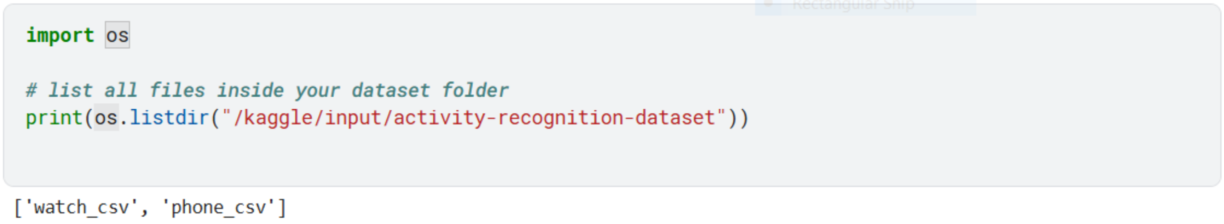
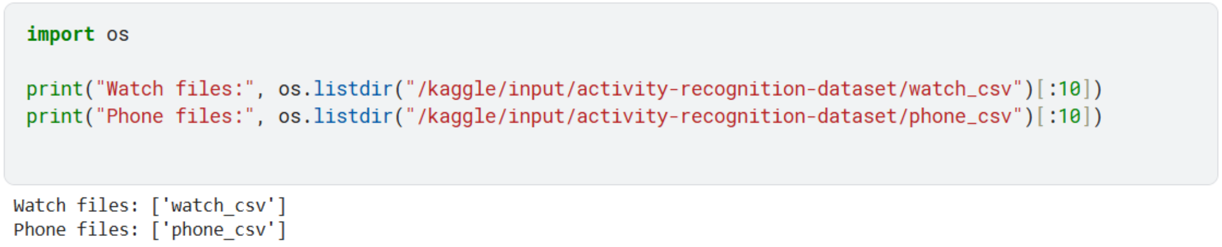
EDA Preparation & Model Planning:

Import Data:



```python
import os

# list all files inside your dataset folder
print(os.listdir("/kaggle/input/activity-recognition-dataset"))
```

```
['watch_csv', 'phone_csv']
```

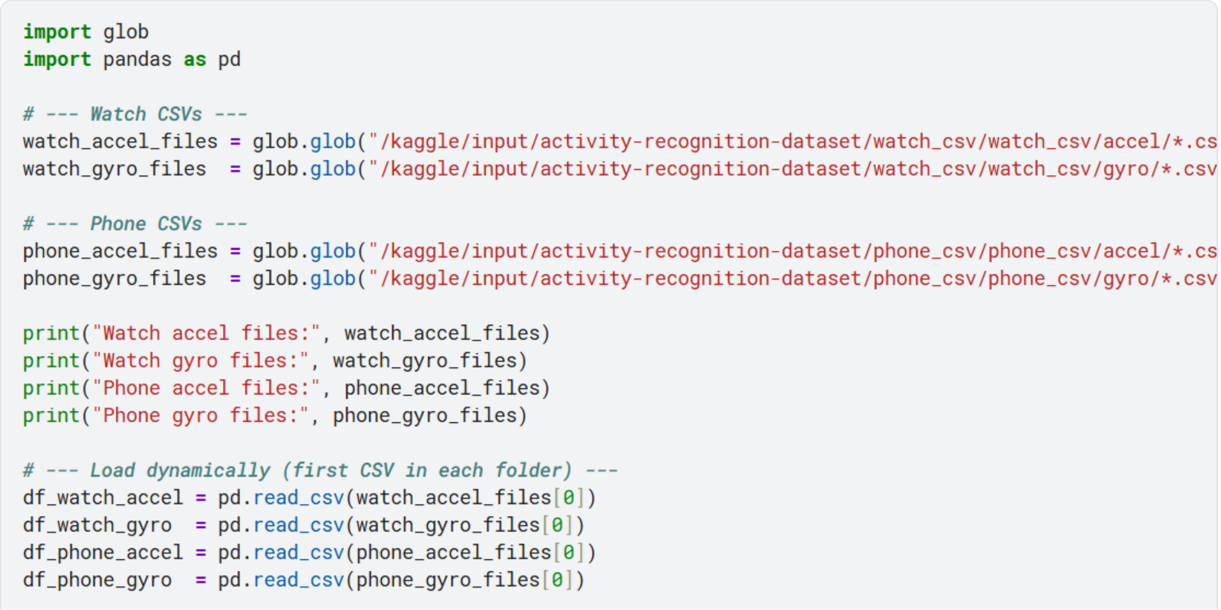


```python
import os

print("Watch files:", os.listdir("/kaggle/input/activity-recognition-dataset/watch_csv")[:10])
print("Phone files:", os.listdir("/kaggle/input/activity-recognition-dataset/phone_csv")[:10])
```

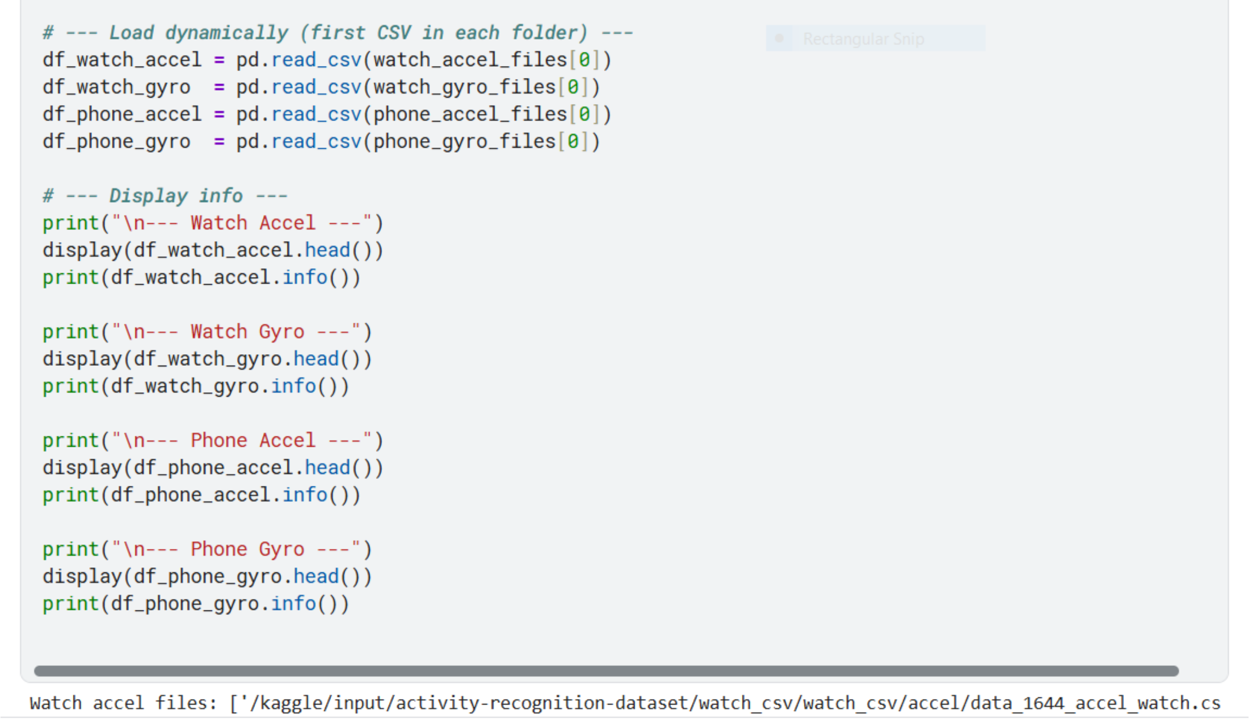
```
Watch files: ['watch_csv']
Phone files: ['phone_csv']
```

```python
# Cell 1 - imports & list files
import os, glob
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

BASE = "/kaggle/input/activity-recognition-dataset"
print("BASE:", BASE)

print("watch_csv list (first 10):")
print(os.listdir(os.path.join(BASE, "watch_csv"))[:20])

print("\nphone_csv list (first 10):")
print(os.listdir(os.path.join(BASE, "phone_csv"))[:20])
```

```
BASE: /kaggle/input/activity-recognition-dataset
watch_csv list (first 10):
['watch_csv']

phone_csv list (first 10):
['phone_csv']
```

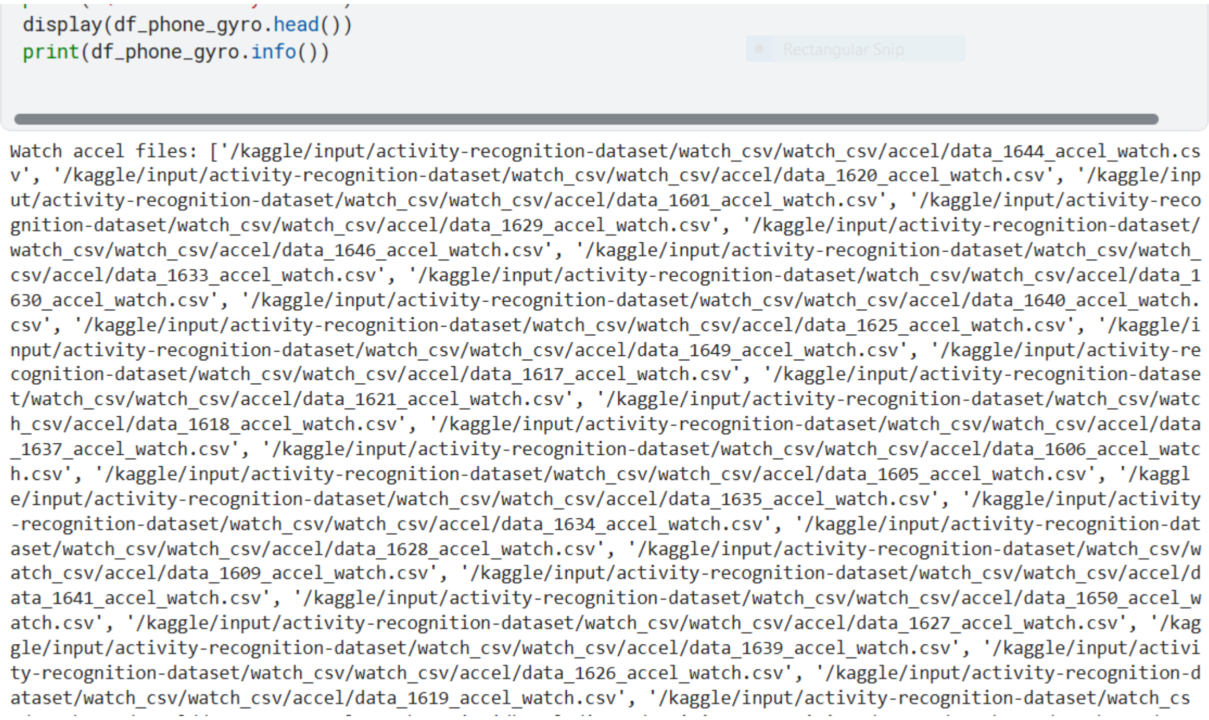


```python
import glob
import pandas as pd

# --- Watch CSVs ---
watch_accel_files = glob.glob("/kaggle/input/activity-recognition-dataset/watch_csv/watch_csv/accel/*.cs
watch_gyro_files  = glob.glob("/kaggle/input/activity-recognition-dataset/watch_csv/watch_csv/gyro/*.csv

# --- Phone CSVs ---
phone_accel_files = glob.glob("/kaggle/input/activity-recognition-dataset/phone_csv/phone_csv/accel/*.cs
phone_gyro_files  = glob.glob("/kaggle/input/activity-recognition-dataset/phone_csv/phone_csv/gyro/*.csv

print("Watch accel files:", watch_accel_files)
print("Watch gyro files:", watch_gyro_files)
print("Phone accel files:", phone_accel_files)
print("Phone gyro files:", phone_gyro_files)

# --- Load dynamically (first CSV in each folder) ---
df_watch_accel = pd.read_csv(watch_accel_files[0])
df_watch_gyro  = pd.read_csv(watch_gyro_files[0])
df_phone_accel = pd.read_csv(phone_accel_files[0])
df_phone_gyro  = pd.read_csv(phone_gyro_files[0])
```



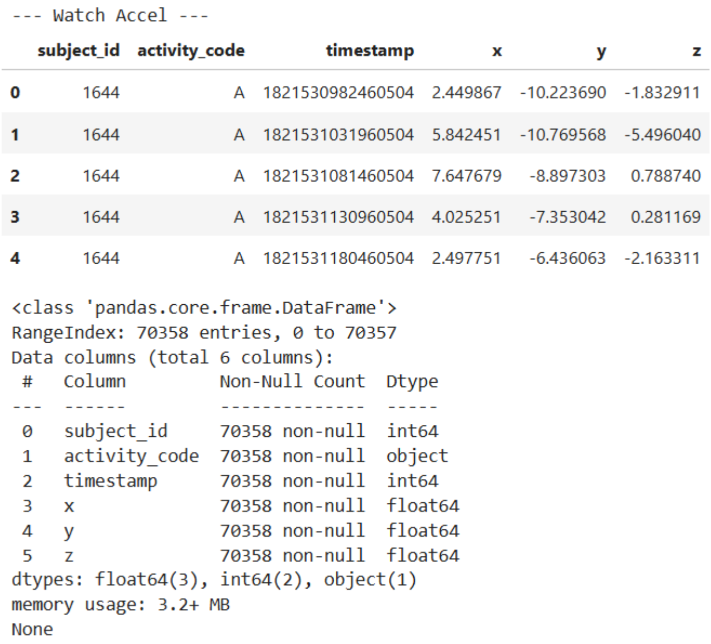
```python
# --- Load dynamically (first CSV in each folder) ---
df_watch_accel = pd.read_csv(watch_accel_files[0])
df_watch_gyro  = pd.read_csv(watch_gyro_files[0])
df_phone_accel = pd.read_csv(phone_accel_files[0])
df_phone_gyro  = pd.read_csv(phone_gyro_files[0])

# --- Display info ---
print("\n--- Watch Accel ---")
display(df_watch_accel.head())
print(df_watch_accel.info())

print("\n--- Watch Gyro ---")
display(df_watch_gyro.head())
print(df_watch_gyro.info())

print("\n--- Phone Accel ---")
display(df_phone_accel.head())
print(df_phone_accel.info())

print("\n--- Phone Gyro ---")
display(df_phone_gyro.head())
print(df_phone_gyro.info())
```

```
Watch accel files: ['/kaggle/input/activity-recognition-dataset/watch_csv/watch_csv/accel/data_1644_accel_watch.cs
```



```
display(df_phone_gyro.head())
print(df_phone_gyro.info())
```

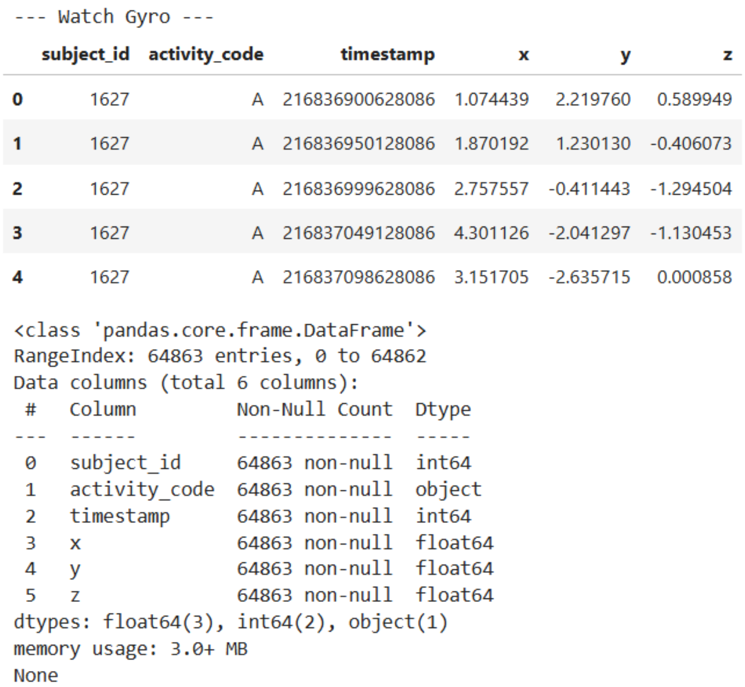
```
Watch accel files: ['/kaggle/input/activity-recognition-dataset/watch_csv/watch_csv/accel/data_1644_accel_watch.cs
v', '/kaggle/input/activity-recognition-dataset/watch_csv/watch_csv/accel/data_1620_accel_watch.csv', '/kaggle/inp
ut/activity-recognition-dataset/watch_csv/watch_csv/accel/data_1601_accel_watch.csv', '/kaggle/input/activity-reco
gnition-dataset/watch_csv/watch_csv/accel/data_1629_accel_watch.csv', '/kaggle/input/activity-recognition-dataset/
watch_csv/watch_csv/accel/data_1646_accel_watch.csv', '/kaggle/input/activity-recognition-dataset/watch_csv/watch_
csv/accel/data_1633_accel_watch.csv', '/kaggle/input/activity-recognition-dataset/watch_csv/watch_csv/accel/data_1
630_accel_watch.csv', '/kaggle/input/activity-recognition-dataset/watch_csv/watch_csv/accel/data_1640_accel_watch.
csv', '/kaggle/input/activity-recognition-dataset/watch_csv/watch_csv/accel/data_1625_accel_watch.csv', '/kaggle/i
nput/activity-recognition-dataset/watch_csv/watch_csv/accel/data_1649_accel_watch.csv', '/kaggle/input/activity-re
cognition-dataset/watch_csv/watch_csv/accel/data_1617_accel_watch.csv', '/kaggle/input/activity-recognition-datase
t/watch_csv/watch_csv/accel/data_1621_accel_watch.csv', '/kaggle/input/activity-recognition-dataset/watch_csv/watc
h_csv/accel/data_1618_accel_watch.csv', '/kaggle/input/activity-recognition-dataset/watch_csv/watch_csv/accel/data
_1637_accel_watch.csv', '/kaggle/input/activity-recognition-dataset/watch_csv/watch_csv/accel/data_1606_accel_watc
h.csv', '/kaggle/input/activity-recognition-dataset/watch_csv/watch_csv/accel/data_1605_accel_watch.csv', '/kaggl
e/input/activity-recognition-dataset/watch_csv/watch_csv/accel/data_1635_accel_watch.csv', '/kaggle/input/activity
-recognition-dataset/watch_csv/watch_csv/accel/data_1634_accel_watch.csv', '/kaggle/input/activity-recognition-dat
aset/watch_csv/watch_csv/accel/data_1628_accel_watch.csv', '/kaggle/input/activity-recognition-dataset/watch_csv/w
atch_csv/accel/data_1609_accel_watch.csv', '/kaggle/input/activity-recognition-dataset/watch_csv/watch_csv/accel/d
ata_1641_accel_watch.csv', '/kaggle/input/activity-recognition-dataset/watch_csv/watch_csv/accel/data_1650_accel_w
atch.csv', '/kaggle/input/activity-recognition-dataset/watch_csv/watch_csv/accel/data_1627_accel_watch.csv', '/kag
gle/input/activity-recognition-dataset/watch_csv/watch_csv/accel/data_1639_accel_watch.csv', '/kaggle/input/activi
ty-recognition-dataset/watch_csv/watch_csv/accel/data_1626_accel_watch.csv', '/kaggle/input/activity-recognition-d
ataset/watch_csv/watch_csv/accel/data_1619_accel_watch.csv', '/kaggle/input/activity-recognition-dataset/watch_cs
```



```
--- Watch Accel ---
```

	subject_id	activity_code	timestamp	x	y	z
0	1644	A	1821530982460504	2.449867	-10.223690	-1.832911
1	1644	A	1821531031960504	5.842451	-10.769568	-5.496040
2	1644	A	1821531081460504	7.647679	-8.897303	0.788740
3	1644	A	1821531130960504	4.025251	-7.353042	0.281169
4	1644	A	1821531180460504	2.497751	-6.436063	-2.163311

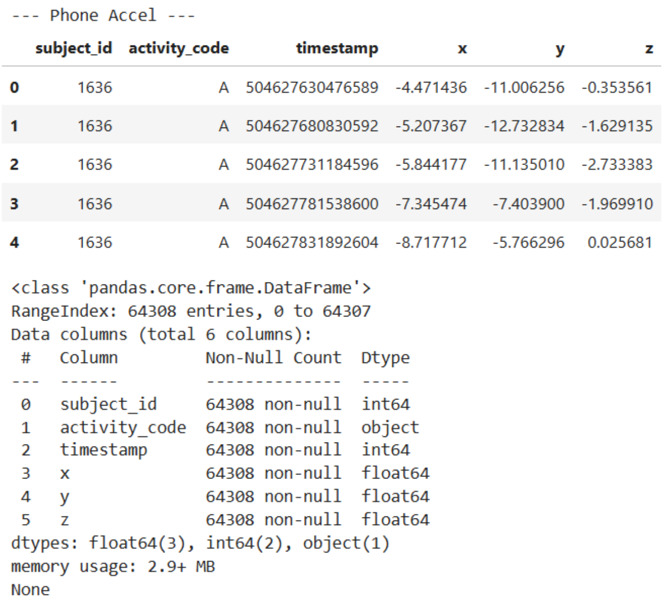
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70358 entries, 0 to 70357
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   subject_id     70358 non-null  int64
 1   activity_code  70358 non-null  object
 2   timestamp      70358 non-null  int64
 3   x              70358 non-null  float64
 4   y              70358 non-null  float64
 5   z              70358 non-null  float64
dtypes: float64(3), int64(2), object(1)
memory usage: 3.2+ MB
None
```



```
--- Watch Gyro ---
```

	subject_id	activity_code	timestamp	x	y	z
0	1627	A	216836900628086	1.074439	2.219760	0.589949
1	1627	A	216836950128086	1.870192	1.230130	-0.406073
2	1627	A	216836999628086	2.757557	-0.411443	-1.294504
3	1627	A	216837049128086	4.301126	-2.041297	-1.130453
4	1627	A	216837098628086	3.151705	-2.635715	0.000858

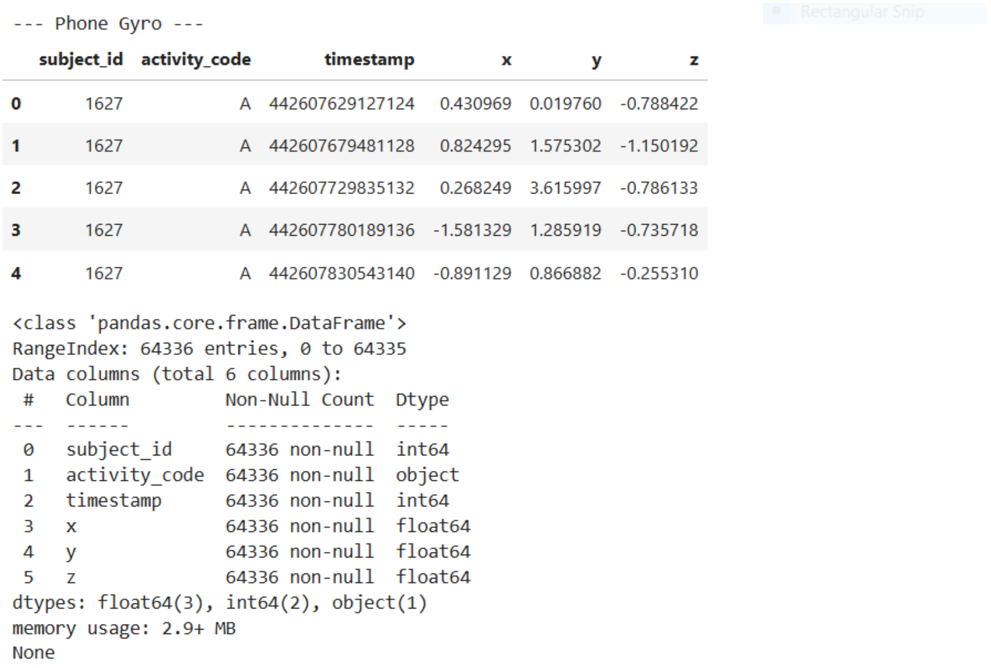
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 64863 entries, 0 to 64862
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   subject_id     64863 non-null  int64  
 1   activity_code  64863 non-null  object 
 2   timestamp      64863 non-null  int64  
 3   x              64863 non-null  float64
 4   y              64863 non-null  float64
 5   z              64863 non-null  float64
dtypes: float64(3), int64(2), object(1)
memory usage: 3.0+ MB
None
```



```
--- Phone Accel ---
```

	subject_id	activity_code	timestamp	x	y	z
0	1636	A	504627630476589	-4.471436	-11.006256	-0.353561
1	1636	A	504627680830592	-5.207367	-12.732834	-1.629135
2	1636	A	504627731184596	-5.844177	-11.135010	-2.733383
3	1636	A	504627781538600	-7.345474	-7.403900	-1.969910
4	1636	A	504627831892604	-8.717712	-5.766296	0.025681

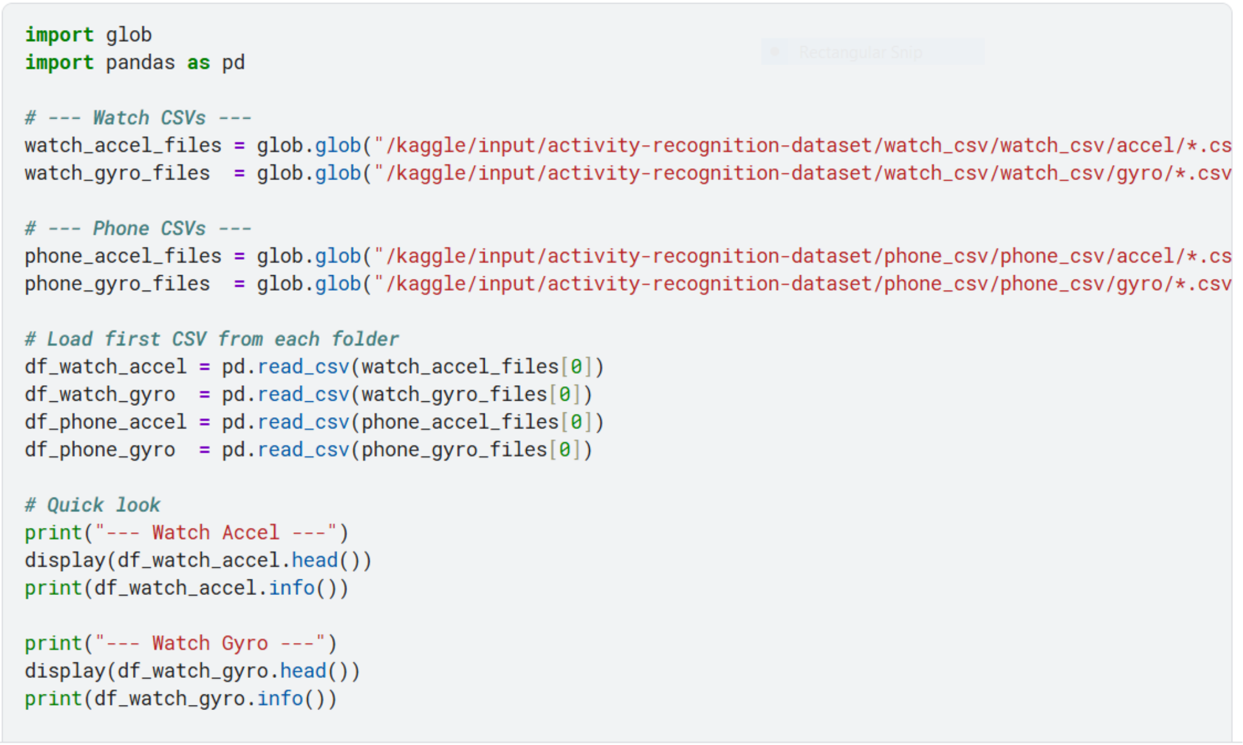
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 64308 entries, 0 to 64307
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   subject_id     64308 non-null  int64  
 1   activity_code  64308 non-null  object 
 2   timestamp      64308 non-null  int64  
 3   x              64308 non-null  float64
 4   y              64308 non-null  float64
 5   z              64308 non-null  float64
dtypes: float64(3), int64(2), object(1)
memory usage: 2.9+ MB
None
```



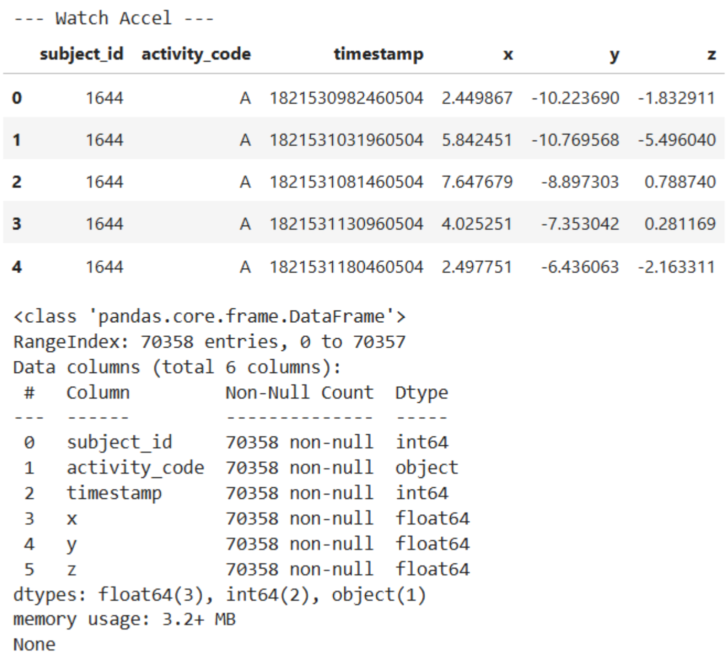
```
--- Phone Gyro ---
```

	subject_id	activity_code	timestamp	x	y	z
0	1627	A	442607629127124	0.430969	0.019760	-0.788422
1	1627	A	442607679481128	0.824295	1.575302	-1.150192
2	1627	A	442607729835132	0.268249	3.615997	-0.786133
3	1627	A	442607780189136	-1.581329	1.285919	-0.735718
4	1627	A	442607830543140	-0.891129	0.866882	-0.255310

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 64336 entries, 0 to 64335
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   subject_id     64336 non-null  int64
 1   activity_code  64336 non-null  object
 2   timestamp      64336 non-null  int64
 3   x              64336 non-null  float64
 4   y              64336 non-null  float64
 5   z              64336 non-null  float64
dtypes: float64(3), int64(2), object(1)
memory usage: 2.9+ MB
None
```



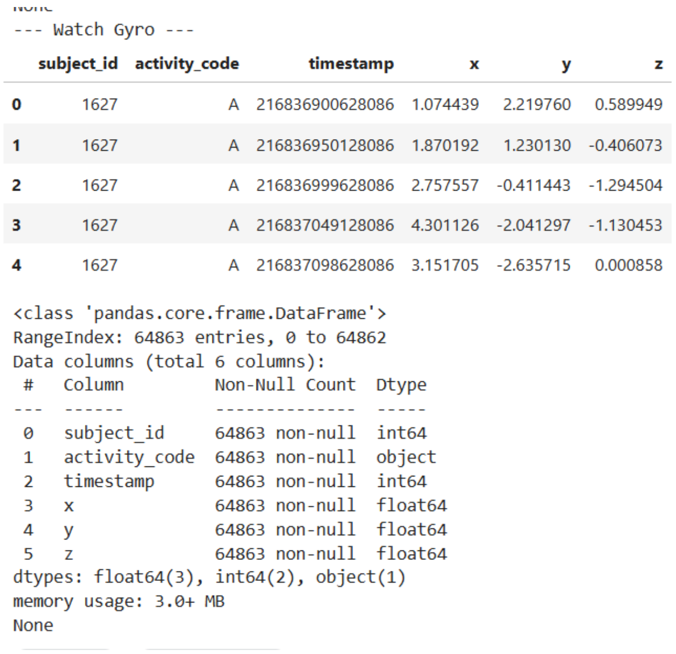
```
import glob
import pandas as pd

# --- Watch CSVs ---
watch_accel_files = glob.glob("/kaggle/input/activity-recognition-dataset/watch_csv/watch_csv/accel/*.cs
watch_gyro_files  = glob.glob("/kaggle/input/activity-recognition-dataset/watch_csv/watch_csv/gyro/*.csv

# --- Phone CSVs ---
phone_accel_files = glob.glob("/kaggle/input/activity-recognition-dataset/phone_csv/phone_csv/accel/*.cs
phone_gyro_files  = glob.glob("/kaggle/input/activity-recognition-dataset/phone_csv/phone_csv/gyro/*.csv

# Load first CSV from each folder
df_watch_accel = pd.read_csv(watch_accel_files[0])
df_watch_gyro  = pd.read_csv(watch_gyro_files[0])
df_phone_accel = pd.read_csv(phone_accel_files[0])
df_phone_gyro  = pd.read_csv(phone_gyro_files[0])

# Quick look
print("--- Watch Accel ---")
display(df_watch_accel.head())
print(df_watch_accel.info())

print("--- Watch Gyro ---")
display(df_watch_gyro.head())
print(df_watch_gyro.info())
```



```
--- Watch Accel ---
```

	subject_id	activity_code	timestamp	x	y	z
0	1644	A	1821530982460504	2.449867	-10.223690	-1.832911
1	1644	A	1821531031960504	5.842451	-10.769568	-5.496040
2	1644	A	1821531081460504	7.647679	-8.897303	0.788740
3	1644	A	1821531130960504	4.025251	-7.353042	0.281169
4	1644	A	1821531180460504	2.497751	-6.436063	-2.163311

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70358 entries, 0 to 70357
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   subject_id     70358 non-null  int64  
 1   activity_code  70358 non-null  object 
 2   timestamp      70358 non-null  int64  
 3   x              70358 non-null  float64
 4   y              70358 non-null  float64
 5   z              70358 non-null  float64
dtypes: float64(3), int64(2), object(1)
memory usage: 3.2+ MB
None
```

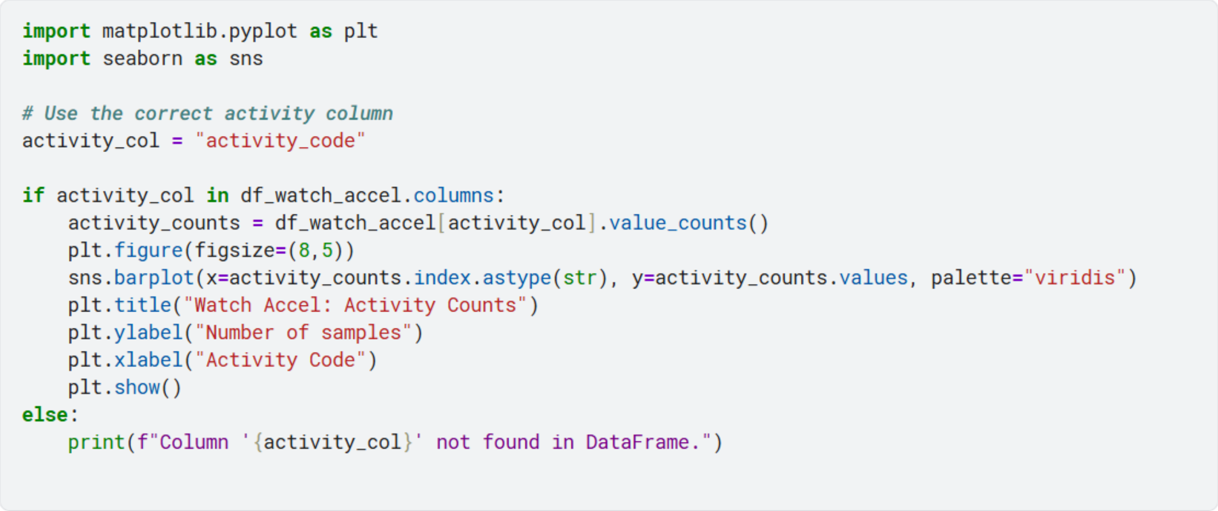


```
--- Watch Gyro ---
```

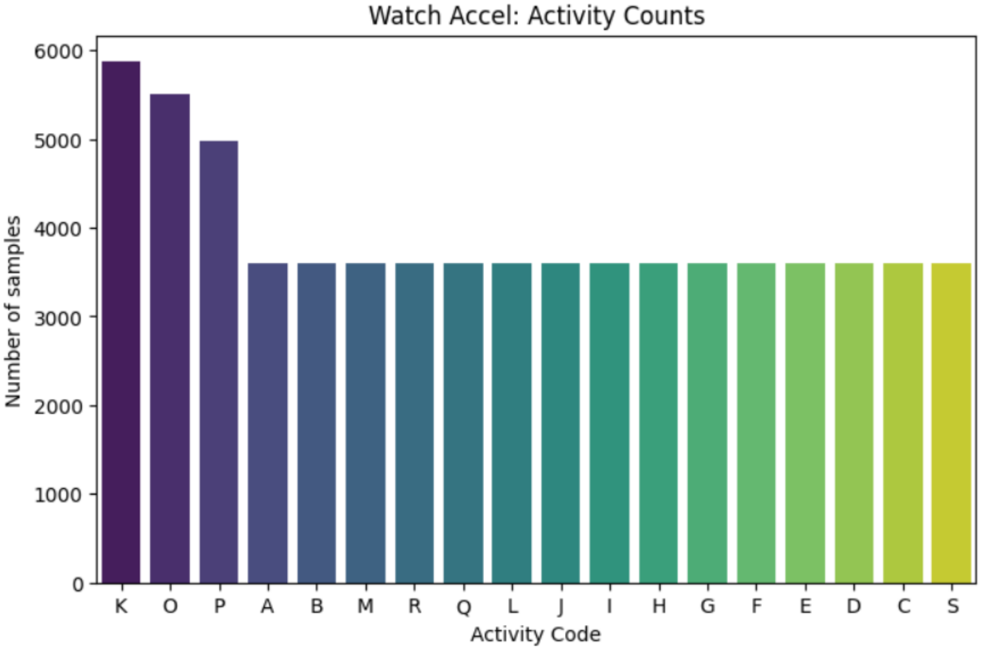
	subject_id	activity_code	timestamp	x	y	z
0	1627	A	216836900628086	1.074439	2.219760	0.589949
1	1627	A	216836950128086	1.870192	1.230130	-0.406073
2	1627	A	216836999628086	2.757557	-0.411443	-1.294504
3	1627	A	216837049128086	4.301126	-2.041297	-1.130453
4	1627	A	216837098628086	3.151705	-2.635715	0.000858

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 64863 entries, 0 to 64862
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   subject_id     64863 non-null  int64  
 1   activity_code  64863 non-null  object 
 2   timestamp      64863 non-null  int64  
 3   x              64863 non-null  float64
 4   y              64863 non-null  float64
 5   z              64863 non-null  float64
dtypes: float64(3), int64(2), object(1)
memory usage: 3.0+ MB
None
```

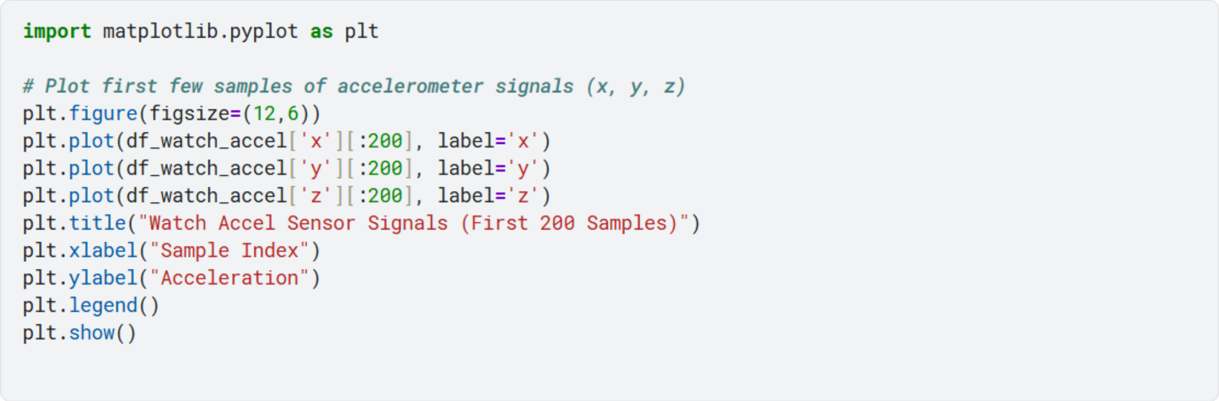
Activity Counts:



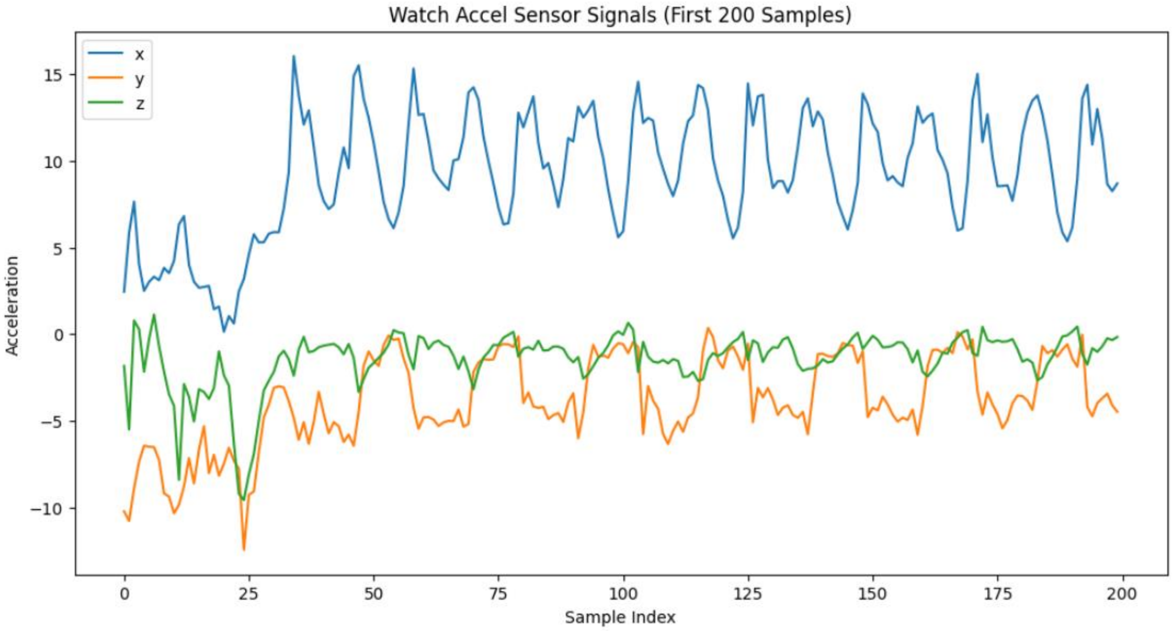
```
import matplotlib.pyplot as plt
import seaborn as sns

# Use the correct activity column
activity_col = "activity_code"

if activity_col in df_watch_accel.columns:
    activity_counts = df_watch_accel[activity_col].value_counts()
    plt.figure(figsize=(8,5))
    sns.barplot(x=activity_counts.index.astype(str), y=activity_counts.values, palette="viridis")
    plt.title("Watch Accel: Activity Counts")
    plt.ylabel("Number of samples")
    plt.xlabel("Activity Code")
    plt.show()
else:
    print(f"Column '{activity_col}' not found in DataFrame.")
```



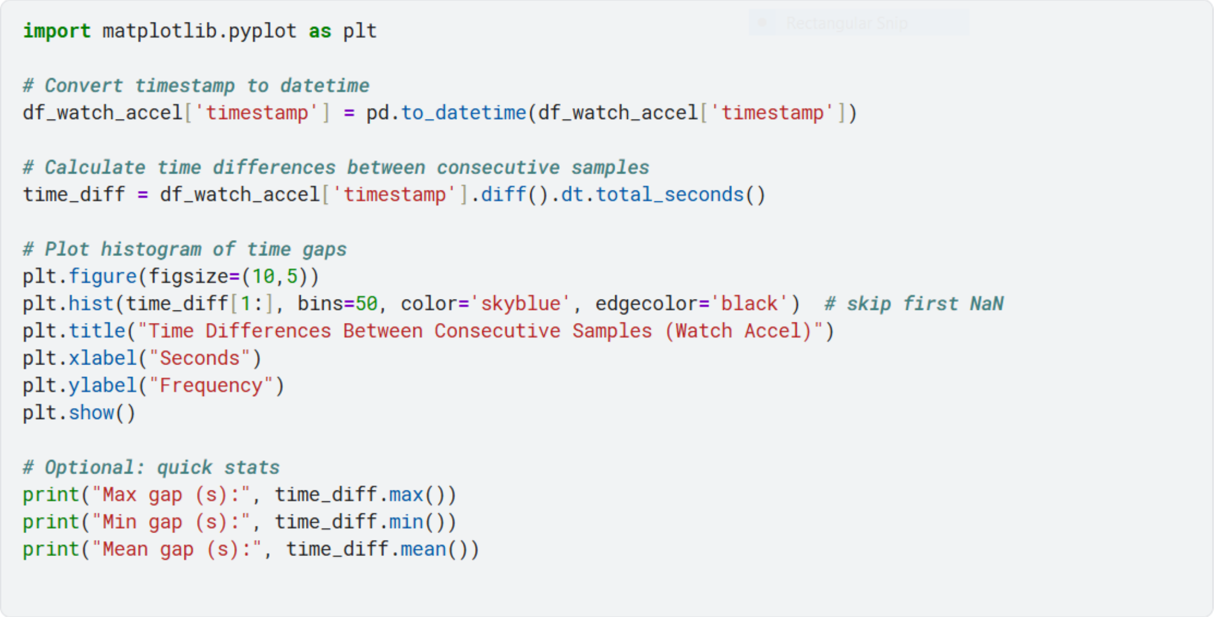
Sensor Noise/Raw Signals:



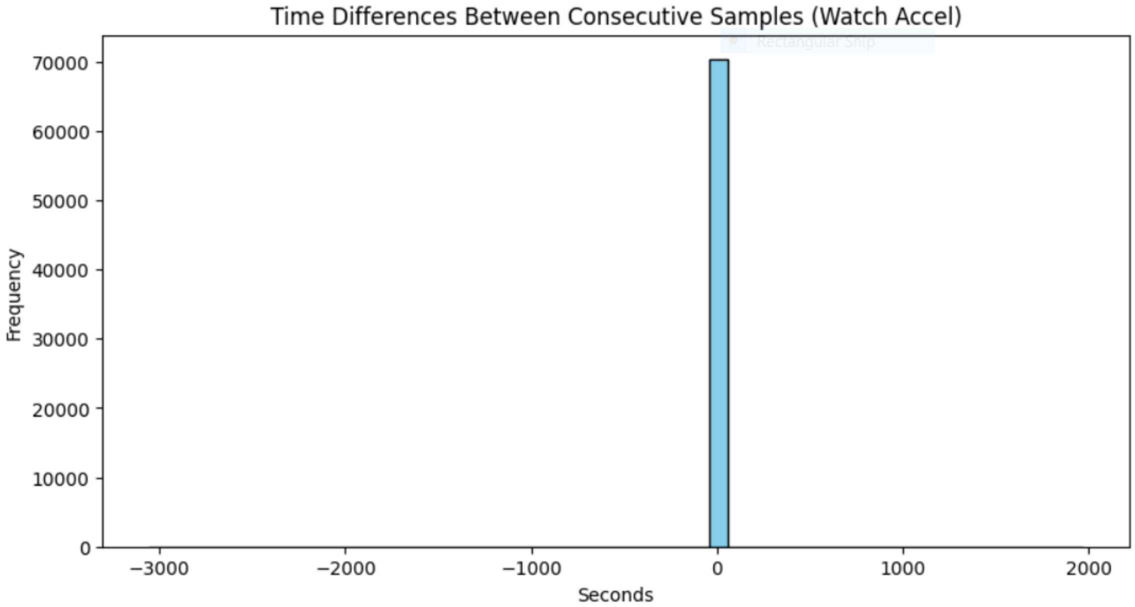
```
import matplotlib.pyplot as plt

# Plot first few samples of accelerometer signals (x, y, z)
plt.figure(figsize=(12,6))
plt.plot(df_watch_accel['x'][:200], label='x')
plt.plot(df_watch_accel['y'][:200], label='y')
plt.plot(df_watch_accel['z'][:200], label='z')
plt.title("Watch Accel Sensor Signals (First 200 Samples)")
plt.xlabel("Sample Index")
plt.ylabel("Acceleration")
plt.legend()
plt.show()
```



Time Gap Analysis:

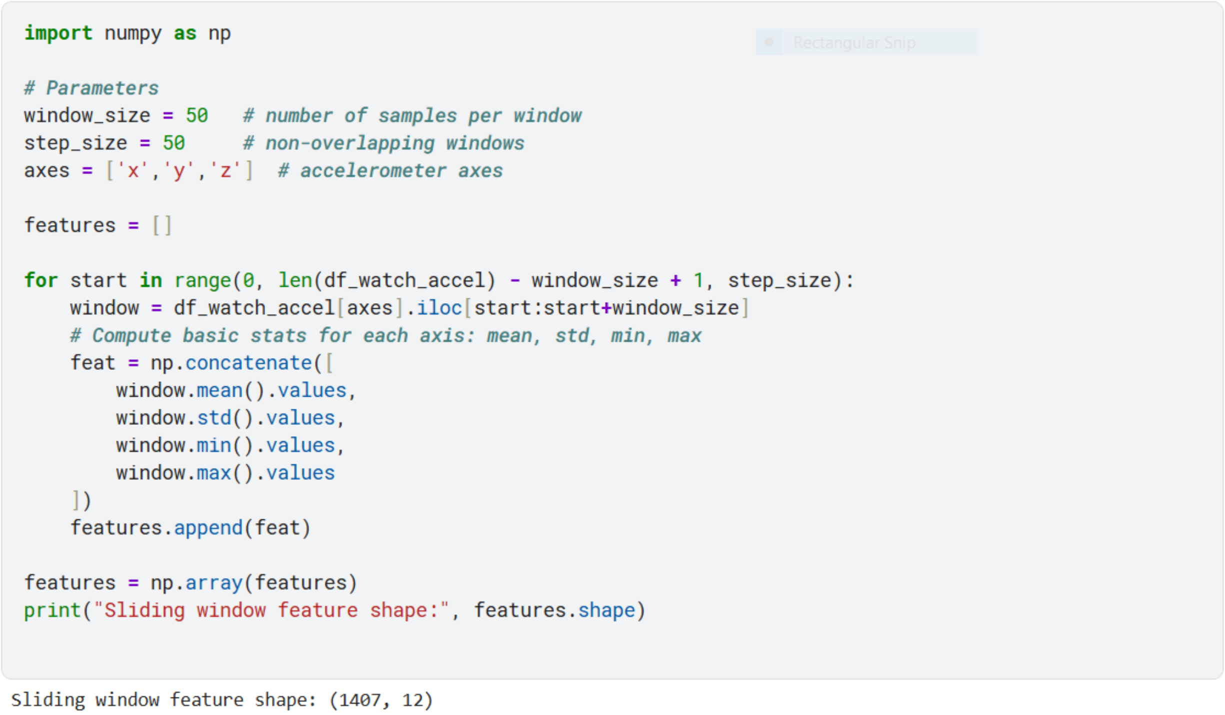


```
import matplotlib.pyplot as plt

# Convert timestamp to datetime
df_watch_accel['timestamp'] = pd.to_datetime(df_watch_accel['timestamp'])

# Calculate time differences between consecutive samples
time_diff = df_watch_accel['timestamp'].diff().dt.total_seconds()

# Plot histogram of time gaps
plt.figure(figsize=(10,5))
plt.hist(time_diff[1:], bins=50, color='skyblue', edgecolor='black')  # skip first NaN
plt.title("Time Differences Between Consecutive Samples (Watch Accel)")
plt.xlabel("Seconds")
plt.ylabel("Frequency")
plt.show()

# Optional: quick stats
print("Max gap (s):", time_diff.max())
print("Min gap (s):", time_diff.min())
print("Mean gap (s):", time_diff.mean())
```



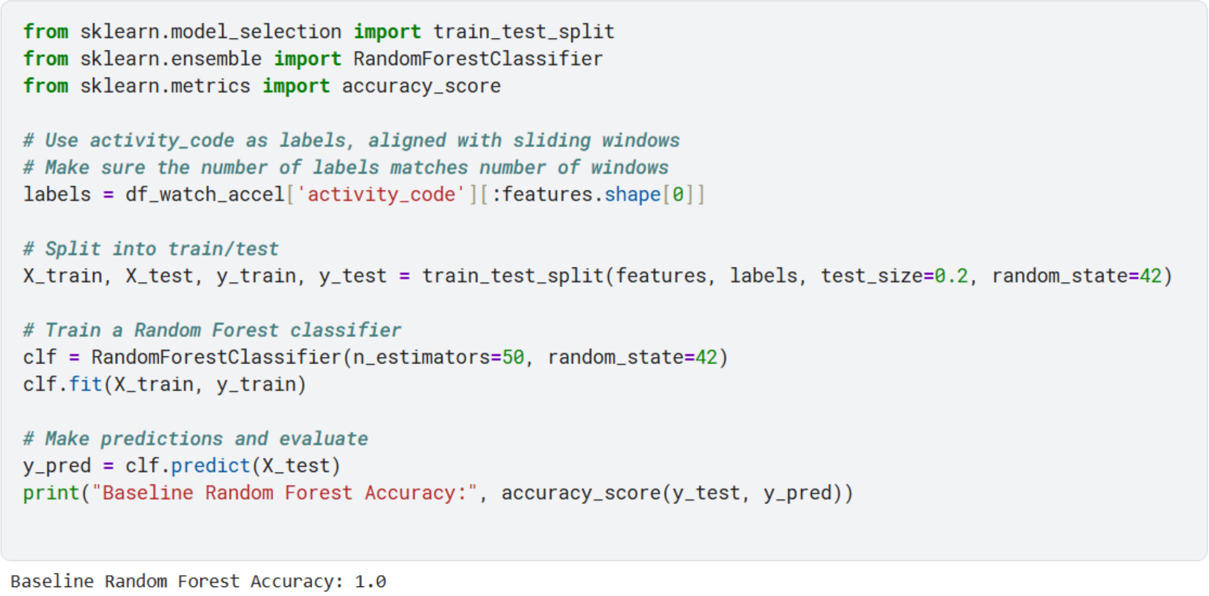
```
Max gap (s): 1969.260901161
Min gap (s): -3056.245672185
Mean gap (s): -0.03735283655174324
```

Sliding Window Feature Extraction:



```python
import numpy as np

# Parameters
window_size = 50   # number of samples per window
step_size = 50     # non-overlapping windows
axes = ['x','y','z']  # accelerometer axes

features = []

for start in range(0, len(df_watch_accel) - window_size + 1, step_size):
    window = df_watch_accel[axes].iloc[start:start+window_size]
    # Compute basic stats for each axis: mean, std, min, max
    feat = np.concatenate([
        window.mean().values,
        window.std().values,
        window.min().values,
        window.max().values
    ])
    features.append(feat)

features = np.array(features)
print("Sliding window feature shape:", features.shape)
```

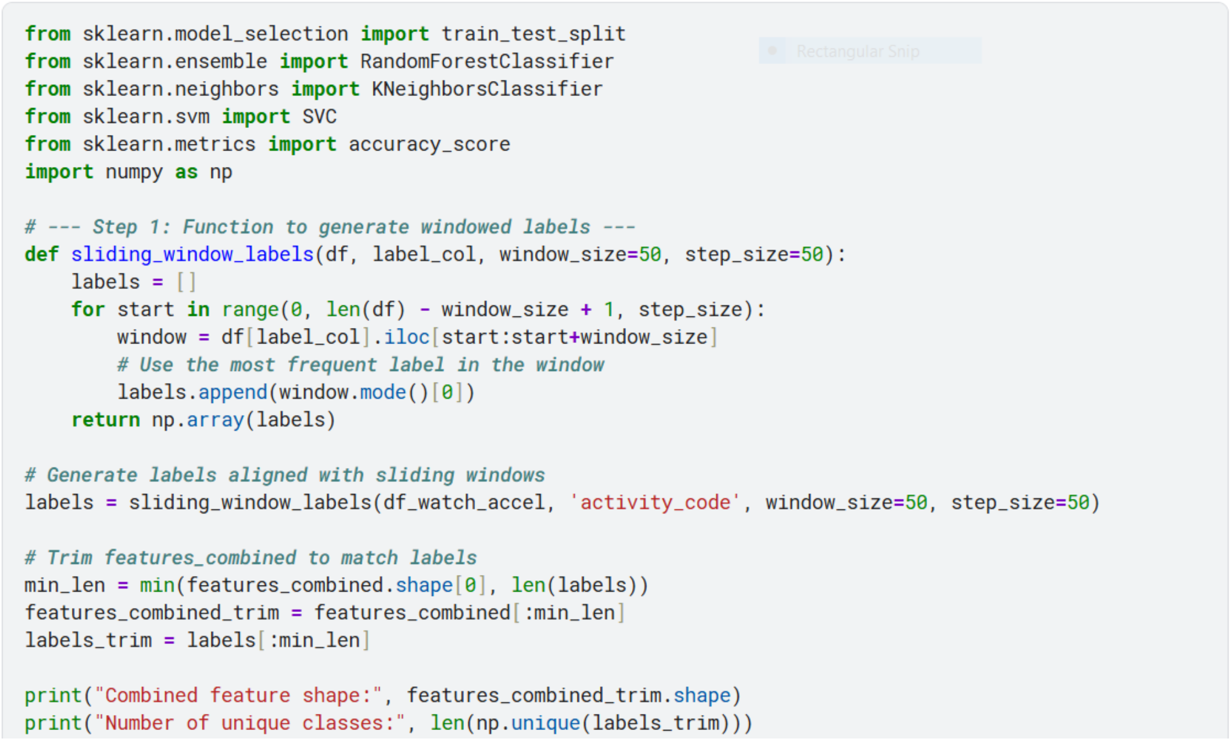
```
Sliding window feature shape: (1407, 12)
```



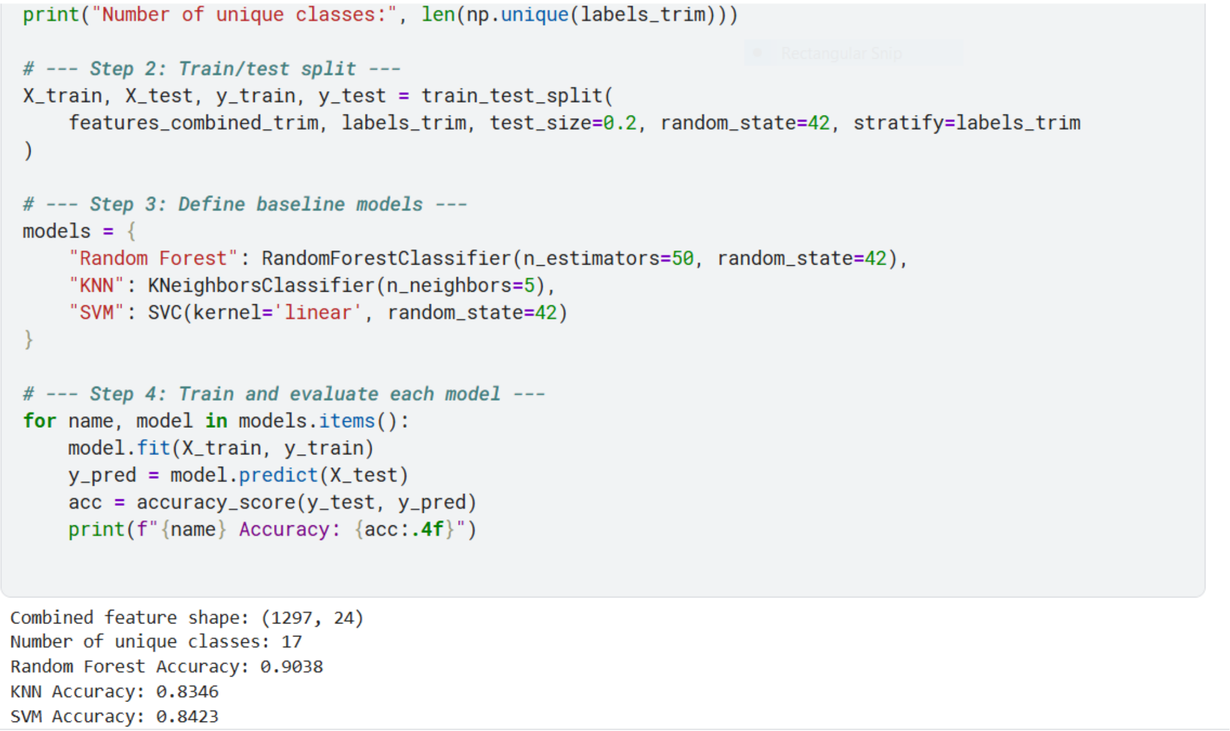
```python
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# Use activity_code as labels, aligned with sliding windows
# Make sure the number of labels matches number of windows
labels = df_watch_accel['activity_code'][:features.shape[0]]

# Split into train/test
X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size=0.2, random_state=42)

# Train a Random Forest classifier
clf = RandomForestClassifier(n_estimators=50, random_state=42)
clf.fit(X_train, y_train)

# Make predictions and evaluate
y_pred = clf.predict(X_test)
print("Baseline Random Forest Accuracy:", accuracy_score(y_test, y_pred))
```

```
Baseline Random Forest Accuracy: 1.0
```

Combine Features:



```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
import numpy as np

# --- Step 1: Function to generate windowed labels ---
def sliding_window_labels(df, label_col, window_size=50, step_size=50):
    labels = []
    for start in range(0, len(df) - window_size + 1, step_size):
        window = df[label_col].iloc[start:start+window_size]
        # Use the most frequent label in the window
        labels.append(window.mode()[0])
    return np.array(labels)

# Generate labels aligned with sliding windows
labels = sliding_window_labels(df_watch_accel, 'activity_code', window_size=50, step_size=50)

# Trim features_combined to match labels
min_len = min(features_combined.shape[0], len(labels))
features_combined_trim = features_combined[:min_len]
labels_trim = labels[:min_len]

print("Combined feature shape:", features_combined_trim.shape)
print("Number of unique classes:", len(np.unique(labels_trim)))
```

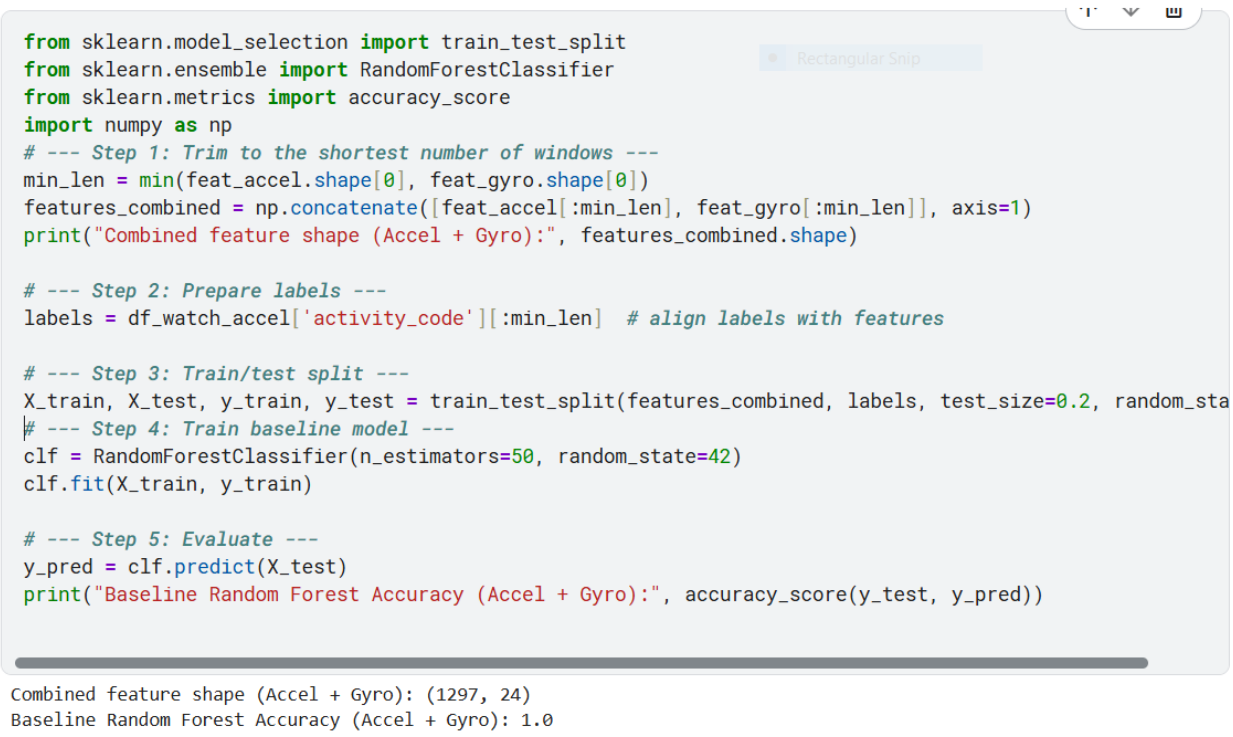


```
print("Number of unique classes:", len(np.unique(labels_trim)))
```

```

# --- Step 2: Train/test split ---
X_train, X_test, y_train, y_test = train_test_split(
    features_combined_trim, labels_trim, test_size=0.2, random_state=42, stratify=labels_trim
)

# --- Step 3: Define baseline models ---
models = {
    "Random Forest": RandomForestClassifier(n_estimators=50, random_state=42),
    "KNN": KNeighborsClassifier(n_neighbors=5),
    "SVM": SVC(kernel='linear', random_state=42)
}

# --- Step 4: Train and evaluate each model ---
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    print(f"{name} Accuracy: {acc:.4f}")
```

```
Combined feature shape: (1297, 24)
Number of unique classes: 17
Random Forest Accuracy: 0.9038
KNN Accuracy: 0.8346
SVM Accuracy: 0.8423
```

Baseline Models:



```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
import numpy as np
# --- Step 1: Trim to the shortest number of windows ---
min_len = min(feat_accel.shape[0], feat_gyro.shape[0])
features_combined = np.concatenate([feat_accel[:min_len], feat_gyro[:min_len]], axis=1)
print("Combined feature shape (Accel + Gyro):", features_combined.shape)

# --- Step 2: Prepare labels ---
labels = df_watch_accel['activity_code'][:min_len]  # align labels with features

# --- Step 3: Train/test split ---
X_train, X_test, y_train, y_test = train_test_split(features_combined, labels, test_size=0.2, random_sta
# --- Step 4: Train baseline model ---
clf = RandomForestClassifier(n_estimators=50, random_state=42)
clf.fit(X_train, y_train)

# --- Step 5: Evaluate ---
y_pred = clf.predict(X_test)
print("Baseline Random Forest Accuracy (Accel + Gyro):", accuracy_score(y_test, y_pred))
```

```
Combined feature shape (Accel + Gyro): (1297, 24)
Baseline Random Forest Accuracy (Accel + Gyro): 1.0
```