2024 5th International Conference on Emerging Trends in Electrical, Electronic and Communications Engineering (ELECOM) | 979-8-3315-3247-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/ELECOM63163.2024.10892156

Human Activity Recognition using WISDM: Exploring Class Balancing and ML Techniques

Maria Hanif*, Rizwan Ahmad*, Shams Qazi*, Waqas Ahmed†, Adnan Kiani‡ and Muhammad Mahtab Alam§
*National University of Sciences and Technology (NUST), Islamabad, Pakistan
†Pakistan Institute of Engineering and Applied Sciences (PIEAS), Islamabad, Pakistan
‡Nottingham Trent University (NTU), England
§Thomas Johann Seebeck Department of Electronics, Tallinn University of Technology, 12616 Tallinn, Estonia

***Abstract*—Human Activity Recognition (HAR) is vital in activity sensing research, particularly with wearable sensors. This study addresses dataset imbalance in the WISDM dataset by employing three class balancing techniques: SMOTE, LoRAS, and ProWRA, integrated with five machine learning models, including Decision Tree, Random Forest, K-Nearest Neighbors (K-NN), CatBoost, and XGBoost. Our experimental results demonstrate that LoRAS consistently outperformed the other techniques, achieving a remarkable accuracy of 98.73% with the Random Forest model, significantly higher than SMOTE (97%) and ProWRA (97.12%). Furthermore, LoRAS also excelled in other evaluation metrics, including recall, precision, and F1-score, reinforcing its effectiveness in enhancing model performance for HAR. In contrast, SMOTE and ProWRA exhibited lower performance, indicating that LoRAS is the most effective technique for mitigating class imbalance and improving the accuracy of machine learning models in HAR applications.**

***Index Terms*—Human Activity Recognition (HAR), LoRAS, Smote, ProWRA.**

I. Introduction

Human Activity Recognition (HAR) analyzes motion data to monitor human movements, driving advancements in healthcare, smart homes, and gait analysis over the past decade [1], [2]. HAR methodologies primarily include video-based and sensor-based approaches. Human activity can be captured as temporal signals using sensors like accelerometers, gyroscopes, and magnetometers, offering privacy and freedom from environmental constraints [3].

With the rise of smartphones and smartwatches, wearable sensor-based health monitoring systems are rapidly developing due to their convenience, cost-effectiveness, and practicality compared to image-based methods [4]–[6]. This technology is also applicable in human-computer interaction, gaming, robotics, and sports. Numerous studies have utilized activity recognition for applications in authentication, medical checks, elderly care, and security using wearable and smartphone systems. HAR research is extensive and varies by sensing modality, with a focus on smartphone sensor data due to its accessibility, cost-effectiveness, and lack of need for specialized setups.

As the world population grows, so do health risks. HAR effectively detects various physical conditions using traditional machine learning methods like K-nearest neighbors and support vector machines [7], [8]. While these methods can automatically recognize simple activities and gestures, their reliance on manual feature extraction limits their performance in capturing comprehensive information about all possible human movements.

The main contributions of this paper include addressing the critical issue of class imbalance in the dataset, which affects the accuracy of human activity recognition models.
- To overcome this challenge, we employed three class balancing techniques: SMOTE, LoRAS, and ProWRA.
- Our comprehensive evaluation of these methods across multiple machine learning models highlights their potential in enhancing model performance. This study emphasizes the importance of addressing class imbalance for effective human activity recognition.

The article is structured as follows: Section II discusses the proposed methodology, covering class balancing techniques and the machine learning models used. Section III presents the results.

II. Proposed Methodology

This section presents our methodology for human activity recognition, using the WISDM dataset as shown in Fig 1. Initial data exploration revealed challenges including class imbalance, datatype inconsistencies, and missing values. These issues were addressed through preprocessing techniques. Subsequently, machine learning algorithms was applied to accurately classify diverse human activities.

A. Dataset

This dataset [9] comprises accelerometer data collected from 29 volunteers who carried Android phones in their front pants pockets. Activities include stationary (sitting, standing) and dynamic (walking, jogging, upstairs, downstairs), with tri-axial accelerometer readings taken every 50 ms, capturing movement along x, y, and z axes. With over one million activity records, it includes user numbers, activity labels, time periods, and corresponding acceleration values. Details of our dataset are as follows:

TABLE I
Description of WISDM Dataset

Total Samples	1,098,207
Total Attributes	6
Label 0 (Walking)	424400
Label 1 (Jogging)	342177
Label 2 (Sitting)	59939
Label 3 (Standing)	48395
Label 4 (Upstairs)	122869
Label 5 (Downstairs)	100427

B. Data Preprocessing

Preprocessing involves cleaning data for compatibility with ML models, addressing null values through removal

979-8-3315-3247-5/24/$31.00 ©2024 IEEE

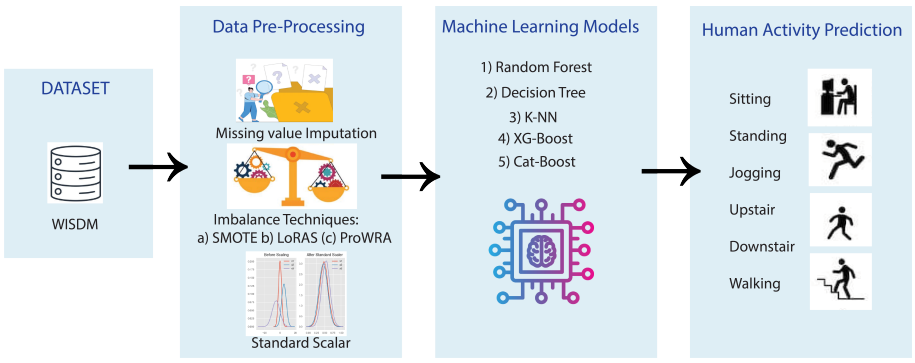

Fig. 1. Proposed Architecture

or imputation, ensuring suitable data types, and optimizing computational complexity. Our approach on the WISDM dataset focused on resolving these issues to ensure consistency and computational efficiency for subsequent analyses.

1) Imputation of Missing Values: The dataset exhibited missing values, primarily in features such as user, time, and spatial coordinates (x, y, z), as illustrated in Fig 2. Imputation was chosen over removal to preserve dataset integrity and solution accuracy. Median imputation was employed, replacing missing values with the median of available samples, maintaining dataset size while reducing unevenness in the data. Additionally, Fig. 3 illustrates the

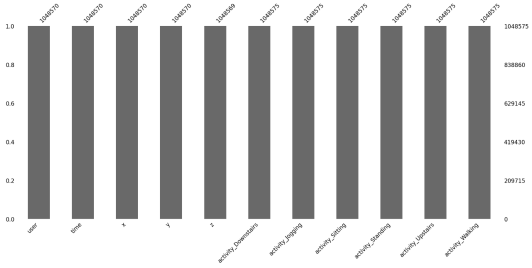

Fig. 2. Missing Value Matrix

outcomes following the median model to address missing values.

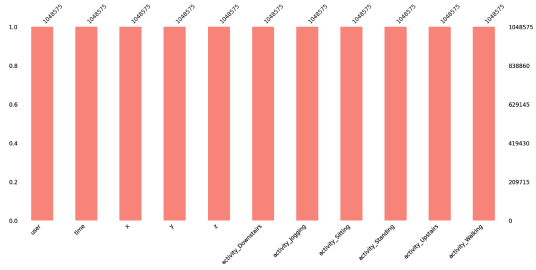

Fig. 3. Results after Median Imputation

2) Imbalanced Dataset: Imbalance in a dataset, as seen in WISDM, results in unequal class distribution, leading to biased classification. In this dataset, class distribution percentages reveal a significant imbalance across classes. To address this issue, we employed three techniques: SMOTE, LoRAS, and ProWRA. The class distribution percentages highlight this disparity: Class 5 comprises the largest portion at 38.793%, followed by Class 1 with 31.087%. Conversely, Class 3 represents only 4.248% of the dataset, while Class 2 accounts for 5.233%. To address this imbalance, we employed three techniques to rebalance the dataset by generating synthetic samples for minority classes, thereby reducing the dominance of majority classes and improving classification accuracy. We have applied three class balancing techniques:

- *Synthetic Minority Over-sampling Technique (SMOTE)*
  SMOTE is a method used to address class imbalances in datasets. It works by generating synthetic samples from the minority class by interpolating between existing minority class instances and their neighbors [10]. By introducing synthetic samples based on a sampling magnification factor, N SMOTE balances class distribution and enhances classifier robustness and generalization on imbalanced datasets using the interpolation formula shown in Equation (1) and illustrated in Fig 4.

$$S_i = X + \text{rand}(0,1) \times (y_i - X) \tag{1}$$

  X refers to a data sample within the minority class samples, $rand(0,1)$ denotes a random number chosen uniformly from the interval (0,1), y_i represents the i_{th} neighboring class, S_i signifies the interpolated sample. Algorithm 1 shows SMOTE steps.

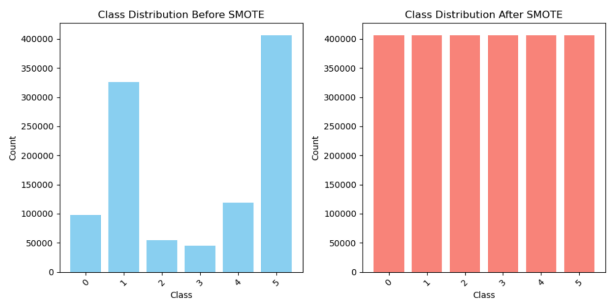

Fig. 4. Class Balancing using SMOTE

- *Localized Randomized Affine Shadow sampling (LoRAS)*
  LoRAS, an oversampling technique, synthesizes minority class samples by introducing Gaussian noise

Algorithm 1 Synthetic Minority Oversampling Technique (SMOTE)

Input: Training data
Training set: Tr
Nearest neighbor: p
Nearest neighbors for data cleaning: k
Output: Augmented training set New_Tr after applying SMOTE
Start
for $i = 1$ to N **do**
    Generate new samples from the minority class and add them to New_Tr
end for
End

within small regions surrounding these instances, followed by constructing final synthetic data points through convex combinations of multiple noisy data points [11]. We applied LoRAS, illustrated in Fig 5, to mitigate class imbalance in our dataset. Algorithm

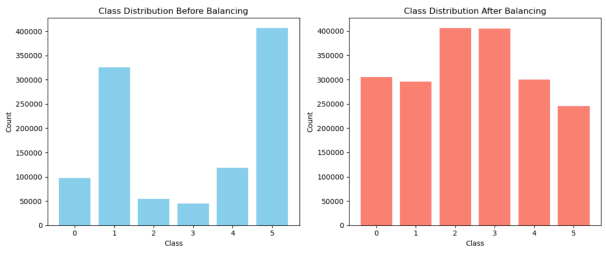

Fig. 5. Class Balancing using LoRAS

2 shows steps of LoRAS.

Algorithm 2 Localized Randomized Affine Shadow Sampling

Inputs:
Majority class: C_{maj}
Minority class: C_{min}
Start
Initialize an empty list named *loras_set*
for each data point p in C_{min} **do**
    Find k nearest neighbors of p
    Initialize *neighborhood_shadow_sample* as empty
    Generate shadow samples S_p from C_{min}
    repeat
        Add a shadow sample to *neighborhood_shadow_sample*
    until desired number of points is reached
end for
Return *loras_set*
End

- *Proximity Weighted Random Affine Shadowsampling (ProWRAs)*
  ProWRAS utilizes synthetic sampling as a key component of its oversampling approach. It begins by partitioning the minority class and forming clusters comprising its members [12]. Algorithm 3 shows steps of ProWRAS. ProWRAS assigns weights to clusters based on proximity to the majority class,

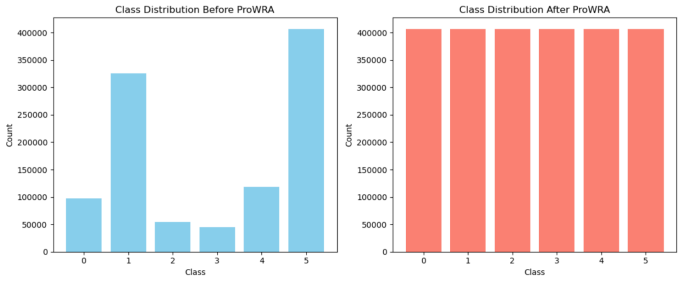

Fig. 6. Class Balancing using ProWRA

normalizes them, and determines sample generation, effectively synthesizing new samples from the largest minority class cluster, as depicted in Fig 6.

Algorithm 3 Proximity Weighted Random Affine Shadow Sampling (ProWRAS)

Input: Training data
ProWRAS-Oversampling: (Dataset)
Start
Cluster the dataset
Initialize an empty set for synthetic samples
for each (Cluster, Weight) in Clusters **do**
    Calculate the number of samples to generate: Num_samples $\leftarrow$ num_samples_generate $\times$ Weight
    Add generated samples to synthetic samples set: Synth_samples $\leftarrow$ Synth_samples $\cup$ synth
end for

C. Machine Learning Models

1) Extreme Gradient Boosting (XG-Boost): XGBoost, an enhanced gradient boosting algorithm, sequentially builds decision trees, refining predictions by iteratively correcting errors [13] [14]. It utilizes parallelization between leaf nodes and features to optimize model training. Equation (2) in XGBoost expresses quadratic functions of one variable, facilitating complex pattern recognition and prediction refinement.

$$L^{(t)} = \sum_{i=1}^{N} \left[g_i f t\left(x_i\right) = 1/2 h_i f_t^2\left(x_i\right) \right] + \Omega\left(f_t\right) \qquad (2)$$

Ω define optimization, g is gradient and L define the loss function. In Algorithm 1, $f(x)$ represents the ensemble model comprising a sum of weak learners for k iterations. The base classifiers, denoted by b_k, are combined to form the ensemble model, and the weight of each tree is indicated by w_k.

2) Cat Boost: CatBoost is a gradient-boosting decision tree (GBDT) framework utilizing oblivious trees with fewer parameters. It uniquely supports categorical variables, enhancing accuracy. By sequentially training learners and aggregating their outputs, it iteratively improves accuracy. Given a training set $D\left\{(X_i, Y_i)_{i=1,2,\ldots,n}\right\}$, where $X_i = \left(x_i^1, x_i^2 \ldots, x_i^m\right)$ denotes input features and $Y_i \in \mathbb{R}$ represents labeled values, the algorithm aims to minimize the loss function L expectation by iteratively updating the strong learner F_{k-1} with a new tree t_k is

Algorithm 4 Extreme Gradient Boosting Classifier

Require: Input Data N, Training samples X, Gain G
Ensure: Final ensemble model

Dataset $N = (x_i, y_i), \ldots, (x_n, y_n)$, where $x_i \in X$ and $y_i \in Y\{0, 1\}$
Initialize the initial prediction function $f(x) = Pb_k(x)\ k = 1, 2, \ldots, M$ where M is the number of base learners
Calculate the first-order gradient $g_k = \frac{\partial L(y,f)}{\partial f}$
Determine the optimal split with highest gain
Calculate loss reduction for split
Update the leaf weights w^* based on loss reduction
Define base learner $b(x)$ of weighted trees
Incorporate newly trained tree to ensemble model
return The final ensemble model

explained in equation(3) from a CART decision tree set T.

$$t_k = \operatorname{argmin} EL\left(y, F_{k-1}(x) + t(x)\right) \tag{3}$$

Here, (x, y) are training samples. Following is the pseudocode of CatBoost:

Algorithm 5 CatBoost Algorithm

Require: Input Data N, Training samples X, Loss function L, Number of iterations M, Learning rate η, Regularization parameters λ, Feature indices I, Categorical feature indices C
Ensure: Final prediction function $f(x)$

Initialize $f(x)$ as 0
for $k = 1$ **to** M **do**
  Calculate gradient g_k using L
  for each feature index i in I **do**
    Calculate second-order gradient h_{ik}
    Update tree node weights based on g_k and h_{ik}
  end for
  for each categorical feature index j in C **do**
    Apply ordered boosting for categorical features
  end for
  Apply regularization to prevent overfitting
  Update $f(x)$ using the new weights
end for
return The final prediction function $f(x)$

3) Decision Tree: DT recursively partition the input space based on features to optimize splits at each node for information gain or impurity minimization. Mathematically, it finds optimal splits θ_t at each node t, typically using entropy or Gini impurity, resulting in a tree structure with leaves representing final decisions or predictions.

4) Random Forest: RF, a supervised ML method, leverages the combined predictive power of numerous independent Decision Trees (DTs) within a bagging ensemble framework to enhance performance and robustness. The mathematical formulation and algorithm for Random Forest are outlined in references [16] and [17]. In Equation (4), the index i represents the number of important features calculated for each tree j, while T denotes the total number of trees in the Random Forest.

$$RFf_i = \sum_J normfi_{ij}/sumT \tag{4}$$

Algorithm 6 Decision Tree Construction

Require: Original dataset S

Begin
Function Predict(tree, S)
if tree is a leaf node **then**
  return Prediction of tree
end if
for each instance n in S **do**
  Choose attribute with lowest entropy and highest gain
  Consider attribute with highest gain as root node A
  while A is not a leaf node **do**
    Calculate output on A using n
    Identify correct output A from n
    Make prediction on n based on labeling of A
  end while
end for

Algorithm 7 Random Forest

Require: Training set X_{train} with n instances, F number of features, A number of classes in target class, B number of trees
Ensure: Trained classifier

for $i = 1$ **to** B **do**
  Generate bootstrap samples $X_{\text{train}}[i]$ from the training set X_{train}
  Create a decision tree using a random sample from $X_{\text{train}}[i]$
  for each selected node t **do**
    Randomly select $m \approx \sqrt{F}$ features
    Find the best splitting point from the subset
    Pass down the data using the best splitting point
    Repeat these steps until termination conditions are met
  end for
  Construct the trained classifier
end for

5) K-Nearest Neighbour: KNN, a versatile non-parametric and supervised technique, serves for both classification and regression tasks by identifying the k-nearest data points and determining their group or mean value.

$$d(\mathbf{X}, \mathbf{Y}) = \sqrt{(X_1 - Y_1)^2 + \cdots + (X_n - Y_n)^2} \tag{5}$$

In Equation (5), where X and Y denote the number of features and data points, respectively, the similarity between two data points is computed.

III. Results & Discussion

To evaluate the activity categories of the model's final output, we use the following identification metrics:

1) Accuracy: Accuracy measures the percentage of instances that are correctly classified out of the total number of instances as expressed in equation (6).

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{6}$$

LoRAS class balancing technique results in the highest

Algorithm 8 K-Nearest Neighbor Construction

Require: Training dataset X_i, Testing dataset X_j, Number of neighbors to consider K

1: Begin
2: **Function** Predict(X_i, X_j, K)
3: **for** each data point x_j in X_j **do**
4: Determine distance $D(x_j, x_i)$ for each data point x_i in X_i
5: **end for**
6: Indices for the K smallest distances $D(x_i, x_j)$ are contained in the computed set
7: **for** each data point x_j in X_j **do**
8: Return majority label or average value based on the K nearest neighbors of x_j
9: **end for**

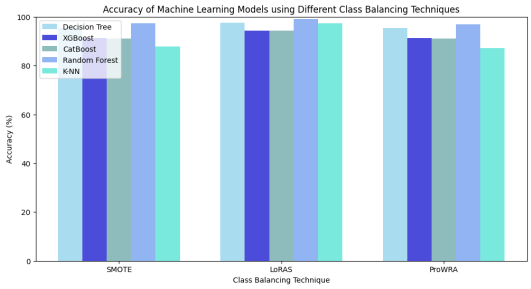

Fig. 7. Accuracy

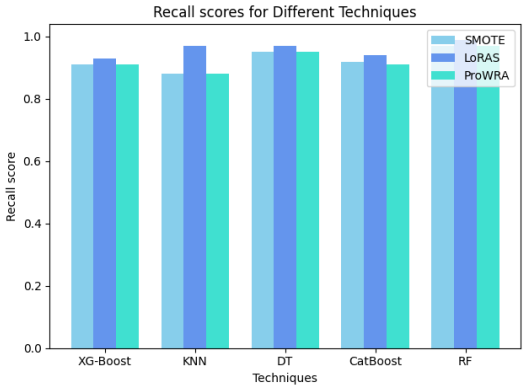

Fig. 8. Recall

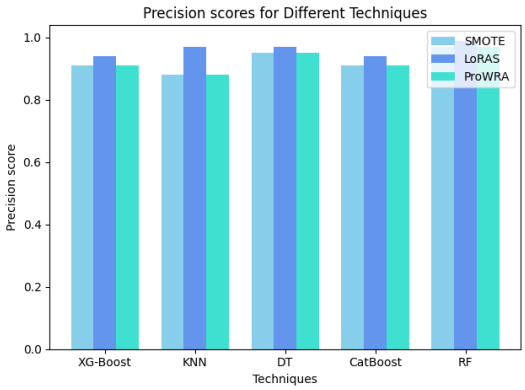

Fig. 9. Precision

accuracy for machine learning models, with Decision Tree and Random Forest achieving better accuracy than other models. SMOTE and ProWRA exhibit similar performance, but ProWRA generally provides slightly better accuracy for Decision Tree and Random Forest compared to SMOTE. XGBoost and K-NN maintain consistent accuracy around 90-95% across all techniques as shown in Fig 7, while CatBoost shows slightly lower accuracy overall, particularly with ProWRA. These results highlight LoRAS as the most effective technique for enhancing model accuracy.

2) Recall: Recall measures the proportion of true positive instances out of all actual positive instances in the dataset.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{7}$$

Fig. 8 shows that LoRAS consistently achieves the highest recall scores compared to SMOTE and ProWRA across all models. Specifically, LoRAS achieves recall scores of 0.93 (CatBoost), 0.97 (DT), 0.97 (KNN), 0.98 (RF), and 0.93 (XGBoost), outperforming both SMOTE and ProWRA in each case.

3) Precision: Precision measures the proportion of true positive instances out of all instances that were predicted to be positive.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{8}$$

Fig. 9 shows precision scores of five machine learning models using SMOTE, LoRAS, and ProWRA. LoRAS consistently achieves the highest precision, often near 1.0, while SMOTE and ProWRA score around 0.85 to 0.9. This highlights LoRAS's superiority in enhancing both model accuracy and precision.

4) F1-Score: The F1-Score is the harmonic mean of precision and recall.

$$F1 - \text{score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{9}$$

Fig. 10 shows the F1-scores of SMOTE, LoRAS, and ProWRA across different ML models. LoRAS consistently achieves the highest F1-scores: 0.93 (CatBoost), 0.97 (DT), 0.97 (KNN), 0.98 (RF), and 0.93 (XGBoost). In comparison, SMOTE and ProWRA both achieve similar and slightly lower F1-scores around 0.91 to 0.97, indicating LoRAS's superior performance in balancing precision and recall.

Table II compares the accuracy of different machine learning models (Decision Tree, XGBoost, CatBoost, Random Forest, KNN) across three balancing techniques: SMOTE, LoRAS, and ProWRA. ProWRA consistently performs well, particularly with XGBoost (94.33%) and Random Forest (92.12%), though SMOTE achieves the highest accuracy with Random Forest (97%) and LoRAS achieves the highest with Random Forest (98.73%). When compared to related work by Essa et al. (2023) and Wang et al. (2023), which report accuracies of 92.51% and 96.90% respectively, the results presented here (with some methods surpassing 97%) show marked improvement in class balancing and model performance.

TABLE II
COMPARISON OF EXISTING MODELS ACCURACY VS. CLASS BALANCING

Accuracy					
Techniques	Decision Tree	XG-Boost	CatBoost	Random Forest	KNN
Smote	95.33%	91.16%	91.5%	97%	87.85%
LoRAS	97.33%	93.5%	93.33%	**98.73%**	96.7%
ProWRA	95.35%	91.33%	91.5%	97.12%	87.75%
Related Work: Essa E et al. (2023) (92.51%) [20], Y Wang et al. (2023) (96.90%) [21]					

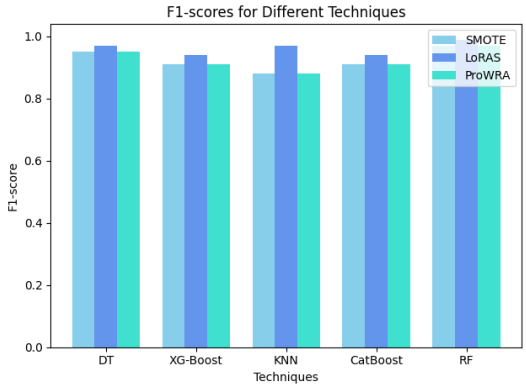

Fig. 10. F1-Score

IV. CONCLUSION

Our study evaluated three class balancing techniques — SMOTE, LoRAS, and ProWRA — on the WISDM dataset for Human Activity Recognition (HAR) using five machine learning models. LoRAS consistently outperformed the others, achieving the highest accuracy (98.73% with Random Forest) and excelling in recall, precision, and F1-score. While SMOTE and ProWRA improved performance over imbalanced data, they fell short of LoRAS's effectiveness. These findings emphasize the importance of selecting optimal balancing techniques to enhance model accuracy in HAR applications. Ultimately, our research highlights the crucial role of advanced class balancing methods in improving machine learning models' precision and generalization in activity recognition tasks.

ACKNOWLEDGMENT

This work has received funding from the Higher Education Commission (HEC), Pakistan under NRPU grant number 15203 and also supported by the North Atlantic Treaty Organization Science for Peace and Security (NATO-SPS) project "PROTECT" under grant number G7699.

REFERENCES

[1] S. Gao, J. Gong, B. Chen, et al., "Use of advanced materials and artificial intelligence in electromyography signal detection and interpretation," Adv. Intell. Syst., vol. 4, no. 10, p. 2200063, 2022.
[2] S. Chen, J. Qi, S. Fan, Z. Qiao, J. C. Yeo, and C. T. Lim, "Flexible Wearable Sensors for Cardiovascular Health Monitoring," Adv. Healthc. Mater., vol. 10, no. 17, pp. 1–23, 2021.
[3] H. Bi, M. Perello-Nieto, R. Santos-Rodriguez, and P. Flach, "Human Activity Recognition Based on Dynamic Active Learning," IEEE J. Biomed. Heal. Informatics, vol. 25, no. 4, pp. 922–934, 2021.
[4] Reem Abdel-Salam, Rana Mostafa AbdElMohsen AbdElMolla, and Mayada Hadhood. Human activity recognition using wearable sensors: Review, challenges, evaluation benchmark. 2021.
[5] Yangyue Zhou and Miaolei Deng. A review of multiple-person abnormal activity recognition. Journal of Image and Graphics, 9(2):55–60, 2021.
[6] Fuqiang Gu, Mu-Huan Chung, Mark Chignell, Shahrokh Valaee, Baoding Zhou, and Xue Liu. A survey on deep learning for human activity recognition. ACM Computing Surveys (CSUR), 54(8):1–34, 2021.
[7] V. Radhika, Ch.Rajendra Prasad, and A. Chakradhar. Smartphone-based human activities recognition system using random forest algorithm,2022.
[8] Saeed Mohsen, Ahmed Elkaseer, and Steffen G. Scholz. Human activity recognition using k-nearest neighbor machine learning algorithm. volume 262 SIST, pages 304 – 313, 2022
[9] R. Kwapisz, G. M. Weiss, and S. A. Moore,"Activity recognition using cell phone accelerometers," ACM SigKDD Explorations Newsletter, vol. 12, no. 2, pp. 74–82, 2011.
[10] A. Arafa, N. El-Fishawy, M. Badawy, and M. Radad, "RN-SMOTE: Reduced noise SMOTE based on DBSCAN for enhancing imbalanced data classification," J. King Saud Univ.-Comput. Inf. Sci., vol. 34, no. 8, pp. 5059–5074, Sep. 2022, doi: 10.1016/j.jksuci.2022.
[11] S. Bej, N. Davtyan, M. Wolfien, M. Nassar, and O. Wolkenhauer, 'LoRAS: An oversampling approach for imbalanced datasets' Mach. Learn., vol. 110, no. 2, pp. 279–301, Feb. 2021, doi: 10.1007/ s10994-020-05913-4.
[12] S. Bej, K. Schulz, P. Srivastava, M. Wolfien, and O. Wolkenhauer,'A multi-schematic classifier-independent oversampling approach forimbalanced datasets' IEEE Access, vol. 9, pp. 123358–123374, 2021, doi: 10.1109/ACCESS.2021.3108450.
[13] D. K. Choi, "Data-driven materials modeling with XGBoost algorithm and statistical inference analysis for prediction of fatigue strength of steels," Int. J. Precis. Eng. Manuf., vol. 20, no. 1, pp. 129–138, Jan. 2019, doi: 10.1007/s12541-019-00048-6
[14] Hanif, M., Shahzad, M. K., Mehmood, V., & Saleem, I. (2023). EPFG: electricity price forecasting with enhanced Gans neural network. IETE Journal of Research, 69(9), 6473-6482.
[15] Decision Tree. Accessed: Apr. 15, 2023. [Online]. Available: https://scikitlearn.org/stable/modules/tree.html
[16] Random Forest. Accessed: Apr. 11, 2023. [Online]. Available: https:// www.datacamp.com/tutorial/random-forests-classifier-python
[17] Random Forest. Accessed: Apr. 9, 2023. [Online]. Available: https:// towardsdatascience.com/random-forests-algorithm-explained-with-areal-life-example-and-some-python-code-affbfa5a942c
[18] Halder, R. K., Uddin, M. N., Uddin, M. A., Aryal, S., & Khraisat, A. (2024). Enhancing K-nearest neighbor algorithm: a comprehensive review and performance analysis of modifications. Journal of Big Data, 11(1), 113.
[19] N. Kumar, N. Narayan Das, D. Gupta, K. Gupta, and J. Bindra, "Efficient automated disease diagnosis using machine learning models," J. Healthcare Eng., vol. 2021, May 2021, Art. no. 9983652, doi: 10.1155/2021/9983652.
[20] E. Essa and I. R. Abdelmaksoud, "Temporal-channel convolution with self-attention network for human activity recognition using wearable sensors," Knowledge-Based Systems, vol. 278, p. 110867, 2023.
[21] Y. Wang, H. Xu, Y. Liu, M. Wang, Y. Wang, Y. Yang, S. Zhou, J. Zeng, J. Xu, S. Li et al., "A novel deep multifeature extraction framework based on attention mechanism using wearable sensor data for human activity recognition," IEEE Sensors Journal, vol. 23, no. 7, pp. 7188– 7198, 2023.