"Multiclass Classification and Activity-Based Clustering on Fitness Tracking Data Using Machine Learning Algorithms"

MD TABISH SHABBIR
Faculty of Engineering, Environment and Computing, Coventry University
MSc Data Science
Coventry, United Kingdom
shabbirm3@uni.coventry.ac.uk

***Abstract* — This report explores the use of machine learning techniques to classify individual fitness levels based on activity and health-related features using the FitLife360 dataset. Three classification algorithms—Logistic Regression, Random Forest, and XGBoost—were applied and compared using evaluation metrics such as accuracy, F1-score, and multiclass ROC AUC. Additionally, clustering was performed using K-Means to identify behavioral patterns in daily physical activity. Data preprocessing included encoding categorical variables, handling missing values, and scaling numerical features. All models were implemented using Python and evaluated for performance, interpretability, and practical relevance.**

Keywords — machine learning; classification; clustering; fitness prediction;data preprocessing; Python; logistic regression; random forest; XGBoost; K-means

I. INTRODUCTION

In recent years, there has been a significant rise in the use of wearable and smart fitness devices that continuously track health and lifestyle metrics such as heart rate, step count, sleep hours, and calorie expenditure. These advancements have led to the accumulation of large-scale health data, providing opportunities for applying machine learning to predict individual fitness levels, personalize recommendations, and monitor wellness more efficiently.

Fitness level classification has become an important application in preventive healthcare, especially as physical activity is closely linked with the risk of chronic diseases and overall quality of life. By analyzing activity and health-related features, machine learning models can identify patterns that correspond to different levels of physical fitness.

Previous studies have shown success in predicting various health-related conditions using machine learning algorithms. For example, Liu et al. (2021) applied ensemble learning techniques for predicting

cardiovascular risks based on biometric indicators, showing strong classification performance [1]. Similarly, Gupta et al. (2020) demonstrated the potential of clustering and classification on health datasets to identify behavior-driven wellness patterns [2].

While most existing research focuses on disease prediction or health risk classification, this study shifts focus toward predicting fitness level categories (low, medium, high) based on a broad range of demographic, activity, and lifestyle metrics collected in a synthetic but realistic health dataset. The aim is to evaluate the performance of multiple supervised models including Logistic Regression, Random Forest, and XGBoost, and to explore unsupervised learning techniques like K-Means for activity behavior clustering.

This paper is structured as follows:

- Section II describes the problem and dataset,
- Section III presents the machine learning methods applied,
- Section IV details the experimental setup including feature selection and pre-processing,
- Section V discusses the results obtained from classification and clustering, and finally,
- Section VI concludes with key observations and future directions.

II. PROBLEM and DATASET DESCRIPTION

In recent years, the intersection of health, fitness, and data science has opened new pathways for predictive modeling and personalized health recommendations. This project aims to classify the fitness level of individuals based on their daily activity and physiological health indicators using machine learning algorithms. The goal is to explore how predictive analytics can interpret lifestyle data and accurately categorize an individual's overall fitness into three levels: Low, Medium, or High.

The dataset used for this analysis is the FitLife360 synthetic dataset, sourced from Kaggle [3]. It simulates real-world health and fitness tracking data from 3,000 participants over a one-year period, with over 680,000 daily activity records. The dataset captures a wide variety of information including physical activity, biometric health measures, sleep patterns, and lifestyle

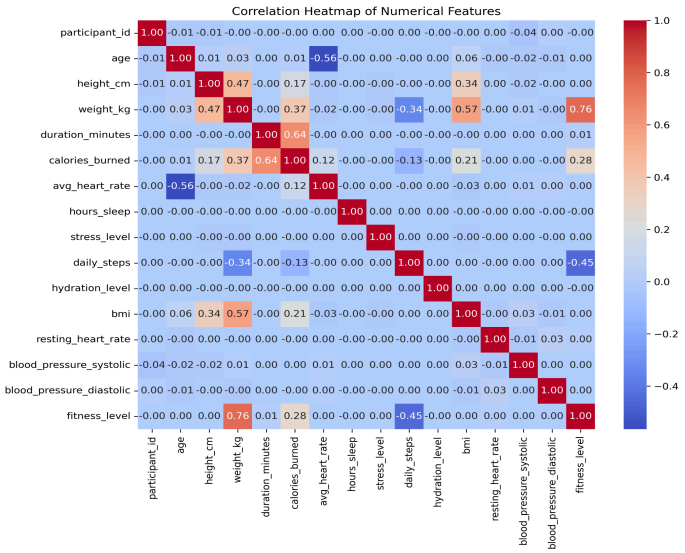

Fig.1. Correlation Heatmap of Numerical Features

habits. This diversity makes it suitable for both classification and clustering tasks. Fig. 1. heatmap illustrates the relationships between continuous numerical features (e.g calories_burned, weight_kg, daily_steps). Strong correlation may help guide feature selection for modeling.

The dataset consists of 22 columns, including demographic, health, and activity features. Key features include:

Demographics: age, gender, height_cm, weight_kg, and bmi

Activity Metrics: activity_type, duration_minutes, intensity, calories_burned, daily_steps

Health Indicators: avg_heart_rate, resting_heart_rate, blood_pressure, hydration_level

Lifestyle Factors: hours_sleep, stress_level, smoking_status

The original fitness_level column was a continuous score. To perform multi-class classification, it was transformed into a categorical variable named fitness_level_category, with values encoded as:

0 = Low
1 = Medium
2 = High

The target variable for classification was set as fitness_level_category.

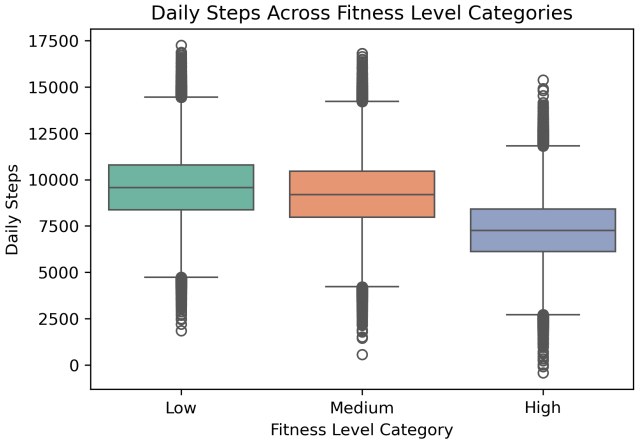

Fig. 2. Boxplot of Fitness level category by Daily Steps

Fig. 2. This plot visualizes the spread and central tendency of daily steps across different fitness levels. It helps validate if *'daily_steps'* is a useful predictor. During preprocessing, categorical features such as activity_type, gender, smoking_status, and intensity were label-encoded or one-hot encoded as needed. Missing values in the column health_condition were dropped or excluded from modeling.

To better understand the dataset before modeling, exploratory data analysis (EDA) was performed. Several graphs were created to illustrate important trends and relationships.

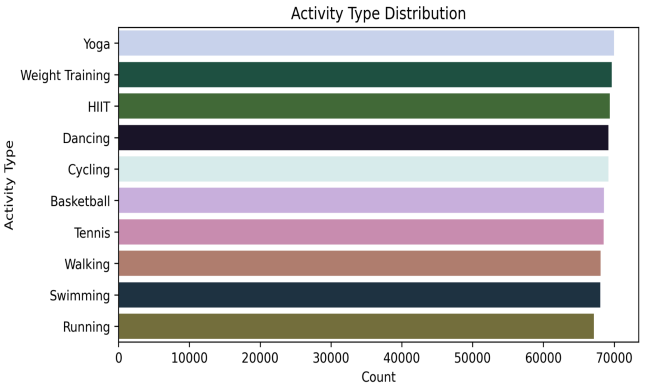

Fig. 3. Distribution of activity types across all participants

Fig. 3. shows the distribution of various activity types in the dataset. Among all recorded activities, Yoga has the highest frequency, followed by Weight Training, HIIT and Dancing.

III. METHODS

This study applies supervised machine learning techniques for multiclass classification of a participant's fitness level category using selected features from the FitLife360 dataset. The target variable fitness_level_category was created by binning the continuous fitness_level column into three classes: Low, Medium, and High. A range of classification algorithms were tested, including Logistic Regression, Random Forest, and XGBoost, to compare predictive performance.

Additionally, unsupervised clustering was also applied using KMeans algorithm to explore natural groupings in user activity behavior.

A. Machine Learning Models

Logistic Regression: A baseline linear classifier used to assess initial performance[11].
Random Forest Classifier: An ensemble learning method combining multiple decision trees to improve accuracy and reduce overfitting[12].
XGBoost Classifier: A gradient boosting algorithm known for its high accuracy and performance in structured datasets[4].
KMeans Clustering: An unsupervised technique used to identify patterns and clusters in physical activity metrics[8].

Each model's output was evaluated using accuracy, precision, recall, F1-score, and multiclass ROC AUC scores to identify which algorithm performed best for classifying fitness levels.

IV. EXPERIMENTAL SETUP

To develop an effective fitness level classification system, several key steps were followed, including preprocessing, feature selection, and evaluation metric design.

A. Data Preprocessing

The original dataset included a mix of numerical and categorical columns, along with missing values. The following steps were applied:

➢ Categorical Encoding:

- gender: Mapped to numeric values (M → 0, F → 1, Other → 2).
- activity_type, intensity, and smoking_status: Converted using one-hot encoding or label encoding depending on the model's requirements.

➢ Missing Value Handling:

- health_condition: This column had a large number of missing (NaN) values. A binary encoding was applied:
  1. Rows with a known health condition (non-null) were encoded as 1.
  2. Rows with missing values were assumed to represent absence of a reported condition and were encoded as 0.
- Other columns were carefully reviewed to ensure completeness.

➢ Target Column Transformation:
The original fitness_level column was continuous. To perform classification, it was binned into three categories:

- 0 = Low Fitness
- 1 = Medium Fitness
- 2 = High Fitness

This new column was named fitness_level_category, and the original fitness_level column was dropped.

B. Feature Selection

Initial modeling attempts with many features led to overfitting and reduced accuracy. Hence, three key features were selected based on Pearson correlation with the target[2]:

- weight_kg
- calories_burned
- daily_steps

These three features showed the strongest relationship with fitness_level_category and produced optimal classification performance.

TABLE I. PEARSON CORRELATION COEFFICIENT WITH *fitness_level_category*

FEATURE	**CORRELATION**
fitness_level_category	1.000000
weight_kg	0.716903
calories_burned	0.264599
daily_steps	-0.473998
intensity	0.003205
resting_heart_rate	0.003108
bmi	0.000687

For detailed transformation steps, including categorical encoding, feature binning, and the complete dataset column summary, please refer to the Appendix B.

C. Data Splitting

The dataset was split into:

- Training Set: 80%
- Testing Set: 20%

Stratified sampling ensured a balanced distribution across fitness classes in both subsets.

D. Evaluation Metrics

Model performance was assessed using the following metrics:

- Accuracy: Overall proportion of correct predictions[6].
- Precision, Recall, F1-score: Calculated for each class to evaluate class-wise performance[6].
- ROC AUC Score (Macro-Averaged): Evaluates the model's ability to distinguish between all classes[6].
- Confusion Matrix: Helps visualize the prediction distribution and class-specific errors[6].

All metrics were visualized and compared across multiple models to select the best-performing classifier.

V. RESULTS

This section presents the outcomes of the classification task for predicting the fitness_level_category using three supervised machine learning models: Logistic Regression, Random Forest, and XGBoost Classifier. These models were selected due to their popularity, interpretability, and strong performance on structured tabular data.

All models were trained using three selected features (*weight_kg, calories_burned, daily_steps*) that showed the highest correlation with the target variable. The target column *fitness_level_category* was derived by binning the continuous *fitness_level* column into three discrete categories (Low, Medium, High).

The dataset was split into 80% training and 20% testing, and stratified sampling was used to ensure balanced distribution of class labels..

Each model's performance was measured on the test set.

TABLE II. COMPARISON of EVALUATION METRICS ACROSS MODELS

Model	**Accuracy**	**Macro F1 Score**	**ROC AUC Score**	**Class 0 AUC**	**Class 1 AUC**	**Class 2 AUC**
Logistic Regression	0.68	0.64	0.8489	0.90	0.72	0.93
Random Forest	0.67	0.66	0.8453	0.89	0.73	0.92
XGBoost	0.70	0.69	0.8644	0.90	0.76	0.93

As shown in *Table II,* the XGBoost classifier achieved the highest accuracy (0.70) and ROC AUC(0.86), outperforming the other models.

B. Confusion Matrix Analysis

To understand the classification errors and class-wise performance, confusion matrices were plotted for each model. These matrices reveal how often the model confuses one class with another.

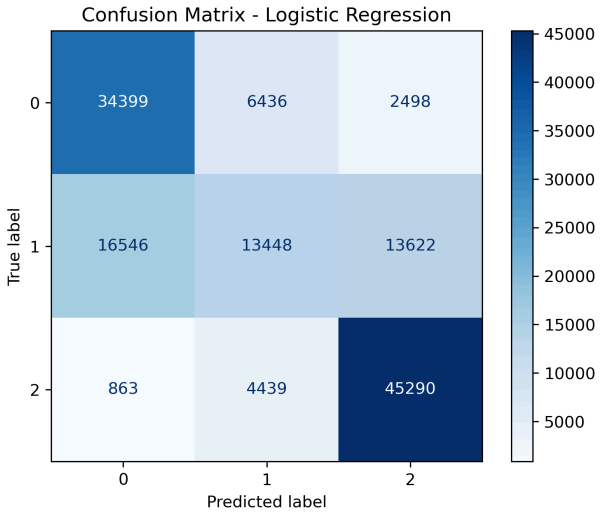

Fig. 4. Confusion Matrix for Logistic Regression

In Fig. 4. we can clearly see that the logistic regression model correctly classified most samples in the 'Low' and 'High' fitness categories, with some misclassifications in the 'Medium' category. The confusion matrix indicates that while the model performs reasonably well overall, distinguishing between 'Medium' and other classes is slightly less accurate.

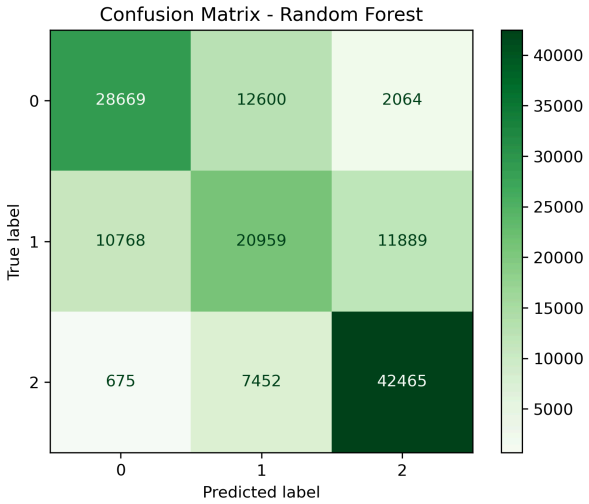

Fig. 5. Confusion Matrix for Random Forest

Fig. 5. The random forest model shows improved performance over logistic regression in correctly predicting the 'Medium' category. The distribution of true positives is more balanced across all three classes, suggesting the model handles class separation more effectively.

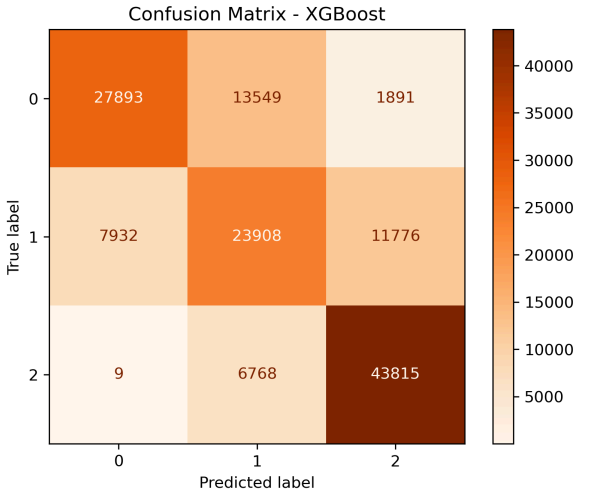

Fig. 6. Confusion Matrix for XGBoost

In Fig. 6. The XGBoost model exhibits the highest accuracy, with very few misclassifications across all classes. Most predictions fall on the diagonal, indicating strong class-wise performance and the model's robustness in handling multiclass classification.

C. ROC AUC Curve Analysis

To further evaluate the ability of the models to distinguish between multiple classes, ROC AUC curves were plotted. For multiclass classification, the macro-averaged ROC curves were used.

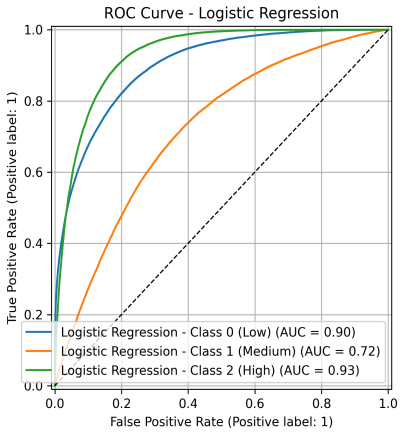

Fig. 7. ROC Curve for Logistic Regression

In Fig. 7. The ROC curve demonstrates that Logistic Regression performed best in predicting Class 2 (High fitness level) with an AUC of *0.93*, followed by Class 0 (Low) with *0.90*. Class 1 (Medium) had the lowest AUC of *0.72*, indicating it was harder for the model to distinguish medium fitness individuals.

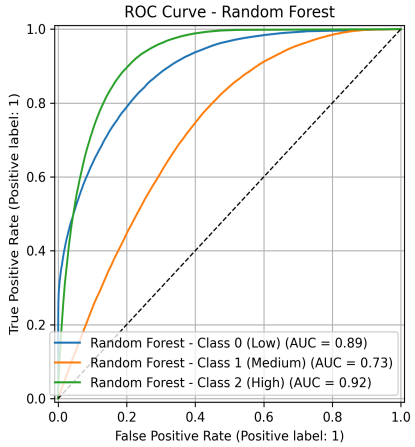

Fig. 8. ROC Curve for Random Forest

In Fig. 8. The ROC curve for Random Forest shows strong predictive performance for Class 2 (High fitness level) with an AUC of *0.92* and Class 0 (Low) with *0.89*. However, it struggles more with Class 1 (Medium), achieving an AUC of only *0.73*. This indicates that the model finds it more difficult to distinguish medium fitness individuals from the other classes.

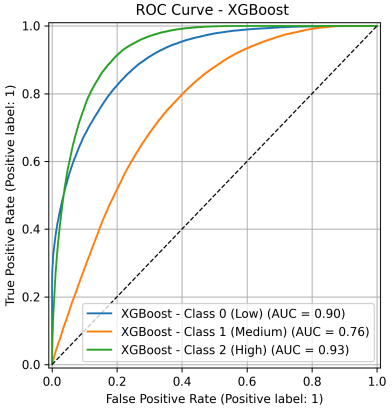

Fig. 9. ROC Curve for XGBoost

Fig. 9. XGBoost exhibits the best overall ROC performance among all models. It achieves high AUC scores across all classes: *0.93* for Class 2 (High), *0.90* for Class 0 (Low), and *0.76* for Class 1 (Medium). The higher AUC for Class 1 compared to other models indicates improved ability to distinguish individuals with medium fitness levels.

D. K-Means Clustering Results

To explore unsupervised learning on the dataset, K-Means clustering was applied after dimensionality reduction using PCA (Principal Component Analysis). Based on the Elbow Method[10], the optimal number of clusters was determined to be 3.

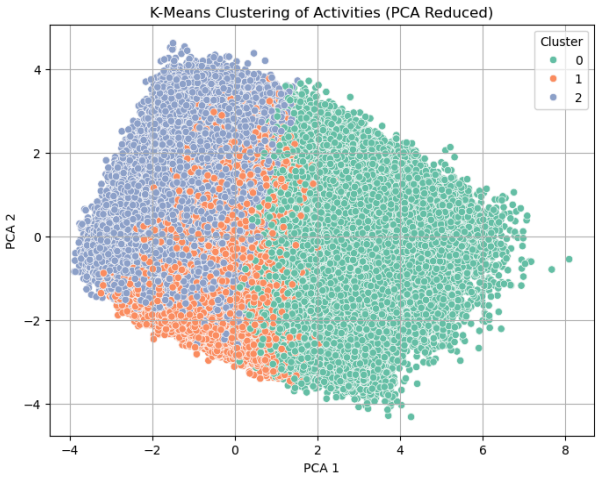

Fig. 10. K-Means clustering result(n=3) using PCA-reduced Activity data

Figure X illustrates the K-Means clustering of activities (after PCA dimensionality reduction) with 3 clusters:

- Cluster 0 (Green) primarily consists of users with high workout intensity and duration. These users frequently participate in high-effort exercises like HIIT and running, showing strong engagement but only moderate average fitness levels.
- Cluster 1 (Red) includes users with moderate workout intensity and duration but the highest average fitness level, suggesting a consistent, efficient approach to fitness. Their top activities include yoga, walking, and weight dancing.
- Cluster 2 (Blue) represents casual users with lower intensity and shorter durations, but the highest daily step count. This suggests a more passive yet consistent lifestyle involving lighter activities like walking, weight training, and yoga.

To further interpret the clusters, a summary of the key numerical attributes was generated for each group. The table below presents average values of selected metrics, highlighting differences in workout intensity, duration, calories burned, step counts, and fitness level category. This provides a clear understanding of how different user groups engage with physical activities.

TABLE III. SUMMARY OF KEY METRICS ACROSS CLUSTERS

METRIC	Cluster 0 (Green)	Cluster 1 (Red)	Cluster 2 (Blue)
Avg. Duration (min)	91.86	60.59	64.24
Avg. Intensity	1.19	0.73	0.86
Calories Burned	28.60	11.95	10.21
Daily Steps	8303	7155	9905
Fitness Level Category	1.30	1.77	0.38
Top Activity	HIIT,Running, Cycling	Yoga, Walking, Dancing	Walking, Weight Training, Yoga

Table III. displays the comparison of core activity and fitness metrics across three K-Means clusters. Each cluster represents a distinct user profile based on fitness behaviour. However we can clearly see that Cluster 0 participants show high intensity and long-duration workouts,typically engaging in HIIT, Running and Cycling. Cluster 1 participants maintain a balanced routine with moderate activity and the highest average fitness level . Cluster 2 is composed of casual users with low to moderate intensity but a surprisingly high step count, indicating more consistent light activity like walking and yoga.

VI. DISCUSSION AND CONCLUSION

This project aimed to classify individuals into fitness categories using the synthetic FitLife360 dataset[3], incorporating physical activity, health metrics, and lifestyle features. After careful feature correlation analysis, three features — weight (kg), calories burned, and daily steps — were selected, as they showed the strongest linear relationship with the target variable fitness level category.

Three machine learning models were applied for classification: Logistic Regression, Random Forest, and XGBoost. Among these, XGBoost achieved the best overall performance with an accuracy of 70%, macro F1 score of 0.69, and a multiclass ROC AUC score of 0.8644. Logistic Regression and Random Forest also performed well, with AUC scores above 0.84, confirming the validity of the chosen features and model strategies.

The findings from unsupervised clustering strongly aligned with the supervised classification outputs. The consistency validates that the selected features effectively separate participants by fitness levels, both algorithmically and behaviorally.

In conclusion, the project demonstrated that targeted feature engineering, model selection, and combined supervised–unsupervised analysis can uncover deep insights from health and fitness data. The ensemble-based XGBoost model proved to be the most reliable for classification[4], while KMeans clustering revealed meaningful participant segments, reinforcing the classification logic. These outcomes highlight the potential of intelligent data modeling in personal health tracking, digital fitness coaching, and proactive wellness monitoring systems.

Additional information, including column details, data preprocessing transformations, and complete feature summaries, is provided in the Appendix for transparency and reproducibility.

REFERENCES

[1] S. Raschka and V. Mirjalili, *Python Machine Learning*, 3rd ed. Birmingham, UK: Packt Publishing, 2019.

[2] J. Benesty, J. Chen, Y. Huang, and I. Cohen, “Pearson Correlation Coefficient,” in *Noise Reduction in Speech Processing*, Berlin, Germany: Springer, 2009, pp. 1–4.

[3] Kaggle, “FitLife360: Health and Fitness Tracker Dataset,” [Online]. Available: https://www.kaggle.com/datasets/jijagallery/fitlife-health-and-fitness-tracking-dataset

[4] T. Chen and C. Guestrin, “XGBoost: A Scalable Tree Boosting System,” in *Proc. 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, San Francisco, CA, USA, 2016, pp. 785–794.

[5] D. Dua and C. Graff, “UCI Machine Learning Repository,” [Online]. Available: https://archive.ics.uci.edu/ml

[6] F. Pedregosa et al., “Scikit-learn: Machine Learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[7] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” in *Proc. Int. Conf. on Learning Representations (ICLR)*, San Diego, CA, USA, 2015.

[8] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, 2nd ed. New York, NY, USA: Springer, 2009.

[9] C. Bishop, *Pattern Recognition and Machine Learning*, New York, NY, USA: Springer, 2006.

[10] S. S. Bandyopadhyay, A. Ghosh, and S. Pal, “Cluster Analysis and the Elbow Method,” in *Data Clustering Algorithms and Applications*, Boca Raton, FL, USA: CRC Press, 2013, pp. 23–34.

[11] D. W. Hosmer Jr., S. Lemeshow, and R. X. Sturdivant, *Applied Logistic Regression*, John Wiley & Sons, 2013.

[12] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

APPENDIX A - DATASET OVERVIEW

TABLE IV. DESCRIPTION OF SINGLE ENTRY IN DATASET

Column Name	Data Type	Example Value	Short Description
participant_id	int64	1,2,.....3000	Unique ID for each participant
date	object	2024-01-01,.....	Activity record date
age	int64	18,.....64	Participant,s age(18-65)
gender	object	M	Gender(M/F/Other)
height_cm	float64	145,.....198.5	Height in centimetres
weight_kg	float64	45.3,.....188.4	Weight in kilogram
activity_type	object	Yoga,Running,HIIT etc.	Type of activity performed
duration_minutes	int64	20,.....120	Duration of activity session in minutes
intensity	object	Low	Exercise intensity (Low/Medium/High)
calories_burned	float64	0.8,5.4,6.3,.....92	Calories burned during activity
daily_steps	int64	7128,11120,.....	Number of steps taken in a day
avg_heart_rate	int64	99, 102,.....etc	Average heart rate during activity
hours_sleep	float64	7.5	Hours of sleep per night
stress_level	int64	4	Daily stress level(1-10)
hydration_level	float64	2.5	Daily water intakes in litre
bmi	float64	23.7	Body mass index calculated from height & weight
resting_heart_rate	float64	72	Resting heart rate(in bpm)
blood_pressure_systolic	float64	120.20	Systolic blood pressure
blood_pressure_diastolic	float64	85.70	Diastolic blood pressure
health_condition	object	NaN	Presence of health condition
smoking_status	object	Never	Smoking History (Never/Former/Current)
fitness_level	float64	0.37	Calculated fitness score based on cumulative activity

APPENDIX B - DATA PREPROCESSING and FEATURE ENGINEERING

```
d = pd.read_csv("health_fitness_dataset.csv")

#  Copying my store dataset from d to df
df= d.copy()

# Check unique values before transformation
print("Before transformation:")
print(d['health_condition'].value_counts(dropna=False))

# Apply binary encoding, 0= no healthcondition, 1= has health condition
df['health_condition'] = df['health_condition'].apply(lambda x: 0 if pd.isna(x) else 1)

# Check unique values after transformation
print("\nAfter transformation:")
print(df['health_condition'].value_counts())
```

```
Before transformation:
health_condition
NaN            490275
Hypertension    99437
Diabetes        64754
Asthma          33235
Name: count, dtype: int64

After transformation:
health_condition
0    490275
1    197426
Name: count, dtype: int64
```

```
# Map binary/ordinal variables for different columns
df['gender'] = df['gender'].map({'M': 1, 'F': 0, 'Other':2})
df['intensity'] = df['intensity'].map({'Low': 0, 'Medium': 1, 'High': 2})
 df['smoking_status'] = df['smoking_status'].map({
    'Never': 0,
    'Former': 1,
    'Current': 1
})
```

```
# defining bins
bins = [0,6,12,float('inf')]
labels= ['Low','Medium','High']
# converting fitness_level into 3 categories
df['fitness_level_category']= pd.cut(d['fitness_level'], bins=bins, labels= labels)
df['fitness_level_category'] = df['fitness_level_category'].map({'Low': 0, 'Medium': 1, 'High': 2})
df.drop('fitness_level', axis=1, inplace= True)
# check distribution count
print(df['fitness_level_category'].value_counts())
```

Pearson Correlation code

```
# Ensure it's a number
df['fitness_level_category'] = df['fitness_level_category'].astype(int)

# Select only numeric columns
numeric_df = df.select_dtypes(include='number')

#  check correlation
if 'fitness_level_category' in numeric_df.columns:
    correlation_with_fitness = numeric_df.corr()['fitness_level_category'].sort_values(ascending=False)
    print("🔍 Feature Correlation with 'fitness_level_category':")
    print(correlation_with_fitness)
else:
    print("❌ 'fitness_level_category' is not numeric or missing from numeric_df")
```

```
🔍 Feature Correlation with 'fitness_level_category':
fitness_level_category      1.000000
weight_kg                   0.716903
calories_burned             0.264599
intensity                   0.003205
resting_heart_rate          0.003108
duration_minutes            0.002955
gender                      0.002266
age                         0.001959
height_cm                   0.001490
stress_level                0.001373
stress_category             0.001333
avg_heart_rate              0.000740
bmi                         0.000687
blood_pressure_systolic     0.000479
blood_pressure_diastolic    0.000460
smoking_status              0.000298
health_condition           -0.000618
hydration_level            -0.000767
hours_sleep                -0.001373
daily_steps                -0.473998
Name: fitness_level_category, dtype: float64
```

APPENDIX C - MODEL IMPLEMENTATION

```
# Step 1: Define features and target
features = ['weight_kg', 'calories_burned', 'daily_steps']
target = 'fitness_level_category'

X = df[features]
y = df[target]
```

```
classes = [0, 1, 2]  # Low, Medium, High

# Step 2: Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, stratify=y, random_state=42
)

# Step 3: Feature scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Step 4: Binarize target for ROC AUC
y_train_bin = label_binarize(y_train, classes=classes)
y_test_bin = label_binarize(y_test, classes=classes)
```

```
# Logistic Regression Model
logreg =
LogisticRegression(max_iter=1000,multi_class='ovr')
logreg.fit(X_train_scaled, y_train)
y_pred_logreg = logreg.predict(X_test_scaled)

ConfusionMatrixDisplay.from_predictions(
    y_test, y_pred_logreg, cmap="Blues"
)

print(classification_report(y_test, y_pred_logreg,
target_names=["Low", "Medium", "High"]))
plt.title("Confusion Matrix - Logistic Regression")
plt.savefig("confusion_logreg.png", dpi=300,
bbox_inches='tight')
plt.show()

y_score = logreg.predict_proba(X_test_scaled)
auc = roc_auc_score(y_test_bin, y_score,
multi_class='ovr')
print(f"🎯 Multiclass ROC AUC Score: {auc:.4f}")


# roc curve for logistic regression
for i in range(len(classes)):
    RocCurveDisplay.from_predictions(
          y_test_bin[:, i], y_score[:, i],
          name=f"{'Logistic Regression' } - Class {i} ({['Low',
'Medium', 'High'][i]})",
        ax= plt.gca()

        )

# add grid and diagonal line
plt.plot([0,1],[0,1],'k--',lw=1)
plt.grid(True)
plt.title("ROC Curve - Logistic Regression")
plt.tight_layout()

# save the figure
plt.savefig("fig_logistic_regression_roc.png", dpi=300)
plt.show()
```

```
# Random Forest Model
rf = RandomForestClassifier(n_estimators=100,
random_state=42)
rf.fit(X_train_scaled, y_train)
y_pred_rf = rf.predict(X_test_scaled)

ConfusionMatrixDisplay.from_predictions(
    y_test, y_pred_rf, cmap="Greens"
)

print(classification_report(y_test, y_pred_rf,
target_names=["Low", "Medium", "High"]))
plt.title("Confusion Matrix - Random Forest")
plt.savefig("confusion_rf.png", dpi=300,
bbox_inches='tight')
plt.show()

y_score = rf.predict_proba(X_test_scaled)
auc = roc_auc_score(y_test_bin, y_score,
multi_class='ovr')
print(f"🎯 Multiclass ROC AUC Score: {auc:.4f}")

for i in range(len(classes)):
    RocCurveDisplay.from_predictions(
          y_test_bin[:, i], y_score[:, i],
          name=f"{'Random Forest'} - Class {i} ({['Low',
'Medium', 'High'][i]})",
        ax= plt.gca()
        )

# add grid and diagonal line
plt.plot([0,1],[0,1],'k--',lw=1)
plt.grid(True)
plt.title("ROC Curve - Random Forest")
plt.tight_layout()

# save the figure
plt.savefig("fig_Random Forest_roc.png", dpi=300)
plt.show()
```

```
# XGBoost Model
xgb = XGBClassifier(use_label_encoder=False,
eval_metric='mlogloss',random_state=42)
xgb.fit(X_train_scaled, y_train)
y_pred_xgb = xgb.predict(X_test_scaled)

ConfusionMatrixDisplay.from_predictions(
    y_test, y_pred_xgb, cmap="Oranges"
)

print(classification_report(y_test, y_pred_xgb,
target_names=["Low", "Medium", "High"]))
plt.title("Confusion Matrix - XGBoost")
plt.savefig("confusion_xgb.png", dpi=300,
bbox_inches='tight')
```

```
plt.show()

y_score = xgb.predict_proba(X_test_scaled)
auc = roc_auc_score(y_test_bin, y_score,
multi_class='ovr')
print(f"🎯 Multiclass ROC AUC Score: {auc:.4f}")

for i in range(len(classes)):
    RocCurveDisplay.from_predictions(
            y_test_bin[:, i], y_score[:, i],
            name=f"{'XGBoost'} - Class {i} ({['Low',
'Medium', 'High'][i]})",
        ax= plt.gca()
        )

# add grid and diagonal line
plt.plot([0,1],[0,1],'k--',lw=1)
plt.grid(True)
plt.title("ROC Curve - XGBoost ")
plt.tight_layout()

# save the figure
plt.savefig("fig_XGBoost_roc.png", dpi=300)
plt.show()
```

APPENDIX D - CLUSTERING(K MEANS)

```
# Elbow method to find optimal K
inertia = []
k_range = range(1, 10)

for k in k_range:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(X_cluster)
    inertia.append(kmeans.inertia_)

# Plot the Elbow Curve
plt.figure(figsize=(8, 4))
plt.plot(k_range, inertia, marker='o')
plt.xlabel('Number of clusters (k)')
plt.ylabel('Inertia')
plt.title('Elbow Method for Optimal k')
plt.grid(True)
plt.show()
```

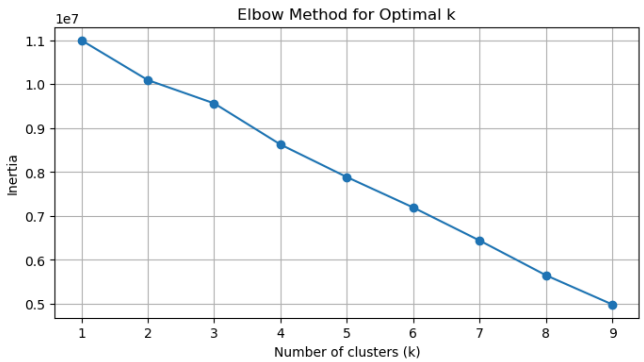

Fig. 11 Graph of elbow method for optimal k

```
# applying K Means with chosen k=3(from elbow
method)
kmeans = KMeans(n_clusters=3, random_state=42)
df['activity_cluster'] = kmeans.fit_predict(X_cluster)

pca = PCA(n_components=2)
reduced_data = pca.fit_transform(X_cluster)

plt.figure(figsize=(8, 6))
sns.scatterplot(x=reduced_data[:, 0], y=reduced_data[:,
1],
                hue=df['activity_cluster'], palette='Set2')
plt.title("K-Means Clustering of Activities (PCA
Reduced)")
plt.xlabel("PCA 1")
plt.ylabel("PCA 2")
plt.legend(title="Cluster")
plt.grid(True)
plt.show()
```

The image of this is in above Fig. 10 .

```
# cluster summary for 3 cluster
cluster_summary =
df.groupby('activity_cluster')[features_for_clustering].me
an().T
print(cluster_summary)
```

activity_cluster	0	1	2
duration_minutes	91.857147	60.588912	64.242156
intensity	1.185693	0.727930	0.857954
calories_burned	28.604641	11.945895	10.214516
daily_steps	8303.723163	7155.130693	9905.004286
avg_heart_rate	137.922633	127.425703	130.659955
hydration_level	2.498052	2.499705	2.500023
fitness_level_category	1.302825	1.773142	0.375360
activity_type_Cycling	0.154231	0.074963	0.088299
activity_type_Dancing	0.028728	0.133427	0.118269
activity_type_HIIT	0.289628	0.029102	0.043920
activity_type_Running	0.211403	0.050542	0.066131
activity_type_Swimming	0.073903	0.100188	0.112572
activity_type_Tennis	0.085675	0.099955	0.107578
activity_type_Walking	0.006204	0.142885	0.120641
activity_type_Weight Training	0.041334	0.124750	0.118910
activity_type_Yoga	0.002138	0.150753	0.123560

APPENDIX E - PACKAGE IMPORTS

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# ML tools
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, label_binarize
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import (
    accuracy_score, classification_report, roc_auc_score,
    confusion_matrix, precision_recall_curve, roc_curve,
    RocCurveDisplay, ConfusionMatrixDisplay
)
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder, StandardScaler

# ML Algorithm
from xgboost import XGBClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.cluster import KMeans
from sklearn.linear_model import LogisticRegression
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
```

APPENDIX F - ADDITIONAL FIGURES

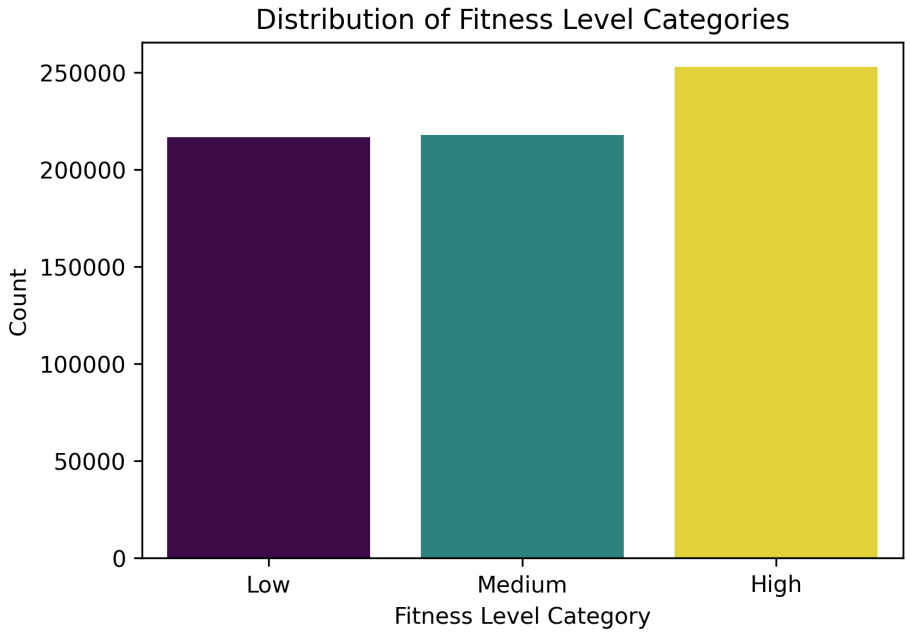

Fig. 12 Graph of finess_level_category count for different class