ELEN0062 - Introduction to Machine Learning
Project 3 - Competition: Human Activity Recognition

DUY VU DINH (S2401627)

1 OVERVIEW

1.1 Introduction

Human activity recognition (HAR) has emerged as a critical area in various domains. By analyzing data from wearable sensors, it becomes possible to understand and classify human activities, enabling applications such as health monitoring, fitness tracking, and elderly care. However, this process is challenging due to noisy data, differences across individuals, and imbalanced activity distributions.

This project aims to tackle the problem of HAR using a dataset comprising time series data collected from multiple sensors (31) attached to subjects performing various activities (14). The primary goal is to develop machine learning models that accurately classify activities based on sensor data while addressing challenges such as missing data, imbalanced activity distribution, feature engineering and feature selection.

1.2 Dataset

The dataset includes:

- Sensors: Data from 31 sensors, each providing 512 time-series points per activity.
- Subjects and activities: 5 subjects performing 14 different activities (e.g., sitting, walking, running).
- Learning and test sets: Split into a learning set for training/validation and a test set for final evaluation.

1.3 Report Outline

- Exploratory Data Analysis (EDA): Overview of data distributions, patterns, and key insights. Includes analysis of missing data and class imbalances.
- Preprocessing: Handling duplicates, missing values, and outliers, along with feature scaling and normalization.
- Data Splitting: Description of the methodology for splitting data into training, validation, and test sets, ensuring subject-wise separation.
- Feature Engineering: Extraction of statistical and domain-specific features from time series data, including mean, standard deviation, and entropy.
- Feature Selection: Methods used for reducing feature dimensionality, including filter-based methods and wrapper techniques like Recursive Feature Elimination.
- Modeling: Overview of machine learning models evaluated, hyperparameter tuning strategies, and training protocols.
- Results: Comprehensive evaluation of model performance on validation and test sets. Includes accuracy trends, confusion matrices, and classifier comparisons.
- Conclusion and Future Work: Summarizing findings, limitations of the current approach, and recommendations for future improvements.

2 EXPLORATORY DATA ANALYSIS (EDA)

2.1 Data distribution

2.1.1 Distribution of activities. Table 1 and Figure 1 show that the data set has a balanced activity distribution between the learning set and the test set. Each activity (IDs from 1 to 14) has exactly 250 samples and a percentage distribution per activity: 7.14%.

Table 1. Distribution of activities in the learning and test sets.

Activity ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Count	250	250	250	250	250	250	250	250	250	250	250	250	250	250
Percentage (%)	7.14	7.14	7.14	7.14	7.14	7.14	7.14	7.14	7.14	7.14	7.14	7.14	7.14	7.14

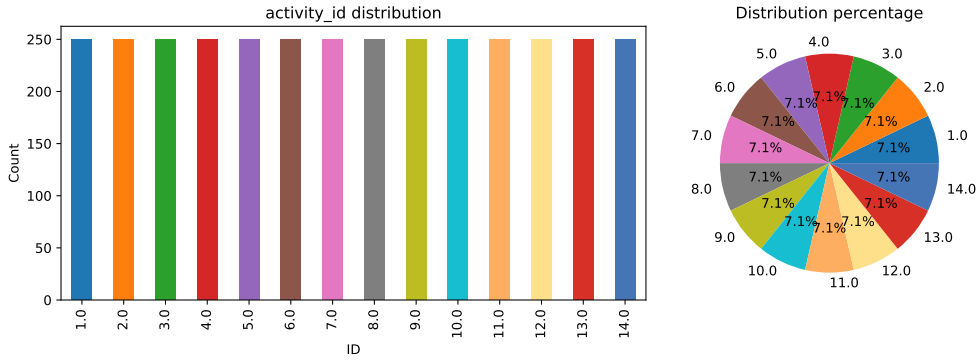


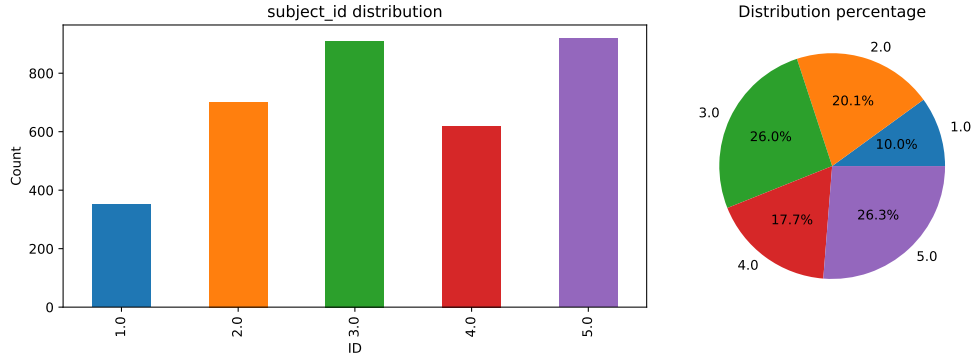
Fig. 1. Distribution of activities in the learning and test sets.

2.1.2 Distribution of subjects. Table 2 shows that there are 5 subjects in learning set (IDs from 1 to 5) and 3 subjects regarding test set (IDs from 6 to 8). The distribution of subjects reveals a notable imbalance in both the learning set and test set (Table 2), which can significantly impact model training and evaluation.

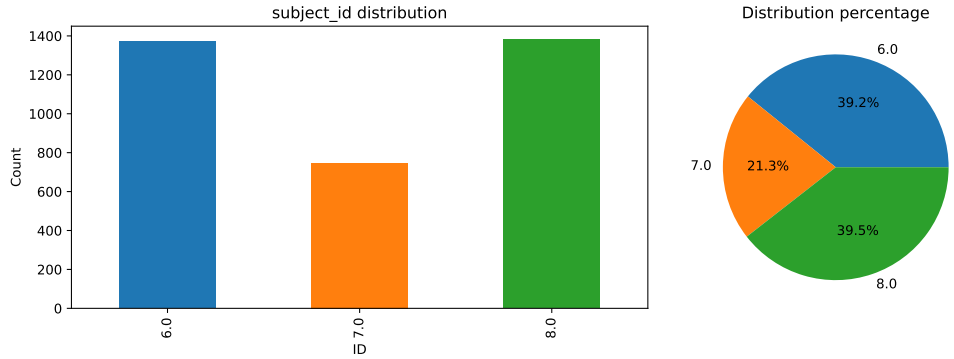
Table 2. Distribution of subjects in the learning set and test set.

Dataset	Subject ID	1	2	3	4	5	6	7	8
Learning set	Count	350	702	910	619	919	-	-	-
	Percentage (%)	10.00	20.06	26.00	17.69	26.26	-	-	-
Test set	Count	-	-	-	-	-	1372	747	1381
	Percentage (%)	-	-	-	-	-	39.20	21.34	39.46

In the learning set (Figure 2a), subjects contribute unevenly to the data, with subjects 3 and 5 dominating at 26.0% (910 samples) and 26.26% (919 samples), respectively, while subject 1 is underrepresented with only 10.0% (350 samples). This imbalance means that subjects with larger sample sizes could disproportionately influence the learning process, potentially leading to subject bias where the model performs better for overrepresented subjects. To mitigate these effects, careful group-based splitting strategies are essential, which ensures that training and validation folds have a



(a) Learning set



(b) Test set

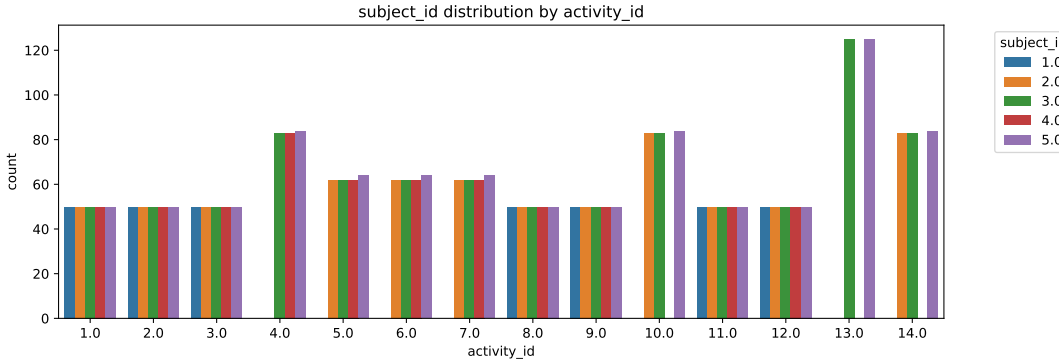
Fig. 2. Distribution of subjects in the learning set and test set.

balanced representation of subjects. By acknowledging and addressing the imbalance, the report ensures transparency in model evaluation and improves the model's ability to generalize across all subjects.

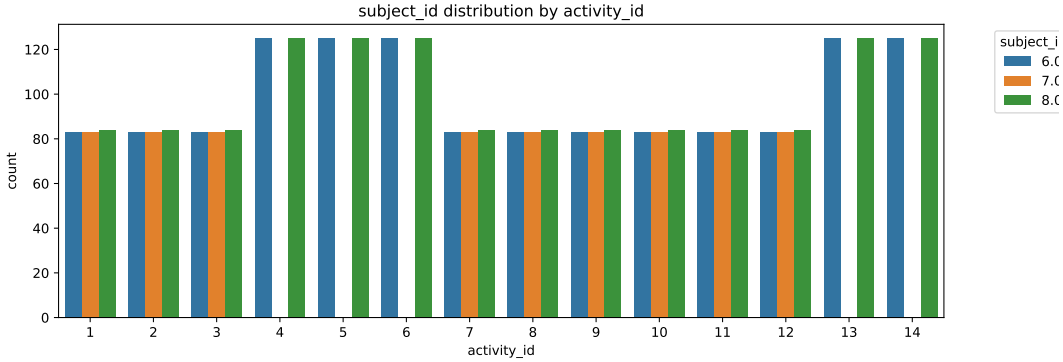
In the test set (Figure 2b), a similar imbalance exists, with Subjects 6 and 8 contributing 39.2% (1372 samples) and 39.46% (1381 samples), respectively, while Subject 7 accounts for only 21.34% (747 samples). This uneven representation in the test set may cause misleading performance metrics, as the model might generalize poorly for underrepresented subjects while appearing strong overall.

2.1.3 Subject distribution by activity. While all activities are represented across subjects in both the learning and test sets, subject-specific imbalances are evident

In the learning set (Figure 3a), subjects perform each of the 14 activities, ensuring that every activity is represented. However, imbalances and missing activities exist across subjects in terms of activity participation, which could introduce subject-specific biases during training. Subjects 3 and 5 participated in all activity and show higher activity counts for certain activities like activity 4 (Walking very slow) and 13 (Rope jumping). Conversely, subjects 1, 2, and 4 display noticeable inconsistencies. Subject 1 participates in the least number of activities compared to other subjects (only 7 activities counted: 1, 2, 3, 8, 9, 11, and 12). This missing participation reduces the coverage for this subject, limiting the



(a) Learning set



(b) Test set

Fig. 3. Distribution of subjects by activity in the learning set and test set.

model's ability to generalize activity patterns for this subject. Similarly, subject 2 has consistently lower activity counts, which may result in insufficient representation for certain activities like 4 (Waling very slow) and 13 (Rope jumping). On the other hand, Subject 4 shows an over-representation in activity 4 (Walking very slow) and 10, while contributing nothing for several activities 10, 13, 14. This imbalance risks skewing the learning process, as the model may overfit activities dominated by specific subjects.

To mitigate these issues, it is essential to split the learning set by subjects, which ensures that no single subject dominates the validation set, and the model is tested on unseen subjects. Additionally, resampling or weighting techniques could help balance the contribution of subjects during training.

In the test set (Figure 3b), subjects 6 and 8 contribute a disproportionately high number of samples, while subject 7 only joins activities 1, 2, 3, 7, 8, 9, 10, 11, and 12.

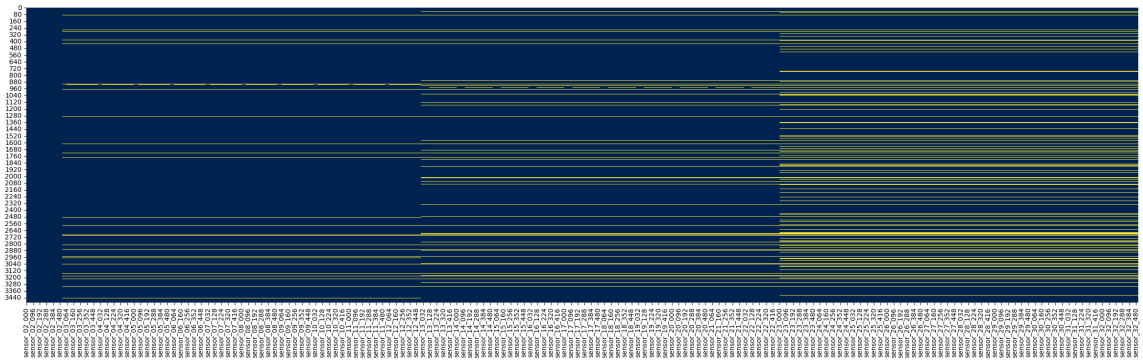


Fig. 4. Missingness.

2.2 Missingness

2.2.1 Overall mssingness. The learning set has a total of 4,957,990 missing values out of 55,552,000 data points, resulting in an overall missingness of 8.92%. The test set witnesses no missing values (0% missingness) in this dataset. Figure shows the distribution of missing values throughout the learning set.

2.2.2 Missingness by columns. Analyzing missing values by column in the learning set reveals significant variation.

Table 3. Missingness by columns in the learning set.

Missing %	0.00	4.63	4.66	4.69	7.06	7.09	7.11	7.14	15.86	15.89	15.91
Count	512	600	1980	2540	1130	1420	1940	630	330	1520	3270
Percentage% over total	3.23	3.78	12.47	16.0	7.12	8.95	12.22	3.97	2.08	9.58	20.60

Based on Table 3, most columns have missing percentages concentrated at specific thresholds:

- 4.63% to 7.14%: Moderate missingness affecting many columns.
- 15.85% to 15.91%: Columns with higher missing percentages, contributing nearly 30% of the total missing values.

This uneven distribution of missing values suggests that some columns are particularly unreliable and need further examination.

2.2.3 Missingness by rows. According to Table 4, most rows are complete, while it can be clearly seen that there are 3 significant thresholds of missing.

Table 4. Missingness by rows in the learning set.

Missing %	0.00	11.59	18.15	28.48	32.26	47.88	50.53	57.40	61.81	64.52	90.54	96.77
Count	2895	1	1	1	352	1	1	1	1	130	1	115
Percentage % over total	82.71	0.03	0.03	0.03	10.06	0.03	0.03	0.03	0.03	3.71	0.03	3.29

A significant portion of the dataset, 82.71% (2,895 rows), has no missing values, providing a strong foundation of reliable data for training and validation. These rows can be directly utilized without any additional preprocessing.

However, the dataset also includes rows with varying levels of missingness, ranging from 11.59% to as high as 96.77%. Rows with 32.26% missingness account for 10.06% of the dataset (352 rows) and represent the largest group of incomplete rows. These rows likely stem from specific sensor or subject-level issues and require careful handling to retain their value for analysis. For rows with very high missing percentages, such as 90.54% and 96.77%, which together make up 3.32% of the dataset (245 rows), imputation may not be effective due to the limited amount of usable data. These rows are strong candidates for removal to reduce noise and improve the quality of the dataset. In addition, rows with the missing rate of 64.52% (including 130 rows) are also considered to be removed. Additionally, rare patterns of missingness occur in single rows at specific percentages, such as 11.59%, 18.15%, 28.48%, 47.88%, 50.53%, 57.40%, and 61.81%. These rows may be anomalies caused by sensor malfunctions or data corruption. Their rarity suggests they are less likely to have a significant impact on the overall analysis and can either be imputed or removed based on their contribution to model performance.

2.2.4 Missingness by sensors. When analyzing missingness across the 31 sensors in the learning set:

- Sensors 03 to 12 exhibit a 5% missing percentage, which is moderate but consistent.
- Sensors 13 to 22 exhibit a 8% missing percentage, which is moderate but consistent.
- Sensors 23 to 32 have 16% missingness, indicating they are less reliable compared to other sensors.
- Some sensors, such as sensor 02, have no missing values.

Table 5. Missingness by Sensors in the learning set.

Sensor	02	03 to 12 13 to 22	23 to 32	
Count	0	83,650	127,183	284,966
Percentage (%)	0.0	5.0	7.1	16.0

This pattern indicates that certain sensors consistently underperform, contributing a higher proportion of missing values across the data. This could affect downstream feature extraction and model performance.

2.2.5 Missingness by subjects. Missingness varies significantly by subject:

- Subject 1, 3, and 5 have 0% missing values, indicating complete data coverage.
- Subject 2 has 22% missingness, and subject 4 has the highest missingness at 25%. This subject-level imbalance suggests that subjects 2 and 4 will require special handling.

Table 6. Missingness by Subjects in the learning set.

Subject ID	1	2	3	4	5
Percentage (%)	0.0	22.0	0.0	25.0	0.0

Imputation strategies (e.g., mean imputation or K-NN imputation grouped by subject) should be applied to retain useful data while addressing the gaps.

3 DATA PREPROCESSING

3.1 Handling duplication

To identify duplicate rows in the learning set (3,500 samples), the following steps were performed:

- Without subject and without activity: When ignoring both subject IDs and activity IDs, **31 duplicates** were identified in the time series data. These duplicates indicate repeated time series patterns across rows, potentially arising from data logging errors.
- With subject but without activity: Including subject IDs while ignoring activity IDs resulted in the same **31 duplicates**. This suggests that duplicate rows were not tied to activity labels but were consistent across subjects.
- With subject and with activity: When considering both subject ID and activity ID, **30 duplicates** were detected. This reduction by one row indicates a specific case where duplicate time series data existed with differing activity labels, highlighting potential data labeling inconsistencies.

To understand the abnormal duplication (conflicting pair of rows), a manual inspection revealed one specific pair of rows (row 414 and row 1002) where identical time series data and subject information were associated with different activity labels. This inconsistency suggests a labeling error, making both rows unreliable for model training.

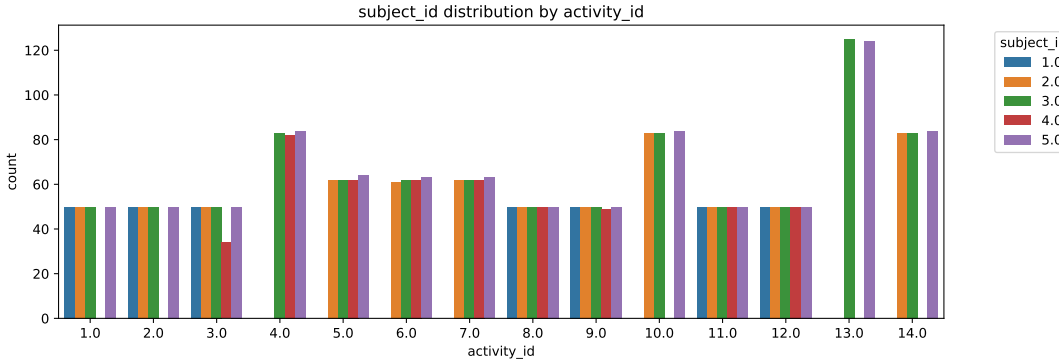
All identified duplicate rows were removed, ensuring that the remaining dataset contained only unique time series data. The duplicates were dropped using a "keep-first" strategy to retain the earliest occurrence of each duplicate time series while discarding subsequent repetitions. In addition, the 414^{th} row is also explicitly removed from the dataset.

After the process of handling duplication, 32 rows were removed. A final check confirmed that there were no remaining duplicates, ensuring data integrity for subsequent analysis and modeling. Therefore, there are 3468 samples left in the dataset.

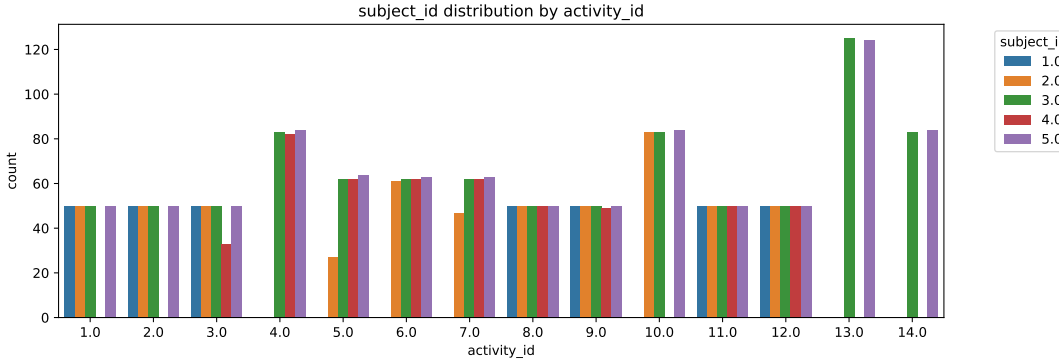
3.2 Handling missingness

3.2.1 Dropping rows with excessive missingness. To address the issue of missingness, the changes in the distribution of subjects by activity across various thresholds for row removal are carefully examined. This analysis helped assess the trade-offs between retaining data completeness and eliminating unreliable rows. In addition, from Sec. 2.2.5, only subjects 2 and 4 experience the heavy missingness.

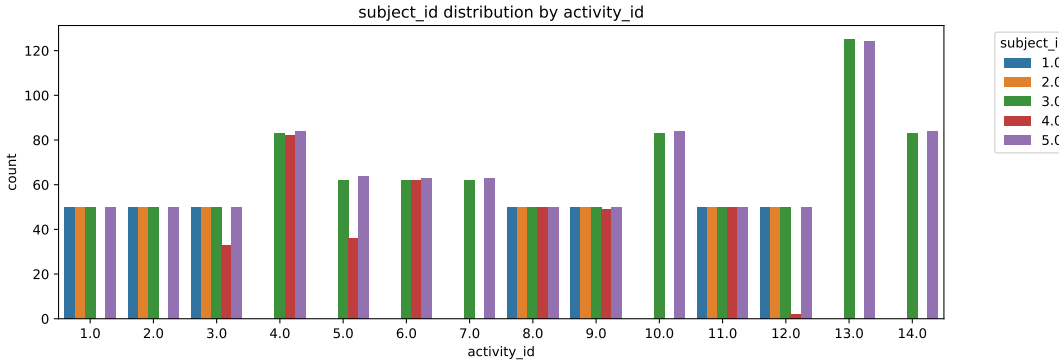
- Original dataset (no dropping) (Figure 3a): The original dataset contains 3,468 rows, and all subjects participate in a balanced manner across activities. However, a significant number of rows exhibit varying levels of missingness, potentially affecting data quality.
- Threshold of 65% missingness (Figure 5a): Dropping rows with more than 65% missing values resulted in the removal of only 90 rows, leaving 3,378 rows. While this threshold retains the largest number of rows, it allows for the inclusion of rows with significant missingness, potentially degrading data quality. Subject 4 is removed from activities 1 and 2, while subject 2 remains its activities.
- Threshold of 33% missingness (Figure 5b): Dropping rows with more than 33% missing values resulted in the removal of 224 rows, leaving 3,244 rows. This threshold achieves a better balance between retaining rows and minimizing missingness. The distribution of subjects and activities is largely preserved, with only minor deviations for certain activities. Subject 2 experiences reductions in activities 5, 6, and 7, then subject 4 does not change its distribution compared to the figures for the threshold of 65



(a) Threshold of 65%



(b) Threshold of 33%



(c) Threshold of 0%

Fig. 5. Distribution of subjects by activity in the learning set when removing missing rows.

- All missing values are dropped (threshold of 0% missingness) (Figure 5c): Dropping rows with more than 0% missing values resulted in the removal of 578 rows, leaving 2,890 rows. While this threshold ensures no

missingness in the retained dataset, the removal of these rows slightly disrupts the distribution of subjects across activities, particularly for subjects with smaller contributions. Subject 2 is not available in activities 4, 5, 6, 7, 10, 13, and 14. Activities 1, 2, 7, 10, 13, and 14 don't contain any subject 4.

In this project, I opted to drop rows exceeding 33% missingness for the following reasons. Firstly, this threshold eliminates rows with excessive missingness while retaining rows that are relatively complete, ensuring that the data remains reliable for feature extraction and modeling. Next, as evident in the subject-by-activity distribution, this threshold strikes a balance by maintaining the overall structure of the dataset. The participation of subjects across activities remains consistent with the original dataset. While a stricter threshold (0%) reduces missingness further, it also risks overfitting the model to a reduced dataset, potentially affecting generalizability. In practice, the dropping missing rows should be examined more.

3.2.2 Data imputation. For the remaining dataset, the missing values were imputed using mean imputation, grouped by subject.

The missing values in each time series column were filled with the mean value of that column, computed separately for each subject. This approach ensures that subject-specific variations are preserved, maintaining the integrity of the data while addressing missingness. Before imputation, the dataset contained 1,806,360 missing values across all rows and columns. After imputation, the dataset became fully complete.

Alternatively, the framework allows for using k-NN imputer, where missing values are filled based on the nearest neighbors' similarity. While not applied here, it remains an option for more sophisticated imputation when needed. However, mean imputation will be used in this project.

3.3 Outlier detection

Outlier detection was omitted in this project due to the complexity of handling time series data and the constraints of limited knowledge and time. Identifying outliers in time series requires advanced techniques like statistical methods, spectral analysis, or model-based approaches, which were beyond the scope of this work. Additionally, removing potential outliers could risk further data loss, especially given the dataset's existing sparsity from missingness. Instead, the project relies on robust algorithms like Random Forest and XGBoost, which are less sensitive to outliers, and preprocessing steps like feature scaling to mitigate their impact.

4 DATA SPLITTING

The learning set was divided into training and validation sets based on the subject to ensure realistic model evaluation and generalization testing. Subjects 1, 2, 3, and 4 were used for the training set, while subject 5 was held out for the validation set. This split ensures that the model is tested on an entirely unseen subject, simulating real-world scenarios where the model encounters new users.

The training set contains 2,328 samples, accounting for approximately 71.8% of the total learning data, while the validation set includes 916 samples, representing 28.2%. By using subject-based splitting, the model's ability to generalize across different subjects is robustly tested, avoiding potential bias or data leakage that could arise from overlapping subjects between the two sets. This approach provides a solid foundation for subsequent model training and evaluation.

5 FEATURE ENGINEERING

5.1 Problem with time series

Raw time-series data from sensors often presents challenges for machine learning due to its high dimensionality, sequential nature, and potential noise. Machine learning models typically perform better on structured, tabular data rather than on raw sequential data. Feature engineering addresses these challenges by summarizing the key characteristics of the time series into meaningful features, enabling models to extract patterns and make predictions more effectively.

Time series data is inherently complex because it captures sequential measurements over time. For example, each sensor generates 512 data points per sample, resulting in a high-dimensional representation for 31 sensors. Directly using such data can lead to issues like overfitting, increased computational complexity, and difficulty in learning temporal relationships. By transforming the raw data into statistical and signal-based features, the dimensionality will be reduced and the most relevant information for modeling will be retained.

5.2 Feature extraction

The goal of feature engineering is to extract statistical summaries and signal characteristics that represent the underlying patterns in the data. The statistical features used in this project were adapted from the methodology [1]. These features allow models to focus on the overall behavior of the time series rather than individual data points, thus improving interpretability and model performance.

The following features were extracted for each sensor's time-series data to capture its key properties:

Statistical Features:

- Mean: $\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$ (average value of the time series).
- Standard Deviation: $\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2}$ (dispersion of values).
- Median: The middle value when data points are sorted.
- Minimum and Maximum: Represent the range of the time series.
- Interquartile Range (IQR): $IQR = Q3 - Q1$, where $Q3$ and $Q1$ are the 75th and 25th percentiles, respectively.
- Skewness: Skewness $= \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i-\mu)^3}{\sigma^3}$, indicating asymmetry in the data.
- Kurtosis: Kurtosis $= \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i-\mu)^4}{\sigma^4}$, capturing the "tailedness" of the distribution.

Signal Features:

- Signal Magnitude Area (SMA): SMA $= \sum_{i=1}^{n} |x_i|$, capturing the overall magnitude of the signal.
- Energy: Energy $= \sum_{i=1}^{n} x_i^2$, measuring the power of the signal.
- Entropy: $-\sum_{i=1}^{n} p_i \log(p_i)$, where p_i is the probability of each value, representing signal randomness.

The feature engineering process transformed the raw 512-point time series data for each sensor into **11 meaningful features**, effectively capturing the key statistical and signal-based properties of the time series. This reduction significantly decreased the data's dimensionality while retaining its most important characteristics.

For the **31 sensors** in the dataset, this transformation resulted in:

$$\text{Total Features} = 31 \times 11 = 341$$

This is a substantial reduction from the original $31 \times 512 = 15,872$ raw data points, making the data much more manageable for machine learning models while preserving the essential patterns and trends.

This transformation demonstrates the power of feature engineering in simplifying complex time-series data while retaining its most critical information. This reduced the number of features by approximately 97.85% (from 15,872 to 341), lowering computational complexity and the risk of overfitting. The resulting features, such as mean, standard deviation, skewness, and entropy, provide clear insights into the nature of each sensor's data, making it easier to interpret model results. The transformed dataset is now structured, tabular, and standardized, which is more suitable for traditional machine learning models.

5.3 Standardization

After extracting features, they were standardized to ensure a consistent scale across all features. This step prevents features with larger numerical ranges from dominating the model training process. The formula used for standardization is:

$$z = \frac{x - \mu}{\sigma}$$

Where z: Standardized value, $x$is the original feature value, μ is the mean of the feature, σ is the standard deviation of the feature.

The transformed dataset is now structured, tabular, and standardized, which is more suitable for traditional machine learning models.

6 FEATURE SELECTION

6.1 Filter technique

The first step of feature selection involved using a filter technique to identify and remove redundant features based on their pairwise correlation. The aim was to retain features that contribute unique information while eliminating those that are highly correlated with others, thereby simplifying the dataset and improving model performance.

The filter technique evaluates the correlation between features and removes those that exhibit a high degree of linear relationship. This approach ensures that the remaining features are less redundant and more informative.

Regarding correlation threshold, features with a correlation coefficient greater than 0.9 were considered highly correlated. If two features were highly correlated, one of them was dropped to avoid redundancy.

6.1.1 Procedure.

- Compute the correlation matrix.
- Identify features to drop. Features with absolute correlation values above the threshold $abs(corr) > 0.9$) were flagged for removal.
- Retain unique features, for each pair of highly correlated features, one feature was retained while the other was dropped

6.1.2 Results. After applying this technique, the feature set was reduced from 341 to 191 features, eliminating 150 redundant features. This represents a significant reduction of approximately 44% in dimensionality, streamlining the dataset while preserving its most informative characteristics.

6.2 Wrapper technique

The wrapper technique was employed as a secondary feature selection step to refine the features retained after the filter-based selection process. Unlike filter methods that rely solely on statistical metrics, the wrapper technique evaluates the performance of a machine learning model to iteratively identify and eliminate redundant or less influential features. This approach ensures that the remaining features not only correlate with the target feature but also contribute directly to improving the model's predictive performance.

For this project, Recursive Feature Elimination (RFE) was adopted as the core methodology for the wrapper technique. RFE progressively refines the feature set by iteratively removing the least important features while monitoring their impact on model performance. This iterative nature ensures that the final feature set balances predictive power and model simplicity.

6.2.1 Procedure. To implement the wrapper technique, the `RandomForestClassifier` from the `Scikit-learn` module [2] was chosen as the base model. This classifier was selected for its robustness, built-in feature ranking capabilities, and ability to handle non-linear relationships in the data. The following structured steps were followed during this process:

The wrapper technique in this project involved the following structured steps:

- Training and validation split:
  - The learning set was divided into training and validation datasets based on subject IDs, as detailed in Section 4. Specifically, subjects 1, 2, 3, and 4 were assigned to the training set, while subject 5 was used as the validation set. This subject-based splitting was critical to maintain the independence between training and validation data.

- Experimental protocol:
  - Sampling from the training set: Due to the imbalance of activities among subjects and missing activities for subjects 2, 4, and especially subject 1, applying traditional cross-validation techniques was deemed unstable. Instead, experiments were conducted by drawing $M = 20$ subsets of fixed size $N = 1,500$ from the training set. This sampling ensured variability in model training, allowing us to capture the impact of feature elimination on model performance across diverse subsets.
  - Training and validation: For each subset LS_i (where i ranges from 1 to M), train the model and make predictions on the validation set T. This setup ensures that predictions are evaluated on data that is not part of the training samples, providing an unbiased estimate of the model's error.
  - Accuracy Estimation: The average training accuracy and validation accuracy were calculated across the M training iterations to monitor the model's performance during the feature elimination process.
- Iterative feature elimination:
  - At each iteration, feature importance scores were computed using the `feature_importances_` attribute of the `RandomForestClassifier` in order to rank the feature.
  - The feature with the lowest importance score was identified and removed.
  - The model was retrained on the updated feature set, and the process was repeated until no features remained.
- Feature selection:
  - During the iterative process, the validation accuracy was monitored closely.
  - The feature set that yielded the highest validation accuracy was selected for final use. This ensured that the retained features maximized predictive power while minimizing redundancy and complexity.

6.2.2 Results. The RFE process effectively highlights the relationship between the number of features retained and the model's accuracy on both the training and validation datasets. Figure 6 shows the results of RFE process. Initially, reducing the feature set improves validation accuracy as redundant and less informative features are removed, enhancing the model's ability to generalize. However, this trend reverses beyond a critical point—when fewer than 40 features are retained—causing a decline in validation accuracy as essential features are excluded. Similarly, retaining more than 60 features does not yield significant improvement, suggesting that additional features contribute little to predictive performance.

Training accuracy remains consistently high across most feature ranges, with values close to 1.0, indicating that the model can effectively fit the training data. However, when fewer than 10 features are retained, training accuracy drops sharply, reflecting the model's inability to capture sufficient information from a severely reduced feature set. This behavior points to underfitting, where the model fails to learn critical patterns necessary for accurate predictions.

Validation accuracy, in contrast, demonstrates an initial sharp increase as irrelevant features are removed, peaking within the range of 40 to 60 features. This range represents an optimal balance, where the retained features are sufficiently informative while minimizing redundancy and noise. Beyond this range, validation accuracy declines when the feature set becomes too small, leading to underfitting.

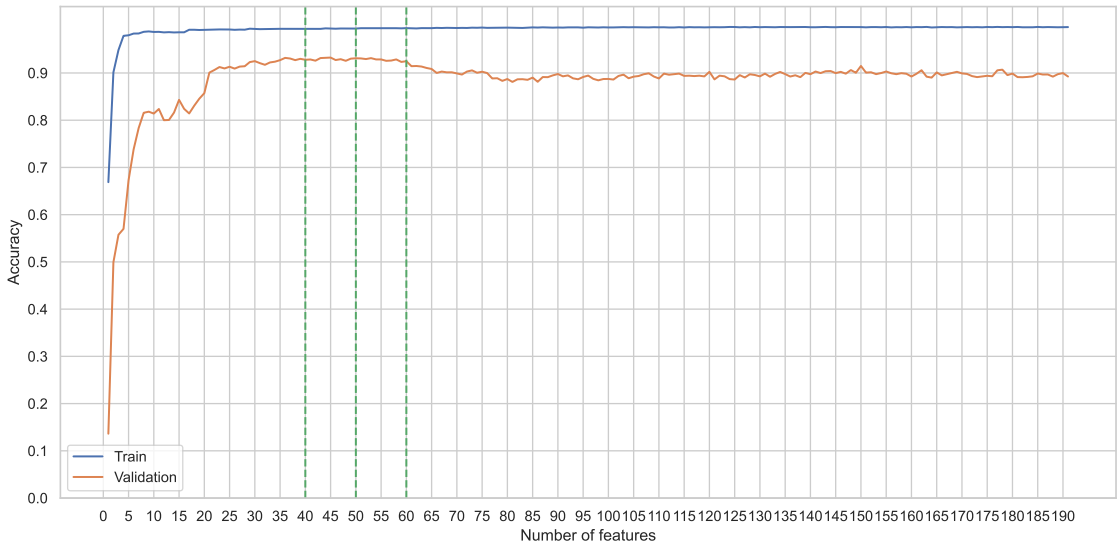


Fig. 6. RFE process.

6.3 Summary

Table 7 provides a comprehensive view of the selected features for the classification problem, highlighting the sensors and the statistical features that contribute significantly to model performance. Each level of feature importance is represented using the following markers:

- *: Features retained in the initial filtering stage (191 features).
- **: Features among the top 60 best-ranked by Recursive Feature Elimination (RFE).
- ***: Features among the top 50 best-ranked by RFE.
- ****: Features among the top 40 best-ranked by RFE.

Acceleration features stand out as the most critical across hand, chest, and foot sensors, reflecting their strong motion-detection capabilities. Temperature sensors, particularly for the heart rate and chest, also provide valuable physiological information. Among groups, hand sensors contribute the most impactful features, followed by chest and foot sensors, reflecting their proximity to central physiological functions.

Among the extracted features, mean and standard deviation appear consistently important, with significant representation across sensors even after feature reduction. Features like mean for the heart rate sensor (Sensor 2), hand accelerometers (sensors 4 to 6), and foot accelerometers (sensors 24 to 26) retain importance even after aggressive feature reduction, indicating their strong predictive value. In contrast, features like SMA and IQR are absent or have minimal importance across all sensors, suggesting limited utility in this dataset for distinguishing between classes.

The heart beat rate sensor (Sensor 2) is consistently important, with parameters such as mean and standard deviation being retained through the most stringent RFE (40 features). This aligns with the physiological relevance of heart rate as a distinct and informative signal. However, other parameters for this sensor, such as kurtosis and entropy, are less relevant, as indicated by their absence in the reduced feature sets.

Table 7. Feature Importance of Sensors

Sensor	Name	mean	std. dev.	median	min.	max.	IQR	Skewness	Kurtosis	SMA	energy	entropy
2	Heart beat rate (bpm)	****	*				*	*	*			
3	Hand temperature (∘C)	****	*					*	*			
4	Hand acceleration X (ms^{-2})	****	****		****	***		*	*		****	***
5	Hand acceleration Y (ms^{-2})	****	****		*	****		****	***			
6	Hand acceleration Z (ms^{-2})	****	**		*	****		*	*		****	
7	Hand gyroscope X (rad/s)	****	**	***	*	*		*	*			
8	Hand gyroscope Y (rad/s)	*	**	*		*		*	*			
9	Hand gyroscope Z (rad/s)	*	****	*				*	*			
10	Hand magnetometer X (µT)	*	*		*	*		*	*		*	
11	Hand magnetometer Y (µT)	*	*			*	*	*	*			*
12	Hand magnetometer Z (µT)	*	*			*	*	*	*			*
13	Chest temperature (∘C)		*					*	*			
14	Chest acceleration X (ms^{-2})	****	****	****				*	*			****
15	Chest acceleration Y (ms^{-2})	*	****		*			*	*		***	**
16	Chest acceleration Z (ms^{-2})	****	***		****	****		*	*			****
17	Chest gyroscope X (rad/s)	****	****		****	***		*	*			****
18	Chest gyroscope Y (rad/s)	*		*				*	*			*
19	Chest gyroscope Z (rad/s)	*	****	*	*	*		*	*			*
20	Chest magnetometer X (µT)	*	*		*	**	*	*	*		*	*
21	Chest magnetometer Y (µT)	*				**	*	*	*			*
22	Chest magnetometer Z (µT)	*	*		*	**	*	*	*		*	*
23	Foot temperature (∘C)	****	*					*	*			
24	Foot acceleration X (ms^{-2})	****	****	***		**		*	*			****
25	Foot acceleration Y (ms^{-2})	**						*	*		****	**
26	Foot acceleration Z (ms^{-2})	****					****	*	*			****
27	Foot gyroscope X (rad/s)	****		*				*	*			
28	Foot gyroscope Y (rad/s)	*		*				*	****			
29	Foot gyroscope Z (rad/s)	*		*				*	***			
30	Foot magnetometer X (µT)	*	*		*	*	*	*	*		*	
31	Foot magnetometer Y (µT)	*	*			*	*	*	*			*
32	Foot magnetometer Z (µT)	***	*		*	****		*	*			*

For the hand group (Sensors 3–12), accelerometers (Sensors 4–6) dominate in importance, with multiple parameters like mean, standard deviation, and maximum being retained. Gyroscopes (Sensors 7–9) show moderate importance but lose relevance in the most reduced feature sets, suggesting that they contribute less uniquely compared to accelerometers. Magnetometers (Sensors 10–12) are largely unimportant, as they do not appear in the reduced feature sets.

The chest sensors (Sensors 13–22) show a similar trend, with accelerometers (Sensors 14–16) being the most valuable. Notably, the chest temperature sensor (Sensor 13) maintains some importance, especially for the mean parameter, reflecting its physiological relevance. Gyroscopes and magnetometers (Sensors 17–22) contribute minimally, with most parameters being removed during feature reduction.

For the foot sensors (Sensors 23–32), accelerometers (Sensors 24–26) again stand out, with parameters like mean and energy retaining their significance. The foot temperature sensor (Sensor 23) shows moderate importance, while gyroscopes (Sensors 27–29) and magnetometers (Sensors 30–32) exhibit limited utility across feature reductions.

Different sensor types reveal unique trends in importance. Temperature sensors, such as the heart rate and chest temperature sensors, provide critical information due to their direct physiological linkage. In contrast, gyroscopes and magnetometers exhibit lower utility, with most of their features being excluded in the reduced sets. Accelerometers emerge as the most critical sensor type, capturing meaningful variations in motion across all body regions.

The reduction process systematically eliminates redundant features, retaining the most significant ones. Parameters like mean, standard deviation persist across reductions, while features like kurtosis and skewness are excluded earlier.

7 MODELING

The primary objective is to classify activities effectively using the selected features (341, 191, 60, 50, 40) and evaluate different classifiers to identify the best-performing model. This section focuses on training, hyperparameter tuning, and evaluating multiple machine learning algorithms to compare their performance.

7.1 Model Selection

The model selection process involves evaluating multiple machine learning algorithms to identify the one that best balances predictive performance, computational efficiency, and robustness for activity classification tasks. The selected classifiers are provided in Table 8. More information for these classifiers and their tunning hyperparameters are shown in Table A.1.

Table 8. Overview of selected models

Classifier Name	Module	Description
Random Forest	`RandomForestClassifier` [2]	A robust ensemble learning method known for its ability to handle high-dimensional data and provide built-in feature importance ranking.
k-Nearest Neighbors	`KNeighborsClassifier` [3]	A distance-based algorithm that predicts class labels based on the majority vote of nearest neighbors.
Decision Tree	`DecisionTreeClassifier` [4]	A tree-structured algorithm useful for interpreting decision-making processes.
Logistic Regression	`LogisticRegression` [5]	A probabilistic model for binary or multiclass classification, effective for datasets with linear separability.
Support Vector Machines	`SVC` [6]	A margin-based classifier that performs well in high-dimensional spaces.
AdaBoost	`AdaBoostClassifier` [7]	An ensemble method that combines weak learners iteratively to improve model accuracy.
Bagging	`BaggingClassifier` [8]	An ensemble technique that aggregates predictions from multiple models trained on random subsets of data.
Multilayer Perceptron	`MLPClassifier` [9]	A neural network-based model that can capture complex relationships in data.
XGBoost	`XGBClassifier` [10]	A powerful gradient-boosting algorithm designed for high accuracy and scalability.

Each classifier's performance is evaluated on multiple feature configurations and subsets to determine the most suitable model for this application.

7.2 Feature Configurations

Feature configurations are critical for managing the high dimensionality of the time-series data. The following subsets of features are considered during the modeling process:

- All Features (341): The complete set of extracted features without any dimensionality reduction, also known as baseline configuration.
- Filtered Features (191): A subset obtained by removing features with correlations above 90% to reduce redundancy.
- Wrapper Technique Features:
  - 60 features: Features selected based on the highest validation accuracy during Recursive Feature Elimination (RFE).
  - 50 features: A refined subset balancing model simplicity and accuracy.
  - 40 features: A minimal yet effective feature set.

By analyzing the performance of each model on these configurations, the impact of feature selection on classification accuracy and computational efficiency is assessed.

7.3 Training Protocol

The training protocol follows a structured approach to ensure robust model evaluation:

- Data splitting: The learning dataset is divided into training and validation sets based on subject IDs to avoid data leakage and ensure independence. The test set is held out for final evaluation.
- Sampling strategy: For each training iteration, $N = 1,500$ samples are randomly drawn from the training set to train the models. This random sampling is repeated times ($M = 20$) for each hyperparameter combination to account for variability in training results.
- Training Iterations: Each model is trained multiple times on different subsets of the data, and its predictions are evaluated on the validation set for unbiased error estimation.
- Final training: The best model (after receiving label data of the test set: regarding each type of classifier) is retrained on the entire learning dataset with its optimal hyperparameters and evaluated on the independent test set.

7.4 Evaluation Metrics

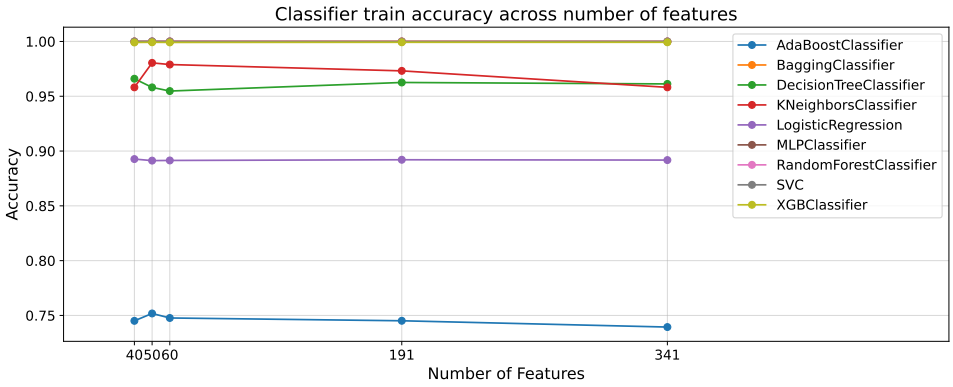
The evaluation of the models is based on the following metrics:

- Accuracy:
  - Training Accuracy: Measures the model's performance on the training data, indicating its ability to learn patterns from the given samples.
  - Validation Accuracy: Assesses the model's generalizability to unseen validation data during the feature selection and hyperparameter tuning stages.
  - Test Accuracy: Evaluates the model's final performance on the independent test set, representing its real-world applicability.
- Confusion Matrix: Provides detailed insights into the classification results by showing true positives, false positives, true negatives, and false negatives for each activity class.
- Classification Report: Includes precision, recall, and F1-score for each activity, offering a more nuanced understanding of the model's performance.
- Comparison Metrics: A summary of train, validation, and test accuracies across different feature configurations and classifiers, helping to identify the best-performing combination.

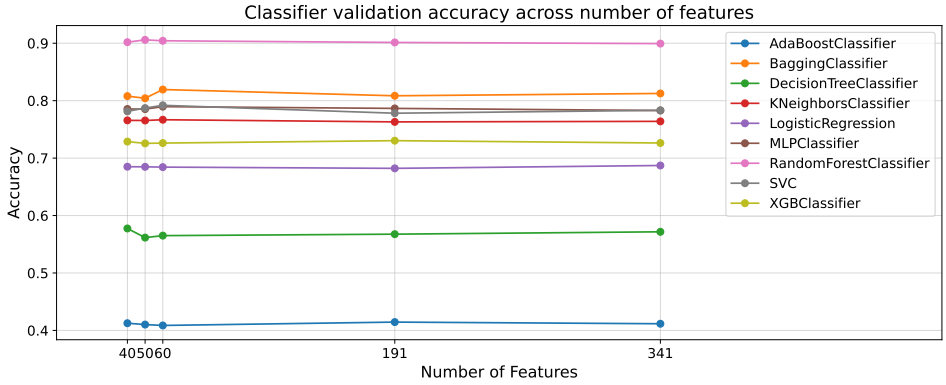
8 RESULTS

8.1 Validation set

The validation results provide a comprehensive overview of how different classifiers perform across various feature configurations, emphasizing their generalization capabilities and sensitivity to feature reduction. The analysis demonstrates notable variations among classifiers and highlights the impact of feature selection on model performance.



(a) Train accuracy



(b) Validation accuracy

Fig. 7. Classifier accuracy across feature sets.

Across the classifiers, Figure 7 illustrates that validation accuracy exhibits consistent patterns, with most models maintaining stable performance as the number of features is reduced. This stability underscores the effectiveness of the feature selection process, which successfully retained the most informative features while discarding redundant or noisy ones. In some cases, such as with RandomForestClassifier and MLPClassifier, reducing features improved validation accuracy, reflecting the elimination of unnecessary complexity in the data. Conversely, models like AdaBoostClassifier showed little improvement, suggesting limited adaptability to the streamlined feature set. Below is the details analysis and the detailed information is shown in Table A.2.

- AdaBoost classifier: Validation accuracy for AdaBoostClassifier hovers around 40%, regardless of the feature count. This relatively low performance indicates that the model struggles with the given dataset, possibly due to its sensitivity to noisy or redundant features. Despite feature reduction, AdaBoost did not show noticeable improvements, suggesting that further hyperparameter tuning or a different base estimator may be needed.
- Bagging classifier: The BaggingClassifier demonstrates robust validation accuracy, achieving around 80% across all feature configurations. The ensemble nature of the model ensures its resilience to feature reduction, although the marginal gains from feature selection suggest that the full feature set already captures the most critical information.
- Decision Tree classifier: Validation accuracy for the DecisionTreeClassifier peaks when using 40 features, highlighting the benefits of aggressive feature reduction. The model's performance, however, remains lower than ensemble methods, likely due to its susceptibility to overfitting when trained on a larger feature set.
- k-NN classifier: The validation accuracy for KNeighborsClassifier remains consistent at around 76%, indicating that the model benefits less from feature selection. This stability reflects the model's reliance on distance-based metrics, which are relatively insensitive to feature reduction when informative features are retained.
- Logistic Regression: LogisticRegression achieves moderate validation accuracy, approximately 68%, across all feature configurations. The linear nature of the model ensures that its performance remains unaffected by feature reduction, as it primarily benefits from datasets with linearly separable features.
- MLP classifier: The MLPClassifier demonstrates the highest validation accuracy, exceeding 78% across feature configurations. The model slightly benefits from feature reduction, with its peak performance observed when using 60 features. This improvement reflects the neural network's ability to capture complex patterns more effectively with a streamlined feature set.
- Random Forest classifier: With validation accuracy consistently around 90%, the RandomForestClassifier emerges as one of the best-performing models. Its robustness to feature reduction and ability to handle high-dimensional data make it a standout choice for this task. The slight improvement in performance with fewer features highlights its efficiency in utilizing relevant information.
- SVC: Validation accuracy for the SVC fluctuates slightly, peaking at around 79% with 60 features. While the model performs well, it is outperformed by ensemble methods such as RandomForest and MLPClassifier. The marginal improvement with feature reduction suggests that SVC benefits from reduced data complexity.
- XGB classifier: The XGBClassifier achieves stable validation accuracy of approximately 73%, showing little sensitivity to feature reduction. This stability reflects the model's inherent ability to leverage all available features effectively, albeit without significant gains from the streamlined feature set.

Ensemble methods such as Random forest and Bagging demonstrate superior performance, achieving the highest validation accuracy across feature configurations. The feature selection process significantly improved or maintained accuracy for most models, particularly those susceptible to overfitting, like Decision tree model. Simpler models such as Logistic regression model and k-NN classifier perform moderately well, while AdaBoost model struggles to adapt to the dataset. Overall, the results underscore the value of feature selection in enhancing generalization performance and the strength of ensemble methods in handling complex, high-dimensional data.

Outlier detection was omitted in this project due to the complexity of handling time series data and the risk of further data loss in an already sparse dataset. Identifying outliers requires advanced techniques beyond the scope of this work. Instead, robust algorithms like Random Forest and preprocessing steps like feature scaling were used to mitigate the

impact of potential outliers. The high validation scores achieved by Random Forest and Bagging classifiers demonstrate the effectiveness of these strategies in managing noise and variability, ensuring competitive model performance without explicit outlier removal.

8.2 Test set

The test results regarding Figure 8 or Table A.2 (for more details) demonstrate clear differences in performance across classifiers and feature subsets. **Random Forest classifier consistently achieved the highest test accuracy, especially when using smaller feature subsets of 40 and 50 features, peaking at approximately 0.86**. This result underscores its robustness to feature selection and its ability to generalize effectively to unseen data. Similarly, Bagging and MLP classifier showed strong and consistent performance, with Bagging model reaching around 0.80 and MLP classifier achieving approximately 0.79 across all feature subsets. These ensemble and neural network methods appear to handle both large and reduced feature sets well, showcasing their flexibility and adaptability.

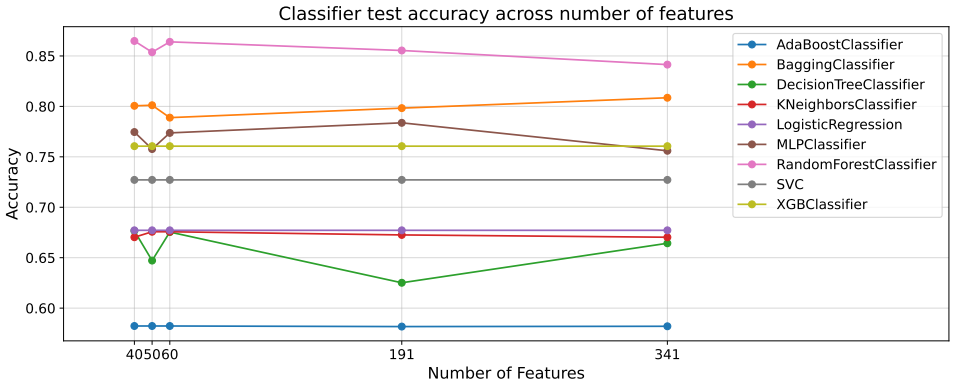


Fig. 8. Classifier test accuracy across feature sets.

In contrast, AdaBoost displayed the weakest performance, maintaining a constant test accuracy of approximately 0.60 across all feature subsets. This indicates its potential limitations in handling the complexity of the dataset. Decision Tree classifier and SVC also demonstrated sensitivity to feature reduction. Decision Tree classifier's accuracy dropped slightly with fewer features, falling to approximately 0.65, likely due to the loss of critical features in the smaller subsets. SVC exhibited moderate performance, peaking at around 0.73 with larger feature sets but declining as the number of features decreased.

Traditional models like Logistic Regression and k-NN delivered moderate results, with Logistic Regression achieving a test accuracy of around 0.68 across all feature subsets. While these models performed adequately, their results highlight their limitations in capturing complex patterns in the dataset. k-NN showed slight improvements with fewer features, reaching up to 0.77, likely benefiting from reduced noise in the dataset.

Ensemble methods, including Random Forest, Bagging, and XGBoost, outperformed most other models, highlighting their robustness to feature redundancy and their ability to leverage diverse subsets of features effectively. Neural network-based models like MLP model exhibited strong and consistent results, showcasing their adaptability across varying feature dimensions. On the other hand, simpler methods like Logistic Regression and Decision Tree were more sensitive to feature selection, with their performance heavily influenced by the choice of features.

In conclusion, the test results emphasize the importance of choosing classifiers suited to the problem at hand. Random Forest emerged as the most effective model, particularly when combined with feature selection techniques to reduce redundancy and complexity. The results highlight the strength of ensemble methods and neural networks in handling complex datasets, while traditional classifiers and single-tree models faced challenges in adapting to reduced feature spaces. These findings provide valuable insights into the trade-offs between model complexity, feature selection, and performance, guiding future work on similar datasets. Therefore, I continue analyzing the confusion matrix for the Random Forest model with the 40-best-feature set, the matrix shown in Figure 9. Other confusion matrice for other classifiers are included in Figure A.1.

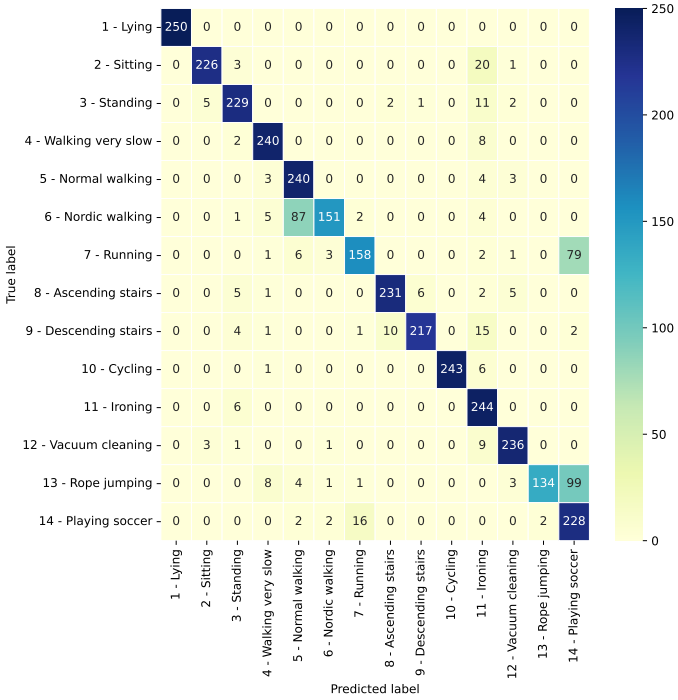


Fig. 9. Confusion Matrix for the Test Set using Random Forest classifier, $n\_features = 40$.

The confusion matrix for the Random Forest classifier on the test set reveals its overall strong performance across various activities, with certain limitations. Many activities, such as Lying (1), Sitting (2), Standing (3), Walking very slow (4), Normal walking (5), Cycling (10), Ironing (11), and Playing soccer (14), exhibit high classification accuracy, reflecting the model's robustness in identifying these activities. For example, static activities like Lying are perfectly classified, likely due to distinct patterns in the corresponding sensor data. Similarly, dynamic activities with clear motion patterns, such as Cycling and Playing soccer, are accurately identified with minimal confusion.

However, the classifier struggles with activities that share similar movement characteristics. For instance, Nordic walking (6) is often misclassified as Normal walking (5) or Running (7), likely due to overlapping acceleration and gyroscope patterns. Additionally, Ascending stairs (8) and Descending stairs (9) show notable confusion, as these

activities involve similar sensor readings due to their physical similarity. Similarly, Rope jumping (13) frequently gets misclassified as Playing soccer (14), which may stem from shared dynamic features in their movement profiles.

The confusion matrix highlights areas for potential improvement. Incorporating advanced feature engineering techniques, such as extracting movement cadence or intensity through spectral analysis, may help distinguish between closely related activities. Additionally, increasing the training sample size for underrepresented classes like Rope jumping and Nordic walking could improve the model's generalization. Furthermore, exploring more advanced classifiers, such as deep learning models, may enhance the ability to capture subtle distinctions in activity patterns.

In summary, while the Random Forest classifier demonstrates strong overall performance, particularly for static and well-defined dynamic activities, its challenges in distinguishing closely related activities suggest room for improvement through feature engineering, expanded datasets, and advanced modeling approaches.

9 CONCLUSION AND IMPROVEMENT IDEAS

9.1 Conclusion

This project explored activity classification using sensor data through comprehensive feature selection, robust modeling, and rigorous evaluation. The Random Forest classifier emerged as the top-performing model, demonstrating high accuracy, particularly on well-defined activities. The feature selection process, including filter-based methods and Recursive Feature Elimination, effectively reduced dimensionality while maintaining model performance. However, challenges persisted in distinguishing between similar activities, such as stair climbing and walking, highlighting areas for further improvement. Despite these limitations, the project underscores the importance of robust preprocessing, effective feature selection, and model evaluation in addressing classification challenges in sensor-based data.

9.2 Improvement idea

- Enhanced Feature Engineering: Introduce advanced features, such as frequency-domain characteristics or derived metrics like cadence and energy consumption, to better capture activity nuances.
- Advanced Modeling Approaches: Explore deep learning architectures like Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs) to capture complex temporal dependencies and patterns in time-series data.
- Data Augmentation: Generate synthetic data for underrepresented classes to improve model generalization and address imbalanced activity distributions.
- Cross-Subject Generalization: Develop strategies to improve model robustness across subjects, such as domain adaptation or transfer learning techniques.
- Outlier Detection: Implement time-series-specific outlier detection methods to refine the dataset further and reduce noise.
- Hyperparameter Optimization: Employ automated tools like Bayesian optimization for more exhaustive hyperparameter tuning to enhance model performance.

These improvements would enhance the classification accuracy, robustness, and scalability of activity recognition systems, paving the way for broader applications in health monitoring, fitness tracking, and human activity analysis.

REFERENCES

[1] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Activity recognition using cell phone accelerometers," in *Proceedings of the Fourth International Workshop on Knowledge Discovery from Sensor Data (at KDD-10)*. Washington, DC, USA: ACM, 2010, pp. 10–18.

[2] S. learn developers, "Randomforestclassifier," *Scikit-learn 1.5 Documentation*, Online, 2024, available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.

[3] ——, "Kneighborsclassifier," *Scikit-learn 1.5 Documentation*, Online, 2024, available: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html.

[4] ——, "Decisiontreeclassifier," *Scikit-learn 1.5 Documentation*, Online, 2024, available: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html.

[5] ——, "Logisticregression," *Scikit-learn 1.5 Documentation*, Online, 2024, available: https://scikit-learn.org/1.5/modules/generated/sklearn.linear_model.LogisticRegression.html.

[6] ——, "Svc," *Scikit-learn 1.5 Documentation*, Online, 2024, available: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html.

[7] ——, "Adaboostclassifier," *Scikit-learn 1.5 Documentation*, Online, 2024, available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html.

[8] ——, "Baggingclassifier," *Scikit-learn 1.5 Documentation*, Online, 2024, available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html.

[9] ——, "Mlpclassifier," *Scikit-learn 1.5 Documentation*, Online, 2024, available: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html.

[10] X. developers, "Xgbclassifier," *XGBoost Documentation*, Online, 2024, available: https://xgboost.readthedocs.io/en/stable/.

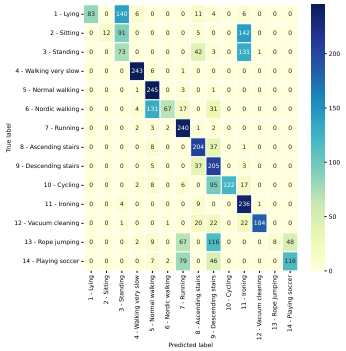
A APPENDIX

Table A.1. Overview of models and hyperparameters

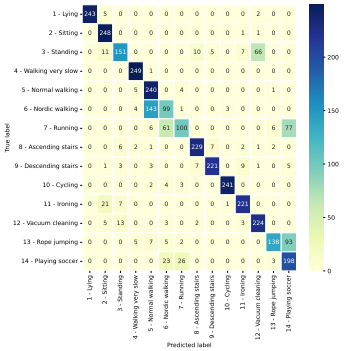
Classifier Name	Module	Parameter	Values
Random Forest	`sklearn.ensemble.RandomForestClassifier` [2]	`n_estimators`	[100, 200, 300]
		`max_depth`	[None, 10, 20, 30]
		`min_samples_split`	[2, 5, 10]
		`min_samples_leaf`	[1, 2, 4]
		`max_features`	['sqrt', 'log2']
		`bootstrap`	[True, False]
k-Nearest Neighbors	`sklearn.neighbors.KNeighborsClassifier` [3]	`n_neighbors`	[5, 10, 15, 20, 50, 75, 100, 200]
		`weights`	['uniform', 'distance']
Decision Tree	`sklearn.tree.DecisionTreeClassifier` [4]	`max_depth`	[5, 10, 15, 20, 25, 30, 35, 40]
Logistic Regression	`sklearn.linear_model.LogisticRegression` [5]	`C`	[0.001, 0.01, 0.1, 1, 10, 100]
		`penalty`	['l1', 'l2']
		`solver`	['liblinear']
Support Vector Machines	`sklearn.svm.SVC` [6]	`C`	[0.1, 1, 10, 100]
		`kernel`	['linear', 'poly', 'rbf', 'sigmoid']
		`degree`	[2, 3, 4]
		`gamma`	['scale', 'auto']
AdaBoost	`sklearn.ensemble.AdaBoostClassifier` [7]	`n_estimators`	[10, 50, 100, 200, 300, 500, 1000]
		`learning_rate`	[0.01, 0.1, 0.2, 0.5, 1]
Bagging	`sklearn.ensemble.BaggingClassifier` [8]	`n_estimators`	[10, 50, 100, 200, 300, 500, 1000]
		`bootstrap`	[True, False]
		`bootstrap_features`	[True, False]
Multilayer Perceptron	`sklearn.neural_network.MLPClassifier` [9]	`hidden_layer_sizes`	[(100,), (100, 100), (100, 100, 100)]
		`activation`	['logistic', 'relu']
		`alpha`	[0.0001, 0.001, 0.01]
		`learning_rate`	['constant', 'adaptive']
		`max_iter`	[200, 300, 400]
XGBoost	`xgboost.XGBClassifier` [10]	`n_estimators`	[100, 200, 250]
		`max_depth`	[3, 5, 7, 9]
		`learning_rate`	[0.01, 0.1, 0.2]

Table A.2. Accuracy across different classifiers and feature sets

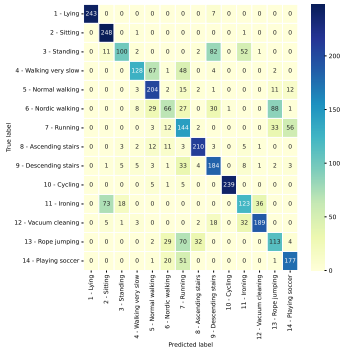
Classifier	No of Features	Train Acc.	Val. Acc.	Test Acc.	Parameters
AdaBoostClassifier	341	0.74	0.41	0.58	'learning_rate': 0.5, 'n_estimators': 1000
	191	0.75	0.41	0.58	'learning_rate': 0.5, 'n_estimators': 1000
	60	0.75	0.41	0.58	'learning_rate': 0.5, 'n_estimators': 1000
	50	0.75	0.41	0.58	'learning_rate': 0.5, 'n_estimators': 1000
	40	0.75	0.41	0.58	'learning_rate': 0.5, 'n_estimators': 1000
BaggingClassifier	341	1	0.81	0.81	'bootstrap': True, 'bootstrap_features': True, 'n_estimators': 500
	191	1	0.81	0.8	'bootstrap': True, 'bootstrap_features': True, 'n_estimators': 500
	60	1	0.82	0.79	'bootstrap': True, 'bootstrap_features': True, 'n_estimators': 500
	50	1	0.8	0.8	'bootstrap': True, 'bootstrap_features': True, 'n_estimators': 500
	40	1	0.81	0.8	'bootstrap': True, 'bootstrap_features': True, 'n_estimators': 1000
DecisionTreeClassifier	341	0.96	0.57	0.66	'max_depth': 40
	191	0.96	0.57	0.63	'max_depth': 35
	60	0.95	0.57	0.68	'max_depth': 25
	50	0.96	0.56	0.65	'max_depth': 35
	40	0.97	0.58	0.68	'max_depth': 35
KNeighborsClassifier	341	0.96	0.76	0.67	'n_jobs': -1, 'n_neighbors': 5, 'weights': 'uniform'
	191	0.97	0.76	0.67	'n_jobs': -1, 'n_neighbors': 10, 'weights': 'distance'
	60	0.98	0.77	0.68	'n_jobs': -1, 'n_neighbors': 5, 'weights': 'distance'
	50	0.98	0.77	0.68	'n_jobs': -1, 'n_neighbors': 5, 'weights': 'distance'
	40	0.96	0.77	0.67	'n_jobs': -1, 'n_neighbors': 5, 'weights': 'uniform'
LogisticRegression	341	0.89	0.69	0.68	'C': 100, 'penalty': 'l2', 'solver': 'liblinear'
	191	0.89	0.68	0.68	'C': 100, 'penalty': 'l2', 'solver': 'liblinear'
	60	0.89	0.68	0.68	'C': 100, 'penalty': 'l2', 'solver': 'liblinear'
	50	0.89	0.68	0.68	'C': 100, 'penalty': 'l2', 'solver': 'liblinear'
	40	0.89	0.69	0.68	'C': 100, 'penalty': 'l2', 'solver': 'liblinear'
MLPClassifier	341	1	0.78	0.76	'activation': 'logistic', 'alpha': 0.0001, 'hidden_layer_sizes': (100, 100), 'learning_rate': 'adaptive', 'max_iter': 400, 'solver': 'adam'
	191	1	0.79	0.78	'activation': 'logistic', 'alpha': 0.0001, 'hidden_layer_sizes': (100, 100), 'learning_rate': 'adaptive', 'max_iter': 400, 'solver': 'adam'
	60	1	0.79	0.77	'activation': 'logistic', 'alpha': 0.0001, 'hidden_layer_sizes': (100, 100), 'learning_rate': 'constant', 'max_iter': 300, 'solver': 'adam'
	50	1	0.79	0.76	'activation': 'logistic', 'alpha': 0.0001, 'hidden_layer_sizes': (100, 100), 'learning_rate': 'adaptive', 'max_iter': 400, 'solver': 'adam'
	40	1	0.79	0.77	'activation': 'logistic', 'alpha': 0.0001, 'hidden_layer_sizes': (100, 100), 'learning_rate': 'constant', 'max_iter': 400, 'solver': 'adam'
RandomForestClassifier	341	1	0.9	0.84	'bootstrap': True, 'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 300
	191	1	0.9	0.86	'bootstrap': True, 'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators': 100
	60	1	0.9	0.86	'bootstrap': True, 'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 300
	50	1	0.91	0.85	'bootstrap': True, 'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 10, 'n_estimators': 100
	40	1	0.9	0.86	'bootstrap': True, 'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators': 300
SVC	341	1	0.78	0.73	'C': 0.1, 'degree': 2, 'gamma': 'scale', 'kernel': 'linear'
	191	1	0.78	0.73	'C': 0.1, 'degree': 2, 'gamma': 'scale', 'kernel': 'linear'
	60	1	0.79	0.73	'C': 0.1, 'degree': 2, 'gamma': 'scale', 'kernel': 'linear'
	50	1	0.79	0.73	'C': 0.1, 'degree': 2, 'gamma': 'scale', 'kernel': 'linear'
	40	1	0.78	0.73	'C': 0.1, 'degree': 2, 'gamma': 'scale', 'kernel': 'linear'
XGBClassifier	341	1	0.73	0.76	'learning_rate': 0.2, 'max_depth': 9, 'n_estimators': 250
	191	1	0.73	0.76	'learning_rate': 0.2, 'max_depth': 9, 'n_estimators': 250
	60	1	0.73	0.76	'learning_rate': 0.2, 'max_depth': 9, 'n_estimators': 250
	50	1	0.73	0.76	'learning_rate': 0.2, 'max_depth': 9, 'n_estimators': 250
	40	1	0.73	0.76	'learning_rate': 0.2, 'max_depth': 9, 'n_estimators': 250



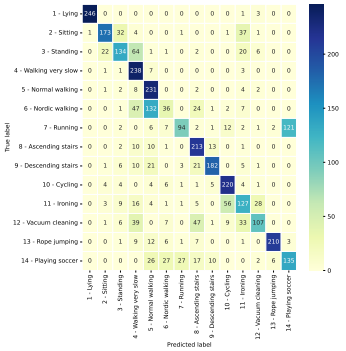
(a) AdaBoost



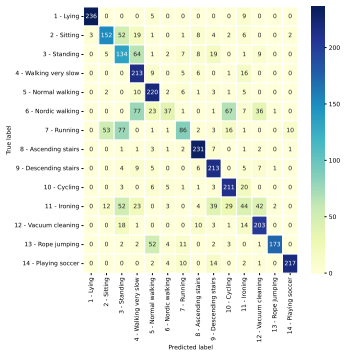
(b) Bagging



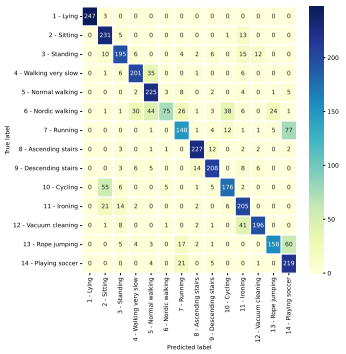
(c) Decision Tree



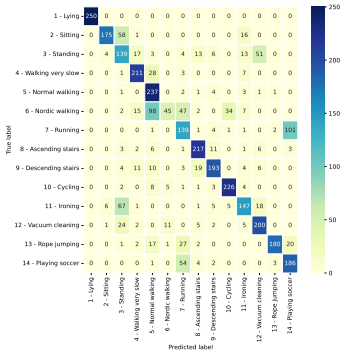
(d) k-NN



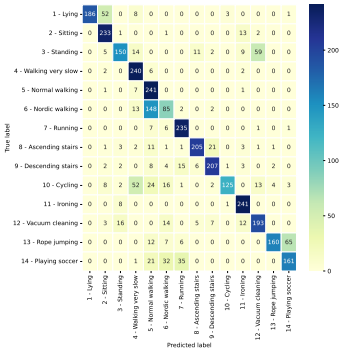
(e) Logistic Regression



(f) MLP



(g) SVC



(h) XGBoost

Fig. A.1. Confusion Matrix for the Test Set using different classifiers, $n\_features = 40$.