Infodeslib: Python Library for Dynamic Ensemble Learning using Late Fusion of Multimodal Data

Firuz Juraev[1], Shaker El-Sappagh[1,2] and Tamer Abuhmed[1,*]

[1] *College of Computing and Informatics, Sungkyunkwan University, South Korea*
[2] *Faculty of Computer Science and Engineering, Galala University, Egypt*

Abstract

There has been a notable increase in research focusing on dynamic selection (DS) techniques within the field of ensemble learning. This leads to the development of various techniques for ensembling multiple classifiers for a specific instance or set of instances during the prediction phase. Despite this progress, the design and development of DS approaches with late fusion settings and their explainability remain unexplored. This work proposes an open-source Python library, Infodeslib, to address this gap. The library provides an implementation of several DS techniques, including four dynamic classifier selections and seven dynamic ensemble selection techniques, all of which are integrated with late data fusion settings and novel explainability features. Infodeslib offers flexibility and customization options, making it a versatile tool for various complex applications that require the fusion of multimodal data and various explainability features. Multimodal data, which integrates information from diverse sources or sensor modalities, is a common and essential setting for real-world problems, enhancing the robustness and depth of data analysis. These data can be fused in two main ways: early fusion, where different modalities are combined at the feature level before model training, and late fusion, where each modality is processed separately and the results are combined at the decision level. The library is fully documented following the Read the Docs standards. The documentation, code, and examples are available anonymously on GitHub at *https://github.com/InfoLab-SKKU/infodeslib*.

Keywords

Ensemble of classifiers, Dynamic classifier selection, Dynamic ensemble selection, multimodal data fusion, Late fusion, Machine learning, Explainable AI, Python.

Ensemble learning is a thriving domain within the fields of machine learning and pattern recognition [1, 2]. With all the diverse ensemble classifiers available, each classifier approaches the problem from a different perspective. The main idea of ensemble learning is to leverage a group of classifiers to provide comprehensive coverage of the learned task [3]. By utilizing diverse models that exhibit distinct decision boundaries, ensemble learning seeks to maximize the accuracy and effectiveness of the overall classification process. As a result, the performance of ensemble classifiers is better than any of its base classifiers [4, 5]. This is because each base classifier concentrates on the specific region of the error space and combining the decisions of these classifiers improves the overall ensemble's decisions. Ensemble learning approaches can be broadly classified into two categories: static and dynamic selection approaches [6, 7]. In static selection [8, 9], a predetermined group of classifiers is selected, and this group is utilized to make decisions for each new test instance. In dynamic selection [10, 11, 12], a new group of classifiers is selected for each test instance, and this group is employed to make a decision for that specific instance.

Since real-world datasets are often complex and consist of multiple feature groups or so-called 'modalities', ensemble learning is a popular candidate to be used to combine multiple models to improve the performance and robustness of predictive models. One approach to ensemble learning is early fusion, where all modalities are merged in a pool for the classifiers to capture the potential interaction and interdependencies among the modalities using either static [13] or dynamic selection [14].

Another approach to ensemble learning is the late fusion or decision fusion, where each classifier in the pool is trained with different feature groups or combinations of feature groups to achieve greater diversity in the model pool. This diversity is crucial for constructing a robust ensemble that can effectively generalize to previously unseen data. Moreover, late fusion provides more flexibility as classifiers are assigned to different modalities considering that certain classifiers are best to model certain modalities [15].

In current literature, late fusion-based ensemble learning is solely available with static classifiers selection [16], and most of these studies show the superiority of late fusion compared to early fusion for static ensemble [17, 18, 19]. This motivates us to explore the performance of late fusion in dynamic selection compared to early fusion; however, to the best of our knowledge, no study or implementation has been conducted to examine the performance of late fusion in dynamic selection settings. This work aims to implement different types of dynamic selection techniques in the late fusion setting. By doing so, we can explore the performance of late fusion-based ensemble learning under dynamic selection modeling, gaining a deeper understanding of its potential advantages and limitations.

Resulting late fusion-based dynamic ensembles are expected to improve the performance of the resulting classifiers. However, these models are black boxes and not understandable. Trustworthy classifiers that are applicable in the real world need to be interpretable. Explainable AI (XAI) has gained significant attention in recent years [20], as it is crucial to provide insights into the decision-making process of machine learning models. However, despite the growing interest in this area, there is a lack of explainability features for ensemble learning techniques, which are increasingly used in complex real-world applications to improve the trustworthiness of resulting models. To the best of our knowledge, no study in the literature and no Python packages are provided to implement XAI capabilities for dynamic ensemble classifiers. This study aims to address this research gap by developing a Python package that offers novel explainability techniques for ensemble models, making them accessible and informative for both domain experts and developers.

KiL'24: Workshop on Knowledge-infused Learning co-located with 30th ACM KDD Conference, August 26, 2024, Barcelona, Spain

[*] Corresponding author.

fjuraev@g.skku.edu (F. Juraev); shaker@skku.edu (S. El-Sappagh); tamer@skku.edu (T. Abuhmed)

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

Table 1
Infodeslib implemented DS methods.

Technique	Selection	Reference
Modified Rank (MR)	DCS	Sabourin et al. [21]
Overall Local Accuracy (OLA)	DCS	Woods et al. [22]
Local Class Accuracy (LCA)	DCS	Woods et al. [22]
Modified Local Accuracy (MLA)	DCS	P.C. Smits [23]
DES-KNN	DES	Soares et al. [24, 25]
K-Nearest Oracles Eliminate (KNORA-E)	DES	Ko et al. [10]
K-Nearest Oracles Union (KNORA-U)	DES	Ko et al. [10]
Weighted KNORA-E (KNORA-E-W)	DES	Ko et al. [10]
Weighted KNORA-U (KNORA-U-W)	DES	Ko et al. [10]
DES Performance (DES-P)	DES	Woloszynski et al. [26]
K-Nearest Output Profiles (KNOP)	DES	Cavalin et al. [27]

The contributions of the study are as follows:

- We extended the literature on dynamic ensemble modeling by implementing four dynamic classifier selection techniques and seven dynamic ensemble selection techniques, incorporating a late fusion of multiple modalities (see Table 1).
- We propose three types of novel explainability that provide deep and suitable XAI for dynamic selection techniques: Case-Based Reasoning, deep-based classifiers contributions, and local feature importance.
- We compare the performance of the proposed techniques with existing approaches on four well-known and real-world multimodal datasets: Alzheimer's Disease Neuroimaging Initiative (ADNI), Credit Card Clients, National Alzheimer's Coordinating Center, and Parkinson's Progression Markers Initiative (PPMI). We also tested the proposed techniques in the Samarkand Neonatal Center dataset which is collected by our team with the help of physicians.
- The implemented techniques have been included in a standard public library called 'Infodeslib' following the industry-standard PEP 8 coding guidelines, and Infodeslib is also clearly documented in accordance with the Read the Docs standards: *https://infodeslib.readthedocs.io/en/latest/*
- We offer a wide range of valuable functions that enable the assessment and evaluation of the excellence and efficacy of the selected pool.

The study is organized as follows. Section 1 highlights the software framework of the proposed late fusion dynamic ensemble learning. Section 2 presents Installation and Usage, Section 4 discusses the performance analysis, and Section 5 introduces possible package extensions. Section 6 concludes the paper.

1. Late Fusion Dynamic Ensemble Framework

In this section, we provide an overview of the late fusion dynamic ensemble framework in algorithmic and visual formats. This encompasses a thorough dissection of the primary stages involved, along with step-by-step explanations of the framework's methodology.

Since late fusion dynamic ensemble utilizes the decision values obtained from each modality and fuses them using a specific fusion mechanism M (such as averaging, weighted averaging, majority voting, etc), let us assume that classifier c_i is applied to modality f_i. The final prediction can be expressed as:

$$p = M(c_1(f_1), c_2(f_2), ..., c_m(f_m)) \tag{1}$$

The proposed concept of dynamic selection with late fusion is illustrated in Figure 1, which outlines a framework consisting of three key stages: training, selection, and prediction. Additionally, the concept is detailed algorithmically in Algorithm 1.

Training Phase. A pool of classifiers is selected and assigned different feature sets. The classifiers within the pool are selected based on their diversity, ensuring a wide range of decision-making capabilities. Each feature set used by selected classifiers is extracted from the same modality to generate a homogeneous feature set. For example, in the medical domain, demographic and MRI features are different modalities that could be used to train two different classifiers. Each classifier in the pool is then trained and optimized with its designated feature set, resulting in a pool of trained classifiers to be utilized in the next phases (1-4 lines in Algorithm 1).

Selection Phase. During the selection phase (5-12 lines of Algorithm 1), a region of competence (RoC) is determined for a given new test instance by selecting the nearest samples from the validation data (DSEL). Subsequently, each classifier in the pool is evaluated on the samples within the RoC, and a measure of competence is calculated for each classifier. The specific method employed to compute the competence varies depending on the chosen DS technique (9-10 lines in Algorithm 1). Once the competencies of each classifier in the pool are calculated, the DS techniques use their own selection criteria to identify the most competent classifiers. These criteria are specific to each DS technique. If no competent classifier satisfies the criteria for a given DS technique, all classifiers in the pool are selected to make the final decision.

Prediction Phase. During the final phase, the selected classifiers are utilized to predict the class of a given test instance, and their individual predictions are combined to generate a final prediction. To provide more accurate decisions, each of the selected classifiers could be weighed based on its level of competence during the aggregation process (line 13 in Algorithm 1).

2. Installation and Usage

Users can conveniently install the most recent version of Infodeslib via pip, the Python package manager, by executing the command `pip install infodeslib`. Alternatively, the library can be installed via the GitHub address, using the command `pip install git+https://github.com/InfoLab-SKKU/infodeslib`.

To use the implemented methods in Infodeslib, a list of classifiers and feature sets must be provided as input. The classifiers in the list can be of any type from the scikit-learn library and should be trained on the corresponding feature set before being used as input.

Once the pool of classifiers and feature sets has been initialized, the method ***fit(X_dsel, y_dsel)*** is applied to fit the Dynamic Selection method, where (X_dsel, y_dsel) is the validation dataset (DSEL) with true labels. Predictions for each test instance x can be obtained using either the ***predict(x)*** or ***predict_proba(x)*** methods. In the example provided below, we demonstrate the steps involved in implementing the KNORA-U technique.

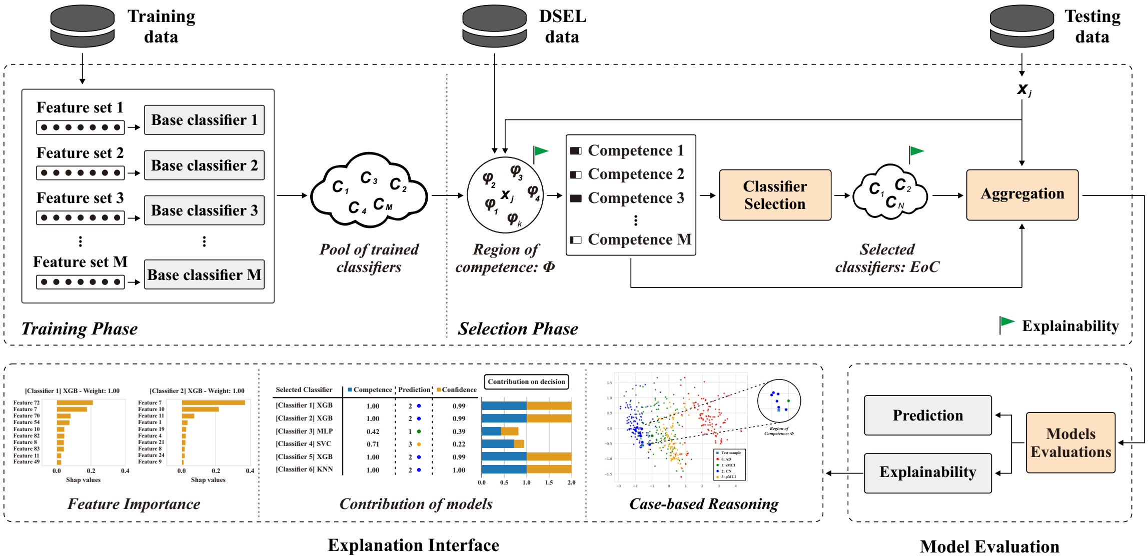


Figure 1: The architecture of the proposed late fusion dynamic ensemble learning framework implemented by Infodeslib.

Algorithm 1 Late fusion DES method

```
    Input: Pool of classifiers $C$, training dataset $D_{tr}$, validation dataset $D_{va}$, testing dataset $D_{te}$, feature set $F$, and neighborhood size $K$
    Output: $EoC_t^*$, an ensemble of classifiers for each testing sample $t$ in $D_{te}$
for each classifier $c_i$ in $C$ do
    Optimize $f_i$ in $F$ for $c_i$;
    Optimize and train $c_i$ on $D_{tr}$ with feature set $f_i$;
end for
for each testing sample $t$ in $D_{te}$ do
    Find $\Psi$ as the K nearest neighbors of the testing sample $t$ in $D_{va}$;
    for each sample $\psi_i$ in $\Psi$ do
        for each classifier $c_i$ in $C$ do
            Calculate competence of $c_i$ on $\Psi$;
            Select ensemble of competent classifiers $EoC_t^*$;
        end for
    end for
    Use the ensemble $EoC_t^*$ to classify $t$;
end for
```

```python
from infodeslib.des.knorau import KNORAU

pool_classifiers = [classifier1, ..., classifierN]
# feature_set1 is a list of columns
feature_sets = [feature_set1, ..., feature_setN]

# Initialize the DS model
knorau = KNORAU(pool_classifiers, feature_sets)

# Fit the dynamic selection model
knorau.fit(X_dsel, y_dsel)

# Predict new examples
knorau.predict(X_test, plot=True)

# Check performance (based on accuracy)
knorau.score(X_test, y_test)
```

When utilizing the ***predict(X)*** method, an additional parameter "plot" can be included to obtain explainability for each test instance. By setting plot=True, explainability for the given test instance can be visualized through a variety of methods (see more details in Section 3).

Infodeslib Methods. Figure 2 provides an overview of the key methods of our library while other supporting methods are available in the documentation of the library. Some of these methods such as *fit()*, *predict()*, *predict_proba()*, and *score()* are well-known and require no detailed explanation; there are several other methods that are particularly useful for pool generation and obtaining information about new test samples. To facilitate pool generation, we have implemented three additional methods: *get_average_accuracy()*, *get_pool_diversity()*, and *get_coverage_score()*. ***get_average_accuracy()*** method computes the average performance of the classifiers in the pool on the validation data. ***get_pool_diversity()*** method calculates the diversity between classifiers in the pool and requires the diversity measure type as a parameter. It supports several diversity functions such as Q-statistic, Correlation Coefficient, Disagreement Measure, Double Fault, Negative Double Fault, and Ratio Errors. ***get_coverage_score()*** method determines the number of samples in the DSEL data that can be accurately predicted by any model in the given pool. This information is particularly useful for evaluating the coverage of the pool and ensuring that all samples are accurately classified by at least one model. The prediction process in machine learning often involves the use of ensemble methods, where multiple classifiers are combined to improve performance. Within these ensembles, three methods play a crucial role: *get_region_of_competence(x)*, *estimate_competence(roc)*, and *select(competences)*.

get_region_of_competence(x) method identifies the region of competence for a given test sample by returning the k nearest neighbors from the validation dataset. This is achieved by applying the k-nearest neighbors algorithm. The ***estimate_competence(roc)*** method calculates the competence of each classifier in the ensemble on the region of competence. The competence calculation differs depending on the technique being used. For example, the k-Nearest Oracle Union (KNORA-U) technique calculates the accuracy of each classifier on the region of competence. The Dynamic

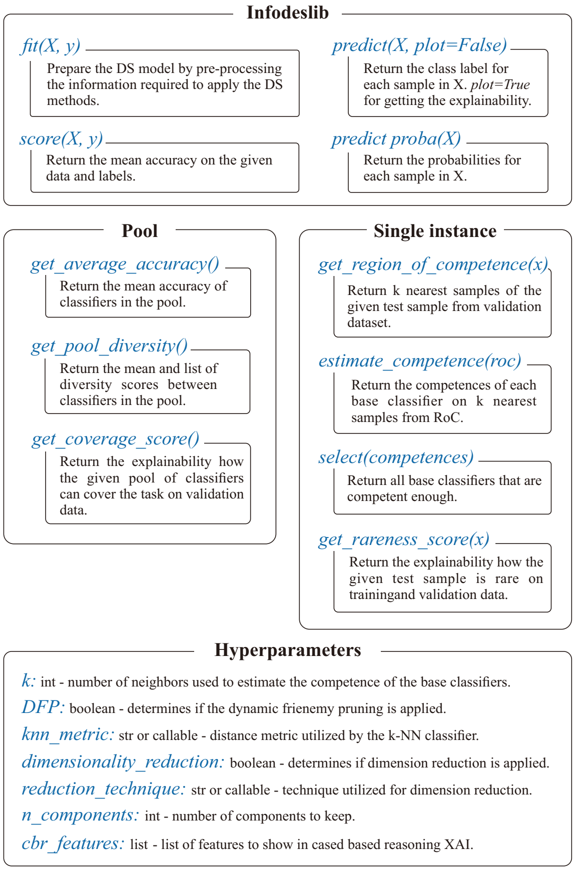

Infodeslib
fit(X, y)
Prepare the DS model by pre-processing the information required to apply the DS methods.
predict(X, plot=False)
Return the class label for each sample in X. plot=True for getting the explainability.
score(X, y)
Return the mean accuracy on the given data and labels.
predict proba(X)
Return the probabilities for each sample in X.
Pool
get_average_accuracy()
Return the mean accuracy of classifiers in the pool.
get_pool_diversity()
Return the mean and list of diversity scores between classifiers in the pool.
get_coverage_score()
Return the explainability how the given pool of classifiers can cover the task on validation data.
Single instance
get_region_of_competence(x)
Return k nearest samples of the given test sample from validation dataset.
estimate_competence(roc)
Return the competences of each base classifier on k nearest samples from RoC.
select(competences)
Return all base classifiers that are competent enough.
get_rareness_score(x)
Return the explainability how the given test sample is rare on trainingand validation data.
Hyperparameters
k: int - number of neighbors used to estimate the competence of the base classifiers.
DFP: boolean - determines if the dynamic frienemy pruning is applied.
knn_metric: str or callable - distance metric utilized by the k-NN classifier.
dimensionality_reduction: boolean - determines if dimension reduction is applied.
reduction_technique: str or callable - technique utilized for dimension reduction.
n_components: int - number of components to keep.
cbr_features: list - list of features to show in cased based reasoning XAI.

Figure 2: The overall schema of the software architecture.

Ensemble Selection KNN (DESKNN) technique, on the other hand, computes each classifier's accuracy and diversity on RoC and uses these metrics to assess its competence.

select(competences) method selects the most competent classifiers from the ensemble to make a prediction. Different techniques may use different criteria for determining the competence of a classifier, such as the number of samples classified correctly within the region of competence. For instance, the KNORA-U technique selects a classifier if it has classified at least one sample within the region of competence. Once the competent classifiers have been identified, their competence values are used as weights in aggregating their predictions. To evaluate a single test instance, our library includes ***get_rareness_score(x)*** method, which provides a detailed description of the instance. The method evaluates whether there are many similar samples to the given instance in the training and validation datasets, allowing users to determine the rarity of the instance. If the instance is an outlier, the method provides information about how far it is from other classes. Furthermore, ***get_rareness_score(x)*** method uses K-means clustering to provide a potential class for the instance and generates tables indicating which features of the instance make it similar to this class. This approach provides valuable insights into the characteristics of the instance and its potential classification, aiding in the development of more accurate models.

Hyperparameters. Optimizing hyperparameters is a critical step for improving the performance of ensemble learning models. This can be achieved through various techniques, including basic approaches such as grid search and random search in Sklearn, as well as more advanced techniques like Genetic algorithms, Bayesian optimization, and others. Our library is designed to work seamlessly with other Python packages such as TPOT [28], Scikit-Optimize [29], Optuna [30], Hyperopt [31], BayesianOptimization [32], GPyOpt [33], Optunity [34], and similar packages that implement these advanced optimization techniques. This allows users to leverage a variety of optimization methods to obtain the best possible hyperparameters for their ensemble models.

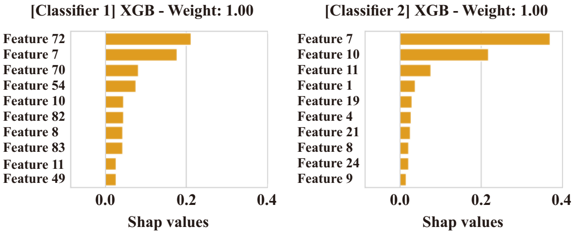
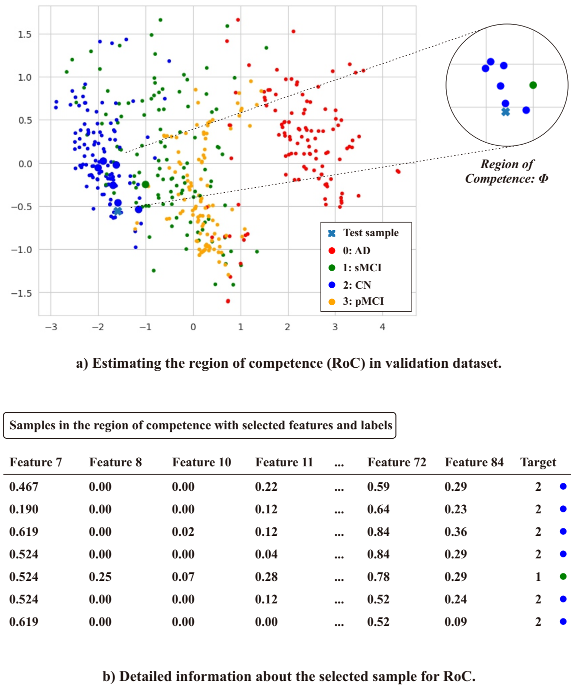


Figure 3: Local feature importance of each selected classifier.

In our library, there are several key hyperparameters that users can adjust to optimize the performance of ensemble learning models. We present these key hyperparameters along with their default values, which have been shown to produce satisfactory results in the majority of cases. One of the main hyperparameters is **k** *(default: 7)*, which represents the number of neighbors to be considered when determining the region of competence. Another important hyperparameter is **DFP** *(default: False)*, which stands for dynamic pruning technique and is particularly useful for imbalanced datasets. In addition, users can also specify the **knn_metric** *(default: 'minkowski')*, which determines the distance metric used when computing distances between the test sample and other samples in the validation dataset. Our library provides several common metrics such as Minkowski, cosine, Manhattan, and Euclidean, as well as the option for users to define their own custom metric function. To handle high-dimensional datasets, we also offer a **dimensionality_reduction** *(default: False)* hyperparameter, which allows users to reduce the number of dimensions used in calculating distances between samples. This can be achieved using either Principal Component Analysis (PCA) or Kernel PCA, or by specifying a custom dimensionality reduction technique using the next **reduction_technique** *(default: 'pca')*. The **n_component** *(default: 20)* hyperparameter determines the number of components to be retained if a reduction technique is selected. Lastly, for those interested in explainability, our library provides the **cbr_features** *(default: None)* hyperparameter, which allows users to specify a list of important features to be included in similar cases data for Case-Based Reasoning.

3. Model Explainability

In the current version of our library, we offer three main XAI techniques: case-based reasoning, deep-based classifier contribution, and local feature importance [20]. The case-based reasoning technique aims to offer domain experts an explanation of the model's prediction process for a given test sample by presenting them with similar samples and their corresponding labels found within the region of



a) Estimating the region of competence (RoC) in validation dataset.

Samples in the region of competence with selected features and labels

Feature 7	Feature 8	Feature 10	Feature 11	...	Feature 72	Feature 84	Target	
0.467	0.00	0.00	0.22	...	0.59	0.29	2	●
0.190	0.00	0.00	0.12	...	0.64	0.23	2	●
0.619	0.00	0.02	0.12	...	0.84	0.36	2	●
0.524	0.00	0.00	0.04	...	0.84	0.29	2	●
0.524	0.25	0.07	0.28	...	0.78	0.29	1	●
0.524	0.00	0.00	0.12	...	0.52	0.24	2	●
0.619	0.00	0.00	0.00	...	0.52	0.09	2	●

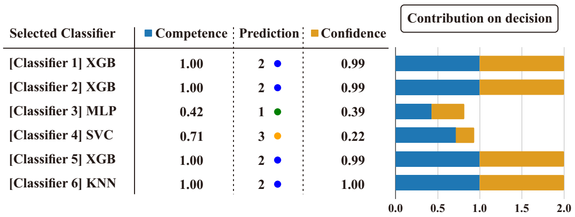
b) Detailed information about the selected sample for RoC.

Figure 4: Estimating a region of competence (RoC) and providing details about the selected sample for RoC.

competence. This approach closely resembles how domain experts make decisions in real-world situations, as they frequently compare current cases with historical ones from their experience. The deep-based classifier contribution technique enables users to comprehend the contribution of each selected classifier in the decision-making process for a given test sample. Finally, the local feature importance technique is a prevalent explainability method that identifies the most crucial features and their corresponding Shap values for each selected classifier.

Case-based reasoning. For example, in the case of the KNORA-U technique, in the selection phase, the nearest neighbors for each test instance are estimated in the validation dataset based on their close similarity to the test sample. The selected samples are used to generate the region of competence for evaluating and selecting classifiers in the pool. Figure 4 a) illustrates an example in which the given test sample (light blue x) falls within the area of class 2, and seven nearest samples are selected, six of which belong to class 2 (blue dots), while one belongs to class 1 (green dot). This finding suggests that, for the given test sample, the chance of it being classified as class 2 is high. Moreover, these samples can also be leveraged for conducting case-based reasoning, which may be particularly valuable for physicians, given that our dataset is in the medical domain. Figure 4 b) provides comprehensive information about all nearest samples within the region of competence, enabling physicians to compare and contrast similar samples and their corresponding labels or diagnoses.

Deep-based classifiers contributions. After selecting the group of classifiers for making the final decision, it may be unclear how each classifier in the pool contributed to the decision or what their individual predictions were for the new test sample. In order to provide a more comprehensive understanding of the decision-making process, an additional level of explainability can be utilized. This is illustrated in Figure 5, which provides detailed information about each classifier in the pool, including their competence level, individual prediction on the new test sample, and confidence level. This explanation provides valuable insight for the development of an ensemble model, as it allows developers to identify classifiers that may have a negative impact on decision-making. For instance, as shown in Figure 5, it is evident that most selected classifiers predict the label of the given test sample as 2 with high confidence, while the SVC classifier predicts it as 3. The SVC classifier demonstrates a higher level of competence in the region of competence, indicating that it has a more significant influence on the decision. If this classifier consistently has a negative impact on many test samples, it may be possible to remove it from the pool of classifiers.



Selected Classifier	■ Competence	Prediction		■ Confidence
[Classifier 1] XGB	1.00	2	●	0.99
[Classifier 2] XGB	1.00	2	●	0.99
[Classifier 3] MLP	0.42	1	●	0.39
[Classifier 4] SVC	0.71	3	●	0.22
[Classifier 5] XGB	1.00	2	●	0.99
[Classifier 6] KNN	1.00	2	●	1.00

Figure 5: The contribution of each selected classifier on the final decision.

Local feature importance. In addition to understanding how the classifiers contributed to the decision-making process, it is also important to identify which features were particularly influential in making those decisions. For the example mentioned earlier, we provide local feature importance for each selected classifier, which can be visualized through Figure 3.

Furthermore, our proposed ensemble models have the ability to provide interpretable explanations using two approaches: surrogate model explainability and post-hoc explainability methods. The surrogate model approach involves creating a simplified model that roughly represents the behavior of the original ensemble model and using this model to explain the ensemble's decisions. On the other hand, post-hoc explainability techniques involve analyzing the ensemble model's decisions after they have been made and providing explanations based on the input features that contributed the most to the decision. Both methods treat our ensemble model as a black box model.

4. Performance Analysis

Within this section, We compare the performance of the proposed architecture with the existing approaches. we provide an overview of the datasets that have been utilized along with a detailed analysis of our proposed techniques.

4.1. Evaluation Datasets

In this section, we outline the five datasets utilized to compare Infodeslib with existing models.

Alzheimer's Disease Neuroimaging Initiative (ADNI) dataset [35]. The study includes a total of 1,371 subjects, with a male gender representation of 54.5%. Participants have been classified into four distinct categories based on their clinical diagnosis, including

Cognitive Normal (CN), Stable Mild Cognitive Impairment (sMCI), Progressive Mild Cognitive Impairment (pMCI), and Alzheimer's Disease (AD) [36]. The distribution of these classes is as follows: 419 CN, 473 sMCI, 140 pMCI, and 339 AD individuals. The dataset has four distinct modalities or feature groups, which contain demographics, cognitive scores, assessment tests, and MRI features.

Credit Card Clients dataset. The study includes a vast participant cohort of 30,000 individuals, with the dataset sourced from the UC Irvine Machine Learning Repository [37]. This is a classification problem that involves determining whether or not a client will make their next payment. The two distinct classes are labeled as 'no' and 'yes', with 23,364 and 6,636 instances, respectively. The dataset has four distinct modalities of features, including demographics, financial, and payment history features.

National Alzheimer's Coordinating Center (NACC) dataset [38]. In this study, we examined a total of 37,547 patients focusing on the Global Clinical Dementia Rating (CDRGLOB) as the primary task. CDRGLOB categorizes patients into five classes based on dementia severity: no impairment (8,253 patients), mild impairment (15,097 patients), moderate impairment (8,346 patients), and severe impairment (5,851 patients). Our analysis included six specific modalities for investigation: demographics, physical health, medications, health history, neuropsychiatric inventory questionnaire, and the geriatric depression scale. These modalities were chosen to comprehensively assess various aspects related to dementia and overall patient health [39].

Parkinson's Progression Markers Initiative (PPMI) dataset [40]. Our study involves 952 patients and focuses on a binary classification task to differentiate between healthy individuals and those diagnosed with Parkinson's disease (PD). Among these patients, 389 are categorized as healthy, while 563 have been diagnosed with PD. The dataset encompasses various information modalities, including subject characteristics, biospecimen data, medical history records, motor function assessments, and non-motor features. This comprehensive dataset enables a thorough analysis to identify potential diagnostic markers and factors associated with PD, facilitating improved understanding and diagnosis of the disease [41].

Samarkand Neonatal Center dataset. Our study involved 347 neonates from the intensive care unit at Samarkand Neonatal Center. The dataset was collected by our team by collaborating physicians in the hospital for a binary classification task to predict whether a neonate survives or passes away. Among these neonates, 303 survived and 44 died during the study period. The dataset comprises a comprehensive set of features categorized into multiple modalities: demographic information, the mother's medical history and information, general notes on the neonate's condition, results from blood tests, and APGAR scores (a standardized assessment of a neonate's health at birth).

4.2. Results

This section contains a comprehensive analysis and comparison of various machine-learning approaches against our proposed late-fusion dynamic ensemble selection model. We collect and present the testing results for each of the considered models. To ensure greater consistency in the results, we have applied the 10-holdout testing method [42]. The results are presented in the form of (mean ± standard deviation). As a pool of classifiers, we utilized the following heterogeneous baseline algorithms: XGboost (XGB), LightGBM (LGBM), Random Forest (RF), Support Vector Classifier (SVC), Multi-Layer Perceptron (MLP), Decision Tree (DT), and k-Nearest Neighbors (KNN).

Table 2
Performance of the different ML approaches using ADNI dataset.

Model Type	Model	Accuracy	Precision	Recall	F1
Single Models	XGB	87.11±2.32	87.50±2.63	87.11±2.32	87.03±2.49
	LGBM	86.74±1.58	87.34±1.96	86.74±1.58	86.70±1.72
	RF	87.11±1.96	87.51±2.22	87.11±1.96	87.08±2.08
[Early] Static Ensemble	Voting	88.08±1.94	88.31±2.07	88.08±1.94	88.05±2.02
	Stacking	86.87±2.02	87.67±2.11	86.87±2.02	86.80±2.15
[Early] Dynamic Ensemble	DESP	88.61±1.96	88.72±2.15	88.61±1.96	88.55±2.08
	KNOP	88.71±1.91	88.80±2.09	88.71±1.91	88.66±2.03
[Late] Static Ensemble	Voting	89.29±1.67	89.39±1.81	89.29±1.67	89.24±1.74
	Stacking	87.65±1.59	88.13±1.64	87.65±1.59	87.60±1.74
[Late] Dynamic Ensemble	KNORAU	89.52±2.01	89.77±2.01	89.52±2.01	89.46±2.10
	KNORAU-W	89.84±1.83	90.29±2.03	89.81±1.83	89.80±1.91

Table 3
Performance of the different ML approaches on Credit Card Clients dataset.

Model Type	Model	Accuracy	Precision	Recall	F1
Single Models	XGB	81.96±0.84	89.05±0.72	72.87±1.28	80.15±1.02
	LGBM	80.43±0.64	86.88±0.66	71.69±0.92	78.56±0.76
	RF	79.90±0.78	87.49±0.70	69.76±1.13	77.63±0.95
[Early] Static Ensemble	Voting	83.94±0.74	88.52±0.72	78.01±1.11	82.93±0.84
	Stacking	82.68±0.69	86.72±0.77	77.17±0.95	81.67±0.77
[Early] Dynamic Ensemble	DESKNN	83.99±0.38	89.34±0.53	77.18±0.60	82.82±0.42
	KNORAE	84.16±0.66	88.81±0.64	78.17±0.97	83.15±0.75
[Late] Static Ensemble	Voting	85.72±0.44	89.83±0.57	80.56±0.73	84.94±0.49
	Stacking	85.08±0.47	89.30±0.36	79.71±0.94	84.23±0.57
[Late] Dynamic Ensemble	KNOP	86.65±0.23	91.64±0.28	80.66±0.52	85.80±0.28
	KNORAU-W	86.73±0.29	91.76±0.33	80.70±0.64	85.87±0.36

Results based on ADNI dataset. Table 2 and Figure 6 a) show the top-performing results achieved by the individual models, as well as the static ensemble with early fusion, the dynamic ensemble with early fusion, the static ensemble with late fusion, and our proposed technique - the dynamic ensemble with late fusion setting. From each group, we selected the best-performed techniques and the results show that our dynamic ensemble techniques, KNORAU and KNORAU-W outperform all existing approaches with 89.52% and 89.84% accuracy. In comparison, a static ensemble with late fusion, voting classifier, achieves an accuracy of 89.29%. This performance is close to the performance of our model and surpasses that of early fusion techniques. This result supports our claim for the significance of late fusion in producing accurate ensemble models.

Results based on Credit Card Clients dataset. Table 3 and Figure 6 b) present the results obtained from the analysis of the Credit Card Clients dataset, following a similar format to the previous dataset. Our proposed techniques have once again outperformed the existing approaches in this instance. Specifically, KNOP and KNORAU-W, utilizing the late fusion setting, have achieved the highest accuracy scores of 86.65% and 86.73%, respectively. In comparison, the static ensemble methods that apply late fusion, specifically the voting and stacking classifiers, demonstrate accuracies of 85.72% and 85.08%, respectively. In contrast, the ensemble methods that employ early fusion achieve the highest accuracy of 84.16%, with the dynamic selection technique known as KNORA-E. These results support our argument regarding the importance of utilizing late fusion for the purpose of producing highly accurate ensemble models.

Results based on NACC dataset. Table 4 highlights the results from the analysis of the National Alzheimer's Coordinating Center dataset, structured similarly to the previous dataset. Among all existing techniques, the dynamic ensemble models with late fusion demonstrate notably su-

Table 4
Performance of the ML approaches on the NACC dataset.

Model Type	Model	Accuracy	Precision	Recall	F1
Single Models	GB	85.70+1.16	85.71+1.16	85.77+1.07	85.56+1.19
	XGB	86.30+1.47	86.30+1.47	86.32+1.48	86.18+1.51
	RF	86.79+0.74	86.79+0.74	86.76+0.82	86.69+0.76
[Early] Static Ensemble	Voting	87.52+0.90	87.53+0.90	87.42+0.93	87.40+0.90
	Stacking	87.17+1.29	87.17+1.29	87.20+1.19	87.11+1.26
[Early] Dynamic Ensemble	DESKNN	87.30+1.16	87.61+1.16	87.37+1.06	87.39+1.14
	KNORAU	88.34+1.44	88.34+1.44	88.34+1.41	88.27+1.45
[Late] Static Ensemble	Voting	89.39+1.34	89.39+1.34	89.43+1.30	89.36+1.33
	Stacking	89.11+0.89	89.11+0.89	89.15+0.95	89.07+0.91
[Late] Dynamic Ensemble	KNORAU-W	90.20+1.10	90.20+1.10	90.30+1.09	90.20+1.11
	DESP	**91.16+0.93**	**91.21+0.93**	**91.14+0.89**	**91.17+0.92**

Table 5
Performance of the ML approaches on the PPMI dataset.

Model Type	Model	Accuracy	Precision	Recall	F1
Single Models	RF	92.40±1.00	93.40±0.90	92.10±1.00	92.10±1.00
	XGB	93.40±1.50	93.60±1.80	93.10±1.40	93.10±1.40
	LGBM	93.90±1.60	93.90±2.00	93.70±1.40	93.70±1.40
[Early] Static Ensemble	Voting	94.20±0.70	94.20±0.80	94.00±0.70	94.00±0.70
	Stacking	94.10±0.90	94.00±1.10	93.90±0.90	93.90±0.90
[Early] Dynamic Ensemble	KNOP	94.20±1.10	94.30±1.20	93.90±1.20	93.90±1.20
	KNORAU	94.30±0.90	94.40±1.10	94.00±0.90	94.00±0.90
[Late] Static Ensemble	Voting	94.60±0.90	94.60±1.10	94.30±0.90	94.30±0.90
	Stacking	94.50±0.80	94.70±0.80	94.20±0.80	94.20±0.80
[Late] Dynamic Ensemble	DESP	95.00±0.90	95.20±0.90	94.70±1.00	94.70±1.00
	KNOP	**95.10±0.60**	**95.40±0.70**	**94.70±0.60**	**94.70±0.60**

perior performance. Specifically, the weighted KNORAU (KNORAU-W) and DESP achieve the highest scores at 90.20% and 91.16%, respectively. Given the substantial dataset size, the results are well-balanced across various metrics.

Results based on PPMI dataset. Table 5 presents the results obtained from the analysis of the Parkinson's Progression Markers Initiative dataset, following a format similar to the previous dataset. Within this dataset, the techniques DESP and KNOP, utilizing late fusion settings, exhibit the most robust performance among other algorithms, achieving accuracies of 95% and 95.1%, respectively. Additionally, static ensemble models with late fusion settings demonstrate strong performance at 94.6% accuracy using a voting technique. These results only marginally exceed those achieved with LGBM alone, which achieved a performance of 93.9%.

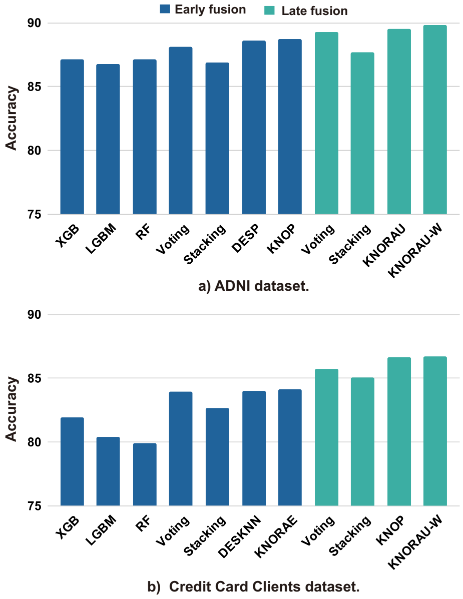
The fact that the LGBM achieved a high accuracy of 93.9% suggests that the task at hand is not very complex. Improving accuracy beyond this point becomes more challenging when a basic technique like LGBM already performs well. Essentially, reaching significantly higher accuracies with more advanced methods might be difficult because the task is relatively straightforward.

Results based on Samarkand Neonatal Center dataset. Table 6 presents the results obtained from analyzing the Samarkand Neonatal Center ICU dataset, following a similar structure to the previous datasets. Due to the dataset's small size, the results may not be consistent or balanced across different metrics. Nonetheless, our proposed late fusion-based dynamic ensemble models achieve notably higher performance compared to other techniques, reaching 77.57% accuracy with the KNOP technique.

Across all five datasets analyzed, the importance of late fusion can be seen in the results. In each dataset, the dynamic ensemble models with late fusion settings outperform other existing models. Combining late fusion with dynamic ensemble learning consistently delivers promising and improved results. This highlights the effectiveness and reliability of employing late fusion techniques within dynamic ensemble models across various datasets.

Table 6
Performance of the different ML approaches on Samarkand Neonatal Center dataset.

Model Type	Model	Accuracy	Precision	Recall	F1
Single Models	RF	69.34+4.66	69.34+4.66	73.70+4.20	67.71+5.44
	XGB	69.74+7.64	69.74+7.64	74.29+6.31	67.77+9.16
	LGBM	70.07+8.58	70.07+8.58	72.65+7.52	68.74+9.62
[Early] Static Ensemble	Voting	73.03+8.03	73.03+8.03	75.50+6.57	72.00+8.95
	Stacking	71.45+5.11	71.45+5.11	75.70+4.39	70.06+5.87
[Early] Dynamic Ensemble	KNORAU	71.64+6.84	71.64+6.84	75.30+5.37	70.28+7.74
	DESP	72.45+6.03	75.95+6.03	72.48+5.24	71.48+7.05
[Late] Static Ensemble	Voting	75.07+6.94	75.07+6.94	77.96+5.19	74.12+7.90
	Stacking	74.21+7.72	74.21+7.72	77.95+5.38	72.84+9.14
[Late] Dynamic Ensemble	KNORAU-W	75.66+7.44	75.66+7.44	78.04+6.09	74.90+8.05
	KNOP	**77.57+5.81**	**77.57+5.81**	**80.58+4.21**	**76.84+6.35**



a) ADNI dataset.

b) Credit Card Clients dataset.

Figure 6: Contribution of each selected classifier to the final decision.

5. Library extension

The primary focus of our paper is to introduce a novel approach to dynamic ensemble selection (DES) that utilizes a late fusion strategy for effectively fusing multi-modal data and offers a high degree of explainability for dynamic selection techniques. Our current library offers implementations of four dynamic classifier selection and seven dynamic ensemble selection techniques that use a late fusion strategy. In addition, the library includes several features and options that enhance its performance and capability. Furthermore, the library provides three different types of explainability to help users gain insights into the decision-making processes of the models. Finally, the library has been designed to be compatible with other important libraries, allowing users to easily integrate it into their existing workflows.

We plan to continue exploring the domain of explainability in ensemble learning by proposing additional techniques for providing comprehensive explanations to domain experts. Our goal is to enhance our library's ability to provide context-based explanations that are tailored to the specific needs of users. Additionally, we aim to incorporate what-if explainability features that enable developers to gain deeper insights into the behavior of their ensemble models. These features will be included in future versions of our library.

Through our experimental evaluations, we have discovered that selecting an appropriate pool of classifiers with matching feature groups is a critical aspect of successful ensemble modeling. However, identifying the ideal combination of classifiers for the pool remains a challenging task. In future versions of our library, we plan to address this issue by developing an automatic optimization process for the selection of the optimal pool of classifiers. We believe this to be a crucial task in the field of ensemble learning, and we are committed to exploring ways to simplify this process and make it more effective.

6. Conclusion

This paper presents a novel approach to dynamic selection using a late fusion setting, which is implemented across four dynamic classifier selection and seven dynamic ensemble selection techniques. This late fusion-based approach is particularly well-suited for complex tasks based on multimodal datasets containing multiple feature groups, which are common in real-world scenarios. As a result, the role of late fusion is crucial in the context of ensemble learning for ensuring diversity in the pool of classifiers. Furthermore, we introduce a novel approach to explainability for dynamic selection techniques. Our proposed approach goes beyond the traditional methods and provides a more in-depth and nuanced understanding of the dynamic selection process. The effectiveness of our proposed techniques is evaluated through a comprehensive comparison with existing baseline approaches. The experimental results demonstrate the superior performance of our proposed techniques over the existing approaches, highlighting the potential of our approach to improving the accuracy and reliability of ensemble learning systems.

Acknowledgments

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT)(No. 2021R1A2C1011198), (Institute for Information & communications Technology Planning & Evaluation) (IITP) grant funded by the Korea government (MSIT) under the ICT Creative Consilience Program (IITP-2021-2020-0-01821), and AI Platform to Fully Adapt and Reflect Privacy-Policy Changes (No.RS-2022-II220688).

References

[1] H. Xiao, Z. Xiao, Y. Wang, Ensemble classification based on supervised clustering for credit scoring, Applied Soft Computing 43 (2016) 73–86.

[2] D. Di Nucci, F. Palomba, R. Oliveto, A. De Lucia, Dynamic selection of classifiers in bug prediction: An adaptive method, IEEE Transactions on Emerging Topics in Computational Intelligence 1 (2017) 202–212.

[3] O. Sagi, L. Rokach, Ensemble learning: A survey, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 8 (2018) e1249.

[4] L. I. Kuncheva, A theoretical study on six classifier fusion strategies, IEEE Transactions on pattern analysis and machine intelligence 24 (2002) 281–286.

[5] M. Fernández-Delgado, E. Cernadas, S. Barro, D. Amorim, Do we need hundreds of classifiers to solve real world classification problems?, The journal of machine learning research 15 (2014) 3133–3181.

[6] A. S. Britto Jr, R. Sabourin, L. E. Oliveira, Dynamic selection of classifiers—a comprehensive review, Pattern recognition 47 (2014) 3665–3680.

[7] X. Dong, Z. Yu, W. Cao, Y. Shi, Q. Ma, A survey on ensemble learning, Frontiers of Computer Science 14 (2020) 241–258.

[8] L. Breiman, Bagging predictors, Machine learning 24 (1996) 123–140.

[9] G. Sakkis, I. Androutsopoulos, G. Paliouras, V. Karkaletsis, C. D. Spyropoulos, P. Stamatopoulos, Stacking classifiers for anti-spam filtering of e-mail, arXiv preprint cs/0106040 (2001).

[10] A. H. Ko, R. Sabourin, A. S. Britto Jr, From dynamic classifier selection to dynamic ensemble selection, Pattern recognition 41 (2008) 1718–1731.

[11] R. M. Cruz, R. Sabourin, G. D. Cavalcanti, Dynamic classifier selection: Recent advances and perspectives, Information Fusion 41 (2018) 195–216.

[12] F. Juraev, S. El-Sappagh, E. Abdukhamidov, F. Ali, T. Abuhmed, Multilayer dynamic ensemble model for intensive care unit mortality prediction of neonate patients, Journal of Biomedical Informatics 135 (2022) 104216.

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: Machine learning in python, the Journal of machine Learning research 12 (2011) 2825–2830.

[14] R. M. Cruz, L. G. Hafemann, R. Sabourin, G. D. Cavalcanti, Deslib: A dynamic ensemble selection library in python, The Journal of Machine Learning Research 21 (2020) 283–287.

[15] K. Liu, Y. Li, N. Xu, P. Natarajan, Learn to combine modalities in multimodal deep learning, arXiv preprint arXiv:1805.11730 (2018).

[16] S. Raschka, Mlxtend: Providing machine learning and data science utilities and extensions to python's scientific computing stack, The Journal of Open Source Software 3 (2018). URL: https://joss.theoj.org/papers/10.21105/joss.00638. doi:10.21105/joss.00638.

[17] S. El-Sappagh, F. Ali, T. Abuhmed, J. Singh, J. M. Alonso, Automatic detection of alzheimer's disease progression: An efficient information fusion approach with heterogeneous ensemble classifiers, Neurocomputing 512 (2022) 203–224.

[18] C. G. Snoek, M. Worring, A. W. Smeulders, Early versus late fusion in semantic video analysis, in: Proceedings of the 13th annual ACM international conference on Multimedia, 2005, pp. 399–402.

[19] F. Juraev, S. El-Sappagh, T. Abuhmed, Explainable dynamic ensemble framework for classification based on the late fusion of heterogeneous multimodal data, in: Intelligent Systems Conference, Springer, 2023, pp. 555–570.

[20] S. Ali, T. Abuhmed, S. El-Sappagh, K. Muhammad, J. M. Alonso-Moral, R. Confalonieri, R. Guidotti, J. Del Ser, N. Díaz-Rodríguez, F. Herrera, Explainable artificial intelligence (xai): What we know and what is left to attain trustworthy artificial intelligence, Information Fusion (2023) 101805.

[21] M. Sabourin, A. Mitiche, D. Thomas, G. Nagy, Classifier combination for hand-printed digit recognition, in: Proceedings of 2nd International Conference on Document Analysis and Recognition (ICDAR'93), IEEE, 1993, pp. 163–166.

[22] K. Woods, W. P. Kegelmeyer, K. Bowyer, Combination of multiple classifiers using local accuracy estimates, IEEE transactions on pattern analysis and machine intelligence 19 (1997) 405–410.

[23] P. C. Smits, Multiple classifier systems for supervised remote sensing image classification based on dynamic classifier selection, IEEE Transactions on Geoscience and Remote Sensing 40 (2002) 801–813.

[24] R. G. Soares, A. Santana, A. M. Canuto, M. C. P. de Souto, Using accuracy and diversity to select classifiers to build ensembles, in: The 2006 IEEE International Joint Conference on Neural Network Proceedings, IEEE, 2006, pp. 1310–1316.

[25] M. C. de Souto, R. G. Soares, A. Santana, A. M. Canuto, Empirical comparison of dynamic classifier selection methods based on diversity and accuracy for building ensembles, in: 2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence), IEEE, 2008, pp. 1480–1487.

[26] T. Woloszynski, M. Kurzynski, P. Podsiadlo, G. W. Stachowiak, A measure of competence based on random classification for dynamic ensemble selection, Information Fusion 13 (2012) 207–213.

[27] P. R. Cavalin, R. Sabourin, C. Y. Suen, Dynamic selection approaches for multiple classifier systems, Neural computing and applications 22 (2013) 673–688.

[28] R. S. Olson, J. H. Moore, Tpot: A tree-based pipeline optimization tool for automating machine learning, in: Workshop on automatic machine learning, PMLR, 2016, pp. 66–74.

[29] T. Head, M. Kumar, H. Nahrstaedt, G. Louppe, I. Shcherbatyi, scikit-optimize/scikit-optimize: v0. 8.1, Zenodo (2020).

[30] T. Akiba, S. Sano, T. Yanase, T. Ohta, M. Koyama, Optuna: A next-generation hyperparameter optimization framework, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2019.

[31] J. Bergstra, D. Yamins, D. Cox, Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures, in: International conference on machine learning, PMLR, 2013, pp. 115–123.

[32] F. Nogueira, et al., Bayesian optimization: Open source constrained global optimization tool for python, URL https://github. com/fmfn/BayesianOptimization (2014).

[33] T. G. authors, Gpyopt: A bayesian optimization framework in python, http://github.com/SheffieldML/GPyOpt, 2016.

[34] M. Claesen, J. Simm, D. Popovic, Y. Moreau, B. De Moor, Easy hyperparameter search using optunity, arXiv preprint arXiv:1412.1114 (2014).

[35] S. G. Mueller, M. W. Weiner, L. J. Thal, R. C. Petersen, C. Jack, W. Jagust, J. Q. Trojanowski, A. W. Toga, L. Beckett, The alzheimer's disease neuroimaging initiative, Neuroimaging Clinics of North America 15 (2005) 869.

[36] N. Rahim, T. Abuhmed, S. Mirjalili, S. El-Sappagh, K. Muhammad, Time-series visual explainability for alzheimer's disease progression detection for smart healthcare, Alexandria Engineering Journal 82 (2023) 484–502.

[37] A. Asuncion, D. Newman, Uci machine learning repository, 2007.

[38] D. L. Beekly, E. M. Ramos, W. W. Lee, W. D. Deitrich, M. E. Jacka, J. Wu, J. L. Hubbard, T. D. Koepsell, J. C. Morris, W. A. Kukull, et al., The national alzheimer's coordinating center (nacc) database: the uniform data set, Alzheimer Disease & Associated Disorders 21 (2007) 249–258.

[39] N. Rahim, S. El-Sappagh, H. Rizk, O. A. El-serafy, T. Abuhmed, Information fusion-based bayesian optimized heterogeneous deep ensemble model based on longitudinal neuroimaging data, Applied Soft Computing 162 (2024) 111749.

[40] K. Marek, D. Jennings, S. Lasch, A. Siderowf, C. Tanner, T. Simuni, C. Coffey, K. Kieburtz, E. Flagg, S. Chowdhury, et al., The parkinson progression marker initiative (ppmi), Progress in neurobiology 95 (2011) 629–635.

[41] M. Junaid, S. Ali, F. Eid, S. El-Sappagh, T. Abuhmed, Explainable machine learning models based on multimodal time-series data for the early detection of parkinson's disease, Computer Methods and Programs in Biomedicine 234 (2023) 107495.

[42] C. Sammut, G. I. Webb (Eds.), Holdout Evaluation, Springer US, Boston, MA, 2010, pp. 506–507. URL: https://doi.org/10.1007/978-0-387-30164-8_369. doi:10.1007/978-0-387-30164-8_369.