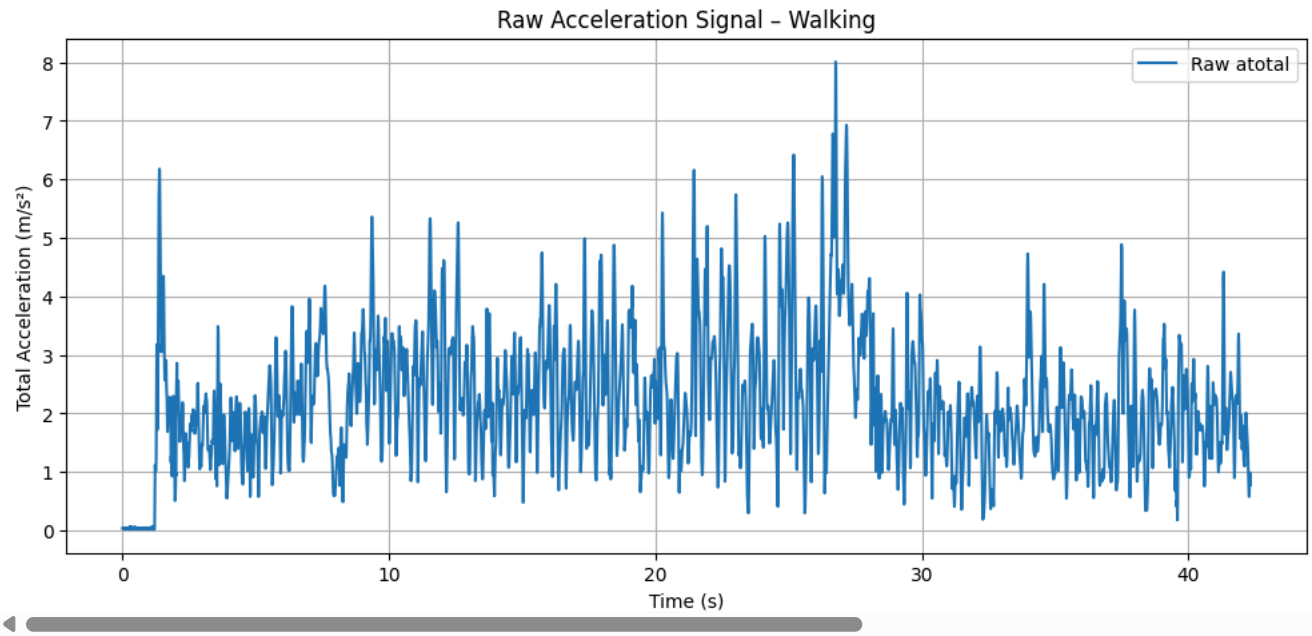
```python
# 📌 Step 1: Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from google.colab import files

# 📌 Step 2: Upload your CSV file
uploaded = files.upload()
```

Dosyaları Seç Dosya seçilmedi Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```python
# 📌 Step 3: Load the CSV File
# Replace 'your_file.csv' with your actual file name, e.g., 'accelerometer_walking.csv'
df = pd.read_csv('accelerometer_walking.csv')

# Preview the data
df.head()
```

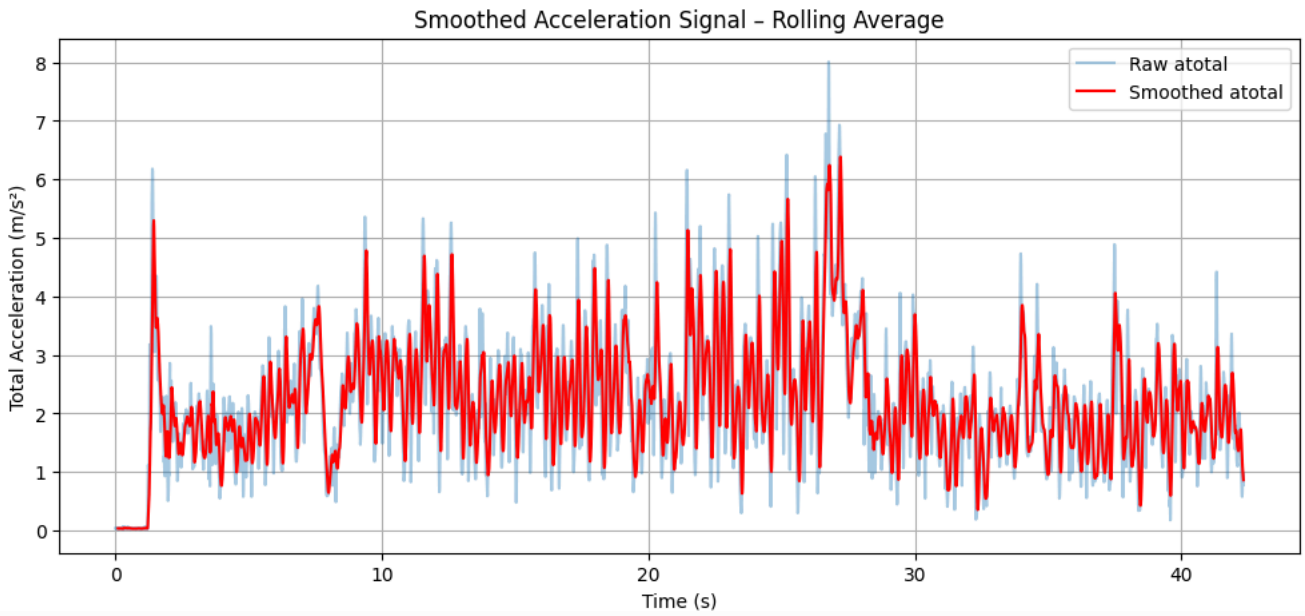
	time	ax	ay	az	atotal
0	0.00	0.01	-0.01	-0.03	0.04
1	0.01	0.02	-0.02	-0.02	0.04
2	0.02	-0.01	-0.02	-0.05	0.05
3	0.03	0.00	-0.02	-0.03	0.03
4	0.04	0.01	-0.01	-0.02	0.03

```python
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 5))
plt.plot(df['time'], df['atotal'], label='Raw atotal')
plt.xlabel('Time (s)')
plt.ylabel('Total Acceleration (m/s²)')
plt.title('Raw Acceleration Signal – Walking')
plt.legend()
plt.grid(True)
plt.show()
```




```python
# Apply rolling average to smooth out the total acceleration
df['atotal_smooth'] = df['atotal'].rolling(window=10).mean()

# Plot both raw and smoothed signals
plt.figure(figsize=(12, 5))
plt.plot(df['time'], df['atotal'], alpha=0.4, label='Raw atotal')
plt.plot(df['time'], df['atotal_smooth'], color='red', label='Smoothed atotal')
plt.xlabel('Time (s)')
plt.ylabel('Total Acceleration (m/s²)')
plt.title('Smoothed Acceleration Signal – Rolling Average')
plt.legend()
```

```
plt.grid(True)
plt.show()
```




```
# 📌 Upload the remaining activity files
from google.colab import files
uploaded = files.upload()
```

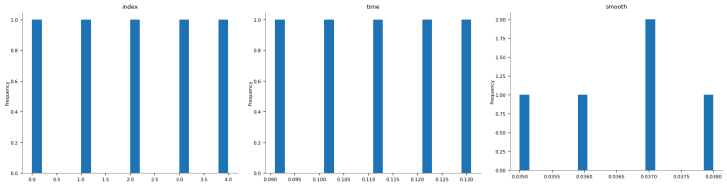
Dosyaları Seç Dosya seçilmedi Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

Saving accelerometer_running.csv to accelerometer_running.csv

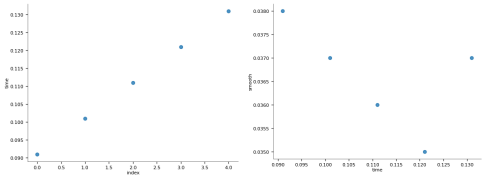
```
# 📌 Step 1: Load all three datasets
df_walk = pd.read_csv('accelerometer_walking.csv')
df_run = pd.read_csv('accelerometer_running.csv')
df_stairs = pd.read_csv('accelerometer_stairs.csv')

# 📌 Step 2: Add labels
df_walk['label'] = 'walking'
df_run['label'] = 'running'
df_stairs['label'] = 'stairs'


# 📌 Step 3: Apply rolling average smoothing
df_walk['smooth'] = df_walk['atotal'].rolling(window=10).mean()
df_run['smooth'] = df_run['atotal'].rolling(window=10).mean()
df_stairs['smooth'] = df_stairs['atotal'].rolling(window=10).mean()

# 📌 Step 4: Drop rows with NaN values created by rolling average
df_walk.dropna(inplace=True)
df_run.dropna(inplace=True)
df_stairs.dropna(inplace=True)

# 📌 Step 5: Combine into one labeled dataset
df_all = pd.concat([df_walk, df_run, df_stairs], ignore_index=True)

# 📌 Optional: Preview the final dataset
df_all[['time', 'smooth', 'label']].head()
```

	time	smooth	label
0	0.091	0.038	walking
1	0.101	0.037	walking
2	0.111	0.036	walking
3	0.121	0.035	walking
4	0.131	0.037	walking

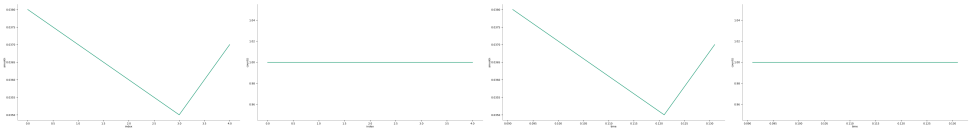
Distributions



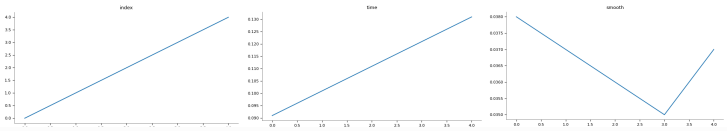
2-d distributions



Time series



Values



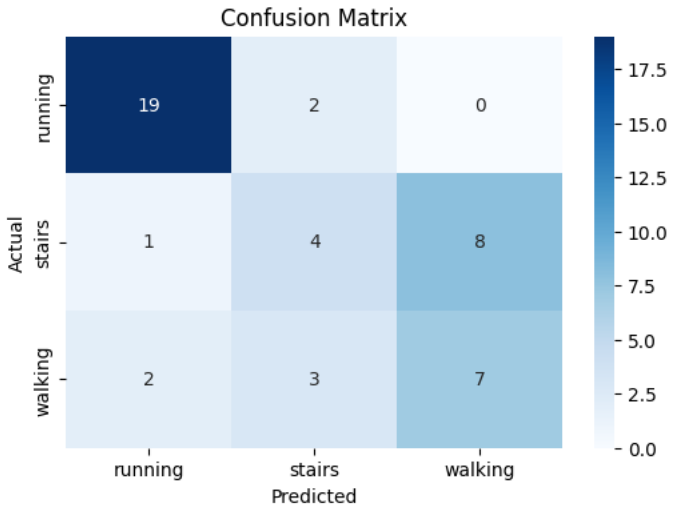
```
# Define window size (e.g., 50 rows per segment ≈ 2–3 seconds depending on sampling rate)
window_size = 50

features = []
labels = []

# Slide through the dataset in steps
for i in range(0, len(df_all) - window_size, window_size):
    segment = df_all.iloc[i:i + window_size]

    # Extract features from this window
    mean = segment['smooth'].mean()
    std = segment['smooth'].std()
    max_val = segment['smooth'].max()

    # Take the most frequent label in this segment as the label
    label = segment['label'].mode()[0]

    features.append([mean, std, max_val])
    labels.append(label)

# Create a new DataFrame with features
import pandas as pd
df_features = pd.DataFrame(features, columns=['mean', 'std', 'max'])
df_features['label'] = labels

# Preview
df_features.head()
```

	mean	std	max	label
0	0.03952	0.002845	0.044	walking
1	0.03608	0.002320	0.042	walking
2	2.33066	1.867286	5.299	walking
3	1.95612	0.553298	3.393	walking
4	1.66276	0.323378	2.447	walking

```python
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Split features and labels
X = df_features[['mean', 'std', 'max']]
y = df_features['label']

# Train/test split (80% training, 20% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and train the model
model = DecisionTreeClassifier()
model.fit(X_train, y_train)

# Predict on test data
y_pred = model.predict(X_test)

# Accuracy & classification report
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.6521739130434783

Classification Report:
               precision    recall  f1-score   support

      running       0.86      0.90      0.88        21
       stairs       0.44      0.31      0.36        13
      walking       0.47      0.58      0.52        12

     accuracy                           0.65        46
    macro avg       0.59      0.60      0.59        46
 weighted avg       0.64      0.65      0.64        46
```

```python
# Visualize confusion matrix
plt.figure(figsize=(6, 4))
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d',
            cmap='Blues', xticklabels=model.classes_, yticklabels=model.classes_)
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```




```python
#CLASS ASSIGMENT
```

```python
# Step 1: Import Required Libraries and give them nicknames.

import pandas as pd
# for handling data
import numpy as np
# used for numerical processing
import matplotlib.pyplot as plt
# for plotting the signals
from google.colab import files
# for uploading files
```

```
11
12 # Step 2: Upload the CSV file. File uploader.
13 uploaded = files.upload()
```

Dosyaları Seç 4 dosya

- **accelerometer_jumping.csv**(text/csv) - 82864 bytes, last modified: 22.04.2025 - 100% done
- **accelerometer_running.csv**(text/csv) - 121750 bytes, last modified: 03.04.2025 - 100% done
- **accelerometer_stairs.csv**(text/csv) - 87659 bytes, last modified: 03.04.2025 - 100% done
- **accelerometer_walking.csv**(text/csv) - 119123 bytes, last modified: 03.04.2025 - 100% done

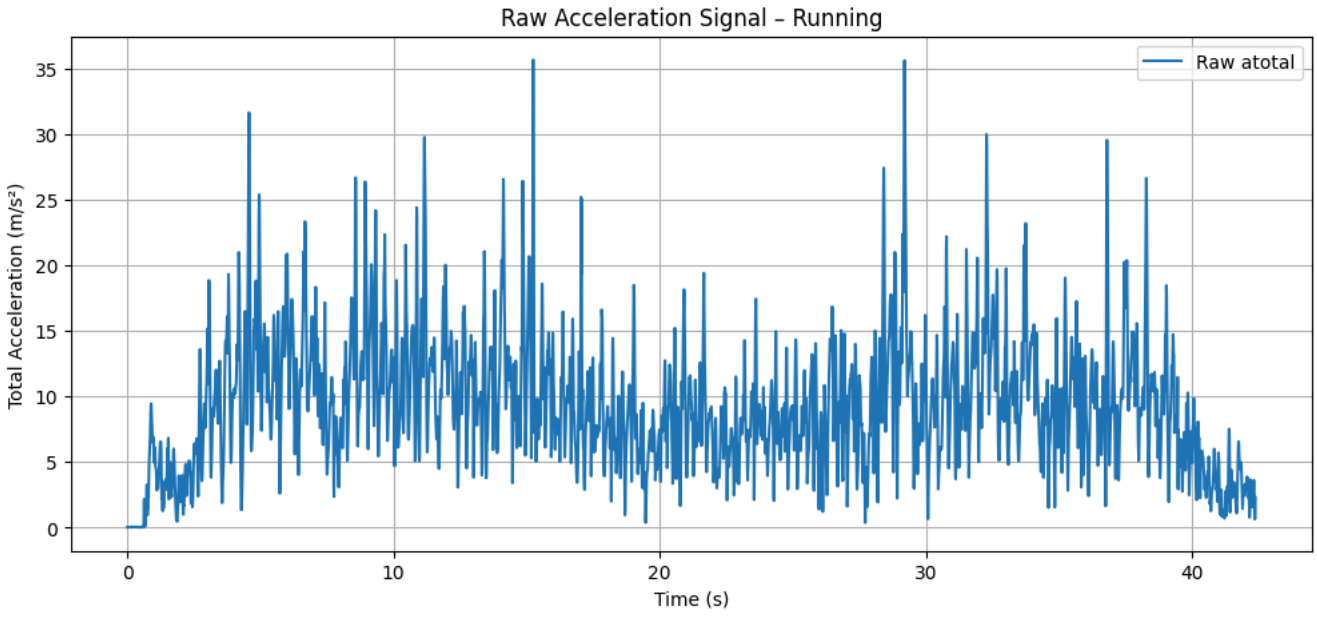
```
Saving accelerometer_jumping.csv to accelerometer_jumping.csv
Saving accelerometer_running.csv to accelerometer_running.csv
Saving accelerometer_stairs.csv to accelerometer_stairs.csv
Saving accelerometer_walking.csv to accelerometer_walking.csv
```

```
1 # Step 3: Load the CSV File, df is data frame. We need to choose the running file, because it's the one mentioned in our homework.
2 df = pd.read_csv('accelerometer_running.csv')
3
4
5 # Step 4: We want to preview the first few rows to ensure there is data in the file. This also helps to see the column headers.
6 # To make sure the headers are there and that they are correct.
7 df.head()
8
```

	time	ax	ay	az	atotal
0	0.00	0.01	-0.01	-0.04	0.04
1	0.01	0.00	-0.00	-0.05	0.05
2	0.02	0.02	-0.03	-0.03	0.04
3	0.03	0.02	-0.01	-0.04	0.05
4	0.04	0.01	-0.00	-0.03	0.03

Sonraki adımlar: `df` ile kod oluşturun | Önerilen grafikleri göster | New interactive sheet

```
1 #Step 5 : We are importing another library and giving it a nickname. Also, we are specifying what it should do (create a graph).
2 #We made a change to the title. plt.title ('Raw Acceleration Signal - Running') is where we changed it.
3
4 import matplotlib.pyplot as plt
5
6 plt.figure(figsize=(12, 5))
7 plt.plot(df['time'], df['atotal'], label='Raw atotal')
8 plt.xlabel('Time (s)')
9 plt.ylabel('Total Acceleration (m/s²)')
10 plt.title('Raw Acceleration Signal – Running')
11 plt.legend()
12 plt.grid(True)
13 plt.show()
```




```
1 # Apply rolling average to smooth out the total acceleration
2 df['atotal_smooth'] = df['atotal'].rolling(window=10).mean()
3
4 # Plot both raw and smoothed signals.
5 plt.figure(figsize=(12, 5))
```