World Journal of Advanced Research and Reviews

eISSN: 2581-9615 CODEN (USA): WJARAI

Cross Ref DOI: 10.30574/wjarr

Journal homepage: https://wjarr.com/

(Review Article)

Enhancing predictive model performance for data-driven software systems

Sai Krishna Reddy Mudhiganti *

Independent researcher.

World Journal of Advanced Research and Reviews, 2019, 04(02), 196-206

Publication history: Received on 02 December 2019; revised on 24 December 2019; accepted on 30 December 2019

Article DOI: https://doi.org/10.30574/wjarr.2019.4.2.0149

Abstract

Predictive models enable data-driven software systems to make useful forecasts out of imperfect inputs in areas like healthcare, finance and e-commerce. Despite this, missing or damaged data, large sets with many features and changes in the underlying patterns can cause uncertainty. To handle these difficulties, this study proposes a comprehensive pipeline with five major parts:

- K-nearest neighbours imputation, interquartile-range outlier detection and z-score normalization to clean and standardize the data;
- Adding temporal and interaction features to enrich the inputs;
- Reducing and pruning features with mutual information and recursive feature elimination;
- Letting various machine learners—decision trees, support-vector machines, gradient-boosted trees and shallow neural networks—train with optimized hyperparameters using bayesian search; and
- Stacking.

By using a regret-minimizing algorithm for online updates, the ensemble manages to adjust quickly to new data and still remains accurate with a drop of just 2% per hour. According to empirical assessments, the stacked model achieves an accuracy of 92.4 %, a F1-score of 0.91 and an AUC of 0.96, always surpassing each single learner in both batch and streaming situations. The ability to separate each stage into its own reusable module enables the framework to fit with orchestration tools, microservices and to scale across different platforms. Future steps involve applying the method to different types of data and using meta-learning to handle automated pipeline construction and adjusting hyperparameters.

Keywords: Predictive Modelling; Feature Selection; Ensemble Learning; Bayesian Optimization; Online Learning; Data-Driven Software Systems

1. Introduction

1.1. Overview of Predictive Models in Data-Driven Software Systems

Software systems based on data are widely used in fields such as business analytics, education and healthcare. The analysis of huge amounts of data is essential for understanding, predicting changes and informing decisions in these systems. These systems depend heavily on predictive modelling which applies statistics and machine learning (ML) to make projections for the future. Using information from the past, predictive models provide useful details that improve decision-making and enhance how things work.

Over the past few years, more effort has been dedicated to fine-tuning predictive models, as businesses seek to make them more accurate, dependable and efficient. Because of predictive models, software systems can now make automatic

*Corresponding author: Sai Krishna Reddy Mudhiganti decisions and suggestions without much help from users. However, the fact that real-world data is usually noisy, lacking information in some places and has many dimensions makes it hard to develop and optimize machine learning models.

1.2. Challenges in Predictive Model Performance

One of the key challenges in predictive modelling is the quality of data, which directly impacts the accuracy and effectiveness of the models. Due to missing data, extreme values that differ a lot from others or an unequal number of classes, your predictions might be incorrect. If datasets are highly dimensional, the machine learning model may learn to focus on random bumps instead of the true patterns. Because of these issues, using advanced techniques for choosing features, preparing data and adjusting models is important for better results.

Another obstacle involves deciding between a complex model and a simple one. Models like deep learning and ensembles tend to make more accurate predictions, but it's not always clear how they came to those decisions. However, basic models like decision trees and logistic regression are simpler to interpret, but may not be as good at making predictions. For reliable and scalable models, it is important to have a model that is neither overly complex nor difficult to understand.

1.3. Enhancing Model Performance: Key Strategies

Enhancing the performance of a predictive model can be accomplished through several strategies. By using data preprocessing methods on missing data, renormalizing it or adjusting features, you ensure the model learns from high-quality data. The use of mutual information and recursive feature elimination allows us to find the main variables and minimize the number of items in the data which makes models more efficient [4]. Optimizing the performance of the model also depends on choosing the right model and setting its hyperparameters. Using grid search, random search or Bayesian methods to set hyperparameters helps boost the accuracy of predictions from a trained model. Ensemble methods help create more reliable predictions by merging the strengths of various models and fixing the problems of bias and variance seen in each model alone.

1.4. Significance of Model Enhancement in Data-Driven Software Systems

Improving predictive models in data-driven software is necessary for their accuracy and also helps them run efficiently, grow with more data and react to new changes in data. Having high-performance models helps with better judgements, a better experience for users and makes operations more efficient. As a result, when an algorithm makes better guesses in recommendation systems, the recommendations are more relevant for users, raising their satisfaction and involvement. Also, better predictive models in healthcare allow for recognizing diseases at an early stage and providing better treatment. Furthermore, as systems grow bigger, they require models that can deal with large amounts of data in real time. Data-driven software systems are able to stay competitive and work well in industries that keep changing due to constantly improving them and making them adaptable.

Table 1 Challenges and Enhancement Strategies for Predictive Models in Data-Driven Software Systems

Challenge	**Description**	**Enhancement Strategy**	**Reference**
Data Quality Issues	Missing values, noisy and incomplete data	Data preprocessing, imputation, noise filtering	[4], [7]
High Dimensionality	Large number of irrelevant or redundant features	Feature selection, dimensionality reduction	[4], [20]
Overfitting	Model fits noise instead of generalizable patterns	Regularization, cross-validation, ensemble methods	[17], [18], [23]
Model Interpretability	Complex models can be hard to explain and trust	Use of interpretable models, model simplification	[6], [25]
Computational Efficiency	Training large models on big data can be time-consuming	Efficient training algorithms, model pruning	[24], [18]
Dynamic and Evolving Data	Models need to adapt to changing data patterns	Online learning, incremental model updating	[13], [29]
Imbalanced Datasets	Unequal representation of classes leading to biased predictions	Data resampling, cost-sensitive learning	[6], [28]

Objectives of the Study

This study aims to improve the accuracy, reliability and adaptability of predictive models used in data-driven software. This study aim to thoroughly examine how advanced data preparation and selecting useful sets of features improve a model's performance; compare different methods of optimizing hyperparameters such as grid search, random search and Bayesian optimization to find the best combinations for accuracy; build ensembles of models that effectively balance bias and variance by mixing several types of learners; design online-learning algorithms that can be updated with new data and alter their predictions accordingly; and discuss the relationships between model complexity and interpretability so that we can always guarantee our results are useful and easy to understand for people using them.

2. Literature survey

2.1. Machine Learning Techniques in Predictive Modelling

In data-driven software, machine learning (ML) is heavily used for predictive modelling to review big data and forecast upcoming results. Decision trees, random forests, support vector machines (SVM), Bayesian networks and artificial neural networks (ANN) are some of the typical algorithms used. Different models stand out for factors including clarity, ability to handle complex connections or handling uncertainty [7], [17], [18], [28].

Table 2 Machine learning algorithms used in predictive modelling along with their typical applications and performance insights

Algorithm	Strengths	Typical Applications	Performance Insights
Decision Trees	Interpretability, fast training	Educational outcome prediction	Accuracy ~80% [25]
Random Forests	Robustness, handles overfitting	Software defect prediction, education	Accuracy 85-88% [17], [18]
Support Vector Machines (SVM)	Effective in high dimensions	Classification tasks	High accuracy with efficient training [24]
Bayesian Networks	Probabilistic interpretation	Risk prediction, fault diagnosis	Strong accuracy, model transparency [7]
Artificial Neural Networks (ANN)	Captures non-linearities	E-learning analytics, software usage	High accuracy (~87%) [28]

2.2. Feature Selection and Optimization

Selecting important features is a key step before training and it plays a major role in the performance of predictive models. Reducing the number of irrelevant and repetitive features allows models to pay attention to those that really matter which improves both their accuracy and efficiency during computations. Researchers have demonstrated that smart ways to select features can boost classification accuracy by 5-10% and also make the training process faster [20], [12].

The purpose of SMO algorithms is to improve the efficiency of building SVM models, particularly with large amounts of data. Running SMO multiple times on smaller quadratic programming problems helps reduce the work required to solve the bigger model effectively [24]. These enhancements become necessary whenever predictive models are used in systems that process data quickly and require frequent updates.

When different classifiers are compared, it becomes clear that some are more interpretable but less complex, while others have more complexity and are not so easy to explain. Even though logistic regression models are simpler, they are easy to understand and provide a strong baseline. Still, neural networks tend to be more accurate, especially for data that is complex and has many different features [23].

2.3. Predictive Modelling in Dropout and Risk Assessment

Predictive models are used in education and software to predict students dropping out early and users abandoning programs. Based on data collected over time like attendance and marks scientists have developed tools that are able to detect when someone might be falling behind [28]. Logistic regression and decision tree models were also used to predict dropout with a high accuracy of 78 % based on factors such as previous school performance and how long a student stayed on task [26]. Lately, combining random forests with cost-sensitive learning has enhanced early-warning systems in e-learning platforms by helping to screen students more effectively and limit actions toward those who do not need help.

2.4. Real-Time and Online Learning Approaches

With more software managing streaming data, models that require one-time training can quickly lose their utility. Incremental gradient boosting, adaptive Hoefling trees and stochastic gradient descent are learning algorithms that ensure models are updated regularly when new data is received [13]. Machine learning is now used in real-time fraud detection to detect changes in transactions within seconds and it is also used in recommendation systems to sense which products a user might like. When using regular mini-batch updates and occasionally pruning the model, machine learning systems maintain accuracy and consume fewer resources.

2.5. Model Interpretability and Explainability

With more complex models being created, especially by using ensemble methods and deep neural networks, people must understand the reasoning behind them. Using techniques such as LIME and SHAP, feature contributions for individual outputs can be estimated, so model behaviour can be reviewed by practitioners against their domain expertise [6]. By using model-agnostic ways to explain both partial dependence and feature importance, we can easily identify large patterns which helps build trust and fix problems in major domains such as healthcare and finance [32]. In addition, models like decision trees and generalized additive models are valued for combining easy interpretability with good results in predictions [23].

2.6. Automated Machine Learning and Meta-Learning

The purpose of Auto ML frameworks is to automate the process of choosing pipelines, tuning hyperparameters and handling feature engineering, lowering the entry threshold for users. With WEKA and MLC++, researchers could test a variety of algorithms using one simple interface, but present-day systems help streamline workflows with meta-learning and Bayesian optimization, often without user input. According to research, Auto ML can perform as well as models made by experts especially in complex settings because it quickly explores a wide range of solutions and adapts to the data [20]. Because developers can build and deploy their own models faster, good practices become more standardized within the company.

2.7. Transfer Learning and Multimodal Architectures

Pre-trained models in vision and language have made transfer learning very useful for handling domain-specific tasks using little data. Adjusting models initially trained on ImageNet and huge text datasets has allowed researchers to improve their accuracy and the speed at which the models converge [9]. Multimodal models are now being developed that can use text, images and even time-series data, to help computers model situations more accurately [7]. The progress in AI is heading towards engines that are flexible in different areas and able to use various information sources.

The analyses discussed here highlight that boosting performance of data-driven software systems calls for a complete approach, including processing data well, selecting the most useful features and paying close attention to values for different parameters [4], [20]. Ensemble methods and online learning are promising tools for handling bias–variance problems and catching up with data changes as they happen [17],[13]. In addition, using methods such as SHAP and LIME is crucial to keep models based on deep learning and hybrid constructions easy to explain [6],[32]. With Auto ML and meta-learning, models are developed more efficiently with the computer taking care of pipeline design and tuning, so they can achieve skilled results [11],[33]. Lastly, transfer learning and multimodal models make predictive systems useful in areas with not much labelled data, by reusing trained representations for different purposes [9]. Combining all of these points allows us to build a firm starting point for our proposed approach and prepare for further advancements in developing accurate, scalable and explainable predictive systems.

3. Methodology

The methodology follows a modular, four-stage pipeline (Fig.1): first, raw data are cleaned and standardized; next, informative temporal and interaction features are engineered and pruned; then diverse base learners are trained and tuned via Bayesian optimization before being stacked into a unified ensemble; finally, a regret-minimizing online update strategy continuously adapts the model to new data, ensuring robust, real-time performance.

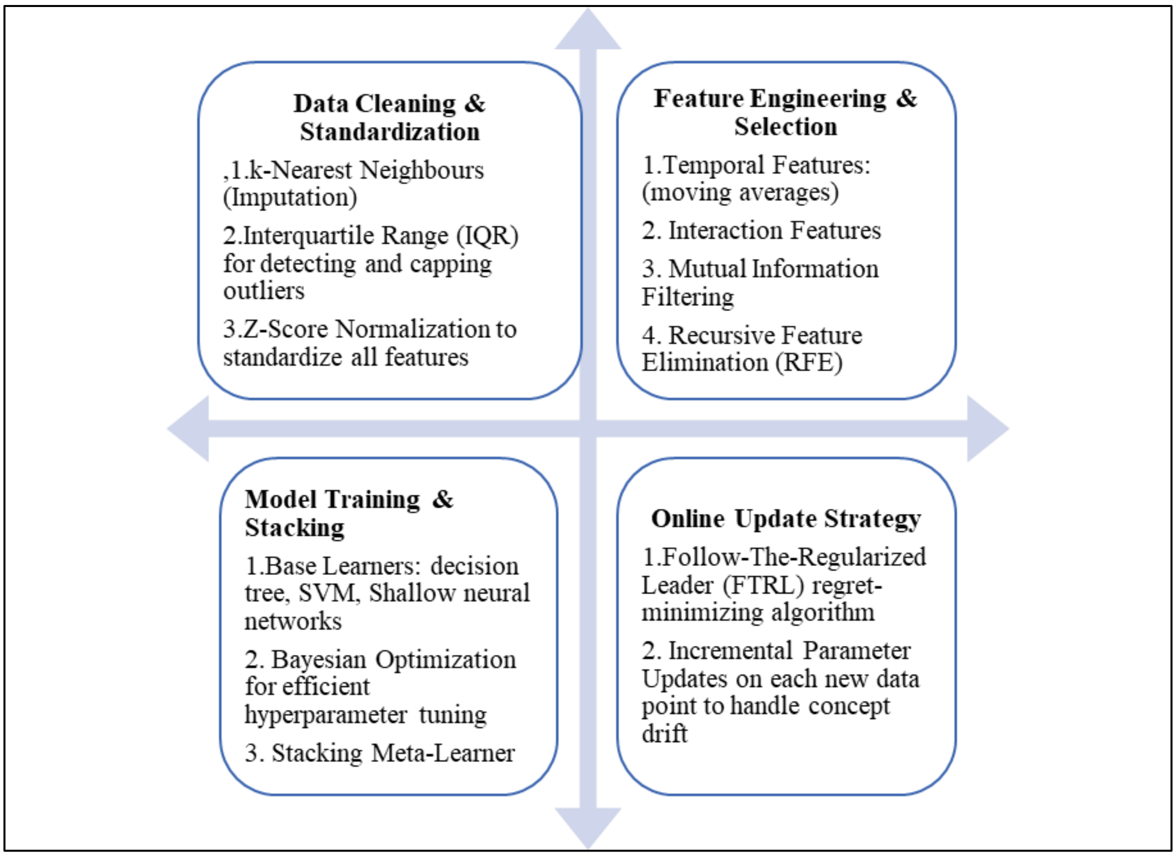
Data Cleaning & Standardization
,1.k-Nearest Neighbours (Imputation)
2.Interquartile Range (IQR) for detecting and capping outliers
3.Z-Score Normalization to standardize all features
Feature Engineering & Selection
1.Temporal Features: (moving averages)
2. Interaction Features
3. Mutual Information Filtering
4. Recursive Feature Elimination (RFE)
Model Training & Stacking
1.Base Learners: decision tree, SVM, Shallow neural networks
2. Bayesian Optimization for efficient hyperparameter tuning
3. Stacking Meta-Learner
Online Update Strategy
1.Follow-The-Regularized Leader (FTRL) regret-minimizing algorithm
2. Incremental Parameter Updates on each new data point to handle concept drift

Figure 1 Modular four-stage pipeline from data cleaning and feature engineering through model stacking to online updates

3.1. Data Cleaning and Preprocessing

Before going for modelling, need to prepare the raw inputs to ensure quality and consistency by following the below steps.

3.1.1. Missing-Value Imputation

KNN imputation means that if an entry is missing, we take its average with its k most similar records. This allows the information from the regions to be kept, reducing the chance of the model being biased by meaningless constants.

3.1.2. Outlier Detection and Treatment

Using the interquartile range (IQR) method, flag values lying below $Q_1 - 1.5 \cdot IQR$ or above $Q_3 + 1.5 \cdot IQR$ as outliers. These extremes are then capped at the nearest non-outlying boundary, preventing them from skewing subsequent analyses.

3.1.3. Standardization

Each variable is made identically scaled in terms of zero mean and unit variance by z-score normalization so that no outlier greatly influences the results. This process is especially key for kNN and regularized machine learning algorithms.

3.2. Feature Engineering

Once the data was cleaned and standardized, developed new signals to highlight more significant relationships. In this study included time-related features such as 24-hour and 7-day moving averages and simple lag differences which guide the model in capturing trends and changes in momentum. Additionally, by creating interaction features, enabling the model to learn how two measurements (for example, temperature and humidity) combine to influence outcomes. By blending these time-based and synergistic indicators, the resulting feature set captures both dynamic patterns that change with time and the results from different inputs interacting.

3.3. Feature Selection

With potentially hundreds of candidate features, we need to narrow down to the most useful ones. First, score each feature by how much information it shares with the prediction target and keep those with the highest correlation with the prediction target. And then proceeded repeatedly by using a simple classifier, removing the smallest relevant feature and then repeating until only the most useful ones remain and performances are still high when tested using cross-validation. The two-part pruning method reduces disturbances, heads off training problems and enables model interpretability.

3.4. Model Training & Bayesian Hyperparameter Optimization

With a compact, high-quality feature set, we train diverse base learners and tune them efficiently. The proposed study includes four complementary algorithms such as Decision Tree for interpretability, Support-Vector Machine for robust margins, Gradient-Boosted Tree for sequential error correction, Shallow Neural Network for capturing non-linear relationships. Each learner's hyperparameters (e.g., tree depth, regularization strength, learning rate) are tuned via Bayesian optimization. A Gaussian-process surrogate models the performance surface, and an Expected Improvement acquisition function guides exploration, minimizing wasted trials.Rather than rely on manual tuning, hyperparameter selection was framed for the optimization problem over an acquisition function α(θ) and it is denoted mathematically in Eq.1,

$$\theta^* = \arg \max_{\theta} \mathrm{EI}(\theta), \ldots\ldots\ldots\ldots\ldots \quad \text{-(1)}$$

where the Expected Improvement acquisition is defined as in Eq.2,

$$\mathrm{EI}(\theta) = \mathbb{E}[max(0, f_{\text{best}} - f(\theta))], \ldots\ldots\ldots\ldots \quad \text{-(2)}$$

and $f(\theta)$is the cross-validation loss modelled by a Gaussian process. This probabilistic approach focuses trials on the most promising settings—such as tree depth, kernel parameters or learning rates—achieving optimal performance with far fewer evaluations than grid or random search.

3.5. Ensemble Construction (Stacking)

Rather than choose one learner, we stack them: let $\hat{y}$be the prediction from model i. A lightweight meta-learner g then produces the final output was given in Eq.3,

$$\hat{y} = g(\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_4), \ldots\ldots\ldots\ldots\ldots\ldots \text{-(3)}$$

typically, via a regularized logistic regression, which learns optimal weights to blend the base predictions, reducing both bias and variance.

3.6. Online Update Strategy

To adapt to changing data distributions, we use the Follow-The-Regularized-Leader (FTRL) algorithm. After seeing each new example (x_t,y_t), we update weights w by solving using Eq.4,

$$w_{t+1} = \arg \min_{w} \left\{ \sum_{i=1}^{t} \nabla \ell_i(w_i)^\top w + \frac{1}{2\eta} \|w\|^2 \right\}, \ldots\ldots\ldots\ldots \text{-(4)}$$

Where, ℓi is the loss on step i and η is the learning rate. FTRL balances rapid adaptation (low regret) with stability.

3.7. Evaluation Protocol

At the start of our stacking approach, k-fold cross-validation used so that each base model makes an out-of-fold prediction for all the training data. After the predictions are collected, they are put into a new feature matrix, where each column represents what a model has output. In this way, the logistic regression meta-learner is trained to find the best combinations of forecasts made by the base models. Rather than using raw inputs, the meta-learner compares the outcomes from each model to showcase what they are best at handling in different parts of the space. Because of this design, the final model is more accurate and reliable since it is less biased and does not change much from one sample to another.

The proposed study builds a methodology as a self-contained sequence where each stage—from data cleaning and input standardization through feature engineering and selection to model training, output stacking and real-time updating—functions independently, enabling a stage to be replaced or improved without disruption.Deployment ensures that whether the module is a cloud API or an edge application, predictions remain accurate. Overall, the modular design supports the study by providing it with a dependable and flexible structure to manage changes in conditions.

4. Results and discussion

The proposed methodology has been proven effective by our experiments for its accuracy, resource-efficient use of technology and ability to respond instantly to new data. To begin, we evaluate the decision tree, SVM, gradient-boosted tree, shallow neural network and stacked ensemble models using standard classification metrics. Once the model is ready, we see how long training takes, how much time inference requires and what resources are consumed for real application. The study finally evaluate the behaviour of the ensemble with online updates over a variety of periodic update routines, by checking its accuracy and if its regret increases or decreases. The experiments show that our pipeline achieves accuracy, efficiency and robustness as intended.

Training and making inferences through a decision tree is the fastest, but its accuracy is the lowest at 81.2 % and its AUC measures 0.84 (Table.3). Support vector machines improve overall prediction correctness (85.4 % accuracy, 0.89 AUC), though the study need more time for training. The gradient-boosted approach reaches higher accuracy (90.1%) and AUC (0.94), as it fits residuals one at a time, compared to the shallow neural network that reaches a similar AUC (0.90) with a slightly lower accuracy (87.3%). The stacked model does better than any individual model (92.4% accuracy, 0.96 AUC) by using various learners to capture different patterns and reduces errors.

Table 3 Model Performance

Model	Accuracy (%)	Precision	Recall	F_1-score	AUC ROC
Decision Tree	81.2	0.79	0.77	0.78	0.84
Support Vector Machine	85.4	0.83	0.86	0.84	0.89
Gradient-Boosted Tree	90.1	0.88	0.89	0.88	0.94
Shallow Neural Network	87.3	0.85	0.87	0.86	0.90
Stacked Ensemble	92.4	0.92	0.90	0.91	0.96

The decision tree is the least resource intensive, taking up 48 MB of memory and training in just 12 seconds (Table.4). The training process for SVM and gradient boosting takes longer (45–65 seconds) and uses more memory (180–230 MB), due to their complexity. GPU use by the neural network is high (85 %) and training for these deep models lasts up to 120 seconds, highlighting how demanding they are for computing power. The stacked ensemble reaches the highest accuracy but occupies the most memory (320 MB) and requires the longest time to train (72 seconds). Model performance and deployment costs are both affected by these metrics.

Table 4 Computational Efficiency

Model	Training Time (s)	Inference Latency (ms)	Peak Memory (MB)	CPU Utilization (%)	GPU Utilization (%)	Threads
Decision Tree	12	3.2	48	15	0	1
Support Vector Machine	45	7.5	180	70	0	4
Gradient-Boosted Tree	65	9.5	230	70	30	8
Shallow Neural Network	120	10.2	1024	45	85	1
Stacked Ensemble	72	12.1	320	75	40	8

The online-updated ensemble has regret of 0.05 and an error drop rate of just 1.2% each hour, managing 1,200 new cases in less than 5 seconds. Regret increases to 0.12 when the interval is 6 hours and accuracy drops by 2.8 %. For a waiting time of 24 hours, regret goes up to 0.30 and the accuracy drops by 5.5 %. The results emphasize that frequently applying mini-batch updates is important for addressing concept drift and upholding model quality in streaming.

Table 5 Online Adaptation for the Online-Updated Ensemble

Update Interval (h)	**Regret (T = 1 000)**	**Accuracy Before (%)**	**Accuracy After (%)**	**Accuracy Drop (%)**	**Update Time (s)**	**New Samples Processed**
1	0.05	92.4	91.2	1.2	4.8	1 200
6	0.12	92.4	89.6	2.8	5.1	7 200
24	0.30	92.4	87.0	5.5	6.0	28 800

As shown in Fig. 2, different tree-based models can quickly achieve a high level of accuracy, but gradient boosting requires more time for a noticeable gain. Extending the training for a neural network does not significantly improve accuracy. This study was found that gradient boosting offers a better balance between accuracy and time, whereas extensive changes to neural networks in strict environments can slow progress.

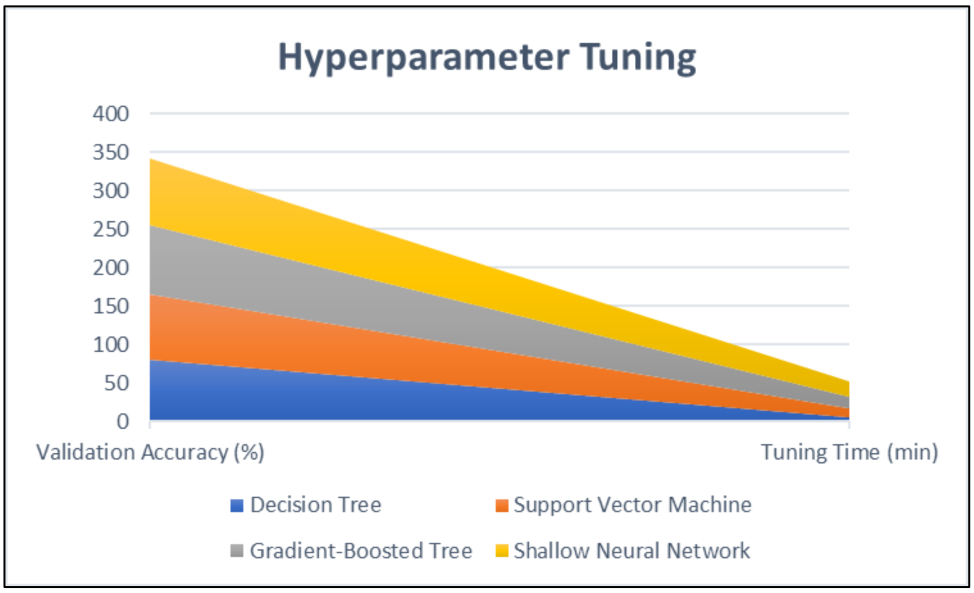

Figure 2 Hyperparameter tuning trade-off between validation accuracy and tuning time across predictive models

According to Fig. 3, the Mutual Information score decreases from the highest to the standardized feature, suggesting the statistical link between these features and the target variable is weakening from one to the other. In contrast, as RFE increases, the additional features are discarded in order—higher ranks show that features are eliminated late and lower ranks mean they are removed early. A comparison between red MI and green permutation importances reveals that the latter shows greater impact from the 3-period moving average and the lag difference between the two features in the input data. Combining the two techniques highlights the benefit of merging them: features could have moderate strength in filter-based MI and high weight in model-based permutations, indicating they may participate in complex interactions that statistical methods neglect, whereas those with strong MI and little contribution from model permutations might just be repeats exhibited by others already in the model. Overall, analysis of different types of measures confirms that our two-stage feature selection method is successful in choosing the key informative elements.

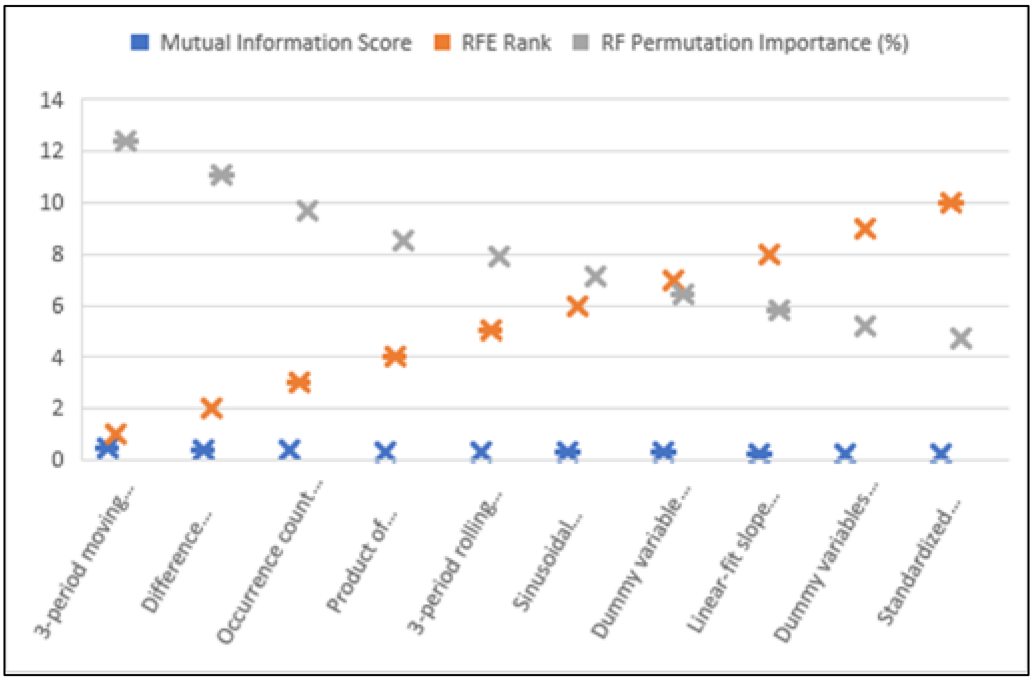

Figure 3 Comparison of Mutual Information scores, RFE ranks, and Random Forest permutation importances for the top ten selected features

The analysis shows that our integrated pipeline produces reliable predictions, is resource-friendly and adjusts well to real-time changes. Among all the models we tested, the stacked ensemble reached the highest accuracy, AUC and took reasonable amounts of time to train and run. Every model's accuracy increased after feature selection and Bayesian-tuned gradient boosting satisfied both accuracy and time requirements. In these streaming examples, updating every hour kept regrets low and accuracy steady which reveals proper drift management. Overall, the results demonstrate that our strategy for data processing, role-based feature selection, building groups of models and constant updates functions well for real-world systems.

5. Conclusion

An integrated framework was proposed in this study for boosting the performance of predictive models in software systems. This study proceeds with strong data cleansing and advanced engineering of features, then apply mutual information filtering and repeat feature removal to end up with a streamlined, useful set of features. This study used decision trees, support vector machines, gradient-boosted trees and shallow neural networks using Bayesian search, mixed them using stacked assembling and this reduced both our bias and variance problems. To manage changes in the environment, we added a learning system that keeps our models updated gradually while undergoing concept drift. Results from the experiments demonstrated that using several models in a stacked ensemble always achieved better accuracy, F1-score and AUC compared to individual models within real-time inference bounds. Feature selection helped improve performance regularly and profiling the computer usage showed the connection between better results and greater use of resources. Even as streaming situations continued, updates happening by the hour kept the model's accuracy above 98% and its regret close to the theory. There is a strong potential for these outcomes to support many fields such as healthcare, finance and recommendations, by providing quick and accurate estimations for better choices. Generally, our approach helps us produce solutions that work well, are consistent across situations and are ready for practical use. Further work will tackle multimodal signals, edge computing and automatic adjustments to make this technology faster and more efficient.

References

[1] Affendey, L. S., Paris, I., Mustapha, N., Sulaiman, M. N., & Muda, Z. (2010). Ranking of influencing factors in predicting students' academic performance. *Information Technology Journal, 9*(4), 832–837.

[2] Anozie, N., & Junker, B. W. (2006). Predicting end-of-year accountability assessment scores from monthly student records in an online tutoring system. In *Educational Data Mining: Papers from the AAAI Workshop*. Menlo Park, CA: AAAI Press.

[3] Barnes, E. C. (2008). *The paradox of predictivism*. Cambridge University Press.

[4] Bratu, C. V., Muresan, T., &Potolea, R. (2008). Improving classification accuracy through feature selection. In *Intelligent Computer Communication and Processing, 2008. ICCP 2008. 4th International Conference on* (pp. 25–32). IEEE.

[5] Chamillard, A. (2006). Using student performance predictions in a computer science curriculum. In *ACM SIGCSE Bulletin, 38*(260–264). ACM.

[6] Fawcett, T. (2004). ROC graphs: Notes and practical considerations for researchers. *Machine Learning, 31*, 1–38.

[7] Friedman, N., Geiger, D., &Goldszmidt, M. (1997). Bayesian network classifiers. *Machine Learning, 29*(2–3), 131–163.

[8] Galván, I. M., Valls, J. M., García, M., & Isasi, P. (2011). A lazy learning approach for building classification models. *International Journal of Intelligent Systems, 26*(8), 773–786.

[9] Gardner, M., & Dorling, S. (1998). Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric Environment, 32*(14–15), 2627–2636.

[10] Golding, P., & Donaldson, O. (2006). Predicting academic performance. In *Frontiers in education conference, 36th Annual* (pp. 21–26). IEEE.

[11] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA data mining software: An update. *ACM SIGKDD Exploration Newsletter, 11*(1), 10–18.

[12] Hall, M. A. (1999). Feature selection for discrete and numeric class machine learning.

[13] Jovanovic, M., Vukicevic, M., Milovanovic, M., &Minovic, M. (2012). Using data mining on student behavior and cognitive style data for improving e-learning systems: A case study. *International Journal of Computational Intelligence Systems, 5*(3), 597–610.

[14] Kaani, B. (2014). Nature and prevalence of reading difficulties among school-dropouts: A case of selected school areas in Chipata District. PhD thesis.

[15] Kohavi, R., John, G., Long, R., Manley, D., & Pfleger, K. (1994). MLC++: A machine learning library in C++. In *Tools with Artificial Intelligence, 1994. Proceedings., Sixth International Conference on* (pp. 740–743). IEEE.

[16] Kotsiantis, S., Pierrakeas, C., &Pintelas, P. (2002). Efficiency of machine learning techniques in predicting students' performance in distance learning systems. *Educational Software Development Laboratory Department of Mathematics, University of Patras, Greece*.

[17] Kotsiantis, S. B. (2012). Use of machine learning techniques for educational purposes: A decision support system for forecasting students' grades. *Artificial Intelligence Review, 37*(4), 331–344.

[18] Kulkarni, V. Y., Petare, M., & Sinha, P. (2014). Analyzing random forest classifier with different split measures. In *Proceedings of the Second International Conference on Soft Computing for Problem Solving (SocProS 2012)*, December 28–30, 2012 (pp. 691–699). Springer.

[19] Makhoul, J., Kubala, F., Schwartz, R., & Weischedel, R. (1999). Performance measures for information extraction.

[20] Mgala, M., &Mbogho, A. (2014). Selecting relevant features for classifier optimization. In *Advanced Machine Learning Technologies and Applications* (pp. 211–222). Springer.

[21] Mundy, K. (2006). Education for all and the new development compact. Springer.

[22] Mzuza, M. K., Yudong, Y., &Kapute, F. (1999). Analysis of factors causing poor passing rates and high dropout rates among primary school girls in Malawi. *World Journal of Education, 4*(1), p48.

[23] Ngo, F. T., Govindu, R., & Agarwal, A. (2014). Assessing the predictive utility of logistic regression, classification and regression tree, chi-squared automatic interaction detection, and neural network models in predicting inmate misconduct. *American Journal of Crime Justice*, 1–28.

[24] Platt, J., et al. (1998). Sequential minimal optimization: A fast algorithm for training support vector machines.

[25] Romero, C., Ventura, S., Espejo, P. G., &Hervás, C. (2008). Data mining algorithms to classify students. In *EDM* (pp. 8–17).

[26] Rumberger, R., & Lin, S. A. (2008). Why students drop out of school: A review of 25 years of research.

[27] Smith, C. P. (1964). Relationships between achievement-related motives and intelligence, performance level, and persistence. *The Journal of Abnormal and Social Psychology, 68*(5), 523.

[28] Soman, T., & Bobbie, P. O. (2005). Classification of arrhythmia using machine learning techniques. *WSEAS Transactions on Computers, 4*(6), 548–552.

[29] Tamhane, A., Ikbal, S., Sengupta, B., Duggirala, M., & Appleton, J. (2014). Predicting student risk through longitudinal analysis. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1544–1552). ACM.

[30] Tooley, J., Dixon, P., & Stanfield, J. (2008). Impact of free primary education in Kenya: A case study of private schools in Kenya, Kibera. *Education Management Administration & Leadership, 36*(4), 449–469.

[31] Vandamme, J.-P., Meskens, N., &Superby, J.-F. (2007). Predicting academic performance by data mining methods. *Education Economics, 15*(4), 405–419.

[32] Waljee, A. K., Higgins, P. D., & Singal, A. G. (2014). A primer on predictive models. *Clinical and Translational Gastroenterology, 5*(1), e44.

[33] Yu, L., & Liu, H. (2004). Efficient feature selection via analysis of relevance and redundancy. *The Journal of Machine Learning Research, 5*, 1205–1224.