



Article

3MuViS, 3-Phase Multi-View Stacking: A Model Selection Algorithm for Multi-View Fusion

Guillermo Villegas-Morales , Enrique Garcia-Ceja * , Salvador Hinojosa , Jesús Arturo Pérez-Díaz and Mahdi Zareei

Tecnologico de Monterrey, School of Engineering and Sciences, Av. Eugenio Garza Sada 2501 Sur, Monterrey 64849, NL, Mexico; A01637169@exatec.tec.mx (G.V.-M.); salvador.hinojosa@tec.mx (S.H.); jesus.arturo.perez@tec.mx (J.A.P.-D.); m.zareei@tec.mx (M.Z.)

* Correspondence: enrique.gc@tec.mx

Abstract

Multi-view learning has emerged as an effective paradigm for integrating heterogeneous data representations in complex classification problems, yet selecting suitable learners for Multi-View Stacking architectures remains computationally challenging and highly dependent on expert decisions. This work proposes 3MuViS, a three-phase methodology for automatic learner selection in Multi-View Stacking that optimizes both view-level learners and the meta-learner according to a target classification metric. The method evaluates candidate machine learning algorithms through cross-validation and constructs optimized stacking configurations. Experiments were conducted on six heterogeneous datasets spanning network intrusion detection, human activity recognition, handwritten digit classification, and transportation mode detection. Performance was evaluated over 25 iterations using the Matthews Correlation Coefficient (MCC), and statistical significance was assessed using Wilcoxon tests. The results show that 3MuViS consistently outperformed stochastic selection strategies and achieved superior or comparable performance to fixed models in five of the six evaluated datasets while frequently approaching the Brute Force upper-bound baseline at a substantially lower computational cost. The findings indicate that jointly optimizing view-level and meta-level learners improves both predictive performance and stability, demonstrating the potential of 3MuViS as a general and efficient framework for multi-view classification problems across diverse domains.

Keywords: multi-view learning; ensemble learning; stacking; optimization; sensor fusion; multi-modal data



Academic Editors: Feiping Nie and Jaroslaw Krzywanski

Received: 2 June 2026

Revised: 10 July 2026

Accepted: 17 July 2026

Published: 3 August 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

The rapid proliferation of distributed sensing, edge computing, and cloud infrastructure has catalyzed the generation of high-velocity, multi-modal data across both digital and physical domains. Traditional machine learning techniques fall short in discerning and comprehending patterns. This escalating complexity necessitates multi-source data fusion frameworks that are not only robust but specifically designed to abstract relations between separate information sources and yield more accurate models than those provided by any single source alone [1].

Multi-view learning (MVL) enhances basic data fusion by explicitly modeling the underlying relationships and correlations between distinct ‘views.’ By moving beyond traditional single-view paradigms, MVL leverages modality-specific properties to mitigate

the curse of dimensionality and uncover complementary insights. These techniques are pivotal in fields requiring multi-faceted understanding, including cybersecurity, behavior analysis, computer-aided medical diagnosis, and autonomous driving, to name a few [2].

Formally, a view is defined as an independent data source capable of describing an observed event from a specific perspective. For instance, in video classification, audio and visual streams constitute separate views. Similarly, in human activity recognition (HAR), heterogeneous sensors—such as accelerometers and gyroscopes—each provide a distinct viewpoint on the subject’s motion [3]. Broadly, datasets can be naturally or synthetically partitioned into independent feature groups, where each group represents a unique view [4].

Ongoing research works on classification methods for multi-view problems report improvements of multi-view specific approaches over information fusion at feature level or using individual sources of information [5]. One of such examples is the Multi-View Stacking algorithm (MVS) [3], which trains specialized *view-level learners* for each individual view. These learners generate class membership scores that serve as input for a high-level *meta-learner*, which synthesizes these generated features into a final prediction.

The MVS framework has proven effective across a diverse spectrum of applications, including image classification using topological features [6], weather forecasting [7], activity recognition [8], and gene expression analysis [9], to name a few. Although MVS has been proven to perform better than other approaches such as feature fusion, a critical bottleneck remains: the selection of the learners for each view and the meta-learner was done, up to this point, manually or guided by domain knowledge. Such ad hoc selection strategies are inherently sub-optimal, as they fail to systematically explore the complex interactions between specific model architectures and view-specific data distributions.

To address this gap, we introduce 3MuViS (3-Phase Multi-View Stacking), an algorithm designed to automate the selection of the optimal learner configuration. Our methodology provides a systematic three-phase approach to model the optimization:

- Phase I (Local Optimization): Selection of the view-level learners with the highest generalization performance for each independent view.
- Phase II (Meta-Optimization): Identification of the optimal meta-learner architecture given the fixed view-level learners from Phase I.
- Phase III (Model Integration): Final construction of the optimized stacking ensemble using the complete training set.

To evaluate the efficacy of the proposed framework, this study addresses the following research questions:

- RQ1: Does the 3MuViS automated selection process significantly outperform Multi-View Stacking models configured with a fixed learner?
- RQ2: How does 3MuViS compare against stochastic (random) selection strategies in terms of classification performance?
- RQ3: Is the performance improvement of 3MuViS statistically significant across different domains (cybersecurity, behavior/activity recognition, and digit recognition)?
- RQ4: To what extent is performance gain driven by the optimization of both the view-level learners and meta-learner versus the optimization of only the meta-learner?
- RQ5: How does 3MuViS perform in subject-independent scenarios?

The main contribution of this paper lies in a novel methodology that, based on the conducted experiments, effectively selects the best learners to construct a superior Multi-View Stacking model compared with traditional feature fusion and manual selection baselines, offering a scalable solution for high-dimensional multi-view problems. In particular, our core contribution is three-fold:

- We propose 3MuViS, a systematic approach for learner selection moving beyond the sub-optimal manual heuristics of current manual approaches.
- We demonstrate that 3MuViS maintains linear scalability $O(|L| \cdot |V|)$, offering a computationally viable alternative to exhaustive search (brute force).
- We demonstrate the robustness of our approach across six heterogeneous datasets, proving its effectiveness in diverse domains from cybersecurity to HAR.

This manuscript is structured as follows: Section 2 provides a detailed framework on multi-view learning and Multi-View Stacking. On Section 3, we explain the 3MuViS methodology thoroughly and provide details on its implementation and the potential of the algorithm's application. The aim of Section 3.2 is to give a brief description of the datasets used for testing the methodology herein described. Performance measurements regarding classification metrics and computational time compared with other methods for choosing learners are reported in Section 4. In Section 5 we discuss the results. Finally, we conclude our work in Section 6.

2. Background

2.1. Related Work

The inherent complexity of MVS necessitates a move beyond static ensembles toward an optimized architectural configuration. Recent advances on Multi-View Stacking focus on designing automated frameworks that improve certain aspects of the model by changing its components. Such is the work by van Loon et al. [9] that addresses the challenge of high-dimensional genomic data by implementing non-negative Lasso, Adaptive Lasso, and Elastic Net as meta-learners. Their approach effectively performs view selection at the meta-layer, identifying the most informative data sources through sparsity-inducing penalties. Another example of constructing a Multi-View Stacking by phases includes the work by Barajas-Montiel et al. [10] on selection of multi-view learning techniques and views on electroencephalogram signals. They take it a step further by incorporating the selection of multi-view learning techniques, cotraining or co-regularization, into their five-step methodology to construct the best-performing ensemble for pattern recognition on this specific signals.

A common factor between these two design methods for Multi-View Stacking is the default setting of the view-level learners to the same machine learning model regardless of the differing statistical properties of the modalities or the choice of meta-learner. In this study, we explore the potential performance increase that a multi-view stacking architecture can have if we select each view-level learner as the best evaluated available machine learning algorithm.

While methodologies for selecting models within an ensemble architecture have successfully improved performance metrics in single-view machine learning [11], MVS addresses a more complex challenge. MVS aims not only for performance gains but also for the strategic separation and aggregation of heterogeneous information sources into a unified outcome.

2.2. Multi-View Learning

Multi-view learning finds its core in the concept of a *view* in a dataset. Multi-View data come from different features or representations that can be extracted from a certain phenomenon. Often, real-world applications have more than one information source that provides insights into the observed event. For instance, a video contains image and sound, both of which independently give information on the video's content; we then say that each one of these is a different view representing the video. However, the views are complementary to one another; therefore using both jointly yields more information

available to describe the phenomenon. Finding the right method to learn from each view, relate them to one another and perform a task based on the derived knowledge is what multi-view learning aims at. Specifically, multi-view learning consists of modeling each view and then optimizing the joint method to improve generalization performance. Opposed to a single-view method, where data are not differentiated by source, a multi-view approach has the opportunity to leverage the statistical properties specific to each view to be learned from [12]. This way, the multi-view paradigm enables to learn from data in a more comprehensive manner.

One of the earlier machine learning algorithms taking advantage on the multi-view paradigm consisted of the cotraining algorithm [13], which was used for classifying web-pages, where two separate information sources were defined as views: (1) the page's corpus (the text content present on the webpage itself) and (2) the text on hyperlinks (the anchor text contained in links pointing to that webpage). This feature partition was used to train a separate classifier for each view. A key mechanism of the cotraining algorithm was that the examples with confident labels predicted by one classifier were subsequently added to the training set of the other classifier, thus effectively augmenting a largely unlabeled dataset.

In a multi-view learning problem, the connection of views requires a certain balance between agreement and contributed information to describe a phenomenon. This balance is referred to as the principle of multi-view learning [4]. (1) Consensus principle: A set of views should exhibit some degree of correlation for the same event. This implies that each view captures essential data on the observation, and consequently, the set of views describe the same event in a coordinate manner. If consensus among views exists, common characteristics and patterns can be derived when using a greater scope of views. For instance, in a video setting, it is expected for the image and sound to bear a relation. (2) Complementary principle: Each view offers insights unattainable by other related views. The basic idea of multi-view learning is that it seeks to combine the information provided by different sources knowing that each one reveals aspects or details of the given source's capabilities. A collection of complementary views widens the scope of information available for any task by providing a richer and comprehensive data representation [4]. Following the example on a video input, visual data uncovers details that audio will not, and vice versa.

2.3. Multi-View Stacking

The Multi-View Stacking (MVS) approach builds upon the concept of stacked generalization originally introduced by Wolpert [14]. It belongs to the family of ensemble learning techniques, where the predictions of multiple base models (learners) are combined to improve predictive performance. In this framework, several models referred to as view-level learners are first trained, and their outputs are subsequently used as features to train a higher-level model known as the meta-learner. The procedure can be summarized as follows:

1. Define a collection $\mathcal{L}$ of view-level learners and a meta-learner.
2. Fit each view-level learner in $\mathcal{L}$ using the original training set $\mathbf{D}$, which consists of n labeled instances.
3. Each trained learner produces predictions over the dataset, generating a prediction vector $\mathbf{p}^i$ of length n .
4. These prediction vectors are concatenated column-wise to construct a matrix $\mathbf{M}_{n \times |\mathcal{L}|}$.
5. A transformed dataset $\mathbf{D}'$ is then created by combining $\mathbf{M}$ with the corresponding ground-truth labels $\mathbf{y}$.
6. The meta-learner is trained with $\mathbf{D}'$ as input.
7. Return the final trained stacked model $Model = (\mathcal{L}, meta\text{-}learner)$.

Figure 1 shows how $\mathbf{D}'$ is generated.

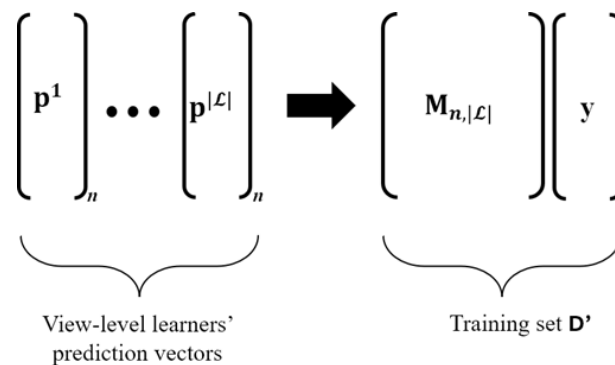


Figure 1. $\mathbf{D}'$ is generated by column-binding the outputs of the view-level learners and true labels $\mathbf{y}$.

To enhance representation, prediction outputs $\mathbf{p}^i$ can be encoded using one-hot encoding. In this case, each prediction vector becomes a matrix of size $n \times k - 1$, where k is the number of classes and one column is omitted to prevent redundancy. Additionally, incorporating class probability estimates (confidence scores) has been shown to further improve classification performance [15]. These scores are aggregated into a matrix $\mathbf{S}_{n,k}$ and appended to the prediction-based features, resulting in an extended feature matrix. Thus the final matrix is $\mathbf{M}_{n \times |\mathcal{L}|(k-1)+k}$.

A key challenge in this process is the risk of overfitting, since predictions used to train the meta-learner are derived from the same data used to train the base learners. To mitigate this issue, k-fold cross-validation is employed during the generation of predictions. After constructing $\mathbf{D}'$, the base learners are retrained using the full dataset to maximize data utilization.

In the context of Multi-View Stacking, each base learner is trained exclusively on a single view rather than on the complete feature set. This specialization allows each model to capture patterns specific to its corresponding data representation. Consequently, different views may be assigned different types of learners—for example, a Random Forest for one view and a Support Vector Machine for another. Similarly, the meta-learner can be chosen independently, such as a neural network or any other suitable model.

3. Materials and Methods

3.1. 3MuViS

The 3MuViS (3-Phase Multi-View Stacking) methodology provides a structured algorithmic approach for optimizing the selection of view-level learners and the meta-learner within a Multi-View Stacking framework. As its name implies, the method consists of three phases. In each phase, a predefined list of learners L is trained and tested at a specific level of the stacking. Choosing learners according to an optimization metric is how the method finds the selection of learners for each view in the list of views V . The three phases are the following:

1. **Selection of view-level learners:** This phase consists of the individual evaluation of each $v \in V$ against every learner in L to identify the optimal view-level learner set L_V^* .
2. **Selection of meta-learner:** Iterative selection of the best-performing aggregator L_{Meta}^* from L , given the fixed view-level learners L_V^* found in the previous phase.
3. **Ensemble Reconstruction:** Final integration and training of the optimized architecture, denoted $MultiViewStacking(L_V^*, L_{Meta}^*)$.

The main goal of this method is to provide a selection of learners without recurring to a Brute Force search or manually selecting the learners. The efficiency of 3MuViS lies

in its computational complexity. While a Brute Force search for the optimal configuration requires training $|L|^{|V|+1}$ models, 3MuViS achieves high performance while reducing the search space to a linear progression. Specifically, the framework requires training only $|L|(|V| + 1)$ models, yielding an algorithmic complexity of $O(|L| \cdot |V|)$. This ensures scalability for complex multi-view problems where the model library or view count is substantial. The following sections provide details of each phase.

Phase 1: Selection of view-level learners. According to the Multi-View Stacking method, each view will predict the label based on a machine learning algorithm fitted on the individual views. Typically, the selection of the view-level learners is not based on a measurable criterion but rather on the specification of the problem or intuition. 3MuViS delivers a systematic and metric-driven approach to select the optimum learners. According to the implementation's performance metric, each view will be trained with each of the learners in L . The learner assigned to a view will be the one with the best evaluation according to the performance metric using k-fold cross-validation. This method essentially chooses the classifier for the view which can predict the best on its own. This is done for every view results in a set of optimal learners L_V^* , whose predictions will be used as input for the meta-learner. To avoid overfitting, the original training set T is partitioned into two subsets: T_v and T_{Meta} . The former subset is used during this phase, and the latter is saved for Phase 2. The size of each subset is a hyperparameter.

Phase 2: Selection of meta-learner. The prediction probability vectors on each view for each instance compound the data on which the meta-learner will learn and predict from, so once the optimum view-level learners are selected, we can proceed to train each view, predict and gather the prediction probability vectors to train the meta-learner. Once again, the method iterates through the predefined list of learners L , training and evaluating each algorithm as the meta-learner, selecting the optimum classifier L_{Meta}^* according to the selected performance metric with k-fold cross-validation and using the subset T_{Meta} .

Phase 3: Ensemble reconstruction. Once the optimum view-level learners and meta-learner are selected in the previous phases, the entire MVS ensemble is trained on the complete training data.

In summary, the 3MuViS algorithm starts from a training dataset T with n labeled observations and a known separation in $|V|$ views and takes as input a list L of learners to choose from. The method then trains iteratively each model in L for each view and chooses the best-performing model for the given view, according to the classification metric selected (precision, recall, etc.). After finding the optimum view-level learners L_V^* , this configuration is used to find the best-performing meta-learner by fitting a Multi-View Stacking model with every available model in L as the meta-learner and fixed L_V^* view-learners. The algorithm described requires to train $|L| \cdot (|V| + 1)$ models. Assuming a fixed number of models in $|L|$, the algorithmic complexity of 3MuViS is then $O(|L| \cdot |V| + |L|)$. For reference, finding the best stacking with brute force requires to train $|L|^{|V|+1}$ models.

Figure 2 shows a flowchart with the overall phases of the 3MuViS algorithm.

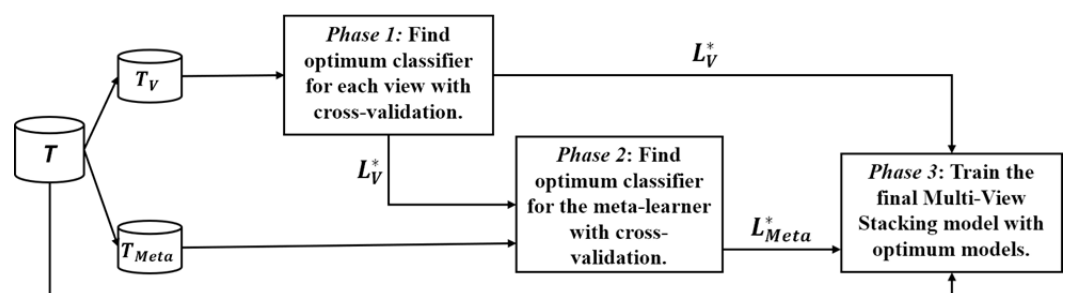


Figure 2. Overall phases of 3MuViS.

Algorithm 1 shows the pseudo code for 3MuViS learner selection and provides a step-by-step explanation of the algorithm.

The algorithm takes as input a training set T ; the classification metric to optimize for, clf_metric ; the list of views V , which groups columns in T based on their view; a catalog list of models, L ; and $percent_views$, which refers to the fraction of data T_v from T used to find the best view-level learners. Accordingly, $1 - percent_views$ is the fraction of T used to find the best meta-learner (T_{Meta}). k_folds refers to the number of folds used throughout Phase 1 and Phase 2 evaluation. Lines 5–18 find the best view-level learners. Lines 19–29 find the best meta-learner. The algorithm outputs the optimized view-level learners and meta-learner.

Algorithm 1 3MuViS learner selection algorithm.

```

1: Input:  $T, clf\_metric, V, L, percent\_views, k\_folds$ 
2: Output:  $L_V^*, L_{Meta}^*$ 

3: # Split  $T$  randomly into two subsets.
4:  $[T_V, T_{Meta}] \leftarrow split(T, percent\_views)$ 

5:  $L_V^* \leftarrow$  empty array
6: for  $i \leftarrow 1$  to  $|V|$  do
7:    $best\_score \leftarrow 0$ 
8:    $best\_learner \leftarrow null$ 
9:   for  $j \leftarrow 1$  to  $|L|$  do
10:     $model \leftarrow copy(L_j)$ 
11:     $score \leftarrow fit\_evaluate\_cv(model, T_{V_i}, clf\_metric, k\_folds)$ 
12:    if  $score > best\_score$  then
13:       $best\_score \leftarrow score$ 
14:       $best\_learner \leftarrow model$ 
15:    end if
16:  end for
17:   $L_{V_i}^* \leftarrow best\_learner$ 
18: end for

19:  $best\_score\_meta\_learner \leftarrow 0$ 
20:  $L_{Meta}^* \leftarrow null$ 
21: for  $i \leftarrow 1$  to  $|L|$  do
22:   $meta\_model \leftarrow copy(L_i)$ 
23:   $MVS \leftarrow MultiViewStacking(L_V^*, meta\_model)$ 
24:   $score \leftarrow fit\_evaluate\_cv(MVS, T_{Meta}, clf\_metric, k\_folds)$ 
25:  if  $score > best\_score\_meta\_learner$  then
26:     $best\_score\_meta\_learner \leftarrow score$ 
27:     $L_{Meta}^* \leftarrow model$ 
28:  end if
29: end for
30: return  $(L_V^*, L_{Meta}^*)$ 

```

3.2. Datasets

In this section we describe the multi-view datasets used to test the proposed methodology. For datasets containing more than two views, the dataset was shortened to the two views with the best class separability as seen in the multidimensional scaling components, except for Berkeley's Multimodal Human Action Database, for which three views were kept. This reduction was done in order to be able to find the best stacking model by brute force within a reasonable time frame for each dataset.

3.2.1. D1: Berkeley Multimodal Human Action Database (MHAD)

The MHAD dataset [16,17] compiles 11 different human activities with different levels of movement of the lower and upper body. The multiple views correspond to the devices used for simultaneous recording, which include: optical motion capture systems, multi-baseline stereo cameras from multiple views, depth sensors, accelerometer and microphones. For the data acquisition, 7 female and 5 male participants performed different activities with no particular instructions on how to execute them. A two Microsoft Kinects set was mounted in order to capture depth data. Audio information was acquired with four different microphones arranged to capture the space, where three of these were placed on tripods 65 cm above the ground and the fourth one was placed on the floor. Six triaxial accelerometers were attached to the subjects' motion capture suits, and these were distributed on the wrists, ankles and hips [16]. For testing 3MuViS on this dataset, the features corresponding to optical motion capture, microphones, and wrist accelerometer were selected. This view division follows the natural sensor streams of the original data collection. While the spatial alignment of the motion capture and microphone arrangement allows for cohesive views representations, the multiple accelerometers distributed across the human body capture localized recordings dependent on the performed action. To ensure a standard evaluation and avoid the arbitrary fusion of independent sensors, we deliberately isolated the wrist accelerometer as a single, independent view. In this manner we got three views, corresponding to depth, audio, and accelerometer data. The raw sensor signals were preprocessed and summarized according to modality. Feature extraction was done for each time frame, corresponding to one repetition of the observed movement. From the microphone recordings, the first 12 Mel-Frequency Cepstral Coefficients (MFCCs) were computed. Accelerometer signals were characterized through descriptive statistics, with the mean, minimum, and maximum computed for each axis. For the Kinect sensor, distance measurements were likewise summarized by calculating their minimum, maximum, and mean values. The described preprocessing yielded a dataset of 155 columns by 659 rows, with the class distribution presented in Table 1. The multidimensional scaling (MDS) plot for the three components is presented in Figure 3 as a visual reference of class separability for each view. Here, we can see that view 2 is better at discriminating class *a02* (orange-colored data points) compared with the other two views. This shows that while some views may have trouble classifying specific classes, others can compensate for those errors and vice-versa.

Table 1. Class distribution for D1.

Classes	Number of Instances
Jumping in place	60
Jumping jacks	60
Bending	60
Punching	60
Waving (two hands)	60
Waving (one hand)	60
Clapping hands	59
Throwing a ball	60
Sitting down	60
Standing up	60
T-Posing	60

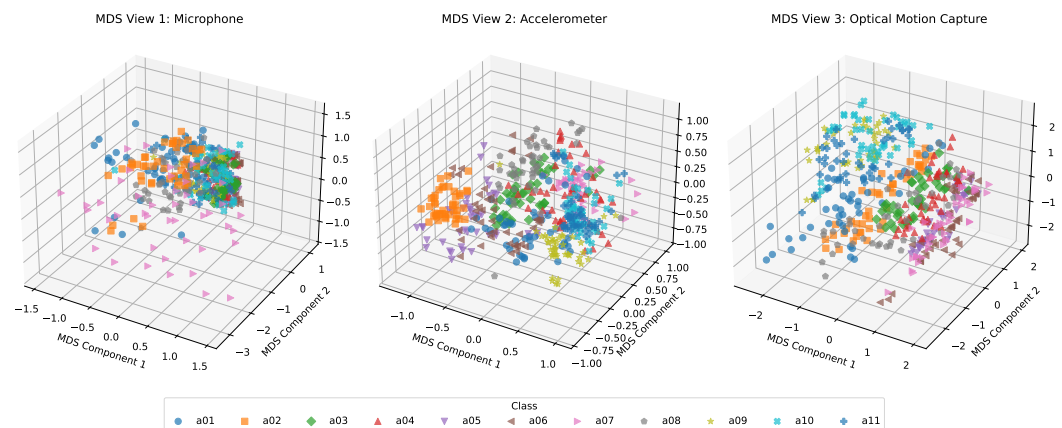


Figure 3. MDS plots of D1.

3.2.2. D2: Multiple Features Digits

The Multiple Features datasets is derived from a collection of handwritten digits (0–9) obtained from Dutch utility maps. Each sample corresponds to a digital image of a single digit, and 200 instances were collected for each class [18]. This yielded a total of 2000 labeled observations with 650 extracted features each, with the class distribution portrayed in Table 2. Six different processing methods were applied to the images to create representation sets. The extracted features included: 76 Fourier coefficients of the character shapes, 216 profile correlations, 64 Karhunen–Love coefficients, 240 pixel averages in 2×3 windows, 47 Zernike moments, and 6 morphological features. Each processing applied corresponds to an independent view.

In Figure 4 the MDS for the two views with most visible separability is shown. These views correspond to the Karhunen–Love coefficients and 2×3 window pixel averaging processing. From the point cloud, vague but distinguishable cluster formations corresponding to the target classes can be observed. This indicates that both representations retain some class-discriminative structure even after dimensionality reduction.

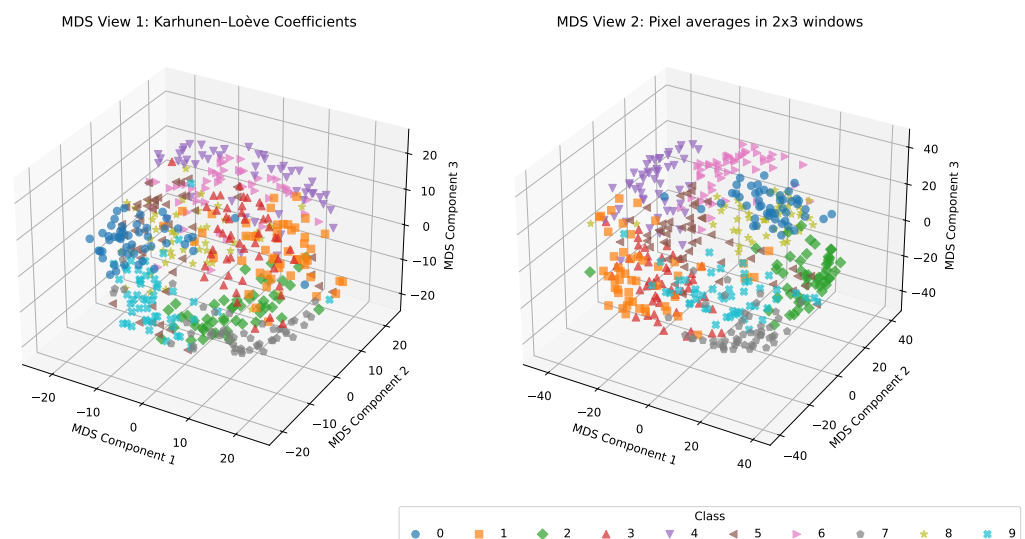


Figure 4. MDS plots of D2.

Table 2. Class distribution for D2.

Classes	Number of Instances
0	200
1	200
2	200
3	200
4	200
5	200
6	200
7	200
8	200
9	200

3.2.3. D3: Transportation Mode Detection

Following the human activity recognition field, the dataset of Transportation Mode Detection [19,20] gathers data from multiple unconstrained smartphone sensors of 3 female and 5 male participants with Android versions ranging from 5.0 to 7.0, who recorded the observations for data collection, with class the distribution reported in Table 3. This dataset has been used to investigate the potential of smartphone sensor features to detect means of transportation as an alternative to relying solely on GPS information. Each of the sensors readings can be understood as a view in the multi-view paradigm: accelerometer, game rotation vector, gyroscope, uncalibrated gyroscope, linear acceleration, orientation, rotation vector, sound and speed. The signal data were aggregated into the average, minimum, max and standard deviation for training the machine learning model, which creates a dataset of 5893 observations with 38 features. For the experiments, only data coming from accelerometer and gyroscope were considered in this study in order to keep two views. Figure 5 shows the corresponding MDS plot of the two views.

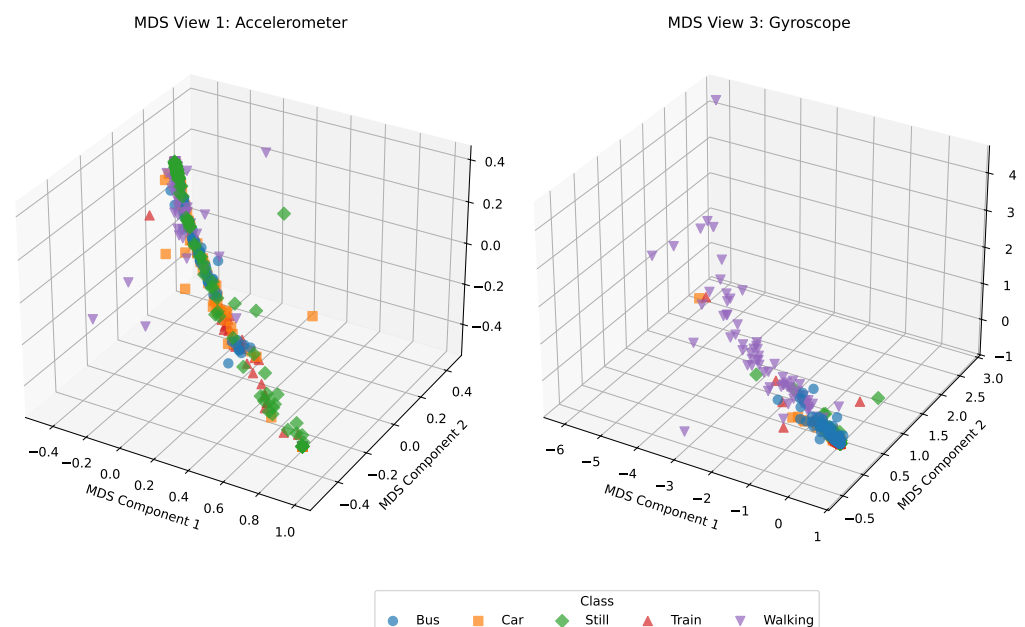
**Figure 5.** MDS plots of D3.

Table 3. Class distribution for D3.

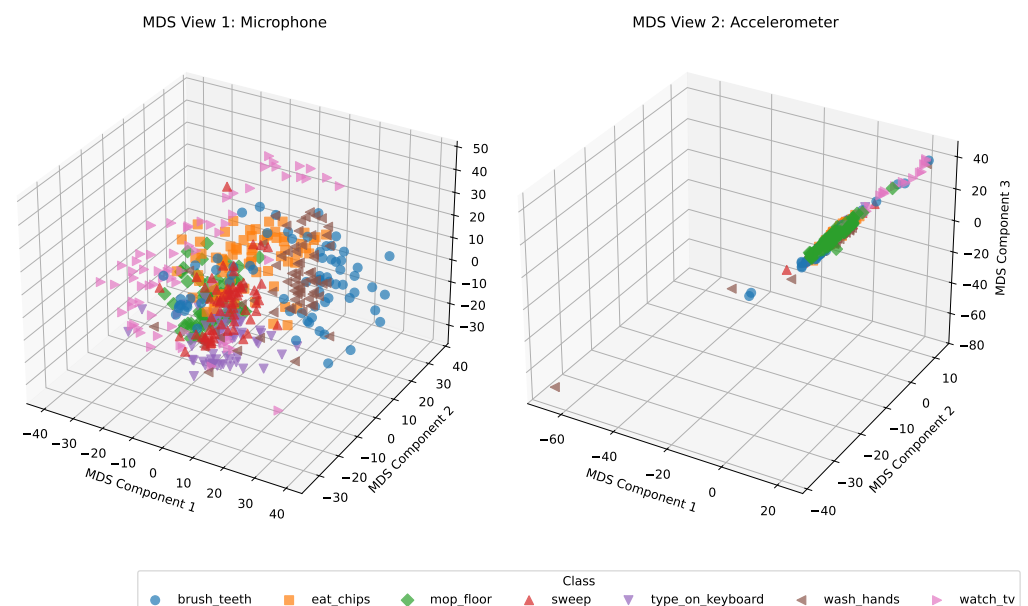
Classes	Number of Instances
Car	1180
Still	1179
Train	1179
Bus	1178
Walking	1177

3.2.4. D4: Home Tasks Activities Dataset (HTAD)

This dataset [21,22] presents a human activity recognition problem focusing on certain home tasks necessary for self-sufficiency, such as brushing teeth, washing hands, sweeping, and so on. Accelerometer and microphone data were recorded by 2 males and 1 female participant, who performed the activities for 3 min each. Raw data were windowed in 3 s segments, from which the first 36 MFCCs were extracted for microphone data, and mean, standard deviation, min, and max were computed for each axis coming from the accelerometer. Other extracted features for the accelerometer data included: correlation between each pair of axes and vector magnitude's mean and standard deviation. The described process yielded a total of 1386 labeled observations with 54 features each. Table 4 shows the class distribution, and Figure 6 shows the corresponding MDS plots.

Table 4. Class distribution for D4.

Classes	Number of Instances
Sweep	178
Type on keyboard	179
Brush teeth	180
Wash hands	180
Mop floor	181
Watch TV	206
Eat chips	282

**Figure 6.** MDS plots of D4.

3.2.5. D5: HAR70+

The HAR70+ database was collected using two tri-axial accelerometers by 18 older adult participants (70–95 years old) [23,24]. One sensor was attached to the right front thigh, and the other one was attached on the lower back of the participant. The dataset consists of seven activities: Walking, Shuffling, Ascending stairs, Descending stairs, Standing, Sitting, and Lying. Since data for activities Ascending/Descending stairs were scarce, those classes were omitted. For each of the sensors, 19 features were computed from 3 s windows. The extracted features included: mean, standard deviation, minimum, and maximum for each axis. The correlations between pairs of axes, mean magnitude, standard deviation of the magnitude, area under the curve, and the mean difference between consecutive readings were computed. A moving average filter of size 10 was applied before feature extraction in order to reduce noise. The resulting database has 11,263 rows and 40 features. Table 5 shows the class distribution. In this case, the dataset consists of two views, one for each sensor.

Table 5. Class distribution for D5.

Classes	Number of Instances
Lying	539
Shuffling	430
Sitting	576
Standing	2532
Walking	7186

Figure 7 shows each of the views' features after reducing their dimension to 3 components using MDS [25]. The plots only show 500 data points selected at random in order to reduce computational costs. From the plot of view 1, it can be seen that it is good at separating the class 'lying' from the rest; however, in view 2 it highly overlaps with 'sitting'.

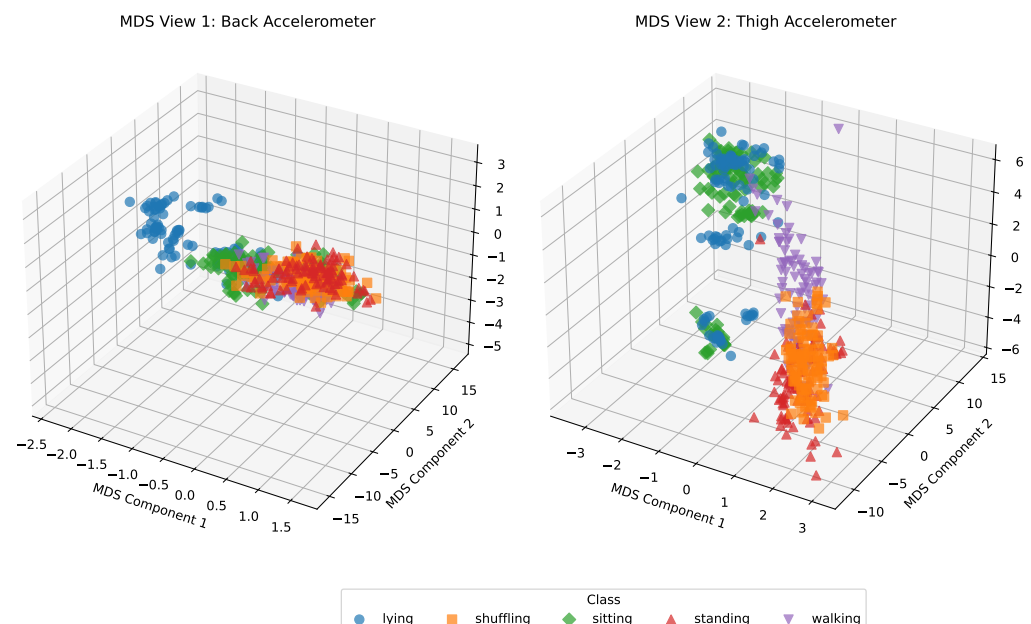


Figure 7. MDS plots of D5.

3.2.6. D6: UNSW-NB15

The UNSW-NB15 dataset [26,27] is a network intrusion detection dataset created by the Australian Centre for Cyber Security (ACCS) at the University of New South Wales

(UNSW). It was developed to provide realistic and diverse network traffic for evaluating intrusion detection systems (IDSs). The data were generated by simulating both normal and malicious behaviors in a realistic network setup. In this work, we considered dividing the features into two views. The first one is what the authors of the original dataset call *basic features*. The second view was built by incorporating what the authors call *content features*. The basic features include: *dur* (the record total duration), *sbytes* (source-to-destination bytes), *dbytes* (destination-to-source bytes), *sttl* (source-to-destination time to live), *dttl* (destination-to-source time to live), *sload* (source bits per second), *dload* (destination bits per second), *sloss* (source packets dropped), *dloss* (destination packets dropped), *spkts* (source packet count), and *dpkts* (destination packet count).

The content features include: *swin* (source TCP window advertisement), *dwin* (destination TCP window advertisement), *stcpb* (source TCP sequence number), *dtcpb* (destination TCP sequence number), *smean* (mean flow packet size transmitted by the source), *dmean* (mean flow packet size transmitted by the destination), *trans_depth* (depth into the connection of http request/response transaction), and *response_body_len* (content size of the data transferred from the server's http service). Given the big size of the dataset, we randomly sampled 10,000 observations such that the experiments could run in a feasible amount of time, particularly with the Brute Force approach. As opposed to the previous datasets, this one only consists of two classes: normal (encoded as 0) and malicious (encoded as 1). Table 6 shows the class distribution, and Figure 8 depicts the MDS plots of the two views.

Table 6. Class distribution for D6.

Classes	Number of Instances
0	37,000
1	45,332

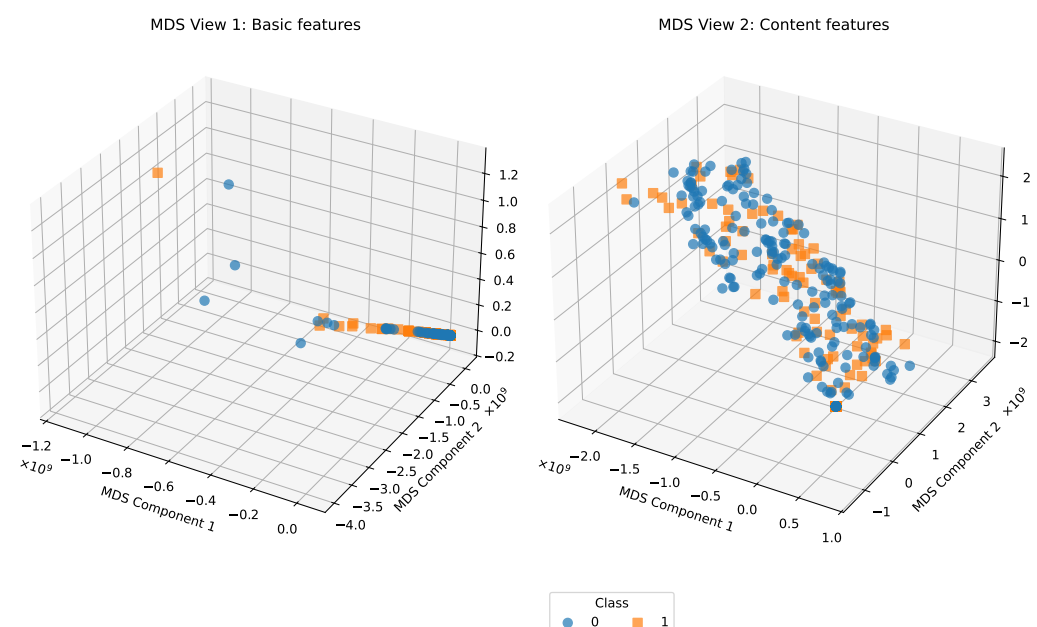


Figure 8. MDS plots of D6.

3.3. Experimental Design

To evaluate the proposed methodology, experiments were conducted to compare the algorithm selection of **3MuViS** against several baselines: (i) **Random Choice**, where learners were chosen randomly for the entire stacking architecture; (ii) **Random Forest MVS**, which corresponds to the Multi-View Stacking algorithm defaulted to Random

Forest as both view-level and meta-learners; (iii) **Brute Force**, which is exhaustively trained and tested on all possible combinations of view-level learners, views, and meta-learners given the available models; and (iv) **Randopt**, in which the view-level learners are chosen randomly and the meta-learner is optimized on top of this selection as proposed in **3MuViS** Phase 2. In the case of **Random Forest MVS**, Random Forest was chosen as the default classifier since it has proven to perform extremely well in diverse domains compared with other classifiers [28].

Brute Force exhaustive search finds the single best combination which is used as an upper-bound baseline. An additional method based on a Multi-View Stacking with random view-level learners and optimized meta-learner was included for the experimentation in order to determine whether the **3MuViS** improved generalization relies solely in the meta-learner optimization or if selecting better performing view-level learners actually contributes significant information in the stack. The discussed method is referred to as **Randopt**, where for a set of V views and L available models, only $|V| + |L|$ machine learning algorithms need to be fitted. On this basis, it is important to evaluate the consistency with which **3MuViS** chooses the algorithms and to conduct hypothesis testing in order to statistically support the results.

In order to test the proposed algorithm, the following parameters were set throughout every experiment for each dataset. To account for variance, the experiments were run for 25 iterations for each dataset. For each iteration, **3MuViS**, **Random Forest MVS**, **Random Choice** and **Randopt** were trained and tested on the same, randomized and stratified training–test split. The training set consisted of 80% of the whole dataset, and 20% was saved for final testing. For **3MuViS**, from the 80% of the training data, half of them (T_V) were used for finding the best view-level learners (Phase 1 of the algorithm). The remaining half (T_{Meta}) were used to find the best-performing meta-learner (Phase 2). Then, the whole training set (80%) was used to fit the final model (Phase 3).

Brute force was implemented to find the best stacking of view-level learners and meta-learners on each dataset according to the available views. Such upper limit baseline was calculated by training and testing all possible permutations according to the required learners (views and meta-learner) and machine learning algorithms available for selection. As the selection of algorithms allows for reposition, the number of permutations to train is $|L|^{|V|+1}$. Therefore, assuming a constant model catalog, the algorithmic complexity of getting the best stacking by brute force is exponential. The **Brute Force** baseline was calculated on a single run, as it was not feasible to include brute force in every iteration due to its computational cost. Multi-View Stacking requires to select a number of folds to train the meta-learner, which was set to five folds for every experiment.

The Matthews Correlation Coefficient (MCC) was selected as the performance metric due to its robustness in evaluating classification models under class imbalance, which is a common characteristic of several datasets, including in cybersecurity, particularly in network anomaly detection tasks. Unlike conventional metrics such as accuracy, MCC makes use of true positives, true negatives, false positives, and false negatives, which provides a more complete view of model performance. This makes it particularly suitable for comparing models in a consistent manner across different datasets.

3MuViS makes use of the classification probability vector of each prediction as input data for the meta-learner; therefore the model catalog, listed in Figure 9, consists only of scikit-learn's machine learning algorithms that implement the method `predict_proba()`, which returns the required vector. Table 7 details the experiments' configuration including hardware, software, and experiment/model hyper-parameters.

Model catalog
Linear Discriminant Analysis
Histogram-based Gradient Boosting Classifier
K-Neighbors Classifier
Nearest Centroid Classifier
Support Vector Classifier
Decision Tree Classifier
Extra Trees Classifier
Random Forest Classifier
SGD Classifier
Gaussian Naive Bayes
Multinomial Naive Bayes
Bernoulli Naive Bayes

Figure 9. Machine learning algorithms available for selection during experiments.

Table 7. Experimental environment and hyperparameter configuration for 3MuViS analysis.

Component	Specification/Value
Hardware and System	
CPU	Intel Core i7-6820HQ @ 2.70 GHz
RAM	32 GB
OS	Windows 10 Pro
Software Environment	
Python	3.12.7
Scikit-Learn	1.6.1
Pandas	2.2.2
NumPy	1.26.4
SciPy	1.11.4
Matplotlib	3.8.2
multiviewstacking	1.0.0
Model Configuration	
Candidate algorithms	12 (sklearn defaults, no tuning; see Figure 9)
k-Folds Multi-View Stacking	5
k-Folds 3MuViS	5
3MuViS performance metric	Matthews Correlation Coefficient
SVC (Support Vector Classifier)	probability=True (sklearn default otherwise)
SGDClassifier	loss=log_loss (sklearn default otherwise)
Random seed models	13
Experimental Parameters	
Iterations	25
Random seed training–test split	13 (+1 per iteration)
Data split	80% Train/20% Test (stratified)
3MuViS view-level/meta-learner split	50%/50% of training set
Data preprocessing	MinMaxScaler (fit on train, applied to test)
Statistical Analysis of Results	
Test	Wilcoxon Signed-Rank (one-sided)
Hypothesis	H_1 : 3MuViS > baseline
Significance level	$\alpha = 0.05$
Paired observations	25 (one per iteration)

4. Results

4.1. Performance Analysis

4.1.1. Classification Performance

Table 8 shows a summary of the performance results in terms of the MCC for each of the evaluated datasets. Our results indicate that **3MuViS** consistently outperforms the baseline **Random Forest MVS** across all datasets except one (D1). Furthermore, the fact

that **3MuViS** exceeds the performance of **Random Choice** and **Randopt** in every dataset confirms that the observed improvements are a direct result of the proposed methodology rather than a stochastic variation. A clearer advantage of **3MuViS** is observed in D4 (HTAD), D5 (HAR70+), and D6 (Attacks), where it achieves the highest mean MCC scores (0.93, 0.95, and 0.85, respectively), outperforming all alternative methods while also maintaining low standard deviation. These results indicate not only strong predictive performance but also consistent behavior across runs.

Baseline approaches such as **Random Choice** and **Randopt** generally present lower mean MCC values and significantly higher variability, particularly in more challenging datasets (e.g., D3 and D6), suggesting reduced reliability. Although the **Brute Force** baseline provides an approximate upper bound, **3MuViS** frequently approaches or matches this limit, confirming its effectiveness.

Table 8. MCC score distribution measurements by dataset and by method, where the highest mean score is in bold—not considering the Brute Force upper-bound baseline.

Dataset	Experiment	Mean	Std	IQR	Min	Max
D1: MHAD	3MuViS	0.984	0.013	0.016	0.943	1.0
	Random Choice	0.922	0.066	0.089	0.746	0.99
	Random Forest MVS	0.990	0.009	0.016	0.961	1.0
	Randopt	0.952	0.042	0.049	0.808	1.0
	Brute Force (Baseline)	0.976	–	–	–	–
D2: Multidigits	3MuViS	0.973	0.007	0.011	0.963	0.976
	Random Choice	0.941	0.028	0.047	0.893	0.986
	Random Forest MVS	0.970	0.009	0.008	0.947	0.978
	Randopt	0.945	0.034	0.03	0.835	0.981
	Brute Force (Baseline)	0.978	–	–	–	–
D3: Transport	3MuViS	0.622	0.017	0.019	0.582	0.657
	Random Choice	0.493	0.075	0.061	0.282	0.591
	Random Forest MVS	0.616	0.014	0.015	0.589	0.640
	Randopt	0.497	0.114	0.112	0.231	0.635
	Brute Force (Baseline)	0.636	–	–	–	–
D4: HTAD	3MuViS	0.933	0.015	0.021	0.912	0.958
	Random Choice	0.776	0.106	0.13	0.491	0.880
	Random Forest MVS	0.921	0.015	0.02	0.879	0.953
	Randopt	0.827	0.087	0.104	0.609	0.936
	Brute Force (Baseline)	0.949	–	–	–	–
D5: HAR70+	3MuViS	0.947	0.005	0.006	0.931	0.962
	Random Choice	0.893	0.05	0.047	0.691	0.938
	Random Forest MVS	0.940	0.055	0.008	0.932	0.948
	Randopt	0.907	0.052	0.042	0.712	0.948
	Brute Force (Baseline)	0.943	–	–	–	–
D6: Attacks	3MuViS	0.850	0.009	0.014	0.832	0.859
	Random Choice	0.645	0.173	0.262	0.312	0.841
	Random Forest MVS	0.823	0.01	0.015	0.810	0.852
	Randopt	0.662	0.13	0.246	0.462	0.828
	Brute Force (Baseline)	0.854	–	–	–	–

4.1.2. Statistical Tests

The performance improvement of **3MuViS** is evaluated with a Wilcoxon test over the distribution of MCC scores. This test has the following hypothesis definition:

$$H_0 : \mu_a \leq \mu_b$$

$$H_a : \mu_a > \mu_b$$

where μ_a corresponds to the mean of the MCCs for **3MuViS** and μ_b is the respective mean for either **Random Forest MVS**, **Random Choice** or **Randopt**. Table 9 shows the resulting statistical significance of the tests. The proposed method was statistically significant superior than the other methods except against **Random Forest MVS** in D1. Regarding our initial RQ1, we can conclude that in the majority of cases, **3MuViS** outperforms MVS with fixed learners. For the case of RQ2, **3MuViS** compares favorably in all datasets against the stochastic selection strategies (**Random Choice** and **Randopt**). Regarding RQ3, we can see that **3MuViS** consistently outperformed or equaled the other methods in all datasets except D1. Based on these results we can also answer RQ4. In all datasets **3MuViS** performed better compared with **Randopt**. This demonstrates that optimizing both view-level and meta-learners offers a competitive advantage as opposed to only optimizing the meta-learner.

Table 9. p -Values for Wilcoxon test, for the alternative hypothesis that 3MuViS performance is greater than that of the other methods. * $p < 0.05$, and *** $p < 0.001$. The symbol + indicates no statistical significance.

Dataset	Random Forest MVS	Random Choice	Randopt
D1: MHAD	+	***	***
D2: Multidigits	***	***	***
D3: HAR70+	***	***	***
D4: Transport	*	***	***
D5: HTAD	***	***	***
D6: Attacks	***	***	***

4.1.3. Distribution Analysis of Classification Performance

Figure 10 shows violin plots for each of the methods. Here, we can see that **Random Choice** and **Randopt** have high variance and a wide distribution. On the other hand, **3MuViS** presents a reduced dispersion in MCC scores, confirming that it provides superior stability, consistently approaching the performance ceiling established by the Brute Force reference. In this case, for D1, the **Random Forest MVS** approach resulted in the highest mean MCC and the lowest dispersion. The violin plots of the remaining datasets are included in Appendix A.

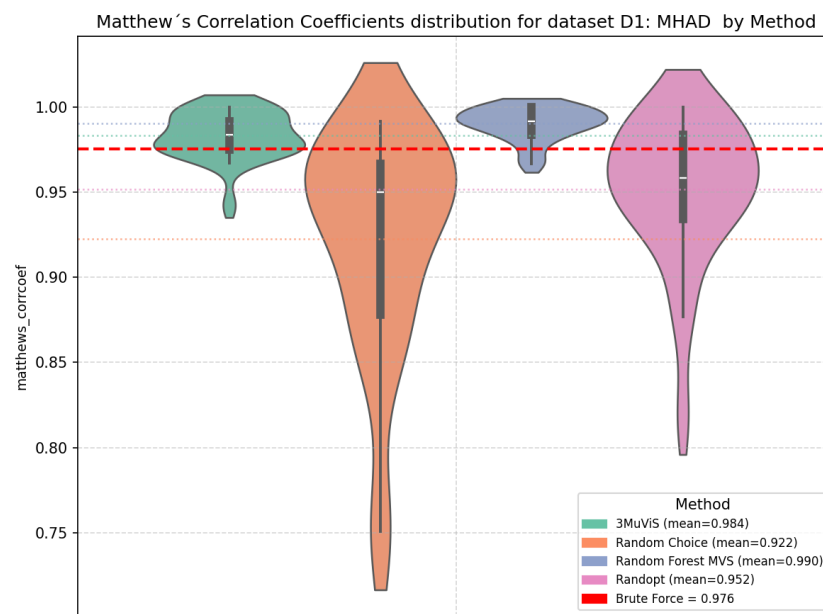


Figure 10. Violin plot of Matthews Correlation Coefficient by method: D1.

4.1.4. Learner Selection Results

To evaluate the stability of the algorithmic selections made by the **3MuViS** algorithm, an UpSet plot [29] (Figure 11) was utilized to visualize the selection frequency of each learner at both the view and meta-levels. The UpSet plot shows the distribution of elements across sets and the distribution of combinations of sets. For example, the most common set appeared six times and consisted of a RandomForestClassifier for the meta-learner, a LinearDiscriminantAnalysis classifier for view 3, and ExtraTreesClassifier for views 1 and 2.

Across the experiments conducted on D1, the selection process for view-level learners exhibited high consistency; specifically, each view demonstrated a clear preference for a particular algorithm across nearly all experimental iterations. For instance, the ExtraTreesClassifier was predominantly selected for both view 1 and view 2, while LinearDiscriminantAnalysis was favored for view 3 (chosen in 22 iterations). RandomForestClassifier was the most frequently selected as the meta-learner. While the Random Forest MVS model performed slightly better than 3MuViS on this specific dataset, the localized selections made by 3MuViS highlight the architectural value of model diversity across different data streams. Notably, an exhaustive, resource-intensive Brute Force search revealed that the absolute optimal ensemble combination for D1 actually consists of a Support Vector Classifier for views 1 and 2, a Histogram-based Gradient Boosting Classifier for view 3, and a Support Vector Classifier as the meta-learner (Table 10). This demonstrates that neither Random Forest MVS nor 3MuViS stacking selection yields the absolute best-performing combination. Consequently, the primary achievement of the 3MuViS methodology is not the estimation of the Brute Force peak combination but rather the efficient generation of a high-performing, view-specific Multi-View Stacking ensemble without the prohibitive computational overhead of an exhaustive search.

While greater variance was observed in the selection of the meta-learner, it is significant to note that the most frequently assigned algorithm for each individual learner is represented within the most recurring global combination set. This indicates that **3MuViS** successfully converges toward a stable and optimal architectural configuration, rather than producing stochastic or disjointed selections. UpSet plots for D2 through D6 are also available in Appendix A. To establish the absolute performance boundaries of Multi-View Stacking on the evaluated datasets, an exhaustive Brute Force search was conducted. Due to the severe computational cost of evaluating every possible permutation of base-learners and meta-learners across multiple views, this combinatorial search was executed only once per dataset to serve as a baseline ceiling. The resulting optimal ensemble configurations and their corresponding MCC scores are detailed in Table 10.

Table 10. Optimal ensemble configurations and MCCs identified via exhaustive Brute Force search.

Dataset	View 1 Learner	View 2 Learner	View 3 Learner	Meta-Learner	MCC
D1	SVC	SVC	HGB	SVC	0.976
D2	SVC	ExtraTrees	—	ExtraTrees	0.978
D3	ExtraTrees	ExtraTrees	—	SGD	0.636
D4	ExtraTrees	ExtraTrees	—	HGB	0.949
D5	ExtraTrees	ExtraTrees	—	HGB	0.943
D6	RandomForest	RandomForest	—	NearestCentroid	0.854

Note: — indicates the absence of a third view.

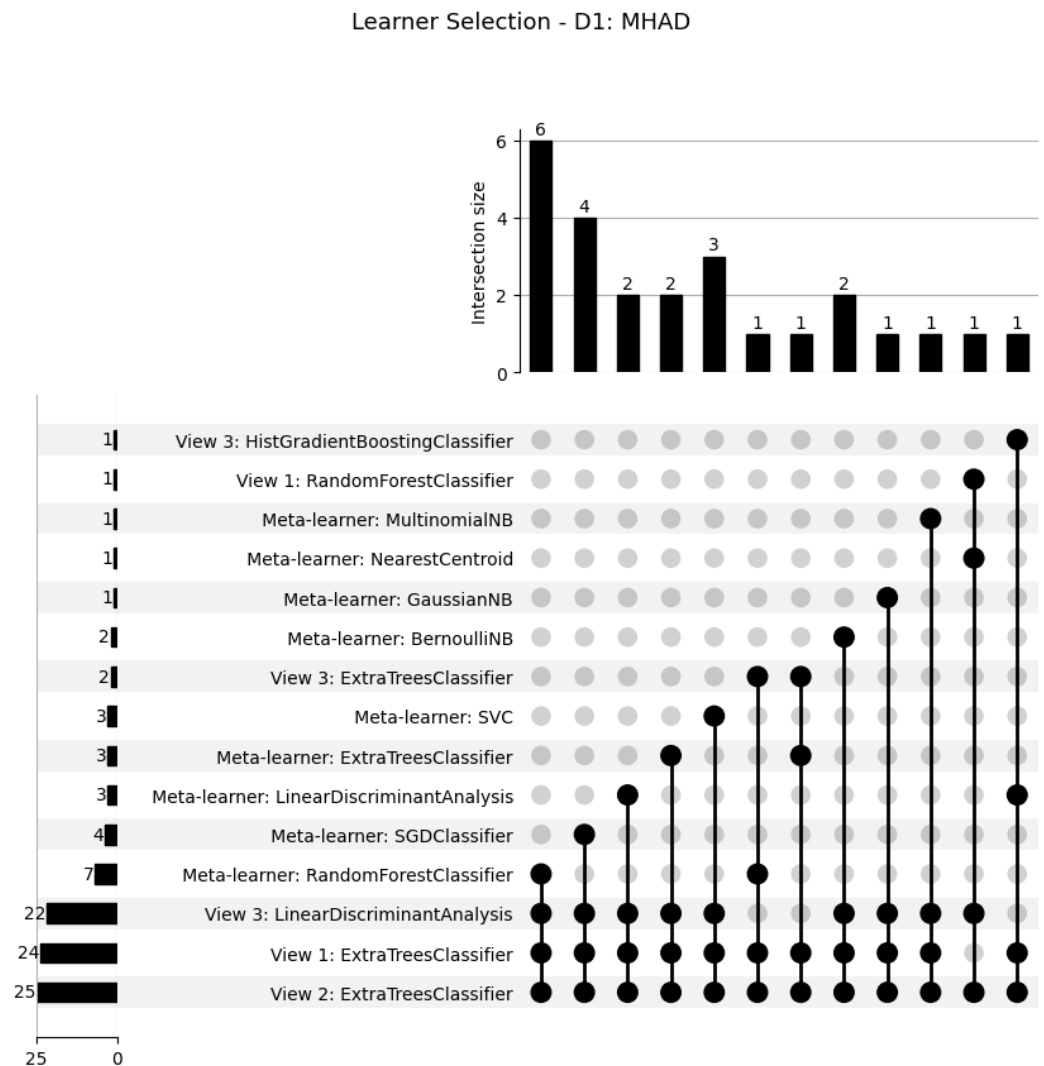


Figure 11. UpSet plot of learner selection for D1. Solid circles indicate the selected learners.

4.2. Computational Efficiency and Resource Allocation

Learner Selection Time

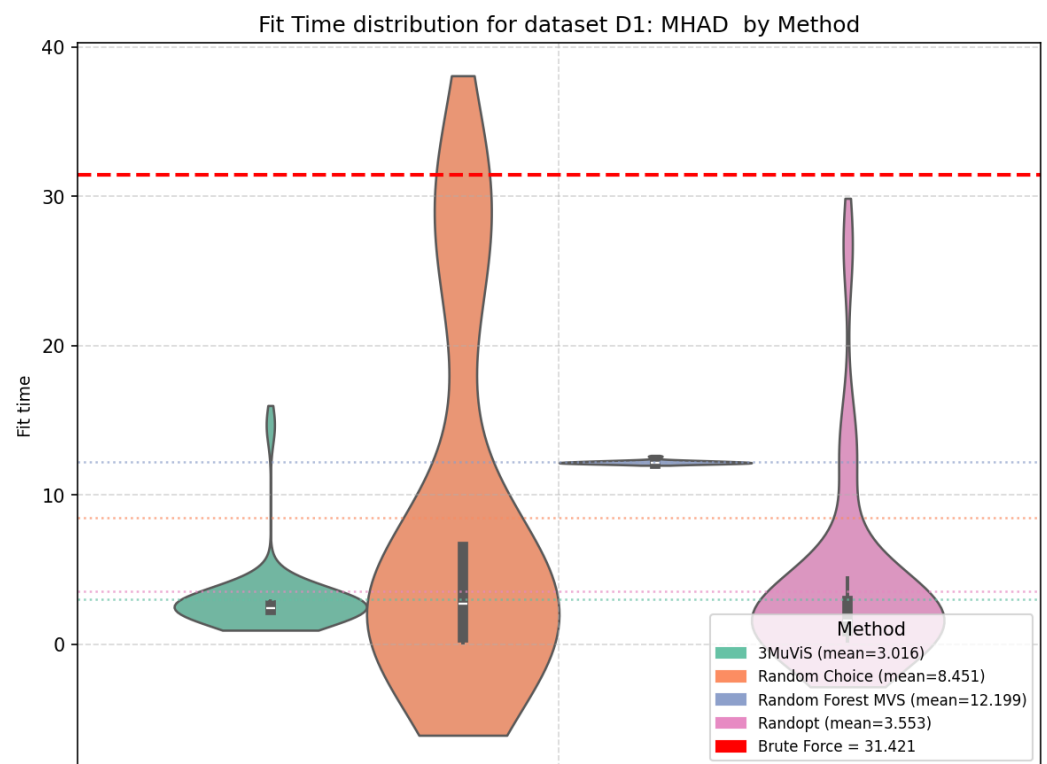
3MuViS represents an advantageous methodology to find a great-performing stacking in terms of computational resources and therefore time, specially when compared against brute force. In Table 11, descriptive statistics from the distribution of search times of both **3MuViS** and **Randopt** are displayed, and the **Brute Force** baseline is added for reference. Search time is understood as the amount of time that the algorithm took to find the optimal solution, that is, selecting the best learners from the catalog, for both view-level learners and meta-learner, based on the performance metric to optimize for.

Overall, **3MuViS** achieved a substantial reduction in computational cost compared with the exhaustive **Brute Force** strategy while still maintaining competitive classification performance. However, **3MuViS** generally required longer search times than **Randopt** due to the additional optimization of both view-level and meta-learners. **Brute force** exhibited extremely high computational demands, requiring between 101 and 1574 min, confirming the exponential cost of exhaustive learner selection. These results demonstrate that **3MuViS** provides an effective trade-off between computational efficiency and predictive performance, enabling near-optimal learner selection at a fraction of the computational cost of exhaustive search.

Table 11. Search time (minutes) for 3MuViS, Randopt and Brute Force.

Dataset	Experiment	Mean	Std	IQR	Min	Max
D1: MHAD	3MuViS	4.7	2.4	0.04	4.0	16.0
	Randopt	0.06	0.1	0.04	0.003	0.45
	Brute Force (Baseline)	1574.4	–	–	–	–
D2: Multidigits	3MuViS	5.1	0.3	0.13	4.02	5.55
	Randopt	0.11	0.21	0.07	0.001	0.98
	Brute Force (Baseline)	261.5	–	–	–	–
D3: Transport	3MuViS	11.2	2.81	2.12	6.64	22.71
	Randopt	7.99	9.59	8.60	0.06	27.21
	Brute Force (Baseline)	244.73	–	–	–	–
D4: HTAD	3MuViS	4.27	0.26	0.03	3.06	4.38
	Randopt	3.14	4.14	2.37	0.25	15.82
	Brute Force (Baseline)	101.1	–	–	–	–
D5: HAR70+	3MuViS	18.47	12.21	16.65	7.25	38.53
	Randopt	8.82	9.42	10.08	0.2	32.28
	Brute Force (Baseline)	306.77	–	–	–	–
D6: Attacks	3MuViS	8.3	0.42	0.36	7.96	9.69
	Randopt	0.27	0.41	0.17	0.001	1.23
	Brute Force (Baseline)	339.98	–	–	–	–

Figures 12 and 13 show the fit and predict times (respectively) for each of the methods after the optimum learners have been selected. Overall, the violin plots highlight that **3MuViS** provides a favorable balance between computational efficiency and optimization capability.

**Figure 12.** Violin plot fit times (seconds) for D1.

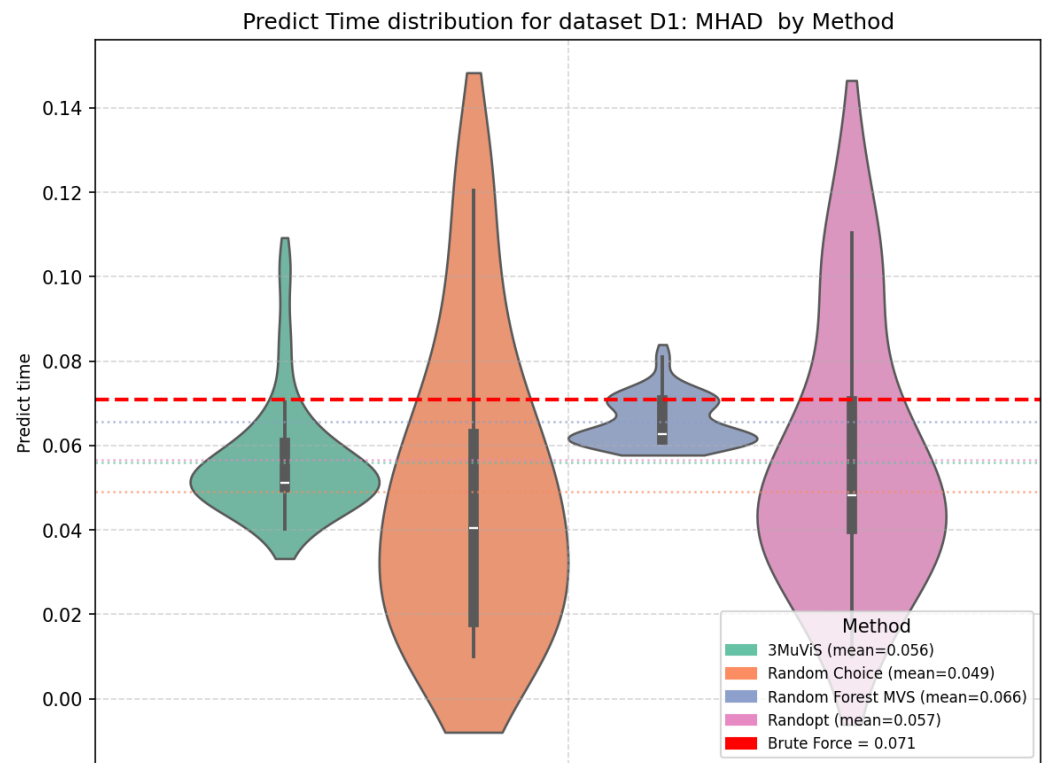


Figure 13. Violin plot predict times (seconds) for D1.

4.3. User-Independent Analysis

In some applications, such as in human activity recognition, the performance of a model is also evaluated in a subject-independent manner [30]. That is, the data from a given user are only included in either the training or test set. This enables to test the generalization capability of the model to new unseen users. In order to test the 3MuViS algorithm under this setting and to answer RQ5, we conducted an additional experiment with user-independent validation.

D1, D4, and D5 feature multi-user human activity recognition data enabling subject-independent evaluation. While we followed the same overall experimental pipeline as that described in Section 3.3, the data partitioning consisted of a random group split assigning 80% of users to training and 20% to testing. Per iteration, this yielded 10 training users for D1, 14 for D5, and 2 for D4. Due to the small number of distinct users in D4 (three users), additional bootstrap sampling was used on the training set to encourage sample variance and prevent overfitting.

Table 12 shows the resulting MCC scores. Here, we can see that **3MuViS** performed better than the other models (except **Brute Force**) on D5. For D1 and D4, **Random Forest MVS** performed better than **3MuViS**. Looking closely at the performance, **Random Forest MVS** exhibits a marginal performance edge on D1 and D4. This behavior can likely be attributed to the inherent robustness of Random Forest. Specifically, the ensemble's feature and bootstrap randomization mitigates overfitting to user-specific traits, facilitating superior generalization to unseen subjects.

Table 12. MCC scores (user-independent) distribution measurements by dataset and by method, where the highest mean score is in bold—not considering the Brute Force upper-bound baseline.

Dataset	Experiment	Mean	Std	IQR	Min	Max
D1: MHAD	3MuViS	0.892	0.047	0.061	0.801	0.97
	Random Choice	0.845	0.076	0.119	0.612	0.950
	Random Forest MVS	0.908	0.039	0.052	0.799	0.953
	Randopt	0.852	0.056	0.005	0.728	0.954
	Brute Force (Baseline)	0.986	—	—	—	—
D4: HTAD	3MuViS	0.578	0.136	0.232	0.487	0.725
	Random Choice	0.591	0.123	0.13	0.191	0.708
	Random Forest MVS	0.597	0.088	0.169	0.487	0.725
	Randopt	0.503	0.136	0.216	0.321	0.732
	Brute Force (Baseline)	0.615	—	—	—	—
D5: HAR70+	3MuViS	0.910	0.026	0.032	0.839	0.958
	Random Choice	0.875	0.052	0.053	0.765	0.943
	Random Forest MVS	0.904	0.027	0.032	0.847	0.939
	Randopt	0.878	0.045	0.062	0.765	0.943
	Brute Force (Baseline)	0.932	—	—	—	—

As detailed in Table 13, the Wilcoxon signed-rank test results demonstrate that **3MuViS** remains a significantly superior alternative to **Random Choice** and **Randopt**. Conversely, the same statistical tests reveal no significant advantage for **3MuViS** over **Random Forest MVS** across these datasets.

Table 13. *p*-Values for Wilcoxon test, for the alternative hypothesis that 3MuViS performance is greater than the other methods. ** $p < 0.01$, and *** $p < 0.001$. The symbol + indicates no statistical significance.

Dataset	Random Forest MVS	Random Choice	Randopt
D1: MHAD	+	**	**
D4: HTAD	+	***	***
D5: HAR70	+	***	***

Table 14 shows the optimal selection of the **Brute Force** approach (which was the best in all datasets) for the user-independent validation tests.

Table 14. Optimal ensemble configurations and MCCs identified via exhaustive Brute Force search.

Dataset	View 1 Learner	View 2 Learner	View 3 Learner	Meta-Learner	MCC
D1	LDA	LDA	LDA	NearestCentroid	0.986
D4	MultinomialNB	HGB	—	GaussianNB	0.615
D5	ExtraTress	HGB	—	RandomForest	0.932

Note: — indicates the absence of a third view.

5. Discussion

In this work, the 3MuViS algorithm was tested extensively on different datasets that presented varied generalization problems. Coupled with this, the Multi-View Stacking found by 3MuViS outperformed those assembled by random strategies (Random Choice and Randopt). Furthermore, on five out of the six datasets, 3MuViS yielded superior results than the Multi-View Stacking consisting of a fixed Random Forest classifier, which is one of the most out-of-the-box, high-performing classifiers in diverse application domains [28].

Another important observation is the stability exhibited by 3MuViS across repeated experimental runs. The proposed method generally produced lower variance and narrower score distributions than stochastic baselines, particularly in challenging datasets such as Transport and Attacks. The UpSet plots further support this observation by showing that the algorithm frequently converged toward similar learner combinations across iterations. Such consistency indicates that the methodology is capable of identifying stable and reliable architectural configurations rather than producing arbitrary or highly variable selections.

Regarding research question 1 (**RQ1**) (see Section 1), which investigated whether 3MuViS improves performance compared with using a fixed learner, the results indicate that the answer is generally positive. Across the evaluated datasets, 3MuViS consistently matched or exceeded the performance of Random Forest MVS, with statistically significant improvements observed in most cases. Only D1 presented no statistically significant difference, suggesting that although fixed learners may perform competitively in some scenarios, adaptive learner selection provides superior generalization in the majority of cases. For **RQ2**, which explored whether 3MuViS performs better than stochastic learner selection strategies, the experimental evidence strongly supports the superiority of 3MuViS. Both Random Choice and Randopt produced lower MCC scores and substantially higher variability across runs. Regarding **RQ3**, which examined whether 3MuViS results generalize across different domains, the results showed that 3MuViS consistently outperformed all stochastic baseline models on all datasets. In the case of MVS with fixed Random Forest, our approach achieved statistically higher MCC scores in five of the six datasets. In the remaining dataset, there were no statistical differences (Table 9). For **RQ4**, which evaluated whether optimizing both view-level learners and the meta-learner provides advantages over optimizing only the meta-learner, the comparison against Randopt provides strong evidence in favor of the complete optimization strategy proposed in 3MuViS. The results suggest that optimizing only the meta-learner is insufficient to fully exploit the representational capacity of multi-view data.

As expected, the overall performance of all methods experienced degradation under the subject-independent evaluation. When addressing **RQ5**, the empirical analysis demonstrates that 3MuViS remains a significantly superior alternative to both Random Choice and Randopt in multi-user scenarios while Random Forest MVS showed superior performance on two out of the three datasets.

An important limitation of the current subject-independent evaluation lies within the internal cross-validation performed by the Multi-view Stacking method and during both the view-level and meta-learner training phases, which does not generate mutually exclusive user partitions. Implementing subject-independent partitions within these internal phases would allow the 3MuViS algorithm to optimize learner selection based explicitly on how effectively it generalizes to unseen subjects across both the view-level and meta-learning stages. While such modification could potentially alleviate the observed performance degradation in multi-user scenarios, the human activity datasets currently available for this study lack a sufficient number of unique subjects to support such fine-grained independent splits effectively. Thus, detailed method optimizations for multi-user settings will be explored in future work.

The variety of domain knowledge and classification cases was the main focus while collecting datasets to test 3MuViS on. While human activity recognition is a common, high-interest domain for multi-source data problems, 3MuViS performance is also outstanding on D2 and D6, which describe handwritten numbers and network intrusion readings, respectively. The collective properties of the selected datasets cover a wide spectrum of classification challenges, including cases of heavy class imbalance and near-perfect balance, binary and multi-class classification tasks, and labels exhibiting both high and low

separability. Given the substantial spread in domains, data characteristics, and inherent classification difficulty represented by these datasets, we are confident that the core 3MuViS methodology exhibits high generality. It is, therefore, well suited for deployment in diverse classification problems, requiring only minimal domain-specific fine-tuning for adaptation.

6. Conclusions

This paper introduced 3MuViS, a three-phase methodology for automatically selecting view-level learners and meta-learners in Multi-View Stacking architectures. The proposed approach addresses a fundamental challenge in multi-view learning: identifying effective learner combinations without relying on manual configuration or computationally expensive exhaustive-search strategies. The experimental evaluation conducted on six datasets from diverse domains demonstrated the effectiveness of the proposed methodology. The results showed that 3MuViS consistently outperformed stochastic learner selection strategies and achieved superior or comparable performance to a fixed Random Forest-based Multi-View Stacking configuration. Furthermore, 3MuViS approached the performance of the Brute Force upper-bound reference while requiring only a fraction of the execution time. The results also revealed that optimizing both the view-level learners and the meta-learner provides a clear advantage over approaches that optimize only the meta-learner.

Despite these promising results, the proposed methodology still presents several limitations. First, the method makes it necessary to specify the view partitions a priori. This requirement poses a limitation when the intrinsic feature grouping is not immediately evident or is not a natural characteristic of a given classification problem. Consequently, the capability to automatically couple a dataset's variables into meaningful, view-like groups becomes a significant feature of interest in the overall multi-modal scope. This view discovery represents a promising future work direction for expanding the applicability and scope of the 3MuViS algorithm. Furthermore, for a given problem, not all views may provide relevant information. Thus, optimizing the view selection is another promising direction.

A second limitation is that due to constraints on time and computational resources, Deep Learning (DL) algorithms, which are significantly more expensive computationally than the employed classical learners, were excluded from the initial model catalog used for the experiments conducted herein. The next step in the algorithm's testing and validation involves integrating and experimenting with DL techniques as learners for selection within the 3MuViS stacking framework. A third limitation is that the proposed method was evaluated in an offline environment. However, in some applications, data distributions change over time. Thus, it would be worth exploring how to dynamically update the model selection over time (online learning). A fourth limitation is that the internal-cross-validation procedures of the method do not support group-based splitting, which could be advantageous when optimizing a model for user-independent scenarios. Finally, hyperparameters were all set to default values throughout the experiments, as taking these values into consideration adds a layer of complexity for model selection. Future work for the 3MuViS algorithm includes taking hyperparameter optimization (e.g., Grid Search or Bayesian Optimization) into consideration, such that the methodology is able to choose beyond the catalog of models with default settings.

Author Contributions: G.V.-M.: Conceptualization, Methodology, Data curation, Software, Validation, Formal analysis, Investigation, Writing—Original Draft, Visualization, and Writing—Review and Editing. E.G.-C.: Conceptualization, Methodology, Data curation, Software, Validation, Formal analysis, Investigation, Writing—Original Draft, Visualization, Writing—Review and Editing, Funding acquisition, Resources, Supervision, and Project administration. S.H.: Conceptualization, Writing—Review and Editing, and Funding acquisition. J.A.P.-D.: Conceptualization, Writing—Review and Editing, and Funding acquisition. M.Z.: Conceptualization, Writing—Review and Editing, and Funding acquisition. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by Fondo Regional para la Innovación Digital en América Latina y el Caribe (FRIDA), coordinated by LACNIC. This research report was prepared by Tec de Monterrey in close collaboration with the Government of the Republic of El Salvador, as represented by the State Cybersecurity Agency (ACE, Agencia de Ciberseguridad del Estado).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All datasets used in this work are publicly available. See Section 3.2 for references on where to download them. The code to reproduce the results is available here: <https://github.com/guillermo-vm/3muvis> URL (accessed on 16 July 2026).

Acknowledgments: The authors used ChatGPT 5.5 to improve the readability, grammar, and structure of this paper. The technical content, experimental design, data analysis, interpretation of results, and scientific conclusions remain entirely the work of the authors. The authors have reviewed and edited the output and take full responsibility for the content of this publication.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

MVS	Multi-View Stacking
MCC	Matthews Correlation Coefficient
MDS	Multidimensional Scaling
SVC	Support Vector Classifier
SGD	Stochastic Gradient Descent
HGB	Histogram-based Gradient Boosting Classifier
HAR	Human Activity Recognition
DL	Deep Learning
IDS	Intrusion Detection System

Appendix A

This section includes the results' plots for all the remaining datasets.

Appendix A.1. Performance Results

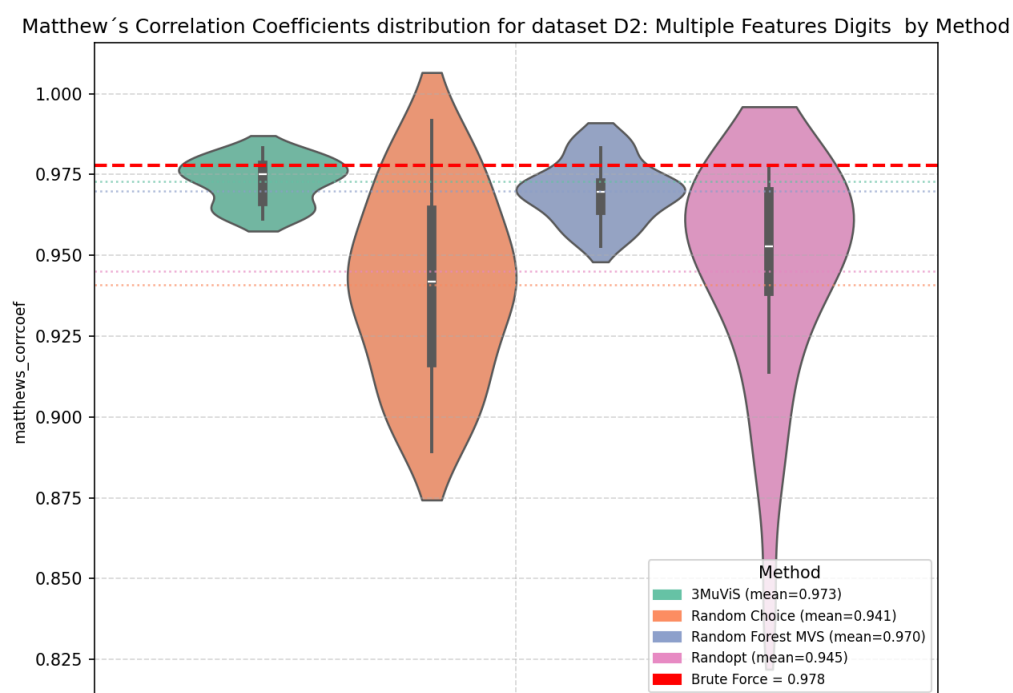


Figure A1. Violin plot with MCC results for D2.

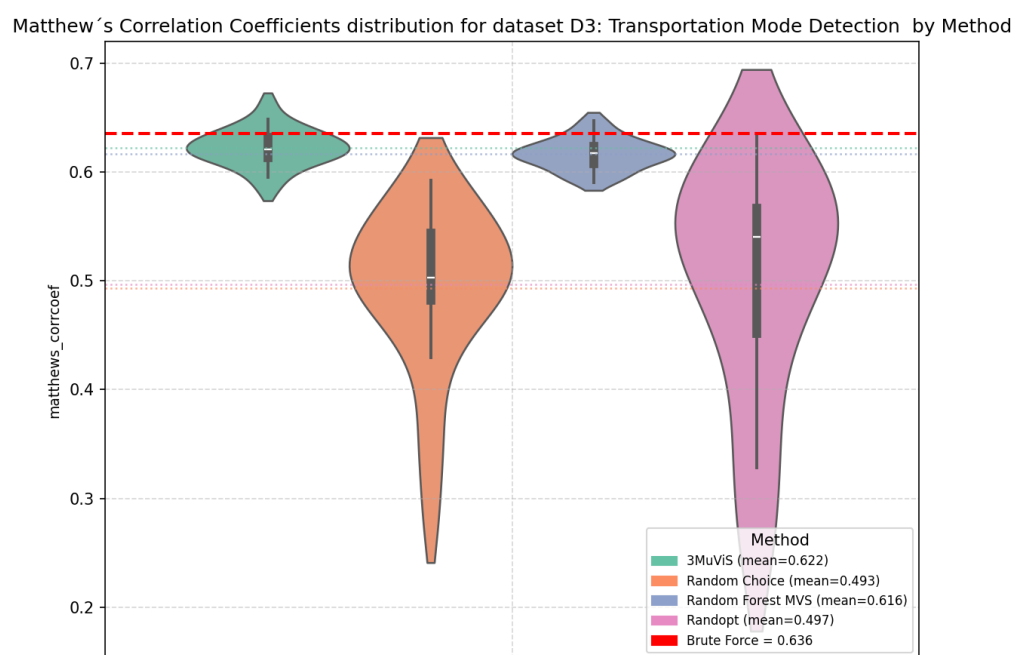


Figure A2. Violin plot with MCC results for D3.

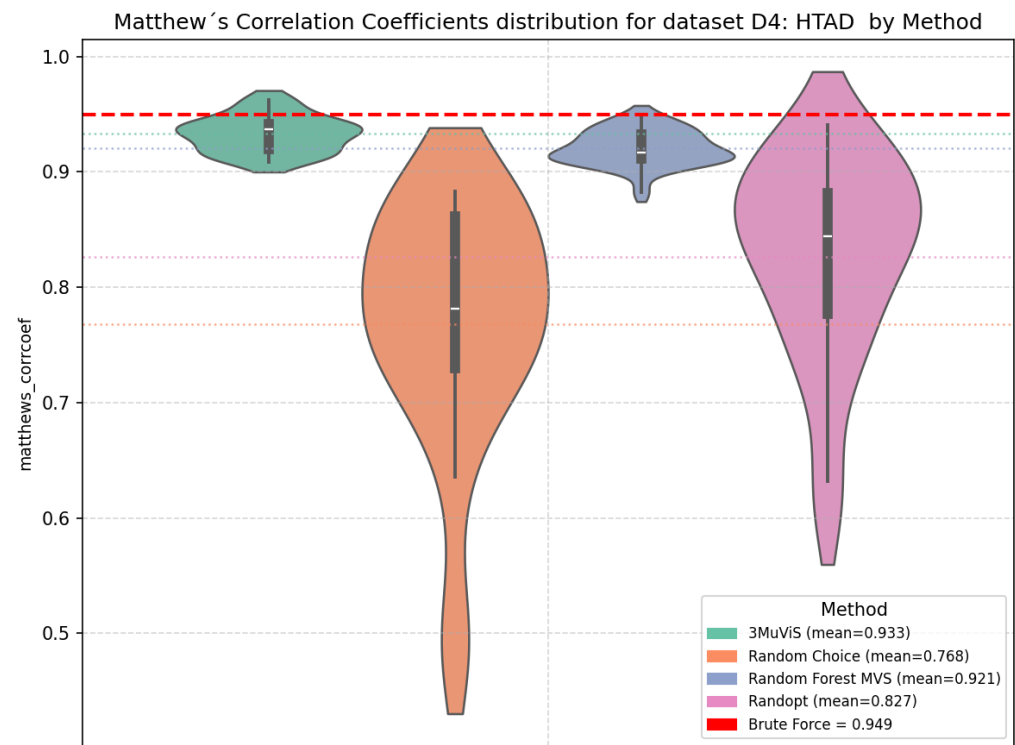


Figure A3. Violin plot with MCC results for D4.

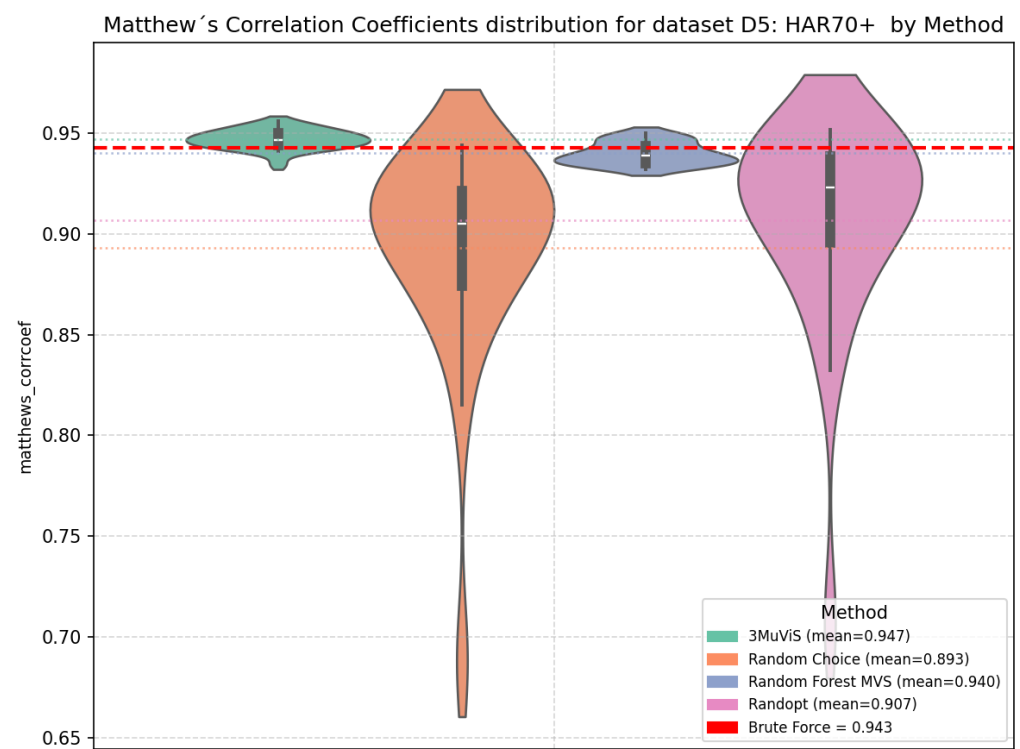


Figure A4. Violin plot with MCC results for D5.

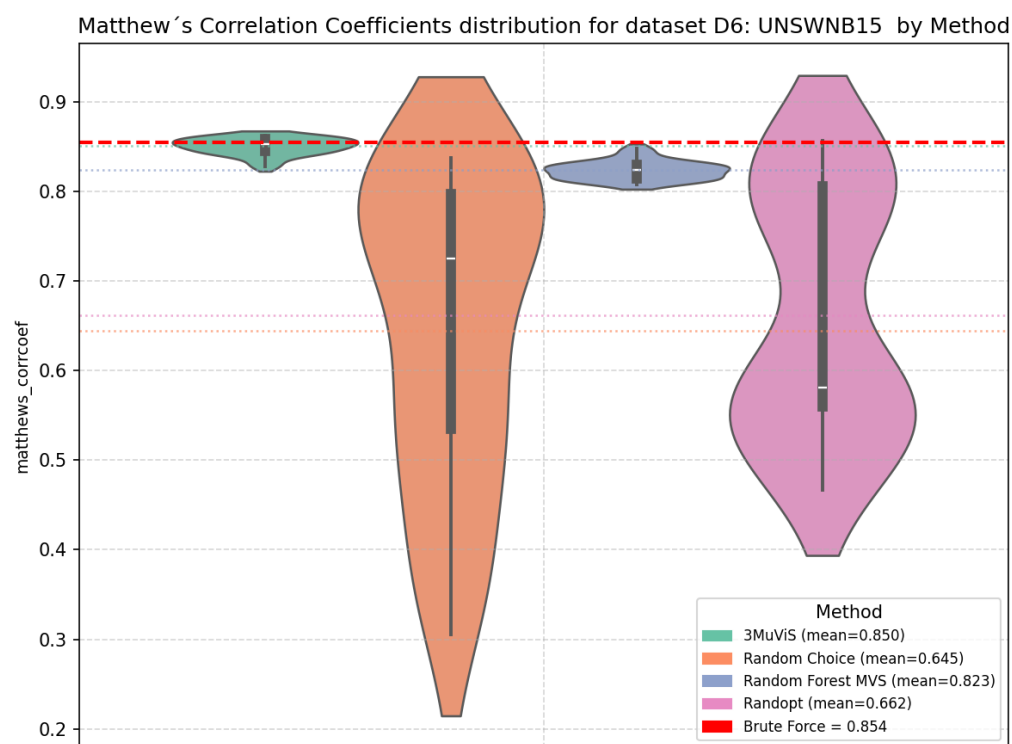


Figure A5. Violin plot with MCC results for D6.

Appendix A.2. Algorithm Selection

Learner Selection - D2: Multiple Feature Digits

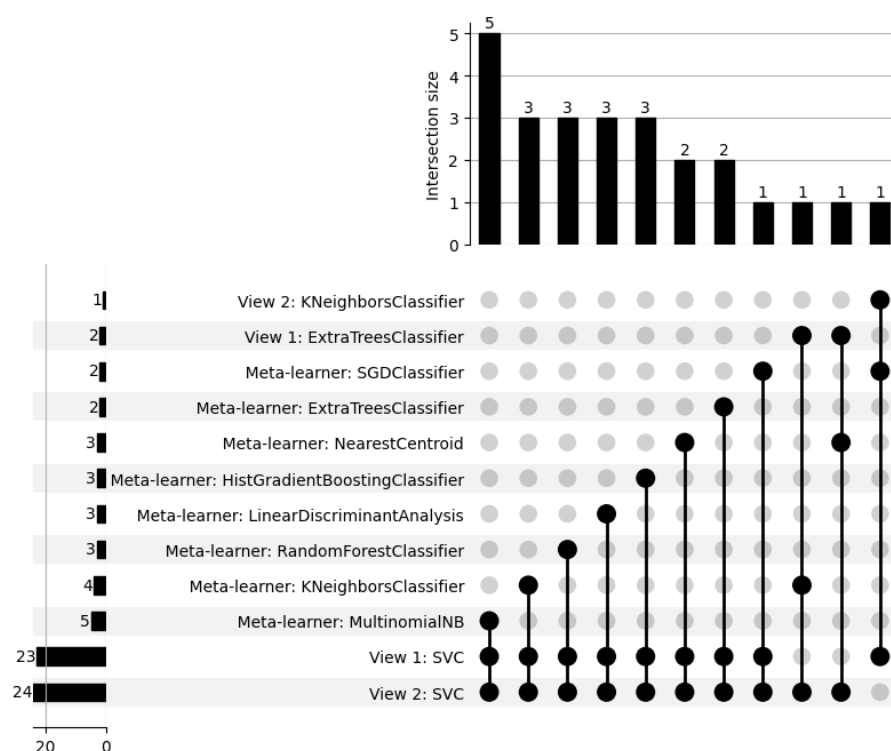


Figure A6. UpSet plot of learner selection for D2. Solid circles indicate the selected learners.

Learner Selection - D3: Transportation Mode Detection

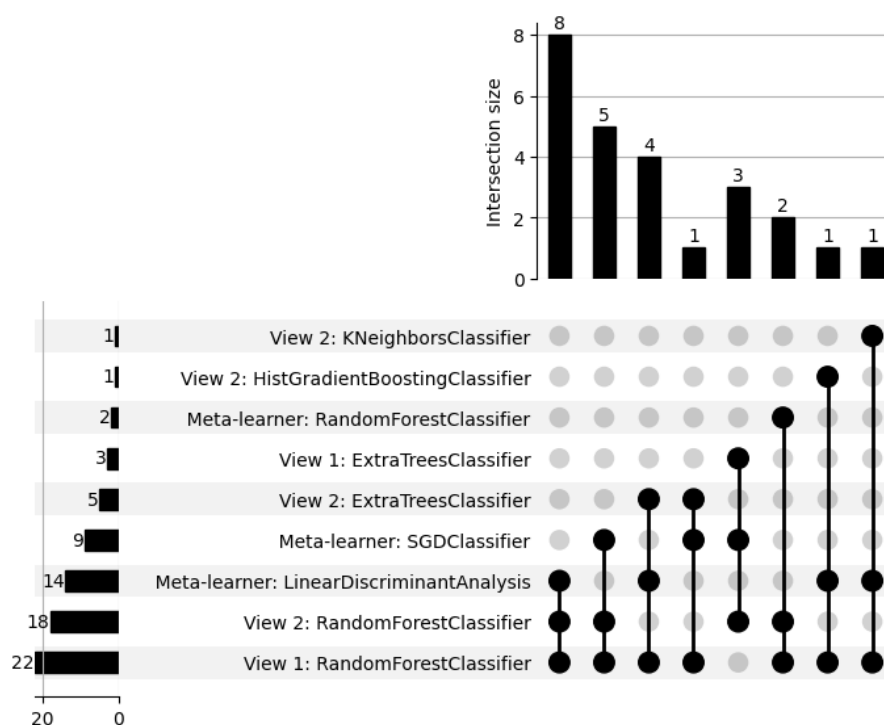


Figure A7. UpSet plot of learner selection for D3. Solid circles indicate the selected learners.

Learner Selection - D4: HTAD

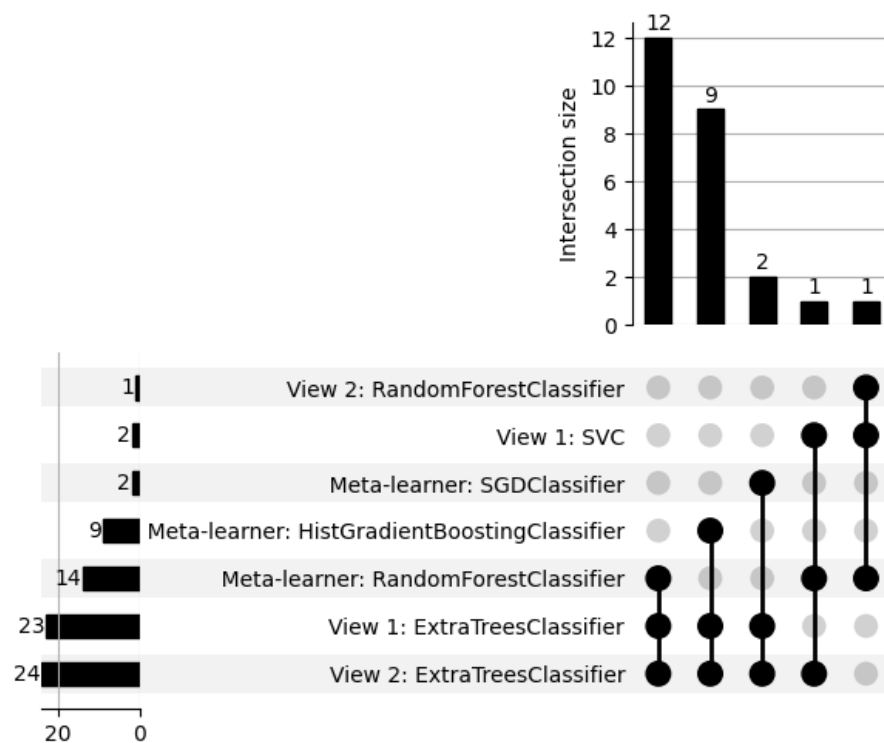


Figure A8. UpSet plot of learner selection for D4. Solid circles indicate the selected learners.

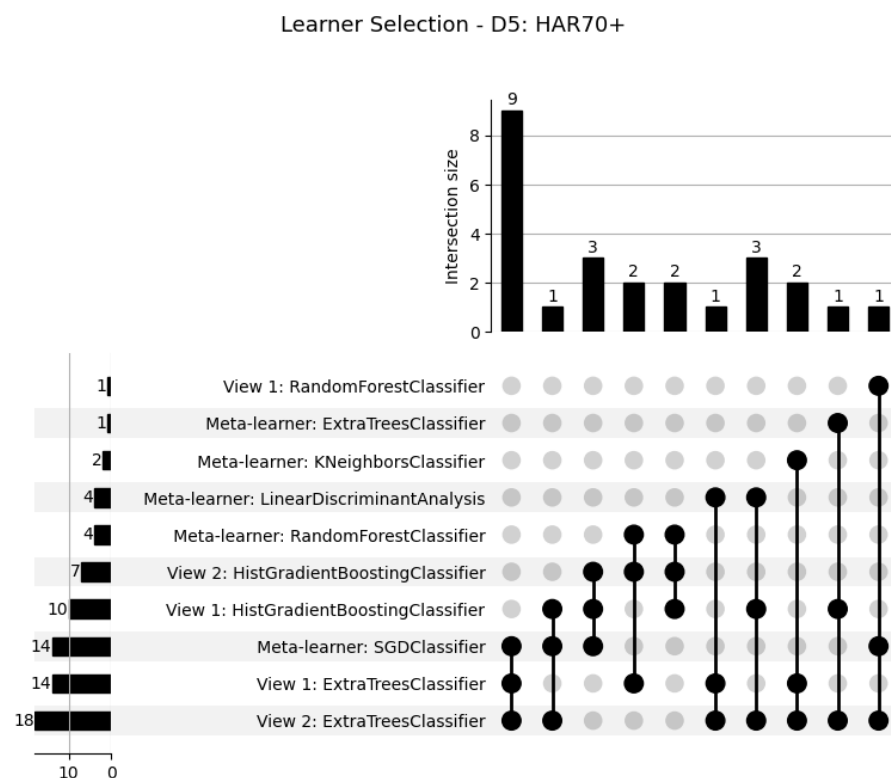


Figure A9. UpSet plot of learner selection for D5. Solid circles indicate the selected learners.

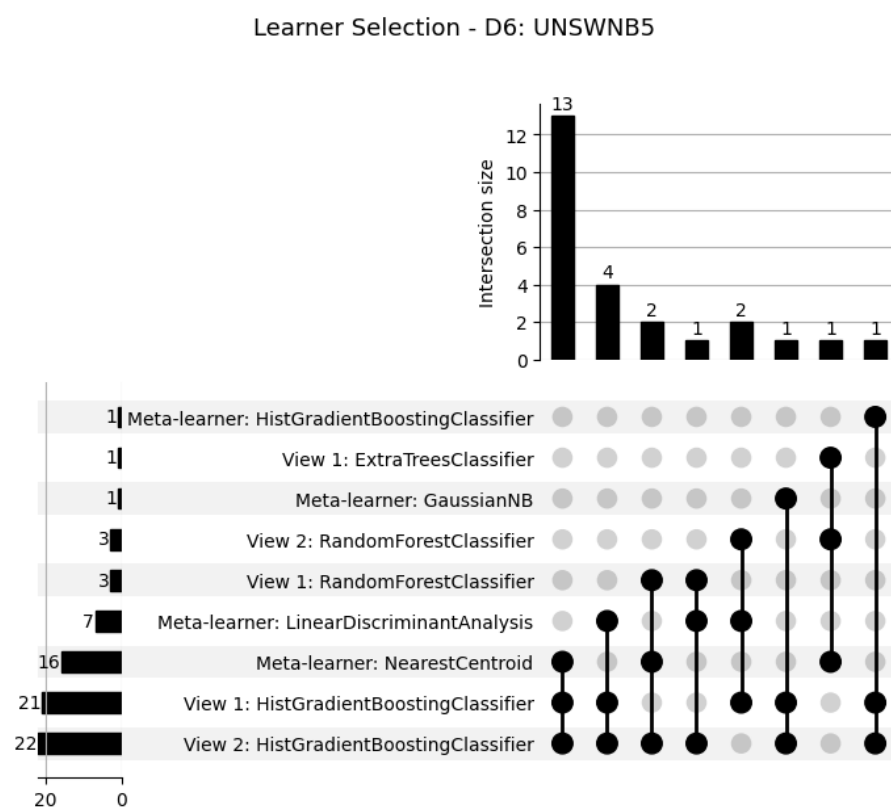


Figure A10. UpSet plot of learner selection for D6. Solid circles indicate the selected learners.

References

1. Wang, X.; Wang, Y.; Yang, J.; Jia, X.; Li, L.; Ding, W.; Wang, F.Y. The survey on multi-source data fusion in cyber-physical-social systems: Foundational infrastructure for industrial metaverses and industries 5.0. *Inf. Fusion* **2024**, *107*, 102321. [CrossRef]
2. Sun, S.; Mao, L.; Dong, Z.; Wu, L. *Multiview Machine Learning*; Springer: Singapore, 2019. [CrossRef]
3. Garcia-Ceja, E.; Galván-Tejada, C.E.; Brena, R. Multi-view stacking for activity recognition with sound and accelerometer data. *Inf. Fusion* **2018**, *40*, 45–56. [CrossRef]
4. Yu, Z.; Dong, Z.; Yu, C.; Yang, K.; Fan, Z.; Chen, C.L.P. A review on multi-view learning. *Front. Comput. Sci.* **2025**, *19*, 197334.
5. Li, M.; Qiao, Y.; Lee, B. Adversarial Robustness Evaluation for Multi-View Deep Learning Cybersecurity Anomaly Detection. *Future Internet* **2025**, *17*, 459. [CrossRef]
6. Corona-Romero, K.; Garcia-Ceja, E.; Mendoza-Montoya, O. MNIST Dataset Classification via Multiview Topological Data Analysis. In *Proceedings of the Mexican International Conference on Artificial Intelligence*; Springer: Berlin/Heidelberg, Germany, 2025; pp. 103–114.
7. El Hafyani, M.; El Himdi, K.; El Adlouni, S.E. Improving monthly precipitation prediction accuracy using machine learning models: A multi-view stacking learning technique. *Front. Water* **2024**, *6*, 1378598. [CrossRef]
8. Garcia-Ceja, E. Multi-View Conformal Learning for Heterogeneous Sensor Fusion. *arXiv* **2024**, arXiv:2402.12307.
9. van Loon, W.; Fokkema, M.; Szabo, B.; de Rooij, M. View selection in multi-view stacking: Choosing the meta-learner. *Adv. Data Anal. Classif.* **2025**, *19*, 579–617. [CrossRef] [PubMed]
10. Barajas-Montiel, S.E.; Morales, E.F.; Escalante, H.J.; Reyes-García, C.A. Automatic Selection of Multi-view Learning Techniques and Views for Pattern Recognition in Electroencephalogram Signals. *Comput. Y Sist.* **2023**, *27*, 211–221. [CrossRef]
11. Caruana, R.; Niculescu-Mizil, A.; Crew, G.; Ksikes, A. Ensemble Selection from Libraries of Models. In *Proceedings of the Twenty-First International Conference on Machine Learning (ICML 2004)*; ACM: New York, NY, USA, 2004; p. 18. [CrossRef]
12. Zhao, J.; Xie, X.; Xu, X.; Sun, S. Multi-view learning overview: Recent progress and new challenges. *Inf. Fusion* **2017**, *38*, 43–54. [CrossRef]
13. Blum, A.; Mitchell, T. Combining labeled and unlabeled data with co-training. In *COLT' 98: Proceedings of the Eleventh Annual Conference on Computational Learning Theory*; Association for Computing Machinery: New York, NY, USA, 1998; pp. 92–100. [CrossRef]
14. Wolpert, D.H. Stacked generalization. *Neural Netw.* **1992**, *5*, 241–259. [CrossRef]
15. Ting, K.M.; Witten, I.H. Issues in stacked generalization. *J. Artif. Intell. Res. JAIR* **1999**, *10*, 271–289. [CrossRef]
16. Ofli, F.; Chaudhry, R.; Kurillo, G.; Vidal, R.; Bajcsy, R. Berkeley MHAD: A comprehensive Multimodal Human Action Database. In *Proceedings of the 2013 IEEE Workshop on Applications of Computer Vision (WACV)*; IEEE: Piscataway, NJ, USA, 2013; pp. 53–60. [CrossRef]
17. Ofli, F.; Chaudhry, R.; Kurillo, G.; Vidal, R.; Bajcsy, R. Berkeley Multimodal Human Action Database (MHAD). 2013. Available online: https://figshare.com/articles/Berkeley_Multimodal_Human_Action_Database_MHAD_/11809470 (accessed on 16 July 2026). [CrossRef]
18. Duin, R. Multiple Features. UCI Machine Learning Repository. 1998. Available online: <https://archive.ics.uci.edu/dataset/72/multiple+features> (accessed on 16 July 2026). [CrossRef]
19. Carpineti, C.; Lomonaco, V.; Bedogni, L.; Felice, M.D.; Bononi, L. Custom Dual Transportation Mode Detection By Smartphone Devices Exploiting Sensor Diversity. In *Proceedings of the 2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, Los Alamitos, CA, USA; IEEE: Piscataway, NJ, USA, 2018; pp. 367–372. [CrossRef]
20. Schwartzter, F. TMD Dataset-5 Seconds Sliding Window. 2018. Available online: <https://www.kaggle.com/datasets/fschwartzter/tmd-dataset-5-seconds-sliding-window> (accessed on 16 July 2026).
21. Garcia-Ceja, E.; Thambawita, V.; Hicks, S.A.; Jha, D.; Jakobsen, P.; Hammer, H.L.; Halvorsen, P.; Riegler, M.A. HTAD: A Home-Tasks Activities Dataset with Wrist-Accelerometer and Audio Features. In *Proceedings of the MultiMedia Modeling*; Lokoč, J., Skopal, T., Schoeffmann, K., Mezaris, V., Li, X., Vrochidis, S., Patras, I., Eds.; Springer: Cham, Switzerland, 2021; pp. 196–205.
22. Thambawita, V.L.; Hicks, S.; Riegler, M. HTAD: A Home-Tasks Activities Dataset with Wrist-Accelerometer and Audio Features. 2023. Available online: <https://osf.io/4dnh8> (accessed on 16 July 2026).
23. Logacjov, A.; Ustad, A. HAR70+ Dataset. UCI Irvine Machine Learning Repository. 2025. Available online: <https://archive-beta.ics.uci.edu/dataset/780/har70> (accessed on 16 July 2026).
24. Ustad, A.; Logacjov, A.; Trollebø, S.Ø.; Thingstad, P.; Vereijken, B.; Bach, K.; Maroni, N.S. Validation of an Activity Type Recognition Model Classifying Daily Physical Behavior in Older Adults: The HAR70+ Model. *Sensors* **2023**, *23*, 2368. [CrossRef] [PubMed]
25. Gower, J.C. Some distance properties of latent root and vector methods used in multivariate analysis. *Biometrika* **1966**, *53*, 325–338. [CrossRef]

26. Moustafa, N.; Slay, J. UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In *Proceedings of the 2015 Military Communications and Information Systems Conference (MilCIS)*; IEEE: Piscataway, NJ, USA, 2015; pp. 1–6.
27. Moustafa, N. The UNSW-NB15 Dataset. 2021. Available online: <https://research.unsw.edu.au/projects/unsw-nb15-dataset> (accessed on 16 July 2026).
28. Fernández-Delgado, M.; Cernadas, E.; Barro, S.; Amorim, D. Do we Need Hundreds of Classifiers to Solve Real World Classification Problems? *J. Mach. Learn. Res.* **2014**, *15*, 3133–3181.
29. Lex, A.; Gehlenborg, N.; Strobel, H.; Vuilleumot, R.; Pfister, H. UpSet: Visualization of Intersecting Sets. *IEEE Trans. Vis. Comput. Graph. InfoVis* **2014**, *20*, 1983–1992. [[CrossRef](#)] [[PubMed](#)]
30. Dehghani, A.; Glatard, T.; Shihab, E. Subject Cross Validation in Human Activity Recognition. *arXiv* **2019**, arXiv:1904.02666. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.