Activity Context Recognition

Implementation Report   
Thi Ha My Nguyen - 32067811

**Contents**

[CHAPTER 1 INTRODUCTION 3](#_Toc135681164)

[CHAPTER 2 PROBLEM STATEMENT 3](#_Toc135681165)

[CHAPTER 3 SOLUTION REQUIREMENTS 3](#_Toc135681166)

[CHAPTER 4 SYSTEM IMPLEMENTATION AND INTEGRATION 4](#_Toc135681167)

[CHAPTER 5 PROGRAM EXECUTION 4](#_Toc135681168)

[CHAPTER 6 APPLICATION STRUCTURE 5](#_Toc135681169)

[CHAPTER 7 EVALUATION 5](#_Toc135681170)

[CHAPTER 8 REFLECTION 5](#_Toc135681171)

[REFERENCES 7](#_Toc135681172)

[APPENDIX A 7](#_Toc135681173)

Figures

Figure 4.1: Retrieve tab to display data. 7

Figure 4.2: Statistic tab display statistical analysis 8

Figure 4.3: Frequencies/Dependencies tab show activity frequencies and correlation of feature 8

Figure 4.4: Pre-processing tab display data information, drop None samples, drop duplicate samples and handle unbalanced samples 9

Figure 4.5: Frequencies/Dependencies after cleaning data 9

Figure 4.6: Train Model and Predict, providing best params of the chosen models. 10

Figure 4.7: Classification Report for the chosen models. 10

Figure 4.8: Confusion Matrix for the chosen models 11

Figure 4.9: Optional Statistics 11

Tables

Table 1: Six classification models 6

# CHAPTER 1 INTRODUCTION

This report presents a concise overview of a proposed problem and its implemented solution. It provides an outline of the program components, software requirements, and implementation details. The report also includes instructions for program execution, a flow chart, and pseudocode for uncharted sections. Reflective evaluation, suggestions for enhancements, and alternative approaches are discussed as well. Chapter 2 focuses on developing an advanced recommendation engine using smartphone sensor data. It aims to accurately recommend activities through machine learning and statistical analysis.

# CHAPTER 2 PROBLEM STATEMENT

The project aims to develop an advanced recommendation engine using smartphone sensor data to identify specific activities. By leveraging machine learning and statistical analysis, the engine provides intelligent and accurate activity recommendations for diverse applications, from remote health monitoring to personalized services.

The program begins with the EDA module, retrieving and pre-processing data from provided datasets. Descriptive statistical analysis and visualizations are performed on observations from seven sensors, including orientation, rotation, accelerometer, gyroscope, magnetic, sound, and light sensors.

The cleaned dataset is split into training and test sets for building and evaluating classification models such as Support Vector Machine, Random Forest, Neural Networks, K-nearest Neighbour, Logistic Regression, and Decision Tree. Evaluation metrics, including confusion matrix, accuracy, precision, recall, and F1-score, are used to compare and assess model performance.

The program utilizes third-party libraries like NumPy, Pandas, Matplotlib, and Seaborn for statistical calculations and data visualization. Error handling ensures program robustness. The user interface, designed with Tkinter, allows users to interact with the system, perform data cleaning, pre-processing, model building, and result analysis for selecting the best model for their data.

# CHAPTER 3 SOLUTION REQUIREMENTS

In EDA module:

* Create an EDA class that includes the following functions.
* Develop a function that extracts the attributes/features from the data and stores the data in a data frame.
* Implement a function that pre-processes the dataset such as checking missing data, applying cleaning techniques, and addressing the issue of unbalanced class distribution (oversampling and under sampling).
* Implement eight functions that computes statistical analysis of the dataset such as mean, median, standard deviation, variance, minimum, maximum, skewness and kurtosis.
  + Each should accept parameters of a list of features.
* Implement two functions that compute frequencies and dependencies of the dataset and plot in appropriate graphs.
  + Each should accept parameters of a list of features.
* Implement six functions that build classification models including Support Vector Machine, Random Forest, Multi-Layer Perceptron Neural Networks, K-nearest Neighbour, Logistic Regression, and Decision Tree
  + Each should accept parameters in grid params.
* Implement a function to report confusion matrix and compare evaluation metrics of the models.
  + Each should accept parameters of user ID.

In the python\_gui\_as2 and python\_gui\_as2\_supported module:

* Implement a class that contains the user interface through which users can query and interact with all functions in eda\_module above.

# CHAPTER 4 SYSTEM IMPLEMENTATION AND INTEGRATION

Firstly, in Figure 1, all the dataset samples are displayed. The data is retrieved from the 'activity\_context\_tracking\_data.csv' file and stored in a data frame. It is shown as a table frame in the 'Retrieve' tab of the Machine Learning - Activities window (Fig 4.1).

Secondly, in the 'Statistics' tab, the program provides statistical figures based on the user's selected features of interest. By selecting the corresponding checkboxes for desired feature ranges and clicking the 'Describe' button, the program generates a table displaying the required statistics (Fig 4.2).

Thirdly, the program displays frequencies and dependencies of the data in the ‘Frequencies/Dependencies’ tab (Fig 4.3). The user can also access information about the dataset in the ‘Data Information’ frame and clean the dataset by clicking the ‘Clean’ button in the ‘Preprocessing’ tab (Fig 4.4 and 4.5).

Next, the program provides 6 models including Support Vector Machine, Random Forest, Multi-Layer Perceptron Neural Networks, K-nearest Neighbour, Logistic Regression, and Decision Tree. Depending on the user's choices by clicking on the checkbox, the user can train and predict their clicked models. After that, the program will execute training models with several different parameters by using ‘GridSearchCV’ to find out the best models displaying on the ‘Best params’ frame, then store the best models in each method in ‘model\_X.pkl’ files. Because the program takes a lot of time to find the best parameters to train models, by saving the models we can reduce the time to train as well as predict new samples (Fig 4.6). Depending on the above-picked models, the ‘Report’ and ‘Confusion Matrix’ tabs will display the Classification Report (Fig 4.7) and Confusion Matrix (Fig 4.8) of those models, respectively.

Finally, the program includes optional statistics in the ‘Optional’ tab. Users can choose their desired ranges, and the program will display statistics such as Mean, Variance, Standard Deviation, Root Square Mean (RMS), Zero Crossing (ZC), Sum of Squares (SOS), and Covariance in the ‘Output’ frame (Fig 4.9)**.**

# CHAPTER 5 PROGRAM EXECUTION

Further information regarding the program's execution is provided below. The flowchart presented in the subsequent section of the report offers an overview of the program's complete execution flow from start to finish. The pseudocode for the three modules can be found in Appendix A, encompassing portions of the module code not explicitly represented in the flowchart. However, it should be noted that the pseudocode does not encompass the entirety of the code contained within the module files.

To ensure the program's proper functioning, it is necessary to have the file named "python\_gui\_as2.py”, “python\_gui\_as2\_supported.py” and the module files named "eda\_module.py" situated within the same directory as the data file titled "activity\_context\_tracking\_data.csv". All these files must be collocated within a single folder to ensure the correct execution of the program.

To execute program, the user just needs to run command ‘**python3 python\_gui\_as2.py**’

# CHAPTER 6 APPLICATION STRUCTURE

A diagram of a train model

Description automatically generated with low confidence

# CHAPTER 7 EVALUATION

Based on the results obtained after training and predicting each model, the following conclusions can be drawn:

* + SVM: The accuracy of this model is 98.1%.
  + Random Forest: The accuracy of this model is 97.4%.
  + MLP: The accuracy of this model is 98.2%.
  + KNN: The accuracy of this model is 98%.
  + Decision Tree: The accuracy of this model is 93.9%.
  + Logistic: The accuracy of this model is 59.7%.

These accuracy values indicate the performance of each model in correctly predicting the activities based on the provided dataset. It can be observed that SVM, MLP, and KNN models achieved high accuracy rates, indicating their effectiveness in activity prediction. On the other hand, the Decision Tree model achieved a slightly lower accuracy, while the Logistic model showed comparatively lower accuracy.

# CHAPTER 8 REFLECTION

Throughout the research and program execution process, I have gained a fundamental understanding of machine learning. This includes familiarizing myself with the steps required for problem-solving, developing algorithms, determining critical algorithm parameters, evaluating models, and visualizing data.

Below is a brief of the result of my research about six Classification models and some of their parameters:

Table 1: Six classification models

|  |  |  |
| --- | --- | --- |
|  | Definition | Parameters |
| SVM | * SVM finds the optimal hyperplane that maximizes the margin between different classes. * SVM can handle high-dimensional data and is effective in cases with a clear separation between classes. | * C: It balances the trade-off between margin width and misclassification (inversely proportion in sklearn * Kernel: Measures the similarity between the original feature space and the enlarged feature space, reducing calculation. * Gamma: Determines the influence of a single training example. |
| Random Forest | * Random Forest is an ensemble learning method that combines multiple decision trees.[1] * It reduces overfitting and improves generalization by aggregating predictions from individual trees. | * n\_estimators: number of decision trees to use in forest (from 64-128 is good choice)[2] * max\_features: how many features to include in each subset.(  for regression problems and  for classification problems)[3] * bootstrap: allow bootstrap sampling of each training subset of features. * obbscore: evaluates the predictions made by each decision tree in the forest on the samples that were not used for training that particular tree. |
| MLP | * MLP is a type of artificial neural network with multiple layers of interconnected neurons. * It is capable of learning complex nonlinear relationships between inputs and outputs. | * hidden\_layer\_sizes: the number of hidden layer. * activation: Activation function for the hidden layer * max\_iter: Maximum number of iterations |
| KNN | * KNN classifies instances based on their proximity to neighbours. * It assigns a label to a data point based on the majority label of its k nearest neighbours. | * n\_neighbors: Number of neighbors to use[4] * metric: Metric to use for distance computation. |
| Logistic Regression | * Logistic Regression models the probability of an instance belonging to a specific class. * It is widely used due to its simplicity and interpretability. | * penalty: the norm of the penalty * max\_iter: Maximum number of iterations * solver: Algorithm to use in the optimization problem. |
| Decision Tree | * Decision Tree builds a tree-like structure to make decisions based on features' information gain. | * criterion: The function to measure the quality of a split * splitter: The strategy used to choose the split at each node * max\_depth: The maximum depth of the tree. |

To evaluate model, there are some evaluation metrics such as precision, recall, and F1-score help measure the performance of classification models.

* Precision: Precision tells us how many of the predicted positive instances are actually correct. It focuses on minimizing false positives.
* Recall: Recall tells us how many of the actual positive instances are correctly predicted. It focuses on minimizing false negatives.
* F1-score: The F1-score is a combined measure of precision and recall. It provides an overall assessment of the model's performance, taking into account both false positives and false negatives.
* Accuracy is a measure of how often a classification model makes correct predictions.

# REFERENCES

1. Umar Iqbal, Charlie Wolfe: KHALEESI: Breaker of Advertising and Tracking Request Chains.
2. Thais M. O., Pedro S. P., and Jos´e A. B.: How Many Trees in a Random Forest?(2012).
3. Leo Breiman: Random Forest(2001).
4. Sukamto, Hadiyanto, Kurnianingsih :A Hybrid Resampling Method with K-Nearest Neighbour (FHR-KNN) for Imbalanced Preeclampsia Dataset(2023).

# APPENDIX A

**FIGURES**

A screenshot of a computer

Description automatically generated with medium confidence

Figure 4.: Retrieve tab to display data.

A screenshot of a computer

Description automatically generated with medium confidence

Figure 4.: Statistic tab display statistical analysis

A screenshot of a computer

Description automatically generated with medium confidence

Figure 4.: Frequencies/Dependencies tab show activity frequencies and correlation of feature

A screenshot of a computer

Description automatically generated

Figure 4.: Pre-processing tab display data information, drop None samples, drop duplicate samples and handle unbalanced samples

A screenshot of a computer

Description automatically generated with medium confidence

Figure 4.5: Frequencies/Dependencies after cleaning data

A screenshot of a computer

Description automatically generated

Figure 4.: Train Model and Predict, providing best params of the chosen models.

A screenshot of a computer

Description automatically generated

Figure 4.: Classification Report for the chosen models.

A screenshot of a computer

Description automatically generated

Figure 4.: Confusion Matrix for the chosen models

A screenshot of a computer

Description automatically generated

Figure 4.: Optional Statistics