Step 1: Mount Google Drive

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Step 2: Load and Parse the Raw Dataset

```
import pandas as pd
import numpy as np

file_path = '/content/drive/My Drive/Colab Notebooks/WISDM_ar_v1.1/WISDM_ar_v1.1_

data = []
with open(file_path) as f:
    for line in f:
        try:
            parts = line.strip().split(',')
            if len(parts) >= 6:
                user_id, activity, timestamp, x, y, z = parts[0], parts[1], parts
                data.append([user_id, activity, timestamp, x, y, z])
        except:
            continue

columns = ['user', 'activity', 'timestamp', 'x', 'y', 'z']
df = pd.DataFrame(data, columns=columns)
print(df.head())
```

```
   user activity       timestamp         x          y         z
0   33  Jogging  49105962326000 -0.694638  12.680544  0.503953
1   33  Jogging  49106062271000  5.012288  11.264028  0.953424
2   33  Jogging  49106112167000  4.903325  10.882658 -0.081722
3   33  Jogging  49106222305000 -0.612916  18.496431  3.023717
4   33  Jogging  49106332290000 -1.184970  12.108489  7.205164
```

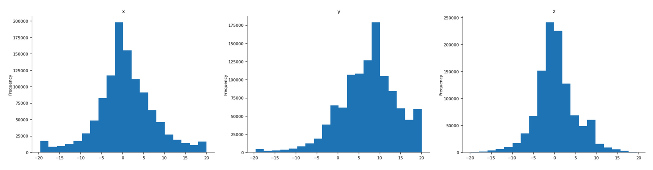
```
df
```

	user	activity	timestamp	x	y	z
0	33	Jogging	49105962326000	-0.694638	12.680544	0.503953
1	33	Jogging	49106062271000	5.012288	11.264028	0.953424
2	33	Jogging	49106112167000	4.903325	10.882658	-0.081722
3	33	Jogging	49106222305000	-0.612916	18.496431	3.023717
4	33	Jogging	49106332290000	-1.184970	12.108489	7.205164
...	...	...	...	...	...	...
1098198	19	Sitting	131623331483000	9.000000	-1.570000	1.690000
1098199	19	Sitting	131623371431000	9.040000	-1.460000	1.730000
1098200	19	Sitting	131623411592000	9.080000	-1.380000	1.690000
1098201	19	Sitting	131623491487000	9.000000	-1.460000	1.730000
1098202	19	Sitting	131623531465000	8.880000	-1.330000	1.610000

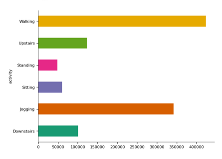
1098203 rows × 6 columns

```
Warning: total number of rows (1098203) exceeds max_rows (20000). Limiting to first (20000) rows.
```

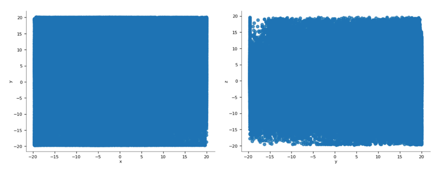
Distributions



Categorical distributions



2-d distributions



Time series

Step 3: Feature Extraction via Sliding Window

```
def extract_features(df, window_size=200):
    X, y = [], []
    for i in range(0, len(df) - window_size, window_size):
        window = df.iloc[i:i+window_size]
        x_vals = window['x'].values
        y_vals = window['y'].values
        z_vals = window['z'].values

        features = []
        for axis in [x_vals, y_vals, z_vals]:
            features += [
                np.mean(axis), np.std(axis), np.min(axis), np.max(axis),
                np.median(axis), np.percentile(axis, 25), np.percentile(axis, 75)
            ]
        label = window['activity'].mode()[0]
        X.append(features)
        y.append(label)
    return np.array(X), np.array(y)

X, y = extract_features(df)
print("Shape of X (Number of samples, Number of features per sample) = ",X.shape)
print("Shape of Y (Number of labels) = ",y.shape)
```

```
X (Number of samples, Number of features per sample) =  (5491, 21)
Y (Number of labels) =  (5491,)
```

Step 4: Encode Labels and Train-Test Split

```
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split

le = LabelEncoder()
y_encoded = le.fit_transform(y)

X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.2, random_state=42)
```

Step 5: Train Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)

print("Accuracy:", accuracy)
print("\nClassification Report:\n", classification_report(y_test, y_pred, target_names=le.classes_))
```

```
Accuracy: 0.9253867151956324

Classification Report:
               precision    recall  f1-score   support

  Downstairs       0.87      0.76      0.81        97
     Jogging       0.94      0.98      0.96       356
     Sitting       0.96      0.95      0.96        58
    Standing       0.98      0.98      0.98        42
    Upstairs       0.88      0.72      0.80       138
     Walking       0.92      0.98      0.95       408

    accuracy                           0.93      1099
   macro avg       0.93      0.89      0.91      1099
weighted avg       0.92      0.93      0.92      1099
```

Step 6: Visualization of Classification Results

```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
import numpy as np

# Generate the confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Plot the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=label_encoder.classes_,
            yticklabels=label_encoder.classes_)

plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```