Data Cleaning & Preprocessing

```
# importing libraries
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

```
# reading raw data file
columns = ['user','activity','timestamp', 'x-axis', 'y-axis', 'z-axis']
df_har = pd.read_csv('WISDM_ar_v1.1_raw.txt', header = None, names = columns)
```

```
# removing null values
df_har = df_har.dropna()
df_har.shape
```

```
(1098203, 6)
```

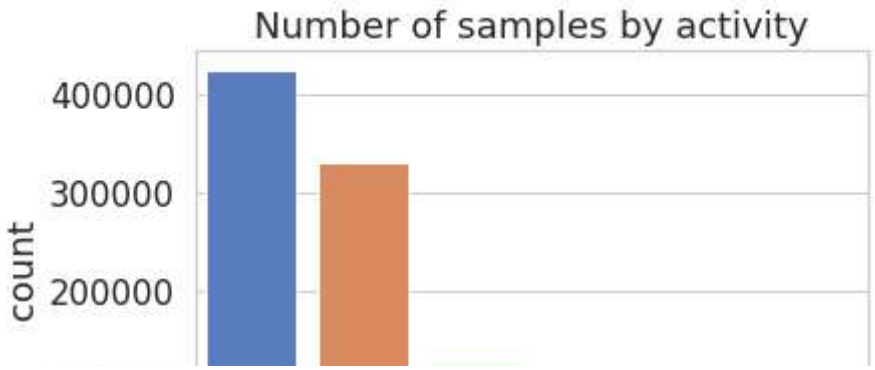
```
# transforming the z-axis to float
df_har['z-axis'] = df_har['z-axis'].str.replace(';', '')
df_har['z-axis'] = df_har['z-axis'].apply(lambda x:float(x))
```

```
# drop rows where timestamp is 0
df = df_har[df_har['timestamp'] != 0]
```

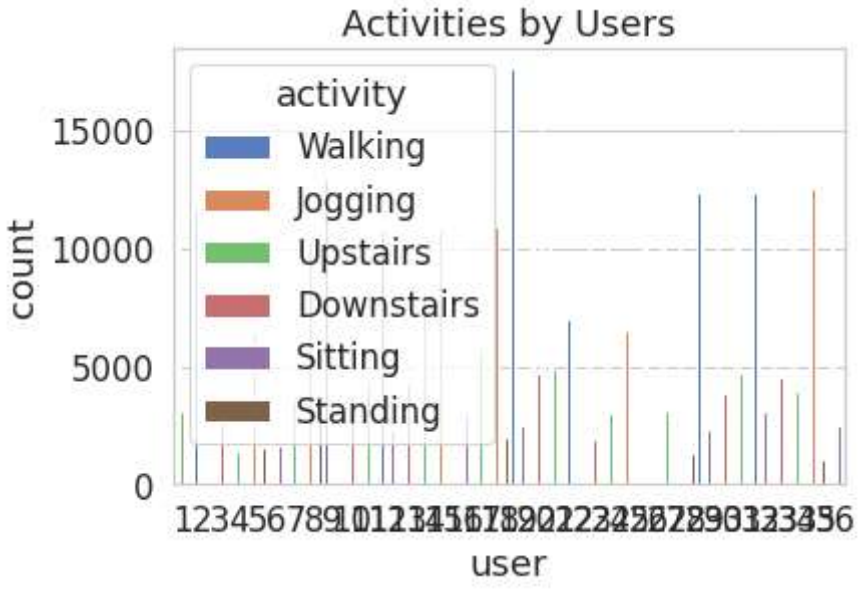
```
# arrange data in ascending order of user and timestamp
df = df.sort_values(by = ['user', 'timestamp'], ignore_index=True)
```

Exploratory Data Analysis

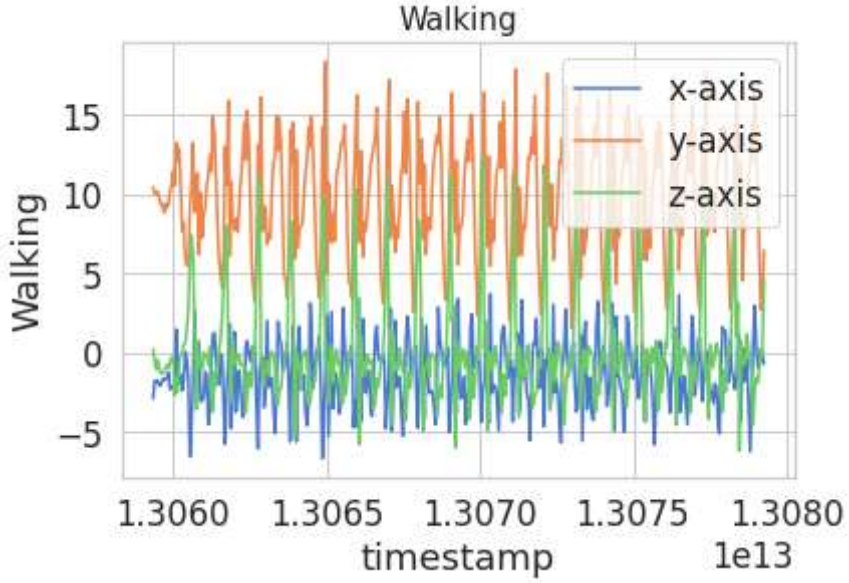
```
sns.set_style("whitegrid")
sns.countplot(x = 'activity', data = df)
plt.title('Number of samples by activity')
plt.show()
```

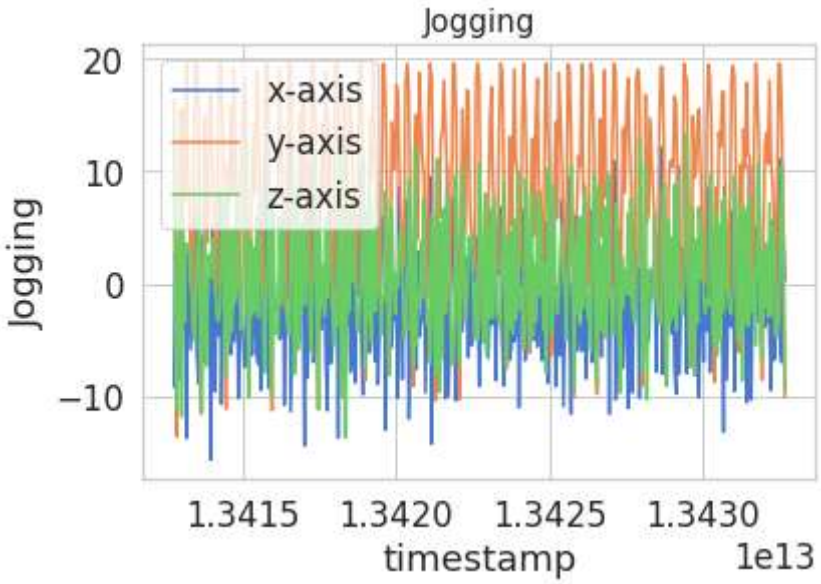


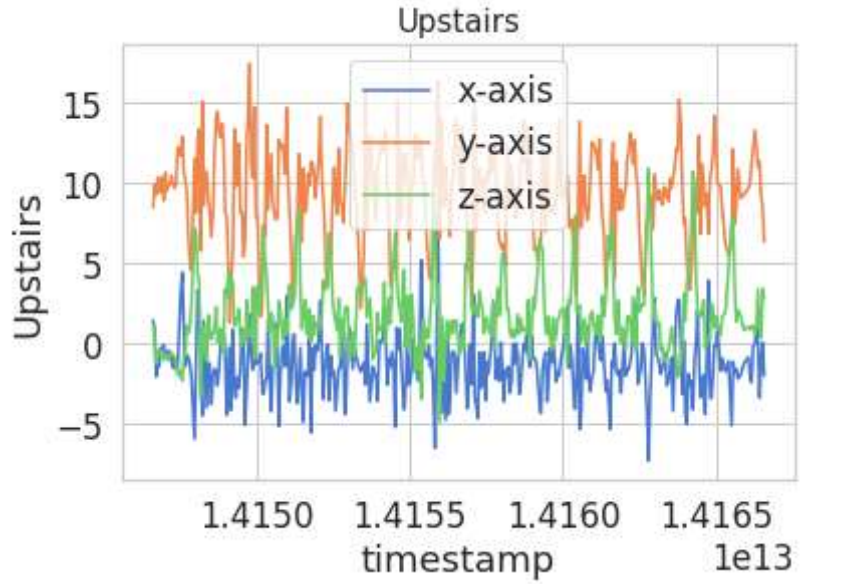

```
sns.countplot(x = 'user',  hue = 'activity', data = df)
plt.title('Activities by Users')
plt.show()
```

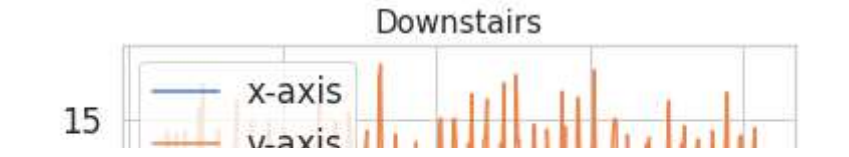


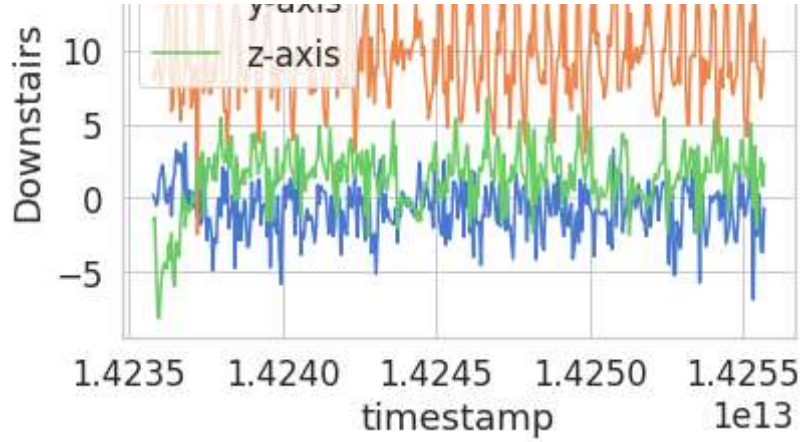

```
activities = ['Walking', 'Jogging', 'Upstairs', 'Downstairs', 'Sitting', 'Standing']
for i in activities:
    data36 = df[(df['user'] == 36) & (df['activity'] == i)][:400]
    sns.lineplot(y = 'x-axis', x = 'timestamp', data = data36)
    sns.lineplot(y = 'y-axis', x = 'timestamp', data = data36)
    sns.lineplot(y = 'z-axis', x = 'timestamp', data = data36)
    plt.legend(['x-axis', 'y-axis', 'z-axis'])
    plt.ylabel(i)
    plt.title(i, fontsize = 15)
    plt.show()
```

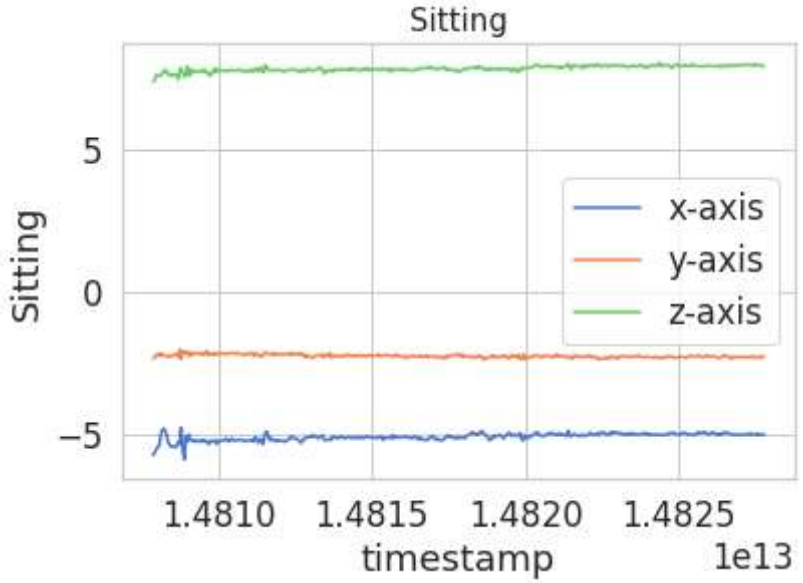

Walking
x-axis
y-axis
z-axis
Walking
15
10
5
0
−5
1.3060
1.3065
1.3070
1.3075
1.3080
timestamp
1e13


Jogging
x-axis
y-axis
z-axis
Jogging
20
10
0
−10
1.3415
1.3420
1.3425
1.3430
timestamp
1e13


Upstairs
x-axis
y-axis
z-axis
Upstairs
15
10
5
0
−5
1.4150
1.4155
1.4160
1.4165
timestamp
1e13


Downstairs
x-axis
15





Preparing Data

```
random_seed = 42
n_time_steps = 50
n_features = 3
step = 10
n_classes = 6
n_epochs = 50
batch_size = 1024
learning_rate = 0.0025
l2_loss = 0.0015
```

```
segments = []
labels = []

```

```
for i in range(0,  df.shape[0]- n_time_steps, step):

    xs = df['x-axis'].values[i: i + 50]

    ys = df['y-axis'].values[i: i + 50]

    zs = df['z-axis'].values[i: i + 50]

    label = stats.mode(df['activity'][i: i + 50])[0][0]

    segments.append([xs, ys, zs])

    labels.append(label)

#reshape the segments which is (list of arrays) to a list
reshaped_segments = np.asarray(segments, dtype= np.float32).reshape(-1, n_time_steps, n_features)

labels = np.asarray(pd.get_dummies(labels), dtype = np.float32)
```

```
reshaped_segments.shape
```

```
(108531, 50, 3)
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(reshaped_segments, labels, test_size = 0.2, random_st
```

Building Model Architecture

```
from keras.models import Sequential
from keras.layers import LSTM, Dense, Flatten, Dropout
```

```
model = Sequential()
# RNN layer
model.add(LSTM(units = 128, input_shape = (X_train.shape[1], X_train.shape[2])))
# Dropout layer
model.add(Dropout(0.5))
# Dense layer with ReLu
model.add(Dense(units = 64, activation='relu'))
# Softmax layer
model.add(Dense(y_train.shape[1], activation = 'softmax'))
# Compile model
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 128)               67584
```

```
 dropout (Dropout)           (None, 128)               0

 dense (Dense)               (None, 64)                8256

 dense_1 (Dense)             (None, 6)                 390

=================================================================
Total params: 76,230
Trainable params: 76,230
Non-trainable params: 0
_________________________________________________________________
```

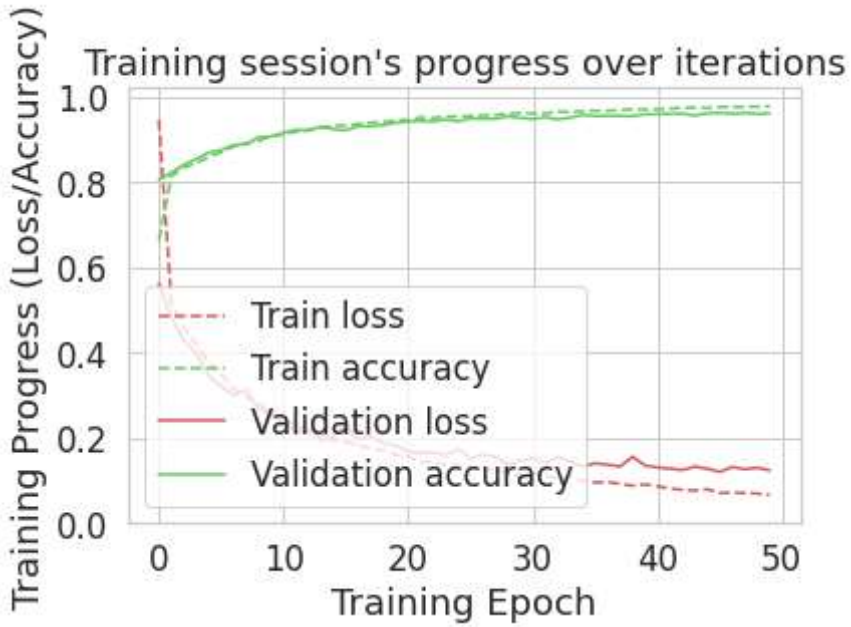
Model Training & Evaluation

```
1 history = model.fit(X_train, y_train, epochs = n_epochs, validation_split = 0.20, batch_size = batch_size
```

```
Epoch 23/50
68/68 [==============================] - 65s 958ms/step - loss: 0.1452 - accuracy: 0
Epoch 24/50
68/68 [==============================] - 66s 973ms/step - loss: 0.1369 - accuracy: 0
Epoch 25/50
68/68 [==============================] - 62s 916ms/step - loss: 0.1362 - accuracy: 0
Epoch 26/50
68/68 [==============================] - 62s 908ms/step - loss: 0.1316 - accuracy: 0
Epoch 27/50
68/68 [==============================] - 64s 949ms/step - loss: 0.1284 - accuracy: 0
Epoch 28/50
68/68 [==============================] - 63s 923ms/step - loss: 0.1240 - accuracy: 0
Epoch 29/50
68/68 [==============================] - 62s 917ms/step - loss: 0.1183 - accuracy: 0
Epoch 30/50
68/68 [==============================] - 64s 949ms/step - loss: 0.1119 - accuracy: 0
Epoch 31/50
68/68 [==============================] - 62s 916ms/step - loss: 0.1166 - accuracy: 0
Epoch 32/50
68/68 [==============================] - 63s 925ms/step - loss: 0.1143 - accuracy: 0
Epoch 33/50
68/68 [==============================] - 65s 952ms/step - loss: 0.1036 - accuracy: 0
Epoch 34/50
68/68 [==============================] - 67s 990ms/step - loss: 0.1084 - accuracy: 0
Epoch 35/50
68/68 [==============================] - 65s 961ms/step - loss: 0.0995 - accuracy: 0
Epoch 36/50
68/68 [==============================] - 74s 1s/step - loss: 0.0956 - accuracy: 0.96
Epoch 37/50
68/68 [==============================] - 72s 1s/step - loss: 0.0969 - accuracy: 0.96
Epoch 38/50
68/68 [==============================] - 68s 994ms/step - loss: 0.0924 - accuracy: 0
Epoch 39/50
68/68 [==============================] - 64s 926ms/step - loss: 0.0881 - accuracy: 0
Epoch 40/50
68/68 [==============================] - 62s 910ms/step - loss: 0.0910 - accuracy: 0
Epoch 41/50
```

```
68/68 [==============================] - 63s 933ms/step - loss: 0.0867 - accuracy: 0
Epoch 42/50
68/68 [==============================] - 61s 903ms/step - loss: 0.0817 - accuracy: 0
Epoch 43/50
68/68 [==============================] - 62s 908ms/step - loss: 0.0781 - accuracy: 0
Epoch 44/50
68/68 [==============================] - 67s 981ms/step - loss: 0.0768 - accuracy: 0
Epoch 45/50
68/68 [==============================] - 61s 899ms/step - loss: 0.0797 - accuracy: 0
Epoch 46/50
68/68 [==============================] - 63s 925ms/step - loss: 0.0706 - accuracy: 0
Epoch 47/50
68/68 [==============================] - 67s 983ms/step - loss: 0.0719 - accuracy: 0
Epoch 48/50
68/68 [==============================] - 64s 940ms/step - loss: 0.0706 - accuracy: 0
Epoch 49/50
68/68 [==============================] - 65s 959ms/step - loss: 0.0698 - accuracy: 0
Epoch 50/50
68/68 [==============================] - 62s 917ms/step - loss: 0.0668 - accuracy: 0
```

```
plt.plot(np.array(history.history['loss']), "r--", label = "Train loss")
plt.plot(np.array(history.history['accuracy']), "g--", label = "Train accuracy")
plt.plot(np.array(history.history['val_loss']), "r-", label = "Validation loss")
plt.plot(np.array(history.history['val_accuracy']), "g-", label = "Validation accuracy")
plt.title("Training session's progress over iterations")
plt.legend(loc='lower left')
plt.ylabel('Training Progress (Loss/Accuracy)')
plt.xlabel('Training Epoch')
plt.ylim(0)
plt.show()
```



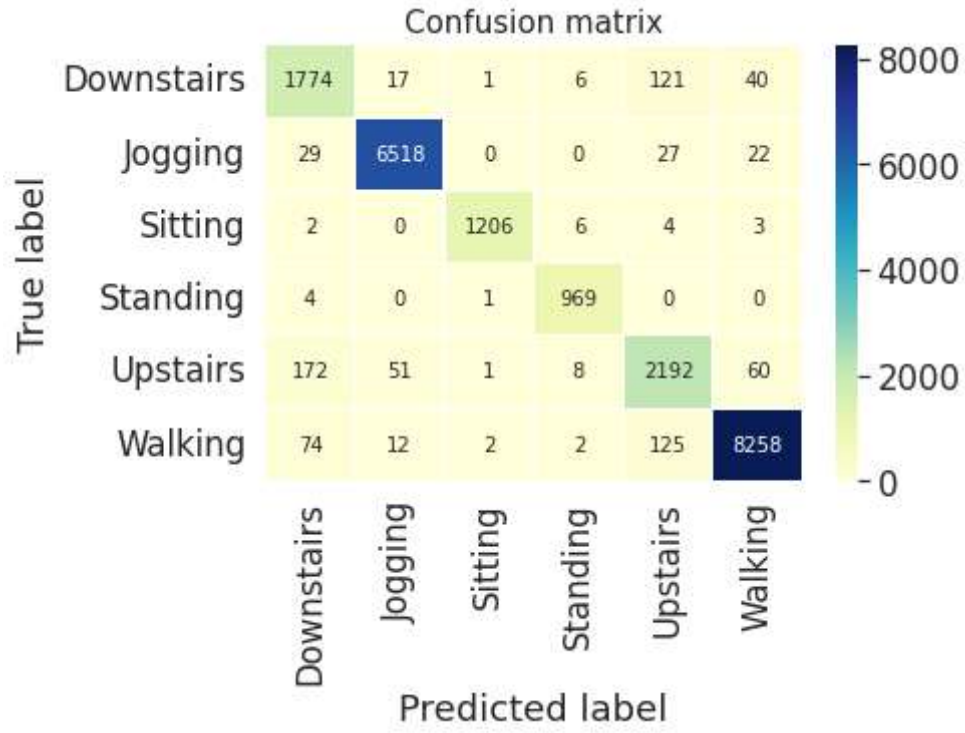

```
loss, accuracy = model.evaluate(X_test, y_test, batch_size = batch_size, verbose = 1)
print("Test Accuracy :", accuracy)
```

```
print("Test Loss :", loss)
```

```
22/22 [==============================] - 7s 300ms/step - loss: 0.1190 - accuracy: 0.963
Test Accuracy : 0.9636062383651733
Test Loss : 0.11902851611375809
```

Confusion matrix

```
predictions = model.predict(X_test)
class_labels = ['Downstairs', 'Jogging', 'Sitting', 'Standing', 'Upstairs', 'Walking']
max_test = np.argmax(y_test, axis=1)
max_predictions = np.argmax(predictions, axis=1)
confusion_matrix = metrics.confusion_matrix(max_test, max_predictions)
sns.heatmap(confusion_matrix, xticklabels = class_labels, yticklabels = class_labels, annot = True, linew
plt.title("Confusion matrix", fontsize = 15)
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.show()
```




```
1
```