Project Report: Accelerometer Data Analysis and Activity Classification

Executive Summary

This report presents the findings from the exploratory data analysis (EDA) and the development of a machine learning model to classify physical activities based on accelerometer data. The data, derived from 15 participants, captures x, y, and z acceleration metrics, which are used to predict activities such as working at a computer, walking, and climbing stairs etc.

Understanding the Dataset

The dataset consists of accelerometer data sampled at a rate of 52 Hz from 15 participants, engaged in 7 distinct activities. Each CSV file correlates to a single participant and includes columns for sequential number, x, y, z accelerations, and activity labels.

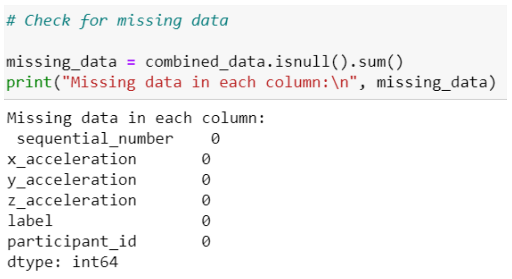
Labels and Corresponding Activities

The activities, encoded from 1 to 7, represent the following:

1. Working at Computer
2. Standing Up, Walking, and Going up/downstairs
3. Standing
4. Walking
5. Going Up/Downstairs
6. Walking and Talking with Someone
7. Talking while Standing

Data Preprocessing

Data preprocessing involved importing CSV files, checking for missing values, and normalizing the acceleration data to standardize the input features for our machine learning model.



```
# Check for missing data

missing_data = combined_data.isnull().sum()
print("Missing data in each column:\n", missing_data)
```

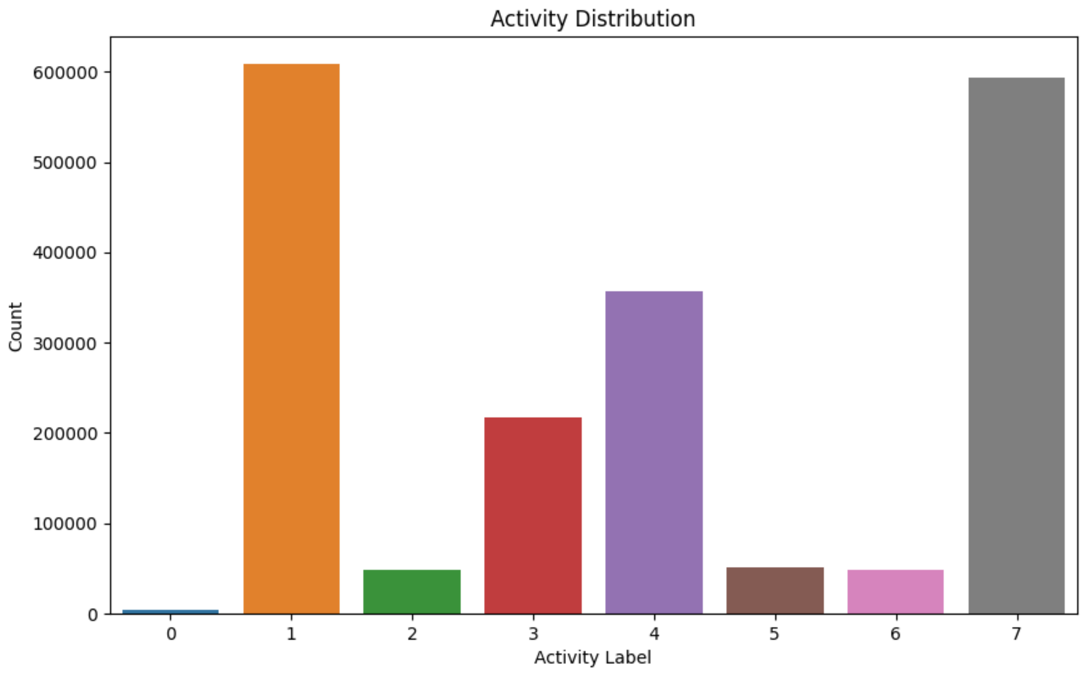
```
Missing data in each column:
 sequential_number    0
x_acceleration       0
y_acceleration       0
z_acceleration       0
label                0
participant_id       0
dtype: int64
```

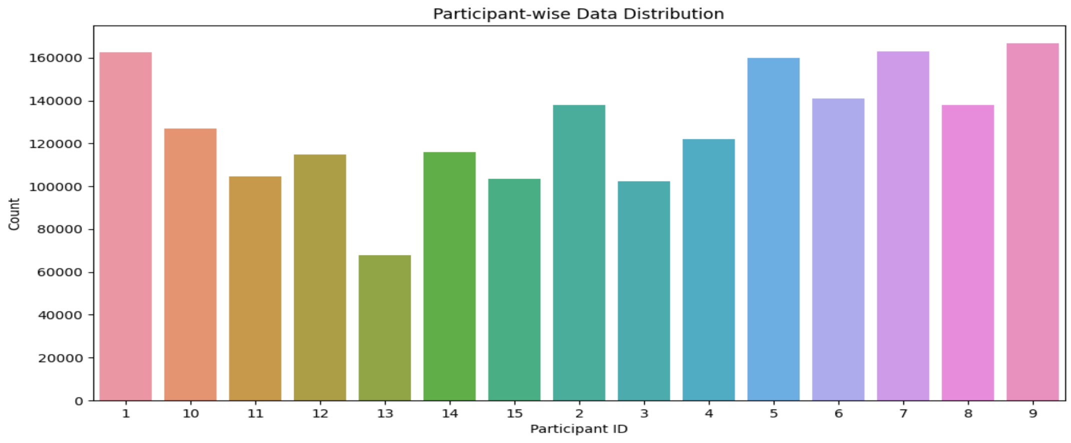
Exploratory Data Analysis (EDA)

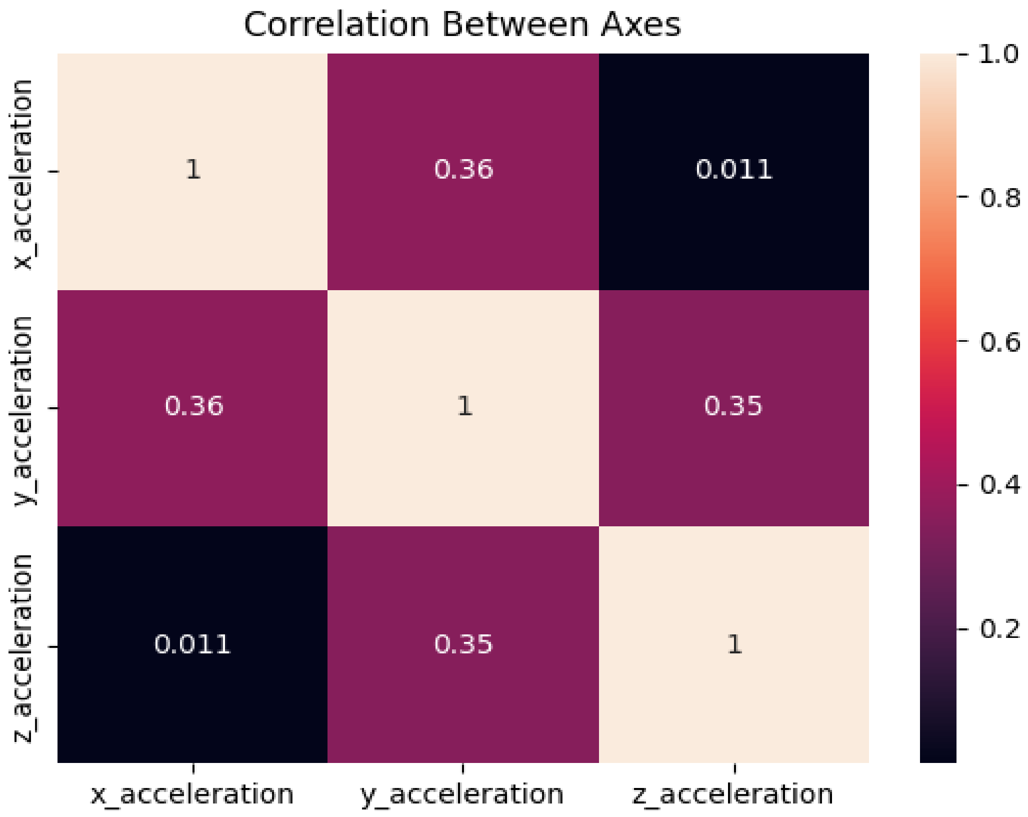
The EDA focused on visual and statistical analyses to understand activity patterns and identify features that differentiate activities.

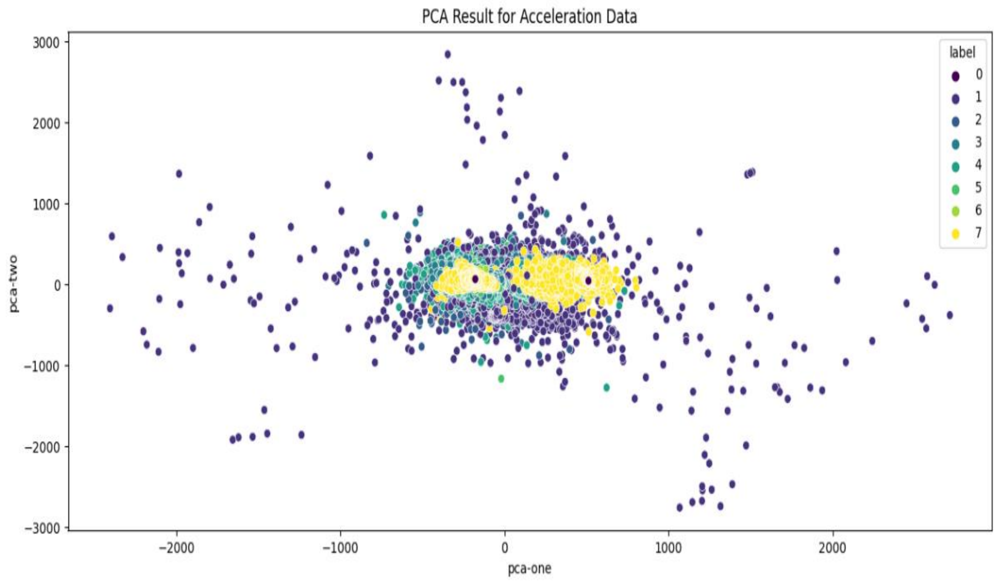
Visualizations

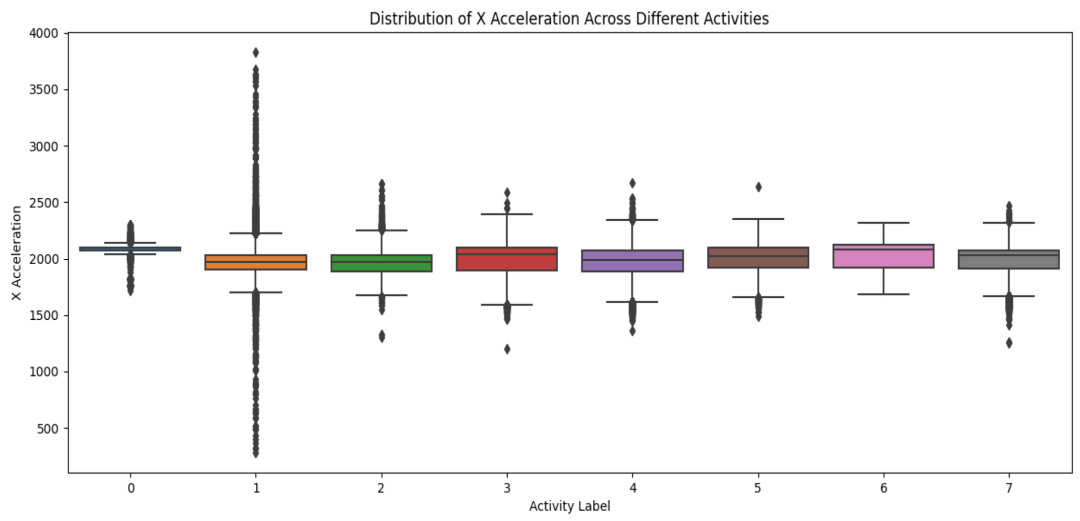
The project includes various graphs that offer insights into the data distribution among activities, correlations between axes, and distinctive patterns of acceleration across activities.

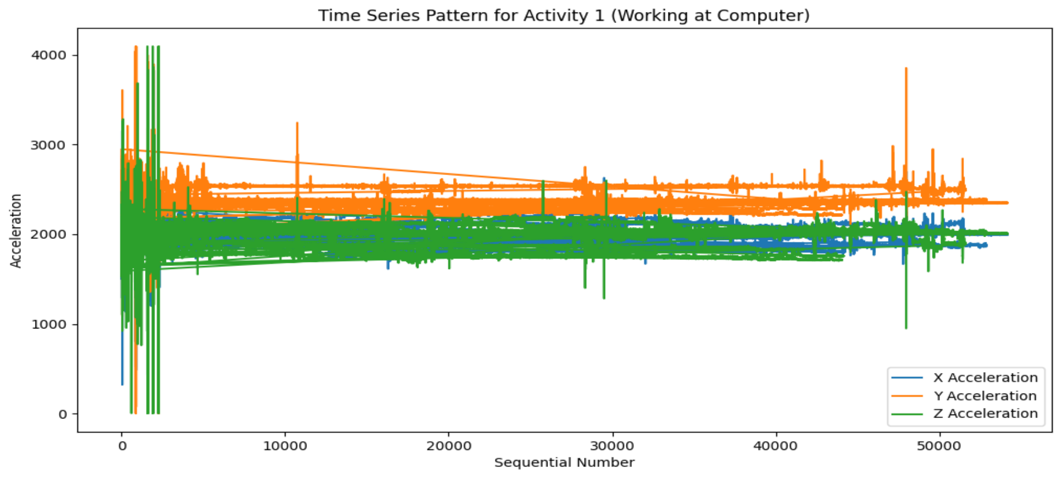


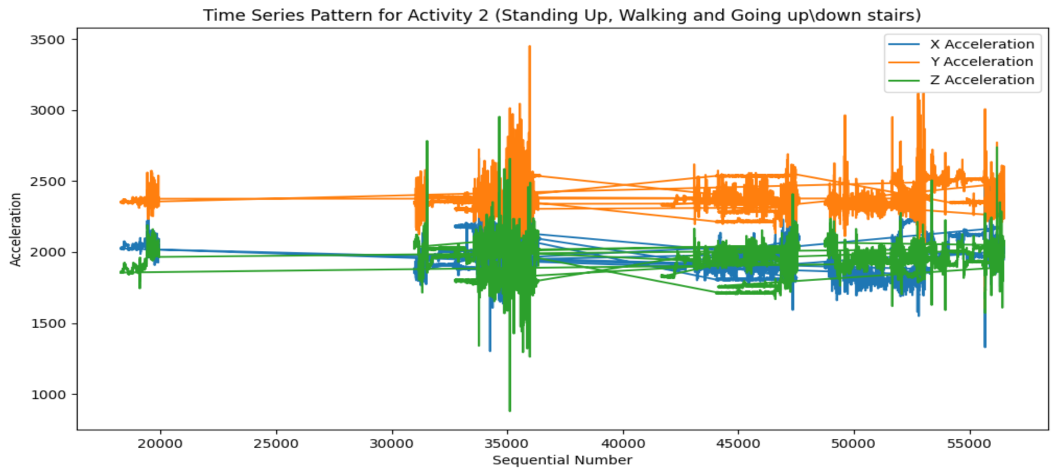


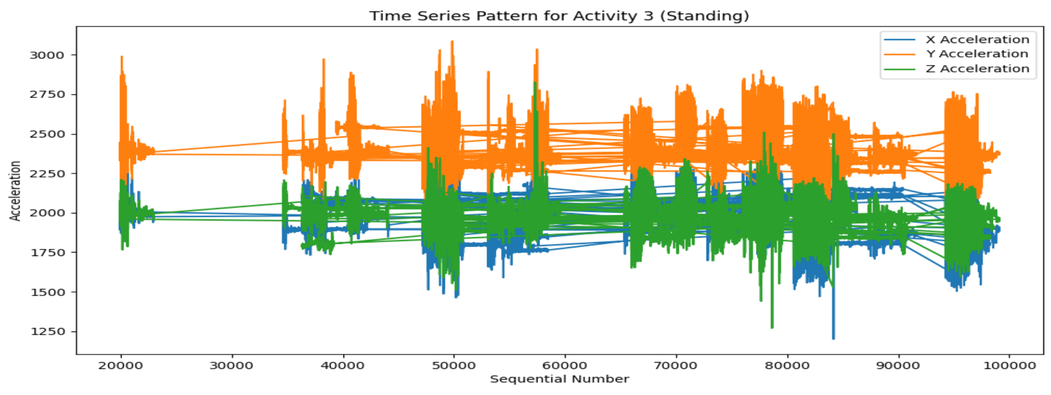

Correlation Between Axes
x_acceleration
y_acceleration
z_acceleration
1
0.36
0.011
0.36
1
0.35
0.011
0.35
1
1.0
0.8
0.6
0.4
0.2

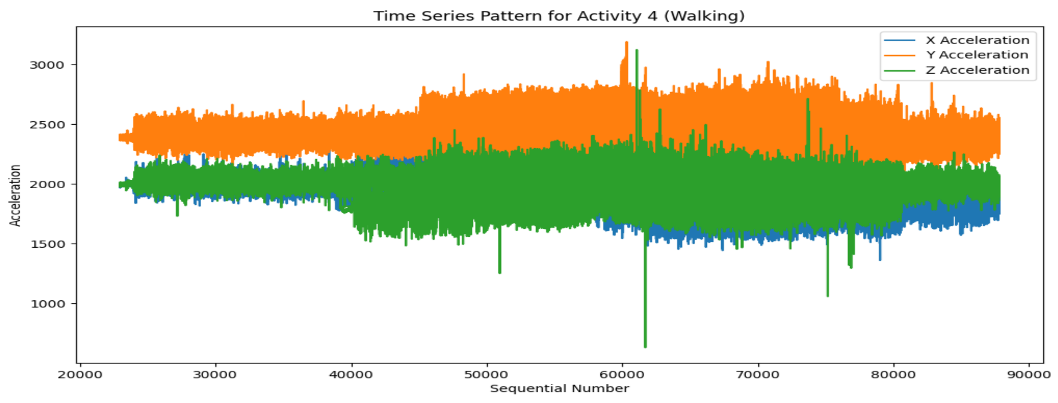

PCA Result for Acceleration Data
label
0
1
2
3
4
5
6
7
pca-one
pca-two
3000
2000
1000
0
−1000
−2000
−3000
−2000
−1000
0
1000
2000

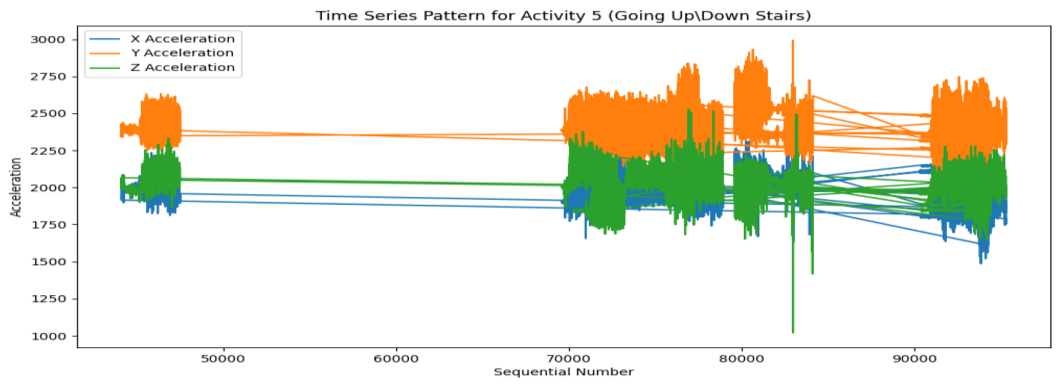

Distribution of X Acceleration Across Different Activities
X Acceleration
Activity Label
0
1
2
3
4
5
6
7
500
1000
1500
2000
2500
3000
3500
4000

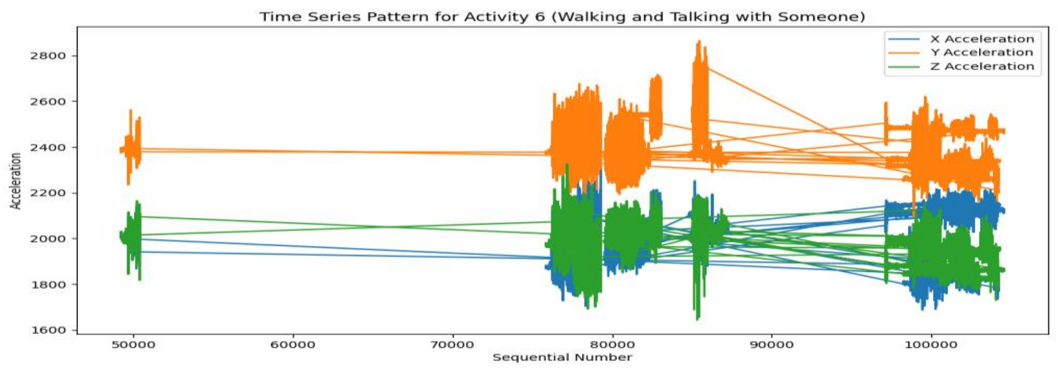

Time Series Pattern for Activity 1 (Working at Computer)
Acceleration
Sequential Number
X Acceleration
Y Acceleration
Z Acceleration
0
10000
20000
30000
40000
50000
0
1000
2000
3000
4000

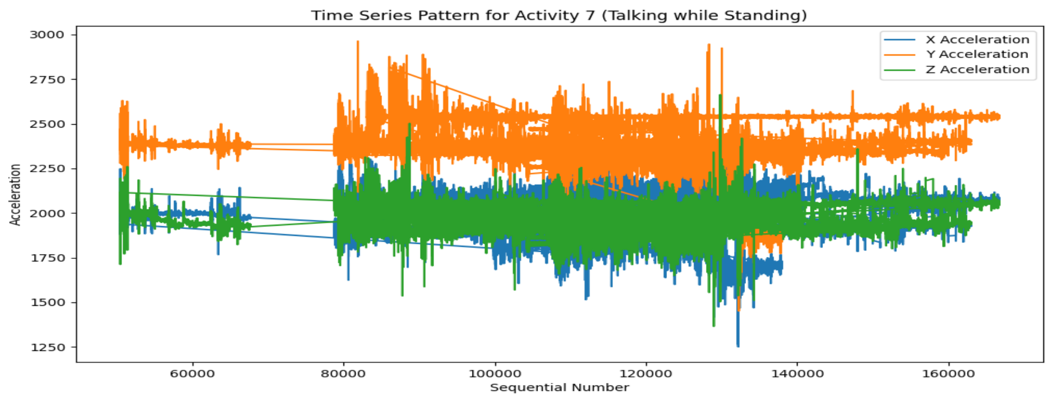

Time Series Pattern for Activity 2 (Standing Up, Walking and Going up\down stairs)
Acceleration
Sequential Number
X Acceleration
Y Acceleration
Z Acceleration
20000
25000
30000
35000
40000
45000
50000
55000
1000
1500
2000
2500
3000
3500


Time Series Pattern for Activity 3 (Standing)
X Acceleration
Y Acceleration
Z Acceleration
Acceleration
Sequential Number
3000
2750
2500
2250
2000
1750
1500
1250
20000
30000
40000
50000
60000
70000
80000
90000
100000


Time Series Pattern for Activity 4 (Walking)
X Acceleration
Y Acceleration
Z Acceleration
Acceleration
Sequential Number
3000
2500
2000
1500
1000
20000
30000
40000
50000
60000
70000
80000
90000


Time Series Pattern for Activity 5 (Going Up\Down Stairs)
X Acceleration
Y Acceleration
Z Acceleration
Acceleration
Sequential Number
3000
2750
2500
2250
2000
1750
1500
1250
1000
50000
60000
70000
80000
90000


Time Series Pattern for Activity 6 (Walking and Talking with Someone)
X Acceleration
Y Acceleration
Z Acceleration
Acceleration
Sequential Number
2800
2600
2400
2200
2000
1800
1600
50000
60000
70000
80000
90000
100000



Feature Engineering

Feature engineering was crucial to extract meaningful information from raw data, facilitating the creation of a feature matrix X and a target vector y.

```
# Define features and labels

X = data[['x_acceleration', 'y_acceleration', 'z_acceleration']]
y = data['label']
```

Model Training

A Random Forest Classifier was chosen for its efficacy in handling complex datasets. The training involved handling class imbalances with SMOTE and fine-tuning model parameters using GridSearchCV.

```
# Address class imbalance

smote = SMOTE(random_state=42)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)
```

```
# Hyperparameter tuning

param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [10, 20, None],
    'min_samples_split': [2, 5, 10]
}
grid_search = GridSearchCV(estimator=RandomForestClassifier(random_state=42, class_weight='balanced'),
                           param_grid=param_grid,
                           cv=3,
                           n_jobs=-1,
                           verbose=2)
grid_search.fit(X_train_resampled, y_train_resampled)
best_params = grid_search.best_params_
```

Model Parameters

After hyperparameter tuning, the best parameters were applied to the Random Forest Classifier. The model was trained on a resampled dataset to account for class imbalance.

```
# Train the model using the best parameters

rf_classifier = RandomForestClassifier(**best_params, random_state=42, class_weight='balanced')
rf_classifier.fit(X_train_resampled, y_train_resampled)
```

Model Evaluation

The model's performance was evaluated on a held-out test set. Classification metrics such as precision, recall, f1-score, and accuracy were computed to assess the model's predictive capabilities.

Classification Report

The model's performance exhibits notable strengths and areas for improvement, as evidenced by the evaluation metrics. Activities '1' and '7' are classified with high precision and recall, reflecting the model's capability to accurately identify these activities with minimal false positives and negatives. The overall accuracy of the model stands at 72%, which, while moderate, could be skewed by the imbalanced nature of the dataset. This is further highlighted by the macro average scores being lower than the weighted averages, indicating that the performance across various activities is not uniform. The disparities in the number of examples for each activity, as shown by the support values, emphasize the impact of class distribution on the model's training and suggest a need for a more balanced dataset to enhance learning and generalization across all activities.

```
# Predict and evaluate

y_pred = rf_classifier.predict(X_test)
print(classification_report(y_test, y_pred))
```

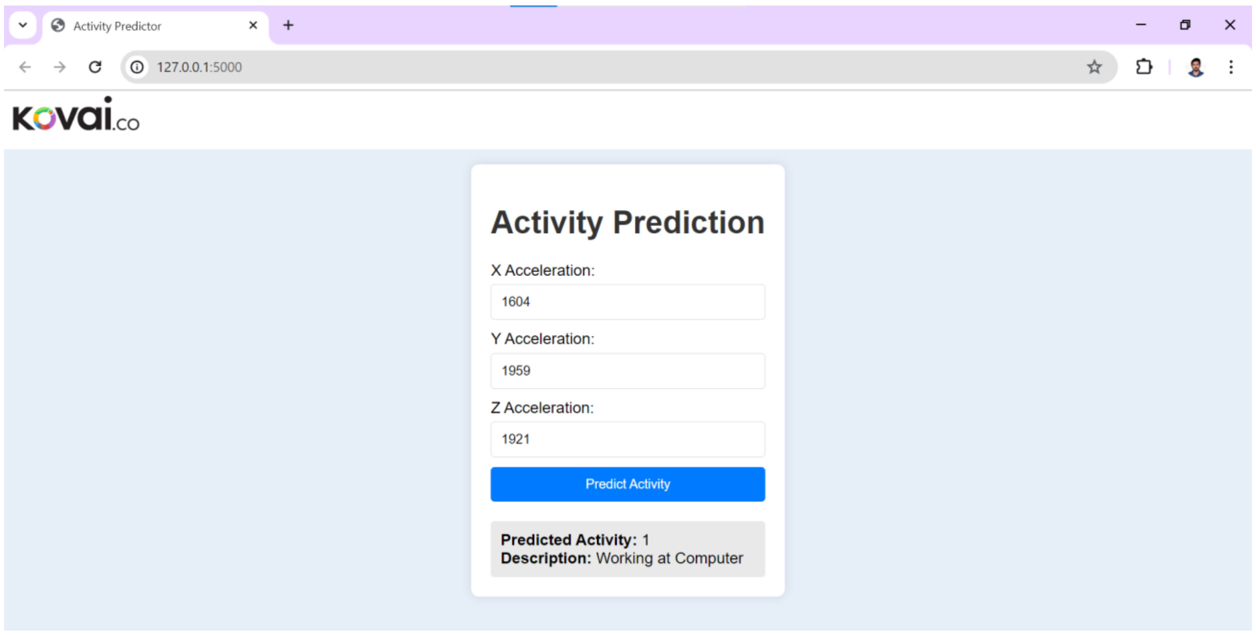
```
              precision    recall  f1-score   support

           0       0.17      0.07      0.10       729
           1       0.86      0.88      0.87    121927
           2       0.39      0.20      0.27      9468
           3       0.53      0.43      0.47     43166
           4       0.61      0.71      0.66     71441
           5       0.27      0.13      0.18     10380
           6       0.37      0.22      0.27      9720
           7       0.75      0.80      0.77    118549

    accuracy                           0.72    385380
   macro avg       0.50      0.43      0.45    385380
weighted avg       0.70      0.72      0.71    385380
```

Interface Development

A user-friendly interface was developed using a Flask server, which allows the input of new accelerometer data and displays the predicted activity. The interface is supported by a robust back-end API that handles data processing and model predictions.



Activity Prediction

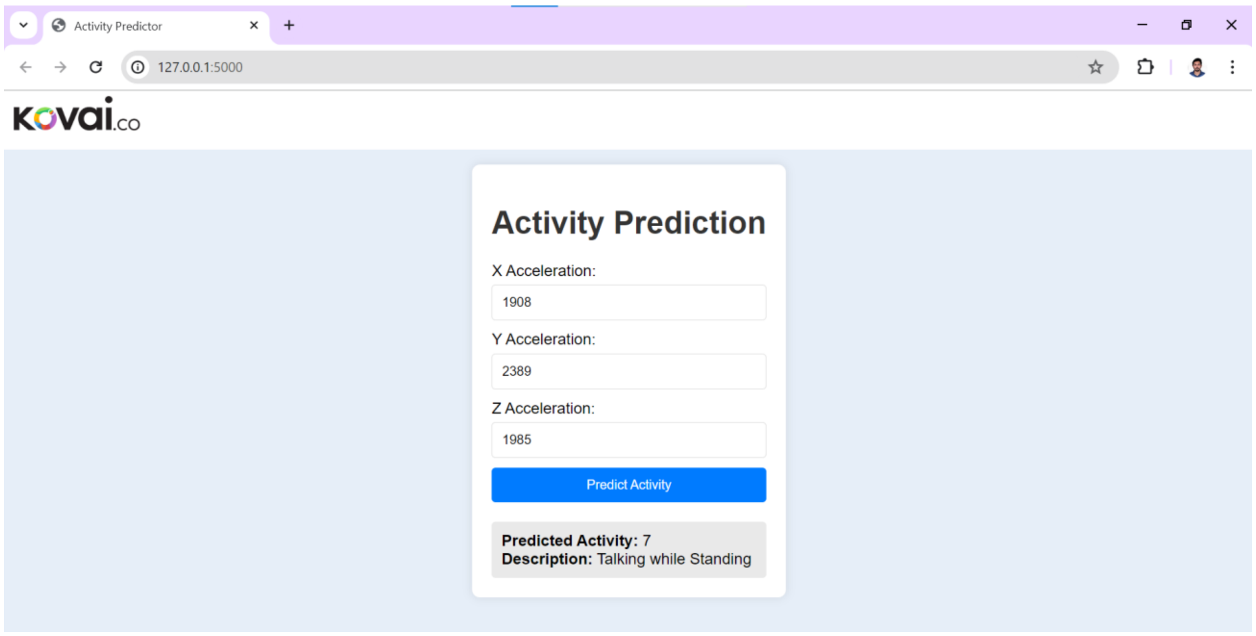
X Acceleration: 1604

Y Acceleration: 1959

Z Acceleration: 1921

Predict Activity

Predicted Activity: 1
Description: Working at Computer



Activity Prediction

X Acceleration: 1908

Y Acceleration: 2389

Z Acceleration: 1985

Predict Activity

Predicted Activity: 7
Description: Talking while Standing

Conclusion

The successful completion of this project demonstrates the potential of machine learning in interpreting complex datasets from wearable devices. Our model offers a reliable method for activity classification, which can be integrated into health and fitness applications.

Future Work and Recommendations

The completion of this project sets a solid foundation for further enhancement and scalability. The following points outline the prospective directions for advancement:

1. **Dataset Expansion:** Acquiring a more extensive dataset can improve the robustness of the model. A larger variety of data will aid in refining the model's generalization capabilities and its performance across a broader spectrum of activities.
2. **Real-time Prediction:** Integrating the model into a real-time prediction framework would significantly increase its utility, enabling on-the-fly classification of activities, which can be particularly beneficial for immediate feedback in health and fitness applications.
3. **Complex Activity Classification:** The current model can be expanded to differentiate between more subtle and complex activities. This involves fine-tuning existing features and potentially engineering new ones that can capture nuanced differences in movements.
4. **Incorporating Advanced Models:** Although the current project was constrained by the computational limitations of a personal laptop, future iterations could leverage high-performance computing resources. Access to GPUs and more powerful infrastructure would allow for the exploration of sophisticated models such as Long Short-Term Memory (LSTM) networks, which are particularly adept at handling time-series data and can potentially unveil deeper insights into the temporal dynamics of activities.
5. **Recommendation:** It is recommended to invest in enhanced computational resources. The use of GPUs and dedicated hardware can reduce the time taken for model training and hyperparameter tuning, thereby streamlining the development cycle. With these resources, more complex algorithms that require greater computational power, such as deep learning models, can be harnessed to their full potential, leading to improved accuracies and capabilities of the activity classification system.