**Table of Contents**

[**Introduction** 3](#_Toc135678684)

[**Methodology** 3](#_Toc135678685)

[Preprocessing the Data: 3](#_Toc135678686)

[Algorithms: 4](#_Toc135678687)

[**Conclusion** 4](#_Toc135678688)

[**Recommendations** 5](#_Toc135678689)

[**Appendices** 6](#_Toc135678690)

**Table of Figures**

[Figure 1: Confusion Matrix 8](#_Toc135675230)

[Figure 2: Plot (A) 8](#_Toc135675231)

[Figure 3: Plot (B) 9](#_Toc135675232)

[Figure 4: Plot (C) 9](#_Toc135675233)

[Figure 5: SVC Evaluation Metrix 10](#_Toc135675234)

[Figure 6: Random Forest Evaluation Metrix 11](#_Toc135675235)

[Figure 7: MLP Evaluation Metrix 12](#_Toc135675236)

# **Introduction**

The goal of this assignment is to create an intelligent activity recognition model for a mobile application for a fitness company. Low-level sensor data from smartphone sensors makes up the dataset. Statistical feature extraction techniques will be applied to the data in order to extract useful information. Custom modules, function definitions, OOP, file processing, and exception handling are just a few examples of programming ideas that will be used. For scientific computing, data analysis, visualization, and machine learning applications, libraries like NumPy, Pandas, Matplotlib, and Scikit-learn will be used. Delivering a precise activity recognition model that can adjust to users' actions and enable personalized fitness monitoring and services is the aim.

# **Methodology**

## **Preprocessing the Data:**

To perform Exploratory Data Analysis, we imported the necessary libraries. Then, loaded a CSV file and assigned it to the ‘data’ DataFrame.

Then performed various data exploration and preprocessing steps on the loaded data, including printing the first few rows, getting the shape of the DataFrame, summarizing basic statistics, checking data types, checking missing values, and dropping unnecessary columns “Sound” and “Light”.

Now all the missing values in the DataFrame were replaced with zeros. To make sure that all blanks have been filled in correctly, the mean is calculated for each column to fill the missing values.

The correlation matrix for the DataFrame's numerical columns is then calculated, by showing their positive or negative correlation, this matrix sheds light on the links between various variables.

Now heatmap was generated using the seaborn library to display the correlation matrix. The correlation coefficients are represented as color-coded cells on the heatmap, with stronger positive correlations appearing in one color, larger negative correlations in another, and no connection (or a correlation very near to zero) appearing in a neutral color. For easier interpretation, the correlation coefficients are annotated on the heatmap. The resulting visualization makes it easier to spot trends and connections between the DataFrame's many components.

Next, we calculated additional statistics such as mean, median, mode, maximum, minimum, standard deviation, and variance for each column in the DataFrame using Pandas. We stored these statistics in a new DataFrame called statistics.

We generated a unique bar plot for each column in the DataFrame by iterating over each one. The bar plots are produced using the sns.barplot() function from the Seaborn library. The y-axis displays the relevant values for each statistic, and the x-axis displays the various statistics (mean, median, mode, maximum, minimum, standard deviation, and variance).

The subplots() function from matplotlib.pyplot is used to build the grid of plots. Based on the total number of columns in the DataFrame, the grid's number of rows and columns is calculated. To guarantee that each plot has the proper amount of space, the figure size is computed.

Labels and titles are added for each plot, and the bars are annotated with the data labels. Setting border colours and widths modifies the plots' appearance. Additionally, the padding and distance between the subplots are changed.

A visual representation of the derived statistics for each column in the DataFrame is produced as a grid of bar charts, making comparison and analysis simple.

Detailed confusion metrix in Appendix-B

All Bar Graphs in Appendix-B

## **Algorithms:**

Three machine learning algorithms were compared, “SVC”, “Random Forest”, and “MLP” and trained using the data. For this, we imported Libraries and modules.

The data was split into features (X) and the target variable (Y). Then the data was split further into training and testing sets. After splitting the features into the training and testing sets it was standardized. This step is to ensure that all features have similar scales.

Now the three models were used, these models were trained on the scaled training data. The trained models were then used on the scaled test data to make predictions. Finally, the prediction from each model was evaluated using various metrics such as Accuracy, Precision, Recall, and F1-score. These metrics assessed the performance of the models by comparing the predicted labels with the actual labels in the testing set.

According to these evaluation metrics, the Random Forest is the best-performing model.

All the libraries and modules are in Appendix-A.

The full result for detailed comparison is in Appendix-B.

# **Conclusion**

The assignment's goal is to use statistical feature extraction and machine learning to create an intelligent activity identification model for a mobile fitness application. The data was captured for this purpose is from smartphone’s built-in sensors. The data consisted of multiple columns and rows in .csv file. To perform analysis, we loaded the data into Python env. using Panda’s library. Then we explored the data first and checked for the basic description of the data set using the panda’s library. We found multiple columns have missing values, so we filled those columns using the means of the columns or with the zeros where necessary. Then we dropped the columns which had no use in the task “light” and “Sound” columns and calculated the correlation among the variables. Then we generated a correlation matrix using the Seaborn library. Then we calculated mean, median, standard, deviation, variance, minimum, and maximum using Pandas Library. At last we use Seaborn which is built upon Mat plot Library to plot for better visualization.

The classification task was performed based on the "activity" column, which contains 13 different attributes or labels. Each algorithm was trained to predict the activity based on the remaining features in the dataset. The performance of the algorithms in classifying the activities was then evaluated using various evaluation metrics. According to these evaluation metrics, the Random Forest is the best-performing model.

# **Recommendations**

I would advise choosing Random Forest for your classification challenge based on the examination of the performance measures, particularly the excellent accuracy, precision, recall, and F1-score obtained by the Random Forest model. The following are the arguments in favour of Random Forest:

* High Accuracy: Random Forest correctly identified the activities in the dataset with an accuracy of 0.9997.
* Balanced Precision and Recall: Random Forest was able to accurately identify positive occurrences (precision) and capture all positive instances (recall) with a precision and recall of 0.9997.
* Strong Performance: Random Forest regularly outperformed other evaluation measures, indicating that it can perform the categorization task.
* Handling Complex Relationships: Random Forest is renowned for its capacity to manage intricate connections and interactions between features, making it appropriate for datasets with numerous attributes.
* Decision Tree Ensemble: Random Forest makes use of a decision tree ensemble to help decrease overfitting and enhance generalization to unobserved data.
* Random Forest offers a measure of feature importance that enables you to learn more about the significance of various features during the classification process.

With great performance and stability, Random Forest emerges as a trustworthy and efficient solution for your classification task.

# **Appendices**

**Appendix-A**

**Libraries:**

* **Pandas**: It is a powerful Python data analysis and manipulation toolkit. It offers data structures and operations for working effectively with structured data, such tabular data.
* **Scikit-learn (sklearn)**: It is a well-known Python machine-learning library. It offers a wide variety of tools for model choice, evaluation, and data preprocessing. It covers several algorithms for dimensionality reduction, clustering, regression, and classification.
* **Seaborn:** This library is built on top of matplotlib and provides a high-level interface for creating informative and attractive statistical graphics.

**Modules:**

* train\_test\_split
* StandardScaler
* SVC
* RandomForestClassifier
* MLPClassifier
* accuracy\_score
* precision\_score
* recall\_score
* f1\_score
* confusion\_matrix
* skew,
* kurtosis
* matplotlib.pyplot

**Appendix-B**

**Correlation Matrix:**

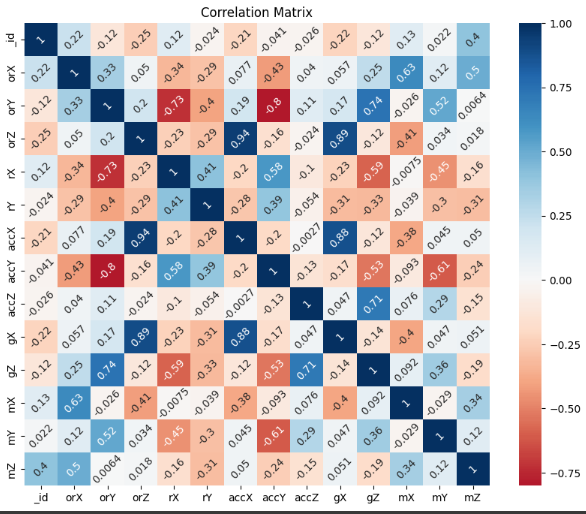


Figure 1: Confusion Matrix

**Plotting of each Column:**

A picture containing diagram, screenshot, line, plan

Description automatically generated

Figure 2: Plot (A)

A picture containing diagram, plan, screenshot, line

Description automatically generated

Figure 3: Plot (B)

A screenshot of a graph

Description automatically generated with low confidence

Figure 4: Plot (C)

**Appendix-C**

**SVC:**

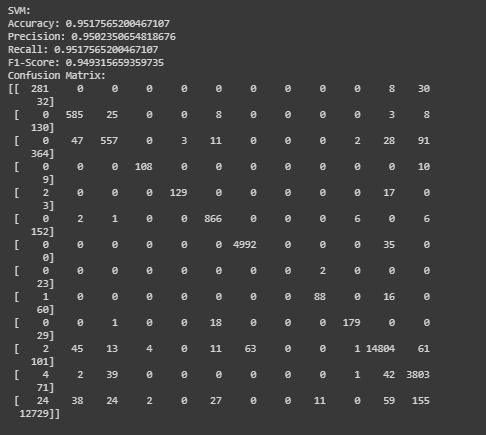


Figure 5: SVC Evaluation Metrix

**Random Forest:**

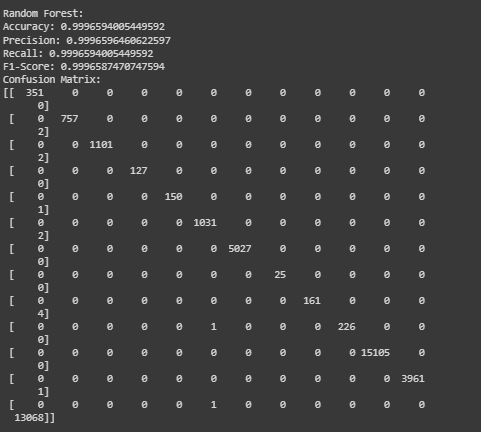


Figure 6: Random Forest Evaluation Metrix

**MLP:**

A screenshot of a computer

Description automatically generated

Figure 7: MLP Evaluation Metrix