Step 1: Mount Google Drive

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

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

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_raw.txt'

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[2], float(parts[3]), float(parts[4]), float(parts[5].split(';')[0])
                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
```

1 to 25 of 20000 entries Filter

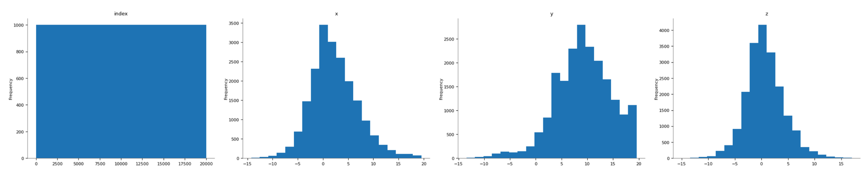
index	user	activity	timestamp	x	y	z
0	33	Jogging	49105962326000	-0.6946377	12.680544	0.50395286
1	33	Jogging	49106062271000	5.012288	11.264028	0.95342433
2	33	Jogging	49106112167000	4.903325	10.882658	-0.08172209
3	33	Jogging	49106222305000	-0.61291564	18.496431	3.0237172
4	33	Jogging	49106332290000	-1.1849703	12.108489	7.205164
5	33	Jogging	49106442306000	1.3756552	-2.4925237	-6.510526
6	33	Jogging	49106542312000	-0.61291564	10.56939	5.706926
7	33	Jogging	49106652389000	-0.50395286	13.947236	7.0553403
8	33	Jogging	49106762313000	-8.430995	11.413852	5.134871
9	33	Jogging	49106872299000	0.95342433	1.3756552	1.6480621
10	33	Jogging	49106982315000	-8.19945	19.57244	2.7240696
11	33	Jogging	49107092330000	1.4165162	5.7886477	2.982856
12	33	Jogging	49107202316000	-1.879608	-2.982856	-0.29964766
13	33	Jogging	49107312332000	-6.1291566	6.851035	-8.158588
14	33	Jogging	49107422348000	5.829509	18.0061	8.539958
15	33	Jogging	49107522293000	6.2789803	2.982856	2.9147544
16	33	Jogging	49107632339000	-1.56634	8.308413	-1.4573772
17	33	Jogging	49107742355000	3.5276701	13.593107	9.425281
18	33	Jogging	49107852340000	-2.0294318	-5.706926	-10.18802
19	33	Jogging	49107962326000	2.7649305	10.337844	-9.724928
20	33	Jogging	49108062271000	3.568531	13.6748295	1.5390993
21	33	Jogging	49108172348000	-0.50395286	3.8681788	3.718355
22	33	Jogging	49108272262000	-2.3018389	1.6889231	0.08172209
23	33	Jogging	49108382370000	-3.568531	19.57244	6.510526
24	33	Jogging	49108492294000	-0.8036005	-3.2961242	-4.630918

Show 25 per page 1 2 10 100 700 790 800

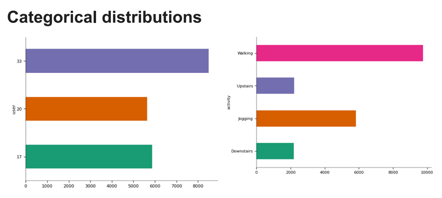
Like what you see? Visit the data table notebook to learn more about interactive tables.

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

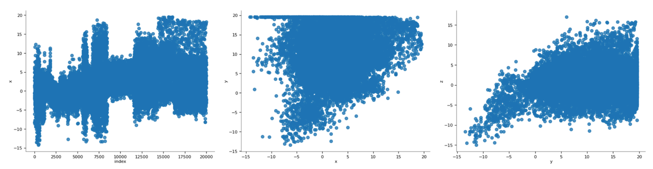
Distributions



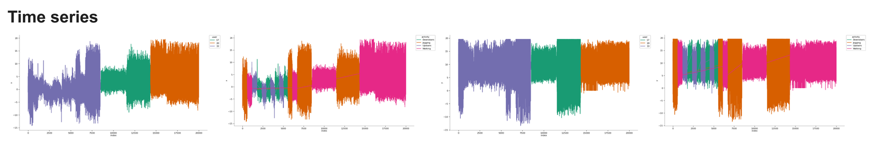
Categorical distributions



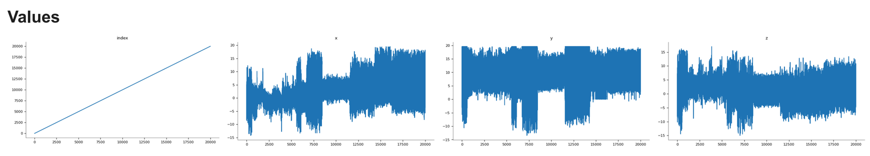
2-d distributions



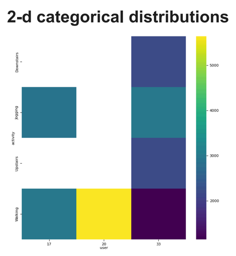
Time series



Values



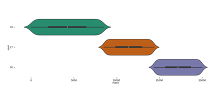
2-d categorical distributions



Faceted distributions

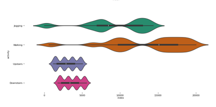
```
<string>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.
```



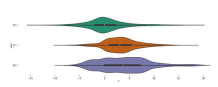
```
<string>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.
```



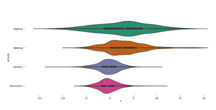
```
<string>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.
```



```
<string>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.
```



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

```
## Step 6: Visualization of Classification Results

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

# 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=le.classes_,
            yticklabels=le.classes_)

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