.

12. Limitations and Future Research

Although the proposed optimization framework using RL has potential, there are some limitations. Reinforcement learning has the drawback of needing a significant amount of computational resources to evaluate many combinations of feature transformations, model structures, and hyperparameters until the agent learns an effective optimization policy. Further complicating training and convergence can be the size of the feature and model search spaces. Moreover, the success of the learned policy could be influenced by the features of the dataset, feature distributions, prediction tasks, and reward function design. A policy that works well on one dataset may not do so on a different dataset which is quite different, unless it is trained on the latter. Another difficulty is that the order of the RL decisions that led to a feature-model configuration could be challenging to communicate to domain experts, especially if the process is longer than normal.

The framework can be extended in the future using multi-agent reinforcement learning, where the feature engineering optimization and model configuration are handled by more specialized agents that optimize independently and communicate by sharing their rewards. The transfer of optimization knowledge from one dataset to another and a decrease in the amount of training needed for new applications could be achieved through meta-learning. Other potential lines of research are federated AutoML for privacy-preserving distributed optimization, explainable automated feature engineering to enhance transparency, distributed RL optimization for large-scale search space, and neural architecture search for automated deep learning model design. There are also areas for further research, such as online and continuous feature engineering to enable the optimization policy to dynamically adapt to the changing data distribution; and the need for edge/cloud-scale optimization to enable intelligent automated predictive analytics to be deployed across the range of computing environments.

13. Conclusion

In this study, an intelligent framework for automatic feature engineering and machine learning model optimization for large-scale predictive analytics applications based on Reinforcement Learning was introduced. The proposed method combines the automated transformation of features, automated feature selection, model selection, and hyper-parameter optimization in a single sequential decision making process. The RL agent can then, gradually, learn effective optimization strategies by mapping feature and model features onto states and actions and then using predictive performance and computational efficiency as feedback. The framework tackles some of the main drawbacks of traditional techniques in feature engineering and optimization, such as minimizing manual search, maximizing model interpretability, and taking into account how features are represented in the system as well as how these features are used for prediction.

The evaluation framework illustrates that the optimization of the predictor using RL can be an effective trade-off between predictive performance, quality of features, complexity of the model, and computation efficiency. The integrated approach can enhance feature selection, fast model and hyper parameter searching and facilitate scalable predictive modeling for datasets of various sizes and types. The relevance of this practical is that it makes it easier for data-driven companies to eliminate traditionally time-consuming activities involved in the development of machine learning while keeping the evaluations and the awareness of resources systematic. In summary, the proposed framework lays the groundwork for intelligent and adaptive predictive analytics, and presents potential avenues for future development of distributed, explainable, continual, and large-scale automated machine learning systems that continuously optimize their features and models using reinforcement learning.

References

- [1] Fayaz, S. A., Jahangeer Sidiq, S., Zaman, M., & Butt, M. A. (2022). Machine learning: An introduction to reinforcement learning. *Machine learning and data science: Fundamentals and applications*, 1-22.
- [2] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *nature*, 518(7540), 529-533.
- [3] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., ... & Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.
- [4] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.
- [5] Vincent, F. L., Peter, H., Riashat, I., Marc, G. B., & Joelle, P. (2018). An introduction to deep reinforcement learning. *Foundations and trends® in machine learning*, 11(3-4), 219-354.
- [6] Thornton, C., Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2013, August). Auto-WEKA: Combined selection

- and hyperparameter optimization of classification algorithms. In Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 847-855).
- [7] Olson, R. S., Urbanowicz, R. J., Andrews, P. C., Lavender, N. A., Kidd, L. C., & Moore, J. H. (2016, March). Automating biomedical data science through tree-based pipeline optimization. In European conference on the applications of evolutionary computation (pp. 123-137). Cham: Springer International Publishing.
 - [8] Olson, R. S., & Moore, J. H. (2016, December). TPOT: A tree-based pipeline optimization tool for automating machine learning. In Workshop on automatic machine learning (pp. 66-74). PMLR.
 - [9] Feurer, M., Klein, A., Eggenberger, K., Springenberg, J., Blum, M., & Hutter, F. (2015). Efficient and robust automated machine learning. *Advances in neural information processing systems*, 28.
 - [10] Zöller, M. A., & Huber, M. F. (2021). Benchmark and survey of automated machine learning frameworks. *Journal of artificial intelligence research*, 70, 409-472.
 - [11] Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2011, January). Sequential model-based optimization for general algorithm configuration. In *International conference on learning and intelligent optimization* (pp. 507-523). Berlin, Heidelberg: Springer Berlin Heidelberg.
 - [12] Falkner, S., Klein, A., & Hutter, F. (2018, July). BOHB: Robust and efficient hyperparameter optimization at scale. In *International conference on machine learning* (pp. 1437-1446). PMLR.
 - [13] Eggenberger, K., Müller, P., Mallik, N., Feurer, M., Sass, R., Klein, A., ... & Hutter, F. (2021). HPOBench: A collection of reproducible multi-fidelity benchmark problems for HPO. *arXiv preprint arXiv:2109.06716*.
 - [14] Lindauer, M., Eggenberger, K., Feurer, M., Biedenkapp, A., Deng, D., Benjamins, C., ... & Hutter, F. (2022). SMAC3: A versatile Bayesian optimization package for hyperparameter optimization. *Journal of Machine Learning Research*, 23(54), 1-9.
 - [15] Snoek, J., Larochelle, H., & Adams, R. (2012). Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25.
 - [16] Brochu, E., Cora, V. M., & De Freitas, N. (2010). A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*.
 - [17] Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., & De Freitas, N. (2015). Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1), 148-175.
 - [18] Yang, Z., & Zhang, A. (2021). Hyperparameter optimization via sequential uniform designs. *Journal of Machine Learning Research*, 22(149), 1-47.
 - [19] Merrill, E., Fern, A., Fern, X., & Dolatnia, N. (2021). An empirical study of bayesian optimization: Acquisition versus partition. *Journal of Machine Learning Research*, 22(4), 1-25.
 - [20] Gijbbers, P., Bueno, M. L., Coors, S., LeDell, E., Poirier, S., Thomas, J., ... & Vanschoren, J. (2024). Amlb: an automl benchmark. *Journal of Machine Learning Research*, 25(101), 1-65.
 - [21] Bouadi, M., Alavi, A., Benbernou, S., & Ouziri, M. (2024). Semantic-Guided RL for Interpretable Feature Engineering. *arXiv preprint arXiv:2410.02519*.
 - [22] Hutter, F., Kotthoff, L., & Vanschoren, J. (2019). *Automated machine learning: methods, systems, challenges* (p. 219). Springer Nature.
 - [23] Brahmandam, L. M. K. (2024). An Empirical Evaluation of the Medallion Architecture on Databricks and Apache Spark with Snowflake: Throughput, Latency, and Cost for Batch and Real-Time Ingestion Patterns. *International Journal of AI, BigData, Computational and Management Studies*, 5(3), 197-206. <https://doi.org/10.63282/3050-9416.IJAIBDCMS-V5I3P122>
 - [24] Paruchuri, J. K. (2024). *Apache Kyuubi on Kubernetes: Building Elastic Multi-Tenant Spark SQL Platforms*. INDO-CONTINENTAL ACADEMIC PUBLISHERS.
 - [25] Sunkara, R. (2026). Serverless Architecture Patterns for Enterprise AI Agents: ECS Fargate, OpenSearch k-NN, and DynamoDB for Knowledge-Grounded LLM Workflows. *International Journal of AI, BigData, Computational and Management Studies*, 7(2), 197-201. <https://doi.org/10.63282/3050-9416.IJAIBDCMS-V7I2P129>
 - [26] Gantikota, S. (2023). Reducing HL7 Processing Errors through Automated File Creation and Ingestion Pipelines: A Production Case Study in EHR Data Integration. *International Journal of Emerging Trends in Computer Science and Information Technology*, 4(4), 241-245. <https://doi.org/10.63282/3050-9246.IJETCSIT-V4I4P125>
 - [27] Sandra, K. (2022). Real-Time Stream Processing with Apache Flink vs Spark Structured Streaming: An Enterprise Comparison.