International Research Journal of Multidisciplinary Scope (IRJMS), 2024; 5(4):755-767

Original Article | ISSN (O): 2582-631X DOI: 10.47857/irjms.2024.v05i04.01373

Designing a Dynamic Weighted Stacking Recommendation System

Nisha Sharma, Mala Dutta*

Department of Computer Technology, Assam down town University, Sankar Madhab Path, Gandhi Nagar, Panikhaiti, Guwahati, Assam, India-781026. *Corresponding Author's Email: mala.dutta@adtu.in

Abstract

To increase accuracy, ensemble approaches have become increasingly popular in designing recommendation systems. More recently, efforts have focused on dynamically integrating base models to improve ensemble performance further. Dynamic integration entails combining multiple base models with dynamically changing contributions to the ensemble. In this paper, we propose a Dynamic-Weighted Stacking (DWS) recommendation model. Bayesian optimization is used to adjust hyperparameters and find the best weight for each base model dynamically. The base models used in the DWS recommendation model are K-Nearest Neighbors (KNN), Linear Regression (LR), and Support Vector Machine (SVM). The proposed ensemble model is tested across various scenarios corresponding to different user populations with varying ratings and user counts, and the results are reported. We also compare the DWS recommendation model to individual static base models. It is observed that the batch dynamic integration approach works better than individual static models and static traditional stacking. The proposed model's flexibility in handling various data distributions highlights its potential for real-world applications in recommendation systems.

Keywords: Bayesian optimization, Dynamic Ensemble learning, Dynamic Weighted Stacking, K-Nearest Neighbors, Linear Regression, Stacking.

Introduction

In recent years, recommendation systems have emerged as indispensable tools for delivering personalized content across a multitude of domains, ranging from e-commerce platforms to social media networks and entertainment services. While these systems have proven their efficacy in enhancing user experience and engagement, traditional recommendation methodologies often rely on static models trained on fixed datasets (1). The dynamic feature of weighted stacking represents a substantial advancement over conventional methods due to its ability to adapt in real-time to changes in the data distribution, user behavior, and external environmental factors (2). In traditional weighted stacking, once the model is trained, the weights assigned to each base model remain fixed, determined solely by the initial training data. This static weighting is problematic in dynamic environments, where user preferences, trends, and contextual factors are constantly evolving. The inability of conventional methods to adjust weights in response to these changes often leads to suboptimal performance, as the model becomes increasingly misaligned with the current state of the data. To address this limitation, we proposed a Dynamic-Weighted Stacking (DWS) model which uses a weight adjustment mechanism using Bayesian optimization, which continuously optimizes the contribution of each base model based on its performance. This ensures that the ensemble adapts dynamically to changing data distributions and user behaviors, leading to improved accuracy, robustness, and responsiveness in environments where static models typically fail. The system continuously updates the weights assigned to each base model based on newly observed data that occur due to changes in external trends (3). This allows the model to recalibrate its weighting strategy dynamically, optimizing the contribution of individual base models according to the most recent and relevant information. As a result, dynamic weighted stacking is better equipped to capture non-stationary patterns in the data, making it particularly effective in environments characterized by temporal variability or context-

(Received 20th June 2024; Accepted 22nd October 2024; Published 30th October 2024)

dependent shifts.

From a technical standpoint, the dynamic nature of this method enables the ensemble to maintain high predictive accuracy in the face of evolving data, where static models would otherwise degrade in performance (4). This adaptability enhances the overall robustness of the system, ensuring that it remains responsive to real-time fluctuations and better aligned with the underlying distribution of user preferences. Consequently, dynamic weighted stacking offers a more sophisticated and responsive approach compared to traditional methods, making it highly applicable in domains requiring continuous adaptation, such as recommendation systems, financial forecasting, and personalized content delivery, all of which are essential in modern systems. Dynamic integration methodologies represent an illustration from the static amalgamation approaches of traditional stacking. These techniques endeavour to imbue recommendation systems with adaptability and responsiveness by dynamically selecting and weighting base models based on their performance and relevance within the prevailing context. Additionally, dynamic weighting strategies, including exponential decay and adaptive boosting, empower recommendation systems to assign higher weights to more informative models while attenuating the influence of less pertinent ones, thereby enhancing the robustness and efficacy of the recommendation framework (5). Consequently, there is a pressing need to develop recommendation systems that exhibit adaptability and responsiveness to these evolving dynamics. The traditional stacking approach represents a seminal technique in the domain of ensemble learning (6). This methodology involves the amalgamation of predictions generated by multiple base models through a meta-learner (6). Each base model operates independently on the dataset, generating individual predictions that are subsequently combined by the meta-learner to produce the final recommendation. Stacking has garnered widespread acclaim for its ability to harness the diverse perspectives offered by individual base models, thereby culminating in enhanced prediction accuracy. However, conventional stacking methods often entail the utilization of fixed weights for amalgamating base model predictions, thus potentially compromising adaptability, especially in dynamic environments characterized by shifting user preferences and evolving trends. The traditional stacking approach has been extensively studied and applied in various domains, including but not limited to finance, healthcare, and marketing. Numerous empirical studies have demonstrated the efficacy of stacking in improving prediction accuracy compared to individual base models. Moreover, researchers have explored different variations and extensions of the stacking approach, such as meta-stacking, heterogeneous stacking, and weighted stacking, to further enhance its performance and robustness (7-9). The innovativeness and applicability of dynamic weighted stacking idea are generated by its ability to improve on the various drawbacks of the traditional static stacking models in recommendation systems. On the other hand, the previous stacking strategies such as stacking based on the use of votes in combining the base model predictions shows good results but tends to be static in nature over the dynamics of the nature of the end user's preferences and behavior. Such inflexibility has been the downside of orthodox stacking models especially in situations when the target users are highly dynamic in their behavior pattern. With respect to modern approaches known as dynamic weighted stacking well defined and scientific processes are enhanced with regard to these model weights for instance on the constructs of feedback relative to users and other external environments (10). As the contribution of the corresponding base model is adjusted not through the base model combining process rather by the current conditions of use, the very structure of ensemble allows following the changing tendencies in the data. Such policies are important to uphold the recommendation accuracy levels, in conditions where time and place of recommendation vary dramatically (11).

In this paper, we introduce a recommendation model termed Dynamic-Weighted Stacking (DWS), wherein both hyperparameter tuning and dynamic optimization of individual base model weights are accomplished through Bayesian optimization. A comprehensive assessment of the ensemble model is carried out across diverse scenarios, reflecting varying user populations characterized by different ratings and user counts. The experimental procedures are detailed, followed by a discussion of the obtained results. The contributions of this work are:

- Development of a Dynamic Weighting Strategy: We provide an approach that dynamically modifies base model weights according to how well they perform on dataset subsets. This method improves overall forecast accuracy by allowing the ensemble model to assign larger weights to more accurate base models.
- Integration of Bayesian Optimization: By utilizing Bayesian optimization methods, we optimize the ensemble model's weighting parameters, facilitating effective weight space exploration and the discovery of ideal weight configurations. This makes it easier for weights to automatically adjust to the properties of the data, which improves generalization performance.
- An Empirical Analysis of Various Data Scenarios: We provide in-depth empirical analyses using datasets that replicate different-sized and complex real-world circumstances. We conducted experiments to show and analyze how our proposed model behaves in terms of increasing predicted accuracy when compared to individual regression models.

The paper comprises the following sections: The initial section, Introduction, overviews our work area and investigates prior research in recommendation systems focusing on dynamic integration. In the subsequent section, Methodology, we commence with an in-depth explanation of the dataset and detail our approach to creating a simulated dataset for experimentation. Subsequently, we outline the methodologies employed in designing the proposed model. The final sections summarize, evaluate, compare, and discuss the findings of our study.

Dynamic integration techniques have gained popularity in various research domains, including personalized medicine, natural language processing, and financial forecasting (12-14). Researchers have proposed innovative algorithms and frameworks for dynamic model selection and weighting, leveraging concepts from reinforcement learning, online optimization, and multi-armed bandit algorithms. Furthermore, studies have investigated the theoretical properties and practical implications of dynamic integration techniques, exploring their convergence properties, regret bounds, and scalability.

Hyperparameter tuning represents a critical facet in the optimization of machine learning models, exerting a profound impact on their performance and generalization ability. Bayesian optimization techniques furnish a principled and efficient framework for automating the search for optimal hyperparameters while concurrently mitigating computational overheads and time expenditure (15, 16). By modelling the objective function and orchestrating an iterative exploration of the parameter space, Bayesian optimization algorithms systematically identify promising regions and refine the search process (17). This iterative refinement engenders tangible enhancements in model performance and generalization capacity, thereby endowing Bayesian optimization with indispensability in the realm of recommendation system design and development. Bayesian optimization has garnered significant attention in both academic research and industrial applications, owing to its versatility and efficacy in optimizing complex and noisy objective functions (18). In the context of recommendation systems, Bayesian optimization has been applied to various tasks, including hyperparameter tuning for collaborative filtering models, matrix factorization techniques, and deep learning architectures. Moreover, researchers have investigated novel extensions of Bayesian optimization, such as multi-objective optimization, asynchronous optimization, and parallel optimization, to address the scalability and efficiency challenges inherent in recommendation system design (19).

Regression models serve as key components in recommendation systems, facilitating the prediction of user preferences predicated on historical interactions with items. Different types of regression models used for designing our proposed model are K-Nearest Neighbours (KNN), Linear Regression (LR) and Support vector machines (SVM). A statistical technique called linear regression (LR) is used to determine the relationship between one or more independent factors and a dependent result. K-nearest neighbors (KNN) is a machine learning method that is simple to implement and effective for both regression and classification applications. Its basic idea is to find the K data points that are closest to a given query point in the feature space, then make predictions based on the average of those

neighbors labels or values or the majority vote (20). Researchers have investigated the strengths, limitations, and trade-offs associated with different regression techniques, exploring their applicability in various recommendation scenarios. Additionally, studies have examined hybrid approaches that combine regression models with other machine learning techniques, such as clustering, dimensionality reduction, and feature engineering, to enhance recommendation performance and interpretability (21, 22). The dynamic weighted stacking approach amalgamates the virtues of dynamic integration methodologies with the robustness inherent in the stacking ensemble paradigm. By dynamically modulating the weights assigned to base models contingent upon their performance and contextual relevance, the recommendation system achieves unparalleled levels of accuracy and adaptability (23). The dynamic weighted stacking framework continually assimilates and accommodates the evolving landscape of user preferences and environmental dynamics, thereby furnishing users with personalized and germane recommendations across diverse application scenarios. The dynamic weighted stacking approach has been validated through empirical studies and real-world deployments across multiple domains, including e-commerce, content streaming, and online advertising. Comparative evaluations against baseline methods and alternative ensemble techniques have demonstrated the superior performance and efficacy of dynamic weighted stacking in various recommendation tasks. Furthermore, researchers have investigated the interpretability, scalability, and computational efficiency aspects of dynamic weighted stacking, exploring strategies for model selection, hyperparameter tuning, and ensemble aggregation (24-26). By harnessing the prowess of dynamic selection, dynamic weighting, Bayesian optimization, and regression modelling techniques, recommendation systems can adeptly navigate the nuanced intricacies of evolving user preferences, thereby proffering tailored and pertinent recommendations with unprecedented efficacy and accuracy.

Methodology

Dataset

To design our recommendation model, we have used MovieLens 100k dataset. To simulate multiple circumstances, the dataset has been divided into distinct situations as shown in Table 1. Each instance represents a distinct state or condition of the data, enabling a dynamic evaluation of the recommendation system in different situations. As an illustration, Situation 1 is the starting state before any user interactions and shows a scenario in which there are no user ratings. Situations 4 to 9 represent a halfway scenario when the dataset is partially populated, while Situations 2 and 3 replicate adding a few user ratings for early training. Finally, Situations 10 through 12 simulate an end scenario when there is a full training set. The dataset divided into different situations simulates a real-world scenario of continuous data. This allows the evaluation process to be more realistic in dynamic and evolving settings (27).

Table 1: Dataset Divided into Different Situations

Situations	Rating	User
1	0	1
2	500	7
3	1500	20
4	3000	32
5	7500	65
6	10500	89
7	20000	157
8	32500	225
9	50000	358
10	62000	437
11	74000	503
12	100000	667

Data Preparation

The methodology begins with data preparation, where a systematic sampling strategy is employed to create subsets of data, referred to as situations. These situations simulate different stages of user engagement and rating activity, representing real-world scenarios where the amount of user interaction with the system varies. The data is separated into training and testing sets [80-20] following sampling, enabling a comprehensive evaluation of the model in every simulated scenario.

Hyperparameter Tuning

Bayesian optimization is used to determine the best hyperparameters for each base model after the training and testing sets have been established for every scenario. Bayesian optimization builds a probabilistic model of the objective function, it intelligently explores the hyperparameter space, making it more efficient than random or standard grid search techniques. The objective function aims to minimize the Mean Absolute Error (MAE) of the individual base models (KNN, XGBoost, Random Forest) on the training data. Each basic model (KNN, XGBoost, Random Forest) has several iterations until the algorithm finds the optimal hyperparameters that reduce the error on the training set.

$$\text{Objective\_Function(hyperparameters)} = \text{minimize}(\text{MAE}_{\text{base\_model}})$$

Dynamic Weight Calculation

The base models (KNN, XGBoost, and Random Forest) are trained on the current subset of data for each scenario once the best hyperparameters are determined via Bayesian optimization. The next stage is to aggregate the predictions made by the underlying models using a weighted ensemble method after they have been trained.

In our proposed DWS model, the process of weight optimization occurs in two key stages. The first step involves using an objective function during the Bayesian optimization phase to determine the initial set of optimal weights for each base model. This function evaluates different combinations of weights and selects the one that minimizes the Mean Absolute Error (MAE) of the ensemble's predictions. The objective function is defined as:

$$\text{Objective\_Function}(w) = \min\left(\text{MAE}\left(y_{\text{train}}, \sum_{i=1}^{n} w_i \times \text{prediction}\right)\right)$$

Where,

- i: An index variable that iterates over each base model from 1 to n.
- w_i: The weight assigned to the i^{th} base model.
- prediction_i: The predicted value from the i^{th} base model

This step ensures that the base models start with a well-optimized set of optimal weights, allowing the ensemble to make accurate predictions based on the training data. Once the initial weights are optimized, the model enters the second phase, where the weights are dynamically adjusted during training as more data is observed. The weights w_i for each base model i are dynamically adjusted using the following formula:

$$W_i^{(t+1)} = W_i^{(t)} \times \exp(-\alpha \cdot \text{MAE}_i)$$

Where,

- $W_i^{(t)}$ is the weight of model i at time t
- $Wi^{(t+1)}$: The updated weight of model i at the next iteration t+1.
- MAE_i_ is the Mean Absolute Error of model i calculated on the current data
- α is a learning rate that controls the sensitivity of weight updates

It updates the weight Wi(t) of each base model i at iteration t based on the model's performance, typically measured by the Mean Absolute Error (MAE), to adjust its contribution to the ensemble's final prediction. In this step, we have dynamically adjusted weights during each iteration based on the performance of base models (MAE) on current data. This dynamic weight adjustment process ensures that the model remains responsive to changes in data distributions and user preferences, continuously improving its performance over time. Together, these two stages, Bayesian optimization for optimal weight selection and dynamic weight adjustment based on real-time performance allow the DWS model to achieve both accuracy and adaptability in diverse and evolving environments.

Meta-Model Training and Evaluation

Once the dynamic weight adjustment process is done, the final ensemble predictions are computed by aggregating the predictions of the base models using the updated weights. The meta-model is trained to further improve the aggregated predictions from the basic models following the weighted ensemble stage. The weighted predictions from the ensemble are fed into the meta-model (such as Linear Regression, KNN, or SVM), which then optimally combines them to get

the desired result. The test data are used to evaluate the meta-model once it has been trained. Then, using the test data for each scenario, the Mean Absolute Error (MAE) is computed to evaluate the performance. Through this approach, the accuracy and responsiveness of the basic models, weighted ensemble, and meta-model are all optimized for the system as a whole. The model's performance is measured by calculating the Mean Absolute Error (MAE) on the test set for each situation:

$MAE = \frac{1}{n}\sum|y_{test} - \text{meta\_model.predict(test\_data)}|$

where:

- y_{test}: Actual true values..
- meta_model.predict(test_data)|: Predicted values from the meta-model.
- $|y_{test} - \text{meta\_model.predict(test\_data)}|$: Absolute error for each situation.

Proposed Dynamic-Weighted Stacking (DWS) Model

The DWS technique presented here differs from previous studies in its utilization of a Bayesian approach for hyperparameter tuning and the dynamic determination of optimal weights. While (26,27) also employed dynamic weight optimization, their method involved a brute force approach for weight selection before system launch, rather than assigning weights based on observed errors of base models as done in this case. The flowchart of our proposed model is shown below in Figure 1.

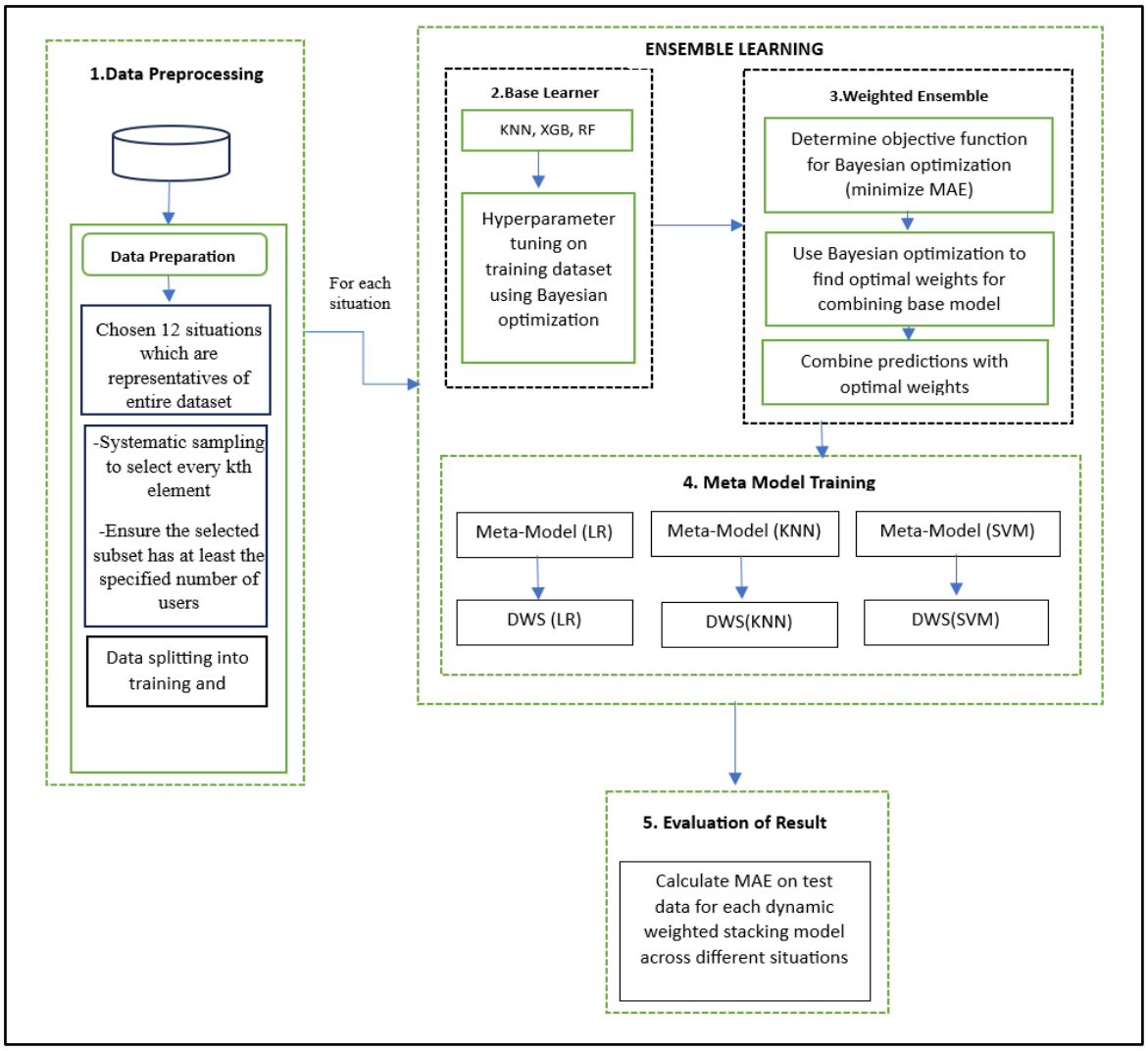

Figure 1: Flow Chart of Our Proposed Model

The pseudocode of our proposed DWS model is outlined as follows

Initialization:

- Define situations with specific ratings and user requirements.
- Initialize necessary libraries and functions.

Iterate through Each Situation:
For Each Situation in Situations
a. Retrieve the number of ratings (num_ratings) and users (num_users) specified for the situation.
b. Data Preparation

- If num_ratings is zero, skip the situation.
- Use systematic sampling to select every kth element based on the total number of ratings.
- Ensure the subset contains at least the specified number of users.
- If the filtered subset has insufficient samples for training and testing, skip the situation.

c. Split Data

- Split the data into training (80%) and testing (20%) sets.

Hyperparameter Tuning for Base Models
a. Define parameter space for K-nearest neighbors (KNN), Linear Regression (LR), and SVM.
b. Use Bayesian optimization (BayesSearchCV) to find optimal hyperparameters for each base model:
Train Base Models
a. Instantiate models with the optimal hyperparameters:
- model_knn, model_lr, model_svm
b. Train each model using the training data.
Initial Predictions
a. Generate predictions for the test set from each base model:
- predictions_knn, predictions_lr, predictions_svm
Find Initial Optimal Weights
a. Define an objective function to minimize Mean Absolute Error (MAE):
- Combine predictions using a weighted sum.
- Return MAE as the objective value.
b. Use Bayesian optimization to find the optimal weights for the ensemble:
- optimal_weights = [w1, w2, w3]
Dynamic Weight Adjustment
a. Initialize updated_weights as optimal_weights.
b. For i from 1 to N:
i.Combine predictions using the current updated_weights.
ii. Calculate MAE for each base model: mae_knn, mae_lr, mae_svm
iii. Update the weights based on the calculated MAEs using the update_weights function:
updated_weights=update_weights(updated_weights, mae_knn, mae_lr, mae_svm)
iv. Print optimal weights, updated weights
Final Ensemble Predictions
a. Combine predictions using the final updated_weights:ensemble_predictions=calculate_ensemble_predictions(updated_weights, predictions_knn, predictions_lr, predictions_svm)
Meta-Model Training
a. Train a meta-model (e.g., KNN, LR) using the final ensemble predictions from the base models: meta_model.fit(ensemble_predictions.reshape(-1, 1), y_test)
Evaluation
a. Predict with the meta-model on the test data: meta_model_predictions_test=meta_model.predict(ensemble_predictions. reshape(-1, 1))
b. Calculate final MAE on test data:final_mae_test = mean_absolute_error(y_test,meta_model_predictions_test)
c. Print final MAE.

Results and Discussion

Table 2 represents the Mean Absolute Error (MAE) values for different models, including Linear Regression (LR), K-Nearest Neighbors (KNN), and Support Vector Machines (SVM), along with the MAE of our proposed Dynamic Weighted Stacking (DWS) using LR, KNN, and SVM as meta-learners across various situations.
Additionally, the table presents the updated weights for each model following the final iteration of the dynamic weight adjustment process. This highlights how the weights of the base models are dynamically updated for each situation based on their performance.

Table 2: MAE Values and Updated Weights of Different Models over Various Situations

Situations	Metrics	LR	KNN	SVM	DWS (LR)	DWS (KNN)	DWS (SVM)
2	MAE	1.02	0.96	0.67	0.69	0.68	0.63
	Updated Weights	0	0.166	0.833			
3	MAE	0.7	0.78	0.55	0.57	0.6	0.57
	Updated Weights	0.001	0.063	0.929			

4	MAE	0.66	0.69	0.72	0.65	0.56	0.61
	Updated Weights	0.936	0.063	0			
5	MAE	0.67	0.66	0.7	0.66	0.6	0.65
	Updated Weights	0.685	0.314	0			
6	MAE	0.75	0.75	0.8	0.74	0.66	0.72
	Updated Weights	0.653	0.346	0			
7	MAE	0.72	0.72	0.75	0.7	0.62	0.68
	Updated Weights	0.63	0.367	0.001			
8	MAE	0.7	0.7	0.74	0.69	0.62	0.67
	Updated Weights	0.683	0.316	0			
9	MAE	0.71	0.7	0.73	0.69	0.62	0.68
	Updated Weights	0.501	0.481	0.017			
10	MAE	0.73	0.72	0.75	0.7	0.63	0.69
	Updated Weights	0.501	0.481	0.017			
11	MAE	0.72	0.7	0.75	0.69	0.61	0.68
	Updated Weights	0.505	0.494	0			
12	MAE	0.72	0.69	0.76	0.69	0.62	0.68
	Updated Weights	0.459	0.537	0.002			

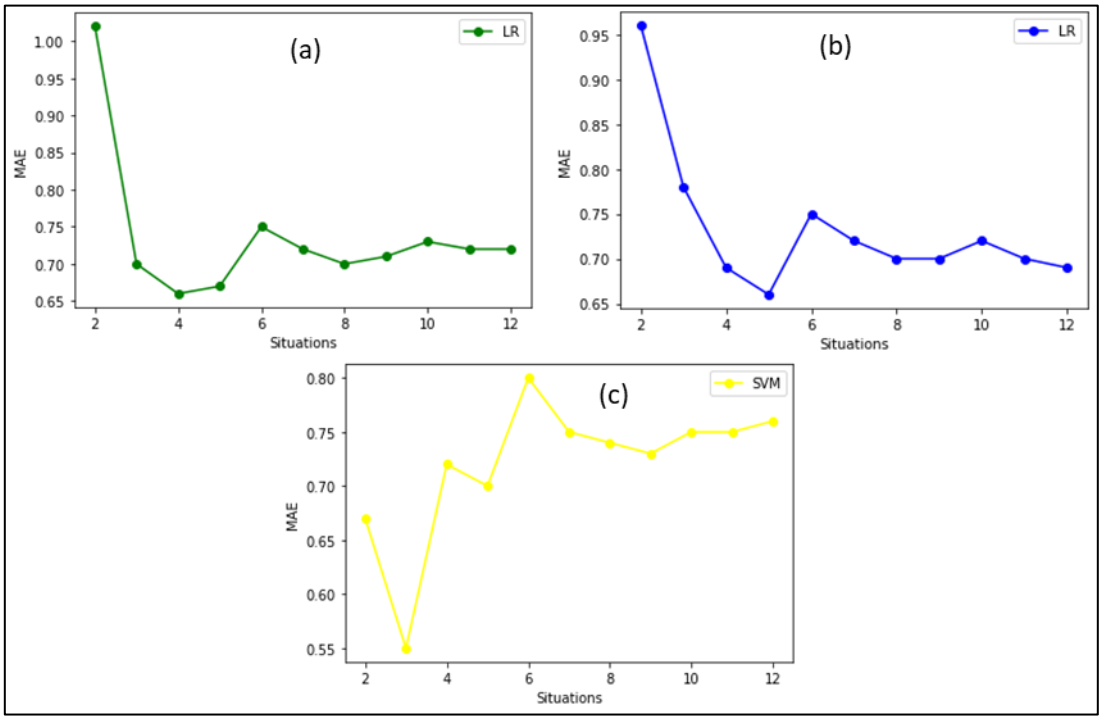

Figure 2: MAE Trend of (a) KNN, (b) SVM, and (c) LR over Different Situations

The graphs shown in Figure 2 represent the Mean Absolute Error (MAE) trends for static individual base models: Linear Regression (LR), K-Nearest Neighbours (KNN), and Super Vector Machine (SVM) across various situations. For situation 1 we have used popularity-based recommendations because of the zero-rating condition. Therefore, the graph-axis represents different situations 2 to situation 12. The y-axis represents the MAE values, indicating the predictive accuracy of each model. Each line on the graph corresponds to a specific model (KNN, SVM, LR).

Linear Regression (LR) shown in Figure 2A demonstrated varied performance, LR starts with a high MAE of 1.02 in Situation 2 and shows improvement in Situation 4 with 0.66 MAE value, indicating an ability to adapt to early training data but LR exhibits consistent performance across

partially populated datasets, with some variability in MAE. LR maintains moderate performance with a slightly higher but consistent MAE in fully populated datasets.

Analyzing the individual base models, we observed that K-Nearest Neighbors Regressor as shown in Figure 2B, the model starts with a high MAE of 0.96 in Situation 2 and shows improvement in Situation 5 with 0.66 MAE, indicating adaptability and learning from early user interactions. The model error increases across partially populated datasets (Situations 6 to 7), but then it maintains relatively stable performance across fully populated datasets (Situations 8 to 12).

Support Vector Machine (SVM) as shown in Figure 2C demonstrates strong initial performance in Situation 3 (0.55) but experiences a high increase in MAE in Situation 4 (0.72), suggesting some variability. SVM shows variability in MAE across different partially populated scenarios, with a peak in Situation 6 (0.80). SVM continues to perform comparatively better in fully populated dataset situations than in previous partially populated dataset situations, showcasing stability.

We conducted experiments with our proposed model using three distinct meta-learners: LR, KNN, and SVM. The models incorporating LR, KNN, and SVM as meta-learners are denoted as DWS_LR, DWS_KNN, and DWS_SVM, respectively. As shown in Figure 3, Figure 4, and Figure 5, our proposed Dynamic Weighted Stacking (DWS) recommendation model consistently outperformed the individual base models across all situations. This suggests that the ensemble model effectively leverages the strengths of each base model and dynamically adjusts their contributions based on the specific characteristics of the given situation. Overall, the DWS ensemble model emerged as the most robust and adaptive, consistently providing improved predictive accuracy compared to the individual base models. This underscores the efficacy of the ensemble strategy in recommendation systems, particularly in handling diverse scenarios with varying numbers of users and ratings.

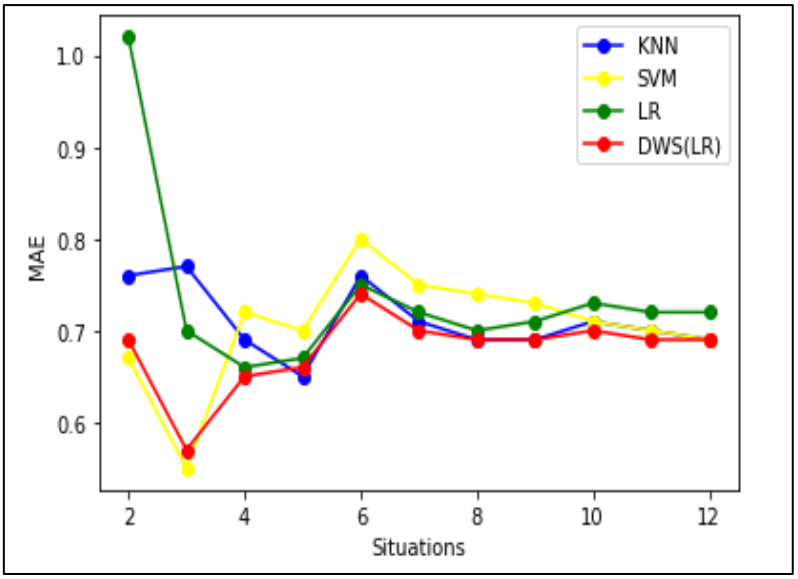

Figure 3: Comparison Graph of DWS (LR) with Other Models over Different Situations

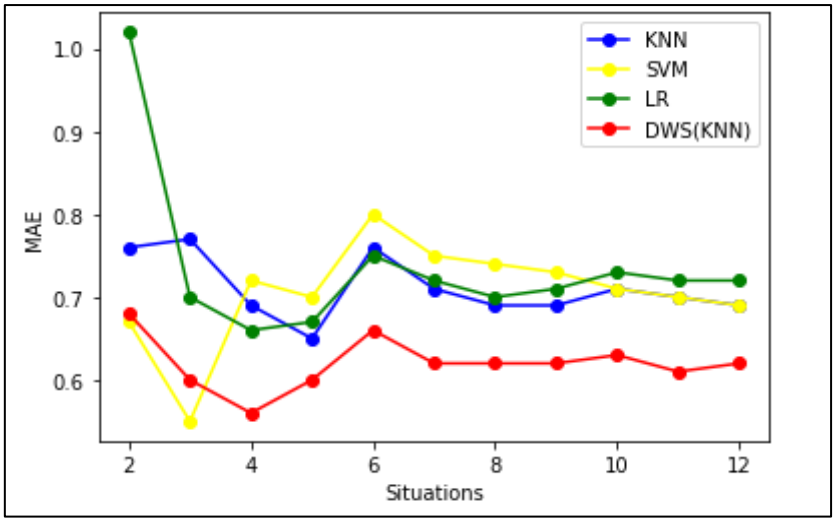

Figure 4: Comparison Graph of DWS (KNN) with Other Models over Different Situations

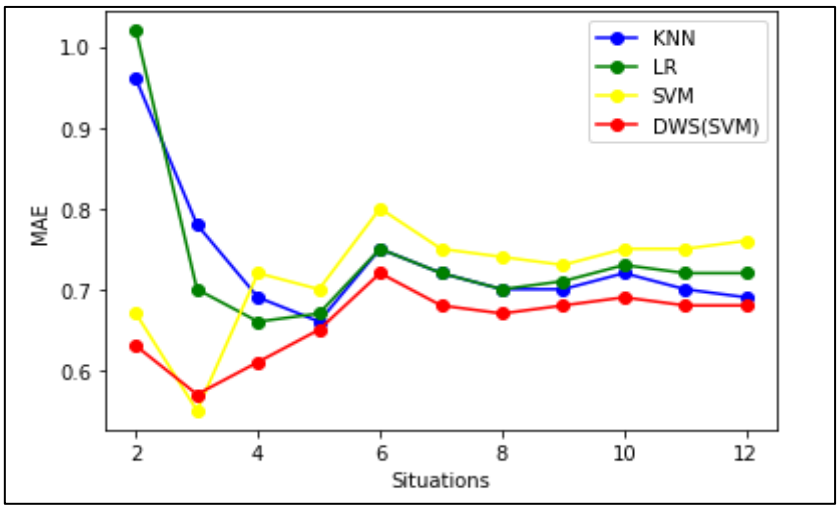

Figure 5: Comparison Graph of DWS (SVM) with Other Models over Different Situations

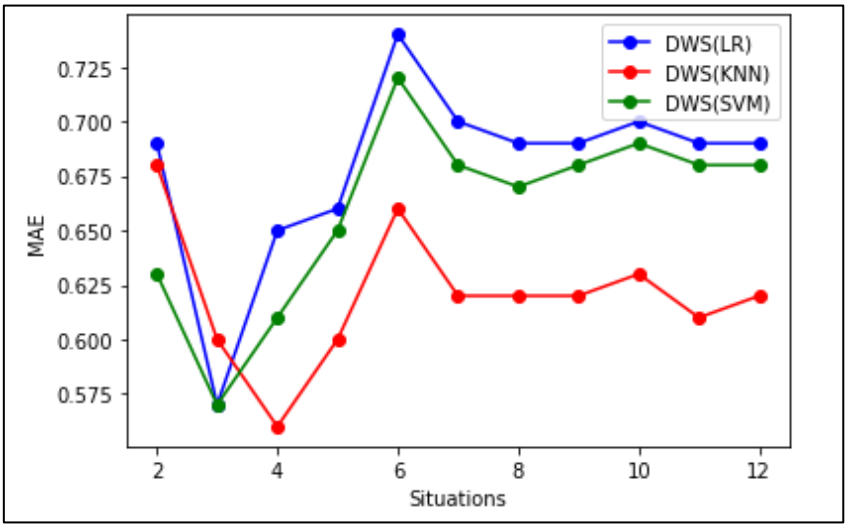

Figure 6: Comparison Graph of DWS_LR, DWS_KNN and DWS_SVM

In Figure 6, we compared the performance of DWS_KNN, DWS_LR, and DWS_SVM across different scenarios. It's evident from Figure 6 that KNN as a meta-learner consistently outperforms LR and SVM in terms of accuracy across all situations.

Afterwards, we conducted experiments using the traditional stacking approach, where we employed

a Bayesian approach for hyperparameter tuning. However, this time, our model remained static, and the final results obtained for each scenario represented the combination of base learner models and meta-learner models without any dynamic weight updates. The results obtained for the traditional static stacking model are presented in Table 3.

Subsequently, we compared dynamic weighted stacking with traditional static stacking and observed that dynamic stacking provided better results by assigning more weight to better-performing models based on observed errors as shown in Figure 6. In contrast, traditional static stacking assigns equal importance to all base learners across all situations. Figure 7 illustrates the comparison graph of traditional stacking and weighted stacking.

Table 3: MAE values of our proposed DWS models and static traditional stacking

Situations	DWS_LR	DWS_KN	DWS_SVM	TRAD_Stack
2	0.69	0.68	0.63	0.62
3	0.57	0.60	0.57	0.62
4	0.65	0.56	0.61	0.65
5	0.66	0.60	0.65	0.66
6	0.74	0.66	0.72	0.75
7	0.70	0.62	0.68	0.70
8	0.69	0.62	0.67	0.70
9	0.69	0.62	0.68	0.69
10	0.70	0.63	0.69	0.71
11	0.69	0.61	0.68	0.70
12	0.69	0.62	0.68	0.70

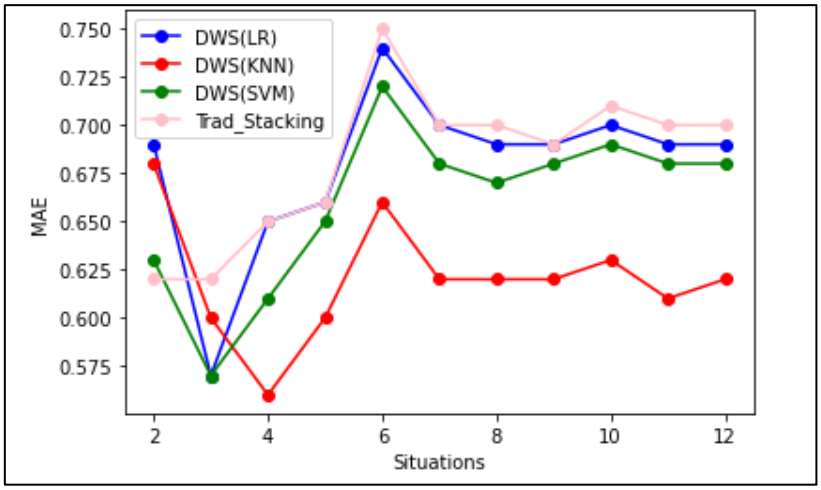

Figure 7: Comparison Graph of Dynamic Weighted Stacking and Static Traditional Stacking

For our proposed model we have utilized three distinct base models KNN, LR, and SVM, and then we examined various scenarios characterized by different user and rating counts. The models are integrated into an ensemble framework, and Bayesian optimization is employed to dynamically optimize weights and hyperparameters. Mean Absolute Error (MAE) is utilized as the evaluation metric. The dynamic approach for adjusting weights enables the model to assign greater weights to models that perform better in real-time, taking into account each scenario's performance. This improves accuracy over the whole range of circumstances by enabling the ensemble to handle both high- and low-rating situations successfully. As data complexity rises, the constant adjustment of weights guarantees that models that contribute more accurate predictions have higher importance, producing better outcomes. This has been shown by the results obtained when comparing individual models with the DWS model. Results indicate that the ensemble model consistently outperforms individual base models and traditional static stacking, showcasing its potential for recommendation system enhancement.

Conclusion

In this study, we have studied the effectiveness of a dynamic ensemble modeling strategy designed for recommendation systems, focusing on several situations with different numbers of users and ratings. Three individual regression models including K-Nearest Neighbours (KNN), Linear Regressor (LR), and Supper Vector Machine (SVM) were used as base models in this experiment. These models were combined into an ensemble framework. Bayesian optimization was used to adjust its hyperparameters and to determine the initial set of optimal weights, while dynamic weight adjustment was applied based on the model's real-time performance. We systematically compared the predictive accuracy of individual base models and static traditional stacking with our proposed DWS model within these different scenarios. The proposed dynamic ensemble model outperformed individual base models and static traditional stacking across the range of conditions examined, continuously producing improved predictive accuracy, demonstrating its flexibility. This adaptability becomes especially advantageous in scenarios with high variability or changing data patterns, where static models struggle to maintain optimal performance. The ensemble's flexibility in handling various data distributions highlights its potential for real-world applications in the field of recommendation systems. This indicates that the dynamic approach is better suited for environments where real-time adaptability is required, such as recommendation systems where user preferences and trends are constantly evolving. The ensemble's flexibility in handling various data distributions highlights its potential for real-world applications in the field of recommendation systems.

Future study could extend this work to a wider range of base models and explore the assessment of ensemble performance on larger and more diverse datasets. Further investigation into the model's effectiveness in other dynamic contexts, such as financial forecasting could provide a deeper understanding of the generalizability of the dynamic method over static substitutes.

Abbreviations

DWS: Dynamic-Weighted Stacking, KNN: K-Nearest Neighbors, LR: Linear Regression, SVM: Support Vector Machine, MAE: Mean Absolute Error, TRAD_Stack: Traditional Stacking, DWS_LR: Dynamic-Weighted Stacking using Linear Regression as meta-learner, DWS_KNN: Dynamic-Weighted Stacking using K-Nearest Neighbors as meta-learner, DWS_SVM: Dynamic-Weighted Stacking using Support Vector Machine as meta-learner.

Acknowledgement

We would like to express our gratitude to Assam down town University and Sunstone School of Technology for providing an enriching research environment.

Author Contributions

Nisha Sharma: Conceptualization, Methodology, Algorithm Design, Code Implementation, Computational Experiments, Data Interpretation, Formal Analysis, Figure Creation, Visualization, Writing - Original Draft, Coordination; Mala Dutta: Supervision, Methodology Validation, Results Validation, Writing - Reviewing and Editing, Conceptual Input, Administration.

Conflict of Interest

Each author works in collaboration without any conflict of interest.

Ethics Approval

Not applicable.

Funding

Nil.

References

1. Guy S, Asela G. Evaluating recommendation systems. In: Recommender systems handbook. Springer. 2010;257-297.
2. Huang D, Liu Z, Wu D. Research on Ensemble Learning-Based Feature Selection Method for Time-Series Prediction. Applied Sciences (Switzerland). 2023 Dec 20;14(1):40.
3. Rao KR, Prasad ML, Kumar GR, Natchadalingam R, Hussain MM, Reddy PCS. Time-Series Cryptocurrency Forecasting Using Ensemble Deep Learning. In: Proceedings of the International Conference on Circuit Power and Computing Technologies, ICCPCT 2023. IEEE. 2023:1446-1451.
4. He K, Yang Q, Ji L, Pan J, Zou Y. Financial Time Series Forecasting with the Deep Learning Ensemble Model. Mathematics. 2023 Feb 20;11(4):1054.
5. Gediminas A, *et al.* Context-Aware Recommender Systems. AI Magazine. 2011;32:67-80.
6. Leo B. Stacked regressions. Machine Learning. 1996;24:49-64.
7. Wolpert DH. Stacked generalization. Neural networks. 1992 Jan 1;5(2):241-259.
8. Wei C, Seung-Taek P. Personalized recommendation on dynamic content using predictive bilinear models.

Proceedings of the 18th international conference on World wide web. Association for Computing Machinery, New York, NY, USA. 2009:691-700.
9. Dietterich TG. Ensemble methods in machine learning. Springer. 2000;1-15. https://doi.org/10.1007/3-540-45014-9_1
10. Carlei V, Adamo G, Ustenko O, Barybina V. Stacking Generalization via Machine Learning for Trend Detection in Financial Time Series. Decision Economics: Minds, Machines, and their Society Studies in Computational Intelligence. Springer International Publishing. 2021; 159-166.https://ricerca.unich.it/handle/11564/808211
11. Sharma AK, Bathula S, Saha K. Method for Improvement of Product Sales Forecast for Long Horizon Using Hybrid Decomposition and Machine Learning on Multi-variate Time Series Data. InICDSMLA 2020: Proceedings of the 2nd International Conference on Data Science, Machine Learning and Applications. Springer Singapore. 2022:303-320. https://link.springer.com/chapter/10.1007/978-981-16-3690-5_27
12. Shalev-Shwartz S. Online learning and online convex optimization. Foundations and Trends® in Machine Learning. 2011;4(2):107–194.
13. D'Addio R, Marcelo M. A Collaborative Filtering Approach Based on User's Reviews. Proceedings - 2014 Brazilian Conference on Intelligent Systems, BRACIS. 2014:204-209.
14. Muandet K, Balduzzi D, Schoelkopf B. Domain generalization via invariant feature representation. In: Proceedings of the 30th International Conference on Machine Learning. 2013;28(1):10-18.
15. Richard SS, Andew GB. Reinforcement learning: An introduction. MIT Press. 2018.
16. Shahriari B, Swersky K, Wang Z, Adams RP, De Freitas N. Taking the human out of the loop: A review of Bayesian optimization. Proceedings of the IEEE. 2016;104(1):148-175.
17. Snoek J, Hugo L, Adams RP. Practical Bayesian optimization of machine learning algorithms. In: Advances in Neural Information Processing Systems. 2012;25(4):2960-2968.
18. Brochu E, Cora, V, Freitas, N. A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning. 2010. arXiv preprint arXiv:1012.2599.
19. Shengbo G. Bayesian Recommender Systems: Models and Algorithms [thesis]. The Australian National University. 2011. http://hdl.handle.net/1885/151280
20. Nisha S, Mala D. An Ensemble Movie Recommender System Based on Stacking. Journal of Theoretical and Applied Information Technology. 2023;100(18):7264-7273.
21. Portugal I, Alencar P, Cowan, D. The Use of Machine Learning Algorithms in Recommender Systems: A Systematic Review. Expert Systems with Applications. 2018;97:205-227.
22. Rashid Al, Albert I, Cosley D, Lam, S, McNee S, Konstan J, Riedl, J. Getting to Know You: Learning New User Preferences in Recommender Systems. International Conference on Intelligent User Interfaces, Proceedings. 2002: 127-134.
23. Wu L, He X, Wang X, Zhang K, Wang M. A Survey on Accuracy-oriented Neural Recommendation: From Collaborative Filtering to Information-rich Recommendation. IEEE Transactions on Knowledge and Data Engineering. 2022 Jan 25;35(5):4425-45.
24. Alamdari PM, Navimipour NJ, Hosseinzadeh M, Safaei AA, Darwesh A. A systematic study on the recommender systems in the E-commerce. IEEE Access. 2020 Jun 16;8:115694-716.
25. Zhang X, Zhao Z, Li C, Zhang Y, Zhao J. An interpretable and scalable recommendation method based on network embedding. IEEE Access. 2019 Jan 16;7:9384-9394. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8613775
26. Rinaldi R, Ridi F, Noor S. A Review Text-based Recommendation System in Text Mining. IEEE International Conference of Computer Science and Information Technology (ICOSNIKOM). 2023: 1-1. https://www.researchgate.net/publication/368542822
27. Henriques PM, Mendes-Moreira J. Combining recommendation systems with a dynamic weighted technique. In: Proceedings of the Eleventh International Conference on Digital Information Management (ICDIM). IEEE. 2016:203-208. https://ieeexplore.ieee.org/document/7829766