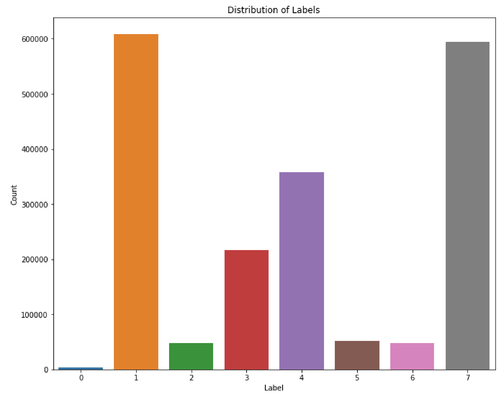
Assessment 2 - Multi class classification

Loading data and feature set

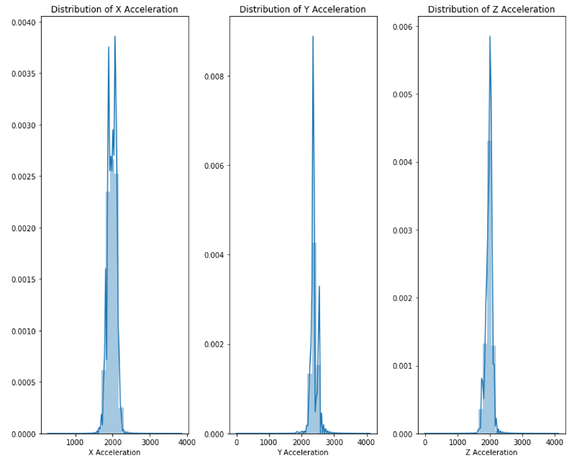
In [2]:
```
import pandas as pd
import os

# Loading the data from the CSV files into a Pandas DataFrame
data = pd.DataFrame()

data_folder = "C:/Users/Desktop/Data Scientist Assignment/Data Scientist Assignment/Activity Recognition/Activity Recognition/"

for participant_file in os.listdir(data_folder

    #Checking for aLL .csv files in the directory and concatenating data
    if participant_file.endswith(".csv"):
        participant_data = pd.read_csv(os.path.join(data_folder, participant_file))
        participant_data.columns = ['sequence', 'x acceleration', 'y acceleration', 'z acceleration', 'label']
        participant_data.drop('sequence', axis=1, inplace=True)
        data = data.append(participant_data, ignore_index=True)


# Splitting the data into input features (x acceleration, y acceleration, z acceleration) to the model and target labels
X = data[['x acceleration', 'y acceleration', 'z acceleration']]
y = data['label']
```

EDA

In [12]:
```
#Distribution of Labels

import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(11, 9))
sns.countplot(data=data, x='label')
plt.xlabel('Label')
plt.ylabel('Count')
plt.title('Distribution of Labels')
plt.show()
```



In [13]:
```
# Distribution of acceleration data

plt.figure(figsize=(11, 9))

plt.subplot(1, 3, 1)
sns.distplot(data['x acceleration'], bins=30, kde=True)
plt.xlabel('X Acceleration')
plt.title('Distribution of X Acceleration')

plt.subplot(1, 3, 2)
sns.distplot(data['y acceleration'], bins=30, kde=True)
plt.xlabel('Y Acceleration')
plt.title('Distribution of Y Acceleration')

plt.subplot(1, 3, 3)
sns.distplot(data['z acceleration'], bins=30, kde=True)
plt.xlabel('Z Acceleration')
plt.title('Distribution of Z Acceleration')

plt.tight_layout()
plt.show()
```



In [14]:
```
# Descriptive statistics
acceleration_stats = data[['x acceleration', 'y acceleration', 'z acceleration']].describe()
print("Descriptive Statistics for Acceleration Data:\n", acceleration_stats)
```

```
Descriptive Statistics for Acceleration Data:
       x acceleration  y acceleration  z acceleration
count    1.926881e+06    1.926881e+06    1.926881e+06
mean     1.987653e+03    2.382524e+03    1.970597e+03
std      1.113573e+02    1.003132e+02    9.445699e+01
min      2.820000e+02    2.000000e+00    1.000000e+00
25%      1.904000e+03    2.337000e+03    1.918000e+03
50%      1.992000e+03    2.367000e+03    1.988000e+03
75%      2.076000e+03    2.413000e+03    2.032000e+03
max      3.828000e+03    4.095000e+03    4.095000e+03
```

Model training and evaluation

In [7]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report

# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardizing the features using standard scaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Using knn classification algorithm for multi class classification
knn_classifier = KNeighborsClassifier()
knn_classifier.fit(X_train_scaled, y_train)

# Predicting on the test set
y_pred = knn_classifier.predict(X_test_scaled)

# Evaluating the model using accuracy and classification report
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.7224873305879697
Classification Report:
              precision    recall  f1-score   support

           0       0.20      0.10      0.13       755
           1       0.85      0.90      0.87    121700
           2       0.38      0.22      0.27      9517
           3       0.50      0.48      0.49     42961
           4       0.62      0.69      0.65     71594
           5       0.30      0.13      0.18     10296
           6       0.40      0.21      0.28      9585
           7       0.78      0.78      0.78    118969

    accuracy                           0.72    385377
   macro avg       0.50      0.44      0.46    385377
weighted avg       0.71      0.72      0.71    385377
```

In [17]:
```
#Prediction output

y_pred
```

Out[17]:
```
array([1, 1, 3, ..., 7, 1, 4], dtype=int64)
```