Feature Engineering and Data Augmentation Report

Introduction

The purpose of this report is to document the feature engineering process for detecting anomalies using smartphone sensor data in crowded environments. Anomalies in this context can include unusual or unexpected movements such as sudden stops, sharp turns, excessive acceleration, or erratic behavior. By analyzing data from smartphone sensors like accelerometers, gyroscopes, and GPS, we can identify and flag these anomalies.

The data collected from sensors includes measurements such as:

```
i.Longitude & Latitude (GPS-based location data)

ii.Speed (derived from GPS data)

iii.Accelerometer data (X, Y, Z) – Measures linear acceleration
along three axes

iv.Gyroscope data (X, Y, Z) – Measures angular velocity around t
hree axes

v.Heading (Direction of movement)

vi.Time (Timestamps of data collection)
```

New Features Created

```
1. Speed Change
```

Description:Measures the difference in speed between consecutive time steps to detect rapid accelerations or sudden stops.

```
In [9]:
```

```
df['Speed_Change'] = df['Speed'].diff()
```

Use Case: Large speed changes within short time periods (e.g., sudden acceleration or deceleration) may indicate abnormal behavior such as running in a crowd or an unexpected halt.

2. Direction Change

Description: Captures the change in heading between consecutive time steps, useful for detecting sharp turns or erratic movements.

```
In [10]:
```

```
df['Direction_Change'] = df['Heading'].diff().fillna(0)
```

Use Case: Sudden or frequent changes in direction can indicate erratic behavior or potentially evasive movement in a crowd.

3. Time Difference

Description: Time difference between consecutive readings.

A large or inconsistent time gap might indicate data collection issues or sudden stops.

In [11]:

```
df['Time'] = pd.to_datetime(df['Time'], format='%H-%M-%S')
df['Time_Change'] = df['Time'].diff().dt.total_seconds().fillna(0)
```

Use Case: Time changes help in calculating time-based features such as speed or acceleration over time.

4. Acceleration Rate and Braking Intensity

Description: Acceleration rate is derived by dividing speed change by time change, while braking intensity isolates negative acceleration (deceleration).

In [12]:

```
df['Acceleration_Rate'] = df['Speed_Change'] / df['Time_Change']
df['Braking_Intensity'] = df['Acceleration_Rate'].apply(lambda x: x if x < 0 else 0)
```

Use Case: Can help in identifying instances of aggressive acceleration and braking behavior, which may be anomalous in crowd movement.

5. Acceleration Magnitude

Description: This feature calculates the magnitude of acceleration using the X, Y, and Z components of the accelerometer.

In [13]:

```
df['Acc_Magnitude'] = np.sqrt(df['Acc X']**2 + df['Acc Y']**2 + df['Acc Z']**2)
```

Use Case:A sudden increase in acceleration magnitude followed by a sharp decline could indicate a trip or a fall in a crowded area.

6. Jerk (Rate of Change of Acceleration)

Description: This feature measures the rate of change of the total acceleration magnitude over time, which can be used to detect sudden changes in motion.

In [14]:

```
df['Jerk'] = df['Acc_Magnitude'].diff() / df['Time_Change']
```

Use Case: Detects sharp or sudden movements, like jerks, which may indicate abnormal or erratic behavior in a crowd.

7. Cumulative Distance

Description: This feature tracks the total distance traveled by summing up the distance values over time.

In [15]:

```
df['Cumulative_Distance'] = df['Distance'].cumsum()
```

Use Case: Useful for tracking overall movement trends and identifying anomalies where there is unexpected stopping or erratic travel behavior

8. Speed Variance

Description: This feature calculates the rolling variance of speed over a defined window (e.g., 5 time points).

In [16]:

```
df['Speed_Variance'] = df['Speed'].rolling(window=5).var()
```

Use Case: Identifies periods of unstable movement by highlighting variations in speed over time.
9.Rolling Mean and Variance for Sensor Data

Accelerometer and Gyroscope Data:

Rolling means and variances were computed for the accelerometer (X, Y, Z) and gyroscope (X, Y, Z) data to smooth the data and reveal trends.

In [17]:

```
window_size = 5
# Rolling mean and variance for accelerometer data
df['Rolling_Acc_X_Mean'] = df['Acc X'].rolling(window=window_size).mean()
df['Rolling_Acc_Y_Mean'] = df['Acc Y'].rolling(window=window_size).mean()
df['Rolling_Acc_Z_Mean'] = df['Acc Z'].rolling(window=window_size).mean()
df['Rolling_Acc_X_Var'] = df['Acc X'].rolling(window=window_size).var()
df['Rolling_Acc_Y_Var'] = df['Acc Y'].rolling(window=window_size).var()
df['Rolling_Acc_Z_Var'] = df['Acc Z'].rolling(window=window_size).var()

# Rolling mean and variance for gyroscope data
df['Rolling_Gyro_X_Mean'] = df['gyro_x'].rolling(window=window_size).mean()
df['Rolling_Gyro_Y_Mean'] = df['gyro_y'].rolling(window=window_size).mean()
df['Rolling_Gyro_Z_Mean'] = df['gyro_z'].rolling(window=window_size).mean()
df['Rolling_Gyro_X_Var'] = df['gyro_x'].rolling(window=window_size).var()
df['Rolling_Gyro_Y_Var'] = df['gyro_y'].rolling(window=window_size).var()
df['Rolling_Gyro_Z_Var'] = df['gyro_z'].rolling(window=window_size).var()
```

10. Total Acceleration and Gyroscope Magnitudes

Description: Magnitudes of acceleration and gyroscope data computed using the Euclidean formula.

In [18]:

```
df['Total_Acc'] = np.sqrt(df['Acc X']**2 + df['Acc Y']**2 + df['Acc Z']**2)
df['Total_Gyro'] = np.sqrt(df['gyro_x']**2 + df['gyro_y']**2 + df['gyro_z']**2)
```

Correlation Analysis

To assess the relationships between the newly engineered features and existing features, a correlation matrix was computed. The correlation matrix helps to identify strong relationships that could indicate potential anomalies in behavior.

In [19]:

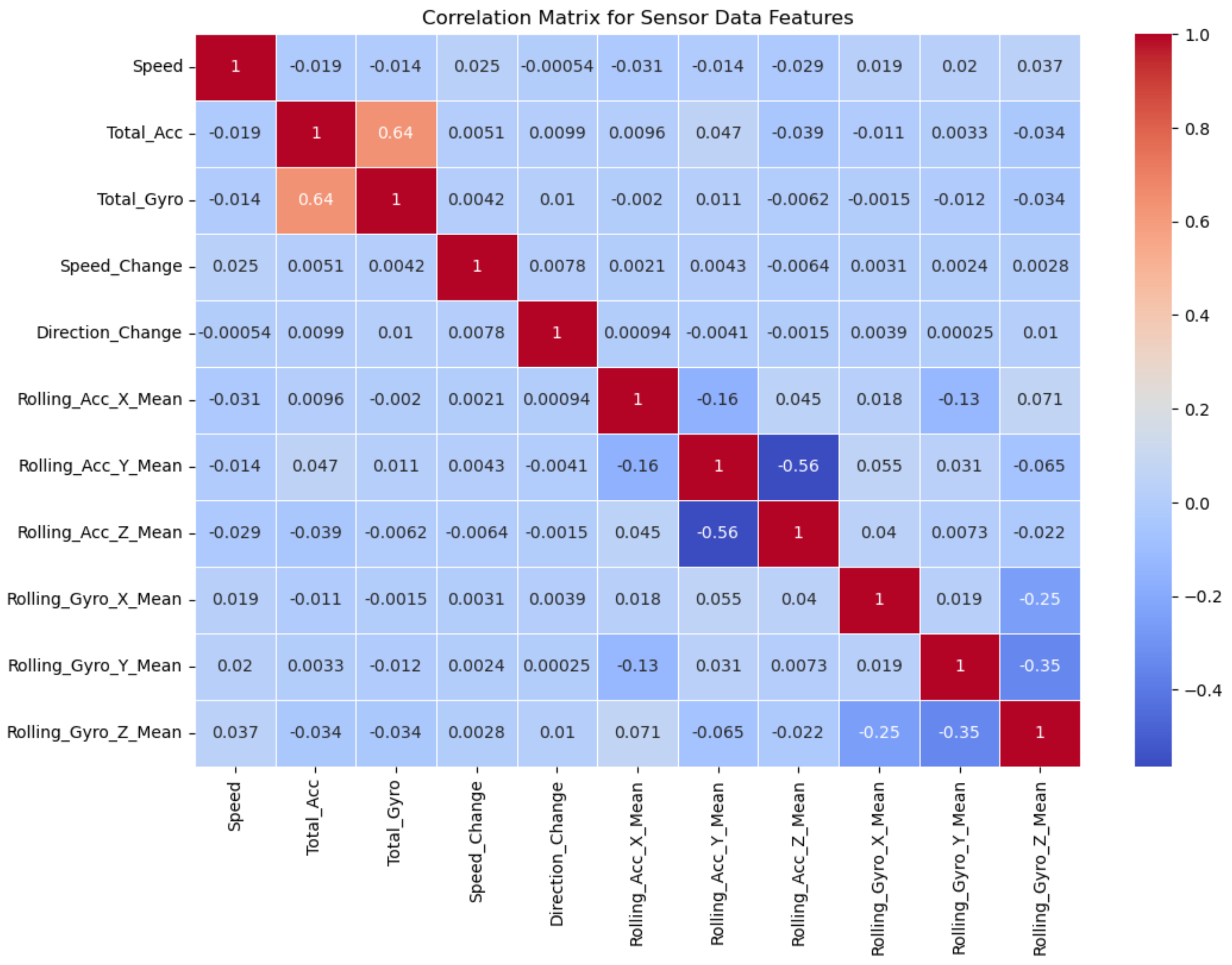
```
correlation_matrix = df[['Speed', 'Total_Acc', 'Total_Gyro', 'Speed_Change',
            'Direction_Change', 'Rolling_Acc_X_Mean', 'Rolling_Acc_Y_Mean',
            'Rolling_Acc_Z_Mean', 'Rolling_Gyro_X_Mean', 'Rolling_Gyro_Y_Mean',
            'Rolling_Gyro_Z_Mean']].corr()
```

Visualization of Correlation Matrix

A heatmap of the correlation matrix was plotted for better visual interpretation of the relationships between features.

In [20]:

```python
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Matrix for Sensor Data Features')
plt.show()
```



Conclusion:

The newly created features provide valuable insights into the movement patterns captured by smartphone sensors. The metrics for speed change, direction change, acceleration rate, braking intensity, and jerk will be instrumental in analyzing dynamic behavior and detecting anomalies in crowds. Further analysis can be conducted to derive actionable insights based on the relationships revealed in the correlation analysis.