In [1]:
```
#191805004 Meltem Altınkaynak

#13-classification_human activity recongnition
```

1) dataset review

In [3]:
```
import pandas as pd

file_path = 'C:\\Users\\Admin\\Desktop\\dataset\\13-classification_human activit
data = pd.read_csv(file_path, delimiter=';')
print(data.head(10))
```

```
    gyro_x   gyro_y   gyro_z   accel_x   accel_y   accel_z  std_acc_30  \
0  0.49875 -0.64750  0.13125  0.685396 -0.630008  0.383141         0.0   
1  0.47250 -0.72625  0.12250  0.684420 -0.630191  0.383690         0.0   
2  0.39375 -0.63875  0.12250  0.687531 -0.629764  0.383507         0.0   
3  0.35875 -0.65625  0.09625  0.686616 -0.628971  0.384056         0.0   
4  0.29750 -0.60375  0.14000  0.685640 -0.631594  0.382714         0.0   
5  0.14875 -0.65625  0.14875  0.685640 -0.630374  0.380152         0.0   
6  0.24500 -0.64750  0.12250  0.685701 -0.630862  0.382348         0.0   
7  0.26250 -0.53375  0.25375  0.688141 -0.631167  0.381189         0.0   
8  0.24500 -0.71750  0.16625  0.686250 -0.631533  0.381982         0.0   
9  0.28875 -0.61250  0.21875  0.686067 -0.630008  0.385032         0.0   

   std_gyro_10  mean_acc_20  mean_gyro_20  max_acc_15  min_acc_20 Output  
0     0.000000          0.0           0.0         0.0         0.0    sit  
1     0.000000          0.0           0.0         0.0         0.0    sit  
2     0.000000          0.0           0.0         0.0         0.0    sit  
3     0.000000          0.0           0.0         0.0         0.0    sit  
4     0.000000          0.0           0.0         0.0         0.0    sit  
5     0.000000          0.0           0.0         0.0         0.0    sit  
6     0.000000          0.0           0.0         0.0         0.0    sit  
7     0.000000          0.0           0.0         0.0         0.0    sit  
8     0.000000          0.0           0.0         0.0         0.0    sit  
9     0.426983          0.0           0.0         0.0         0.0    sit  
```

In [4]:
```
print(data.shape)
```

```
(37161, 13)
```

In [5]:
```
print(data.columns)
```

```
Index(['gyro_x', 'gyro_y', 'gyro_z', 'accel_x', 'accel_y', 'accel_z',
       'std_acc_30', 'std_gyro_10', 'mean_acc_20', 'mean_gyro_20',
       'max_acc_15', 'min_acc_20', 'Output'],
      dtype='object')
```

In [6]:
```
print(data.dtypes)
print("Sütun sayısı:",len(data.columns))
```

```
gyro_x          float64
gyro_y          float64
gyro_z          float64
accel_x         float64
accel_y         float64
accel_z         float64
std_acc_30      float64
std_gyro_10     float64
mean_acc_20     float64
mean_gyro_20    float64
max_acc_15      float64
min_acc_20      float64
Output           object
dtype: object
Sütun sayısı: 13
```

In [7]:
```
print(data.isnull().sum())
```

```
gyro_x          0
gyro_y          0
gyro_z          0
accel_x         0
accel_y         0
accel_z         0
std_acc_30      0
std_gyro_10     0
mean_acc_20     0
mean_gyro_20    0
max_acc_15      0
min_acc_20      0
Output          0
dtype: int64
```

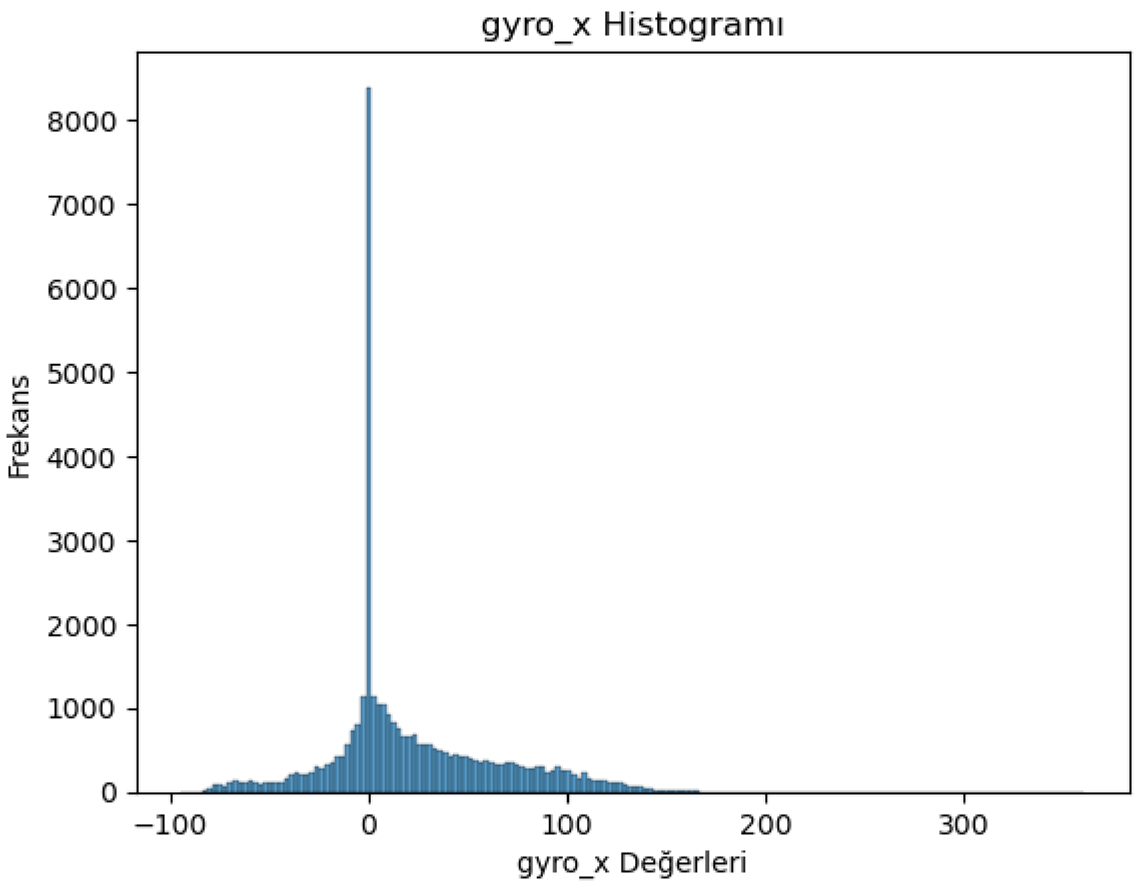
In [8]:
```
print(data.isna().sum())
```

```
gyro_x          0
gyro_y          0
gyro_z          0
accel_x         0
accel_y         0
accel_z         0
std_acc_30      0
std_gyro_10     0
mean_acc_20     0
mean_gyro_20    0
max_acc_15      0
min_acc_20      0
Output          0
dtype: int64
```

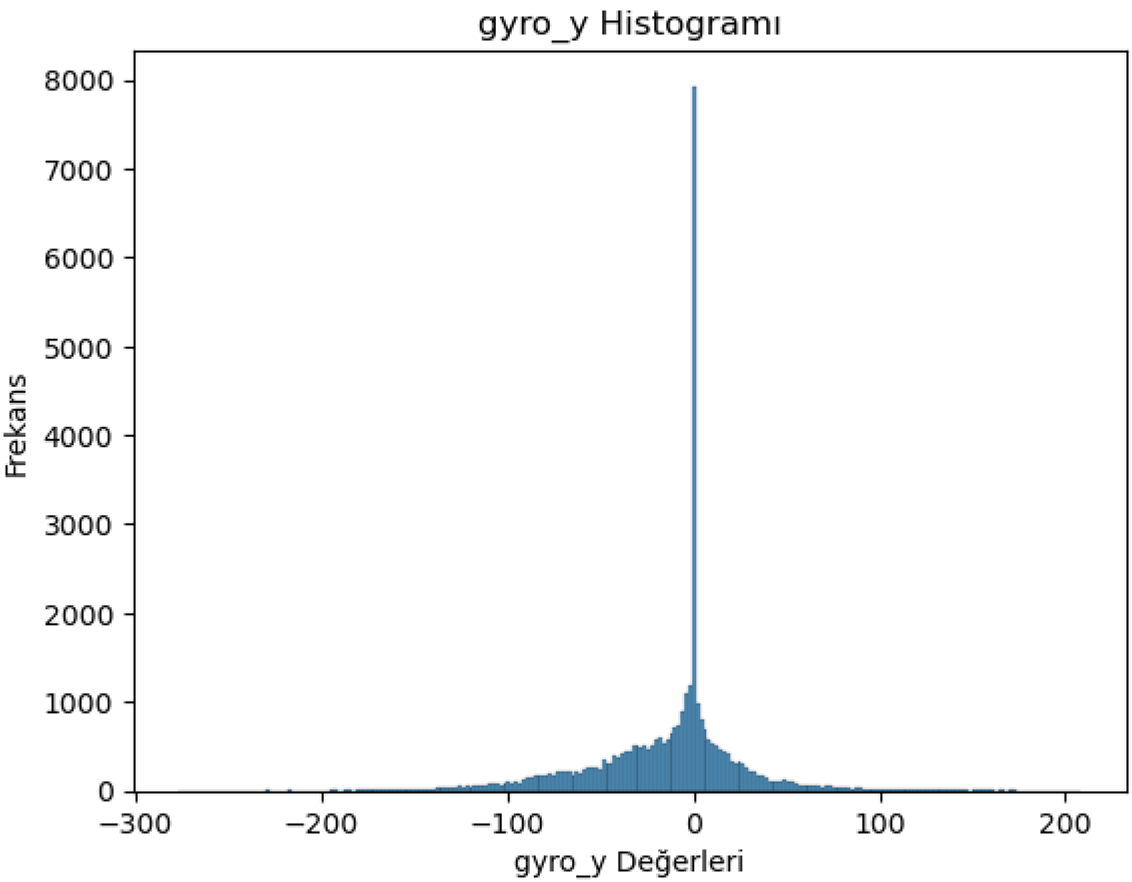
In [9]:
```
yeni_data = data.copy()
```

In [10]:
```
# gyro sütunları
import seaborn as sns
import matplotlib.pyplot as plt


sns.histplot(data=yeni_data, x='gyro_x')
plt.xlabel('gyro_x Değerleri')
plt.ylabel('Frekans')
plt.title('gyro_x Histogramı')
plt.show()


sns.histplot(data=yeni_data, x='gyro_y')
plt.xlabel('gyro_y Değerleri')
plt.ylabel('Frekans')
plt.title('gyro_y Histogramı')
plt.show()


sns.histplot(data=yeni_data, x='gyro_z')
plt.xlabel('gyro_z Değerleri')
plt.ylabel('Frekans')
plt.title('gyro_z Histogramı')
plt.show()
```

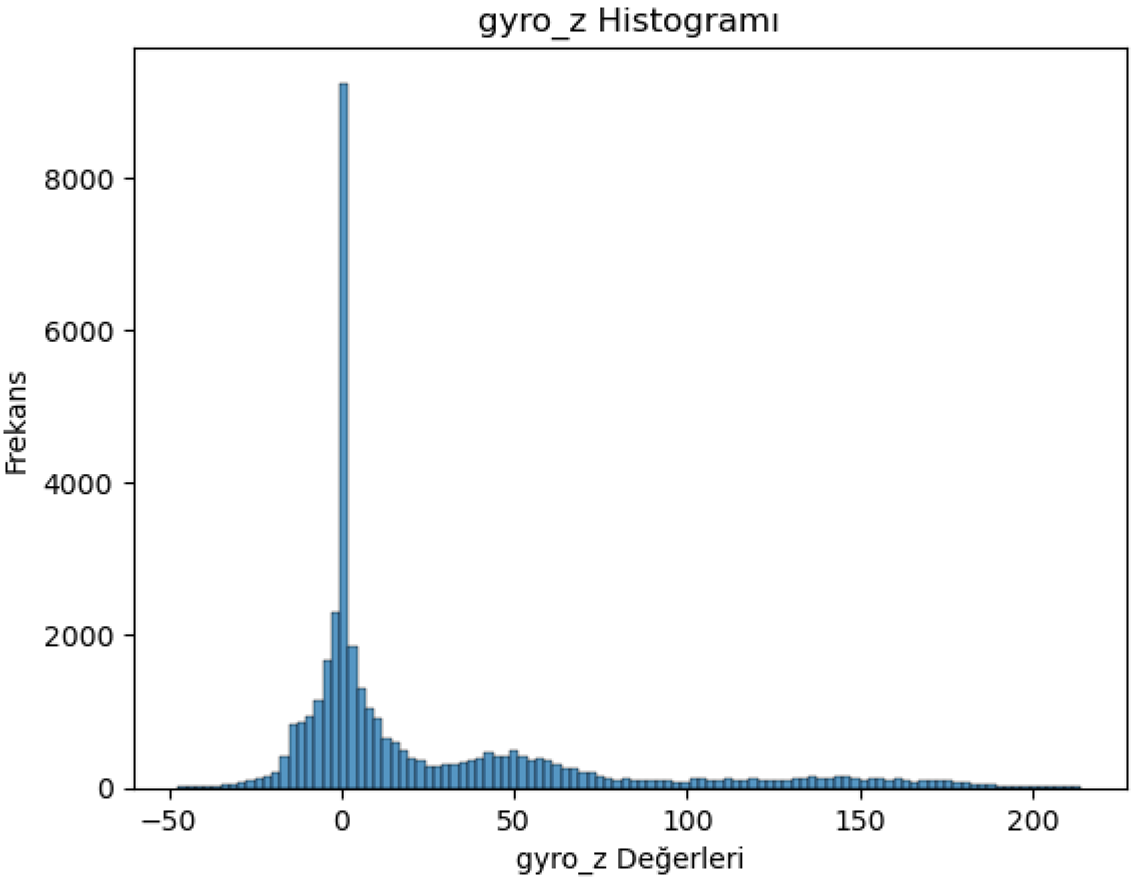
```
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarnin
g: use_inf_as_na option is deprecated and will be removed in a future version. Co
nvert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



```
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarnin
g: use_inf_as_na option is deprecated and will be removed in a future version. Co
nvert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



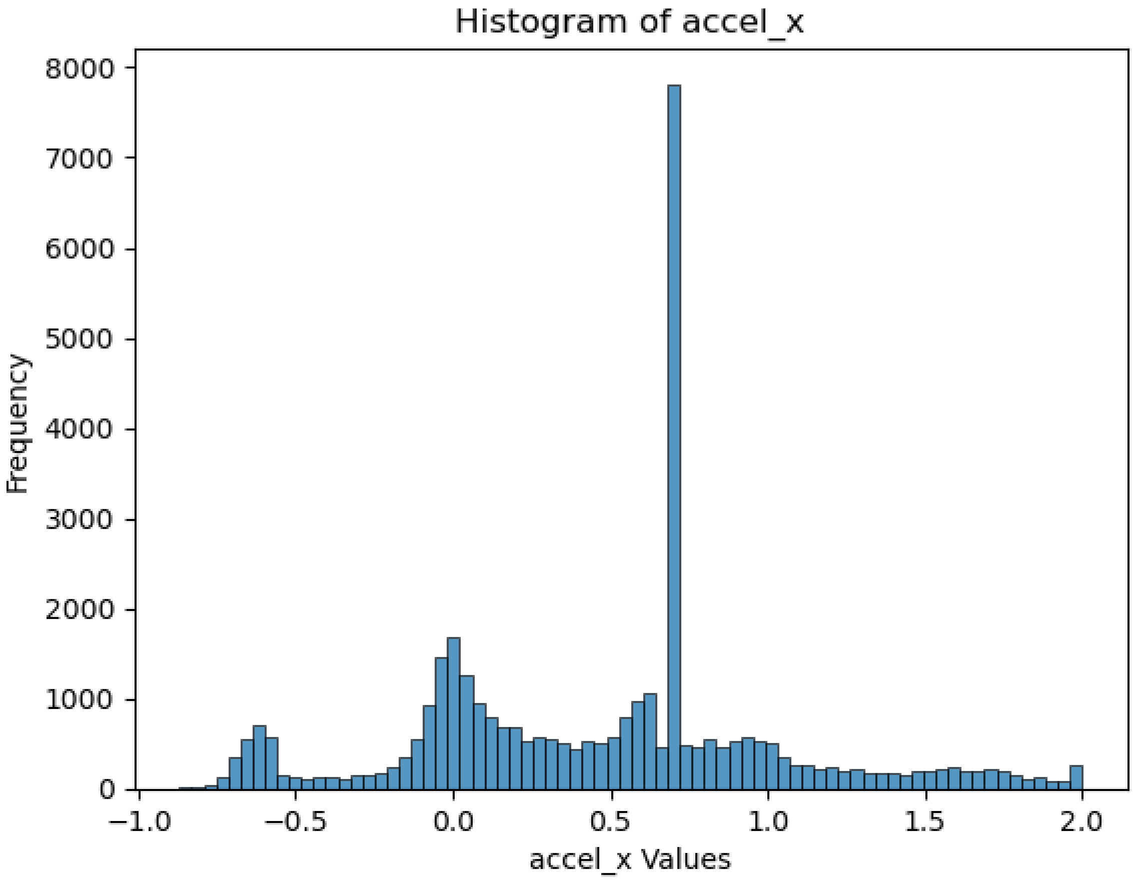
```
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarnin
g: use_inf_as_na option is deprecated and will be removed in a future version. Co
nvert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



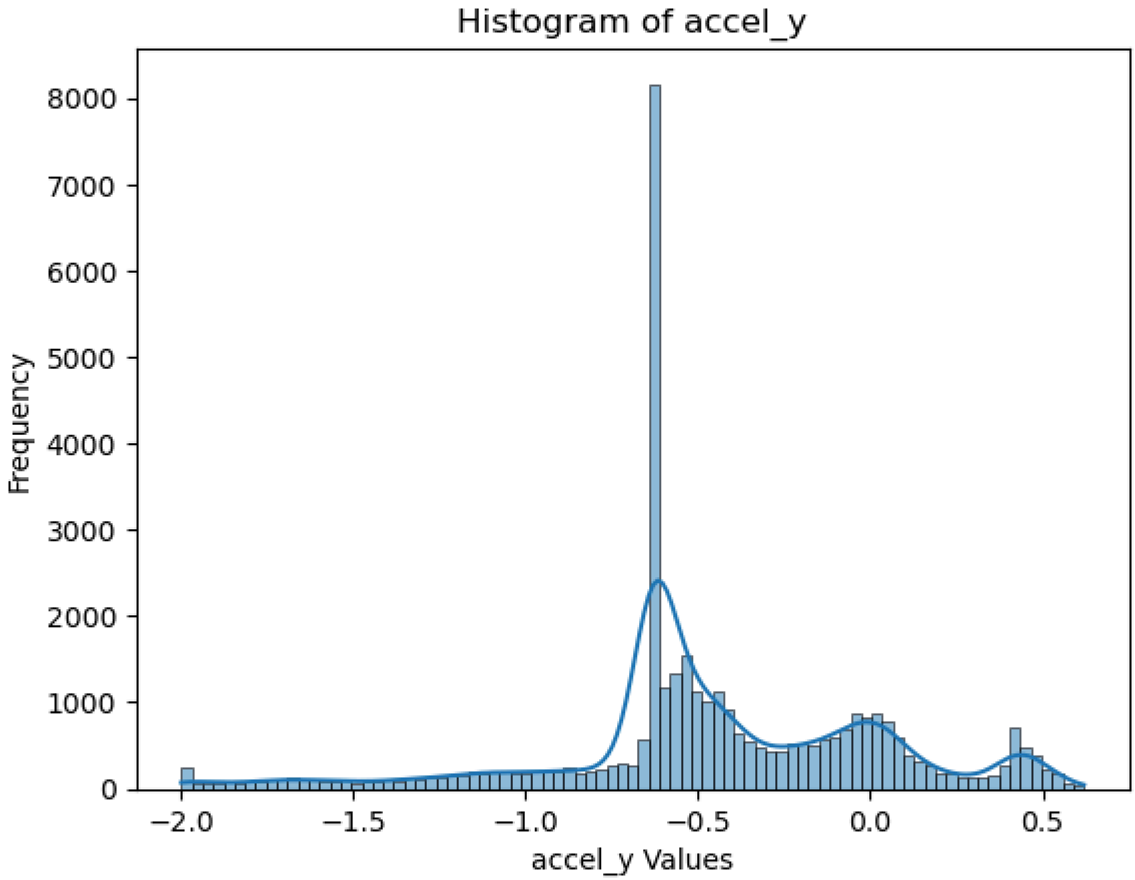
In [11]:

```
# accel sütunları
sns.histplot(data=yeni_data, x='accel_x')
plt.xlabel('accel_x Values')
plt.ylabel('Frequency')
plt.title('Histogram of accel_x')
plt.show()

sns.histplot(data=yeni_data, x='accel_y', kde=True)
plt.xlabel('accel_y Values')
plt.ylabel('Frequency')
plt.title('Histogram of accel_y')
plt.show()

sns.histplot(data=yeni_data, x='accel_z', kde=True)
plt.xlabel('accel_z Values')
plt.ylabel('Frequency')
plt.title('Histogram of accel_z')
plt.show()
```

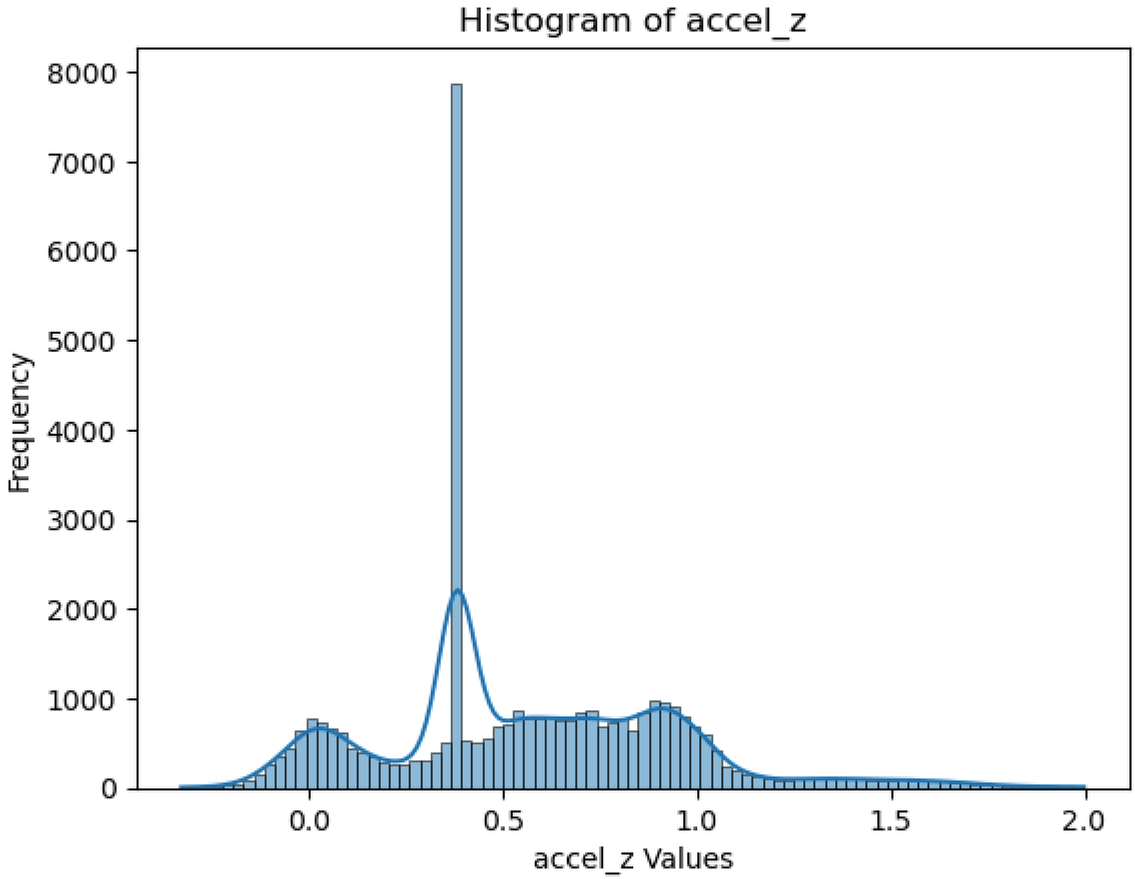
```
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarnin
g: use_inf_as_na option is deprecated and will be removed in a future version. Co
nvert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



```
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarnin
g: use_inf_as_na option is deprecated and will be removed in a future version. Co
nvert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

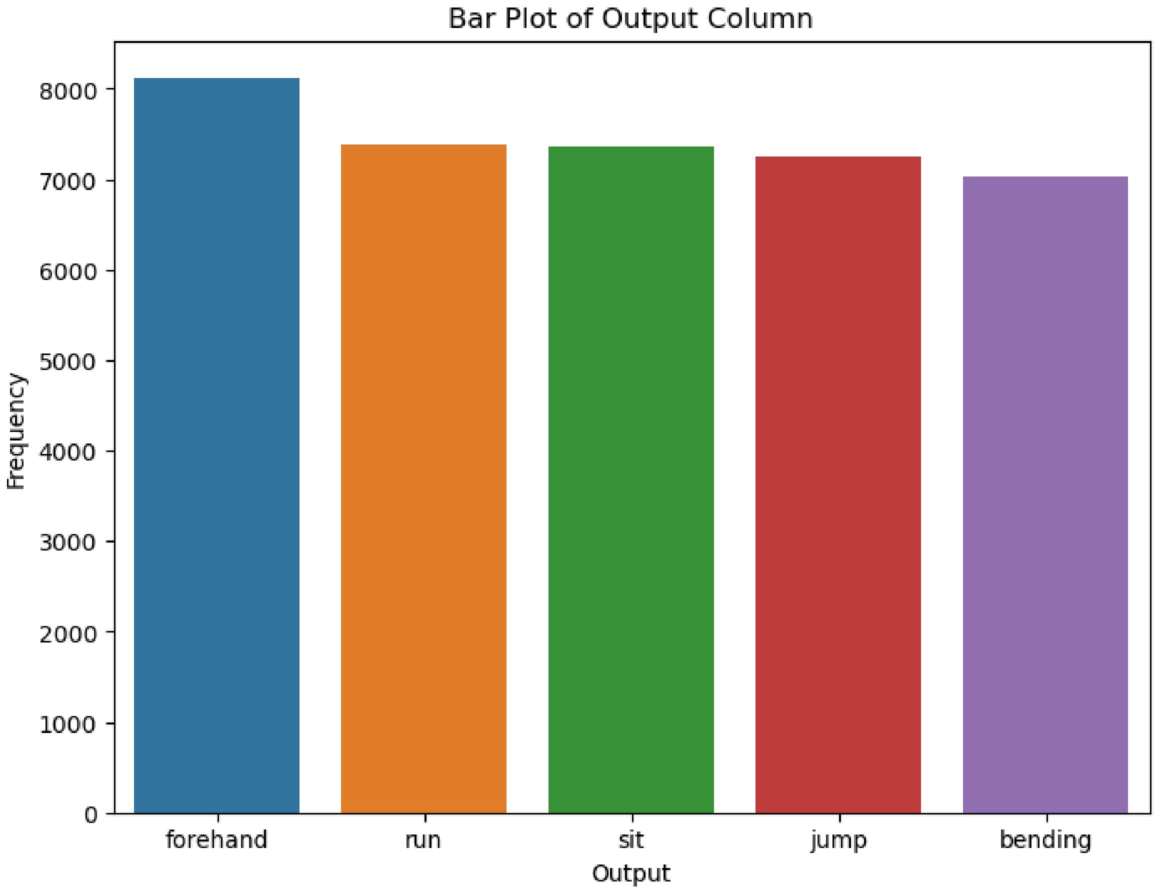


```
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarnin
g: use_inf_as_na option is deprecated and will be removed in a future version. Co
nvert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



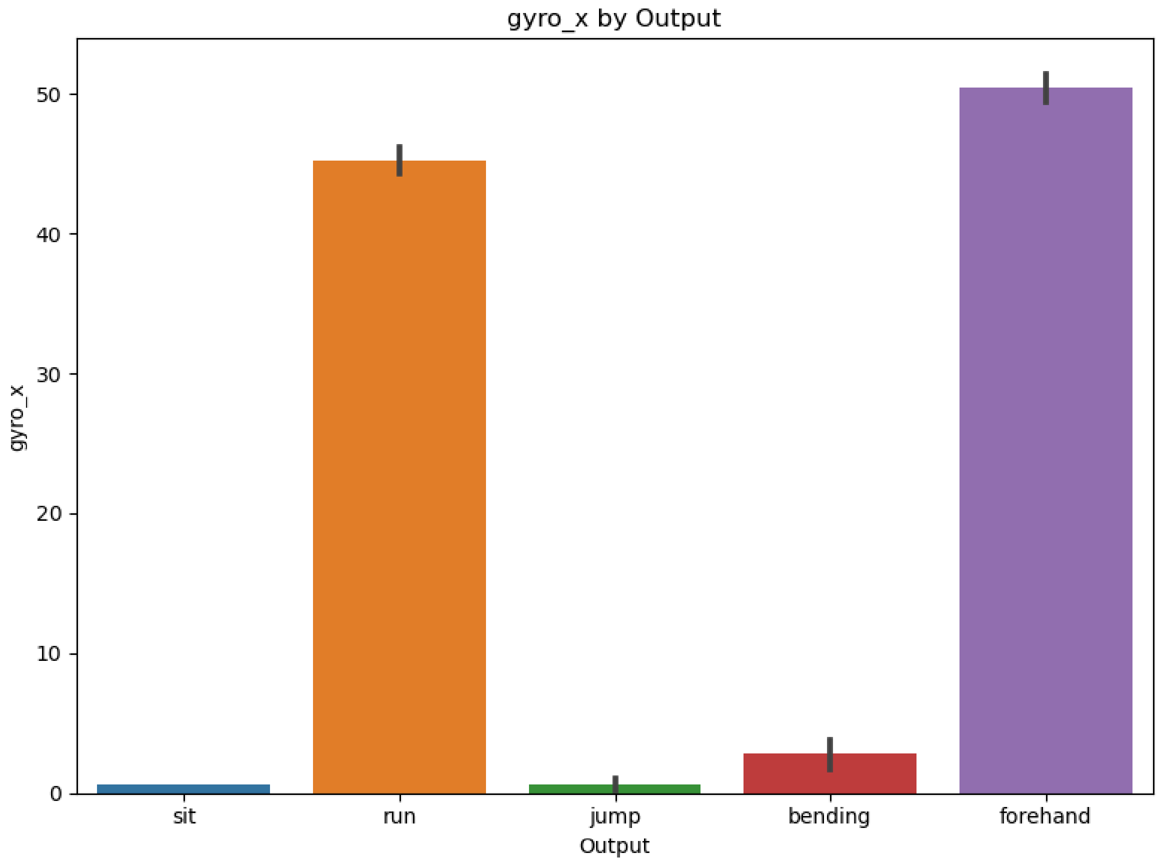
In [12]:

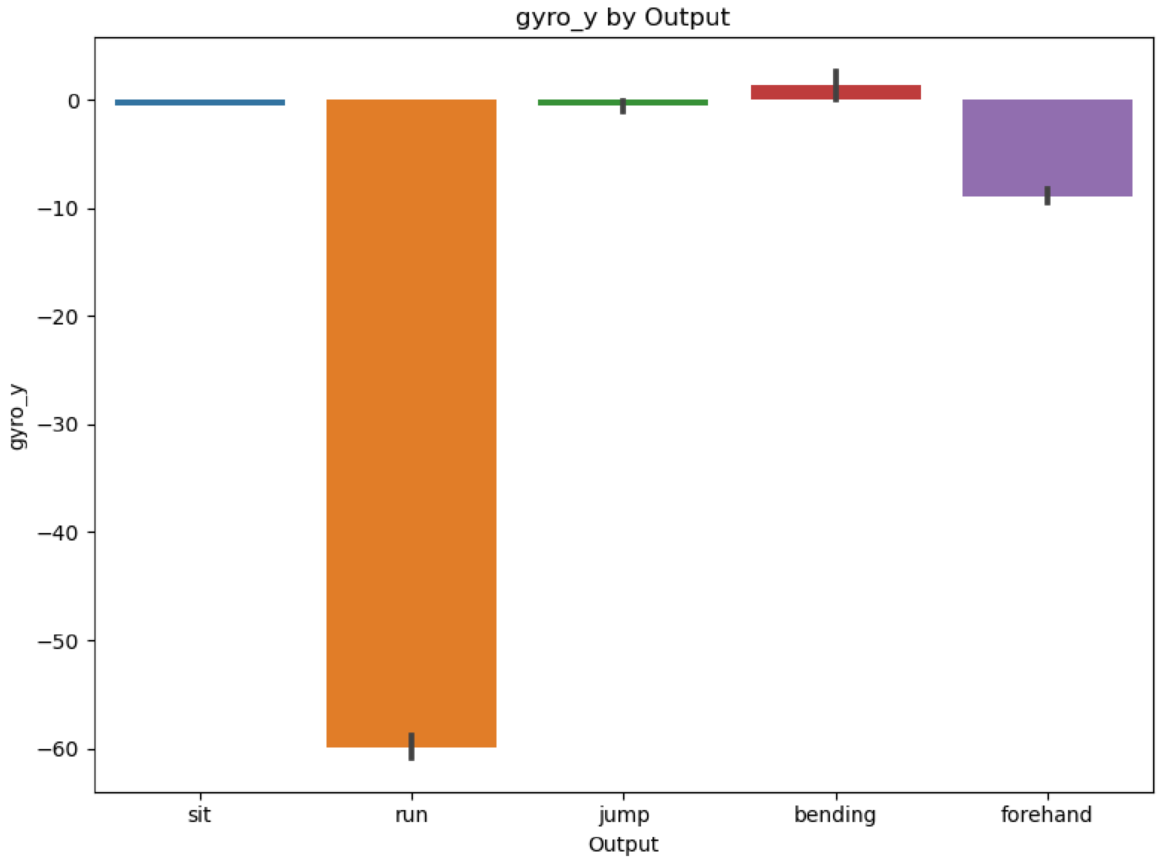
```
# output sütunu

output_counts = yeni_data['Output'].value_counts()

plt.figure(figsize=(8, 6))
sns.barplot(x=output_counts.index, y=output_counts.values)
plt.xlabel('Output')
plt.ylabel('Frequency')
plt.title('Bar Plot of Output Column')
plt.show()
```

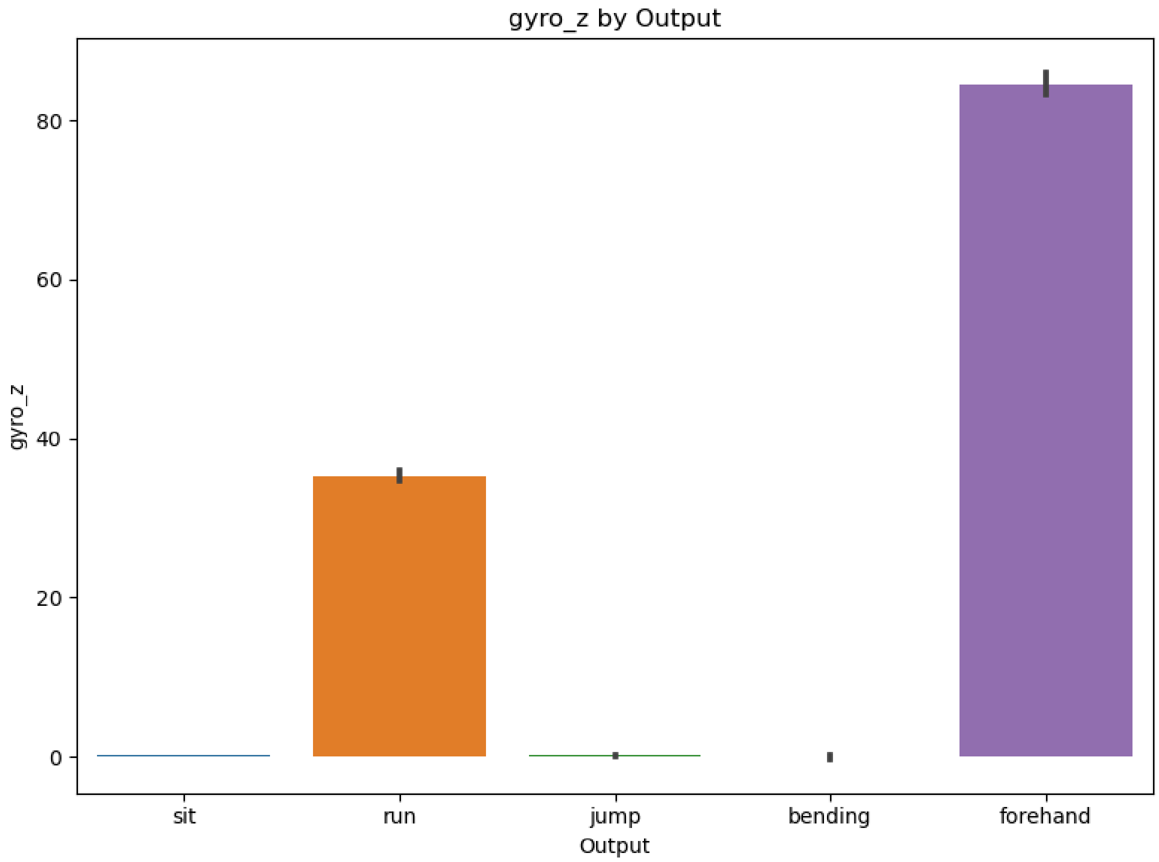


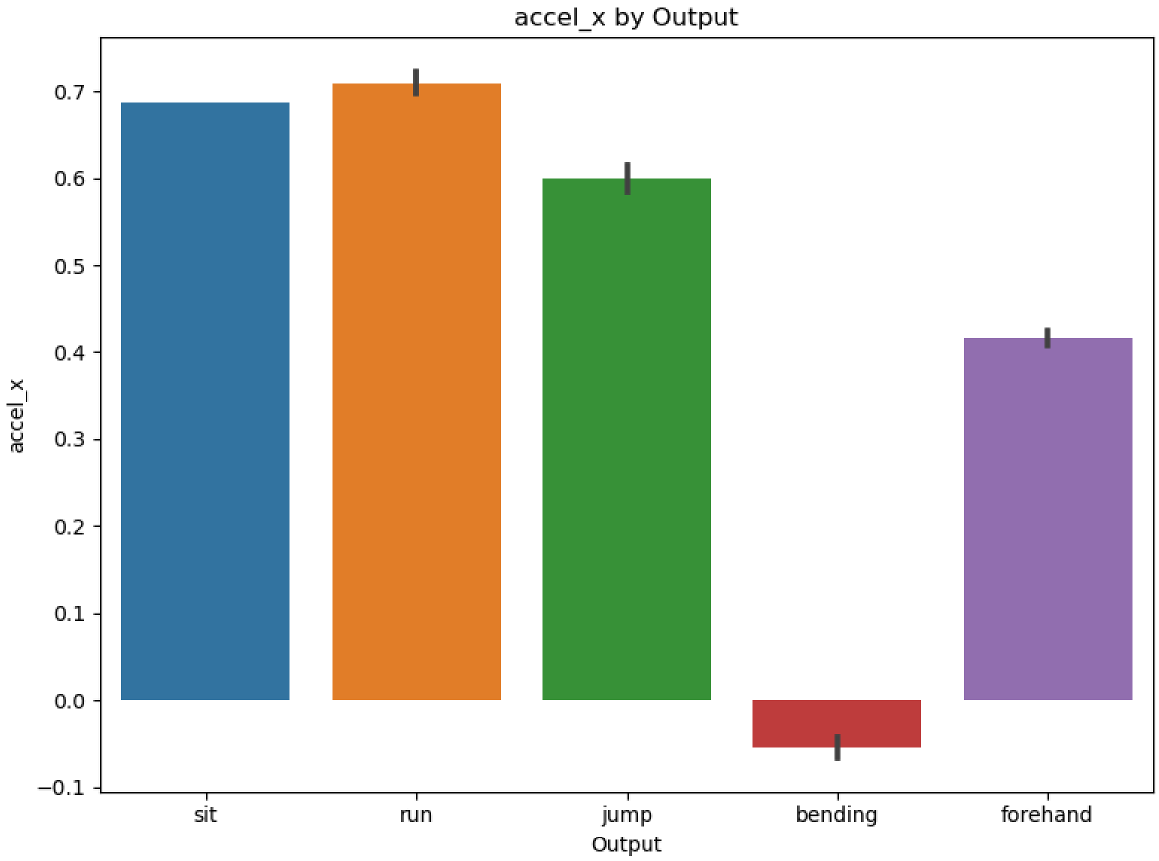
In [13]:

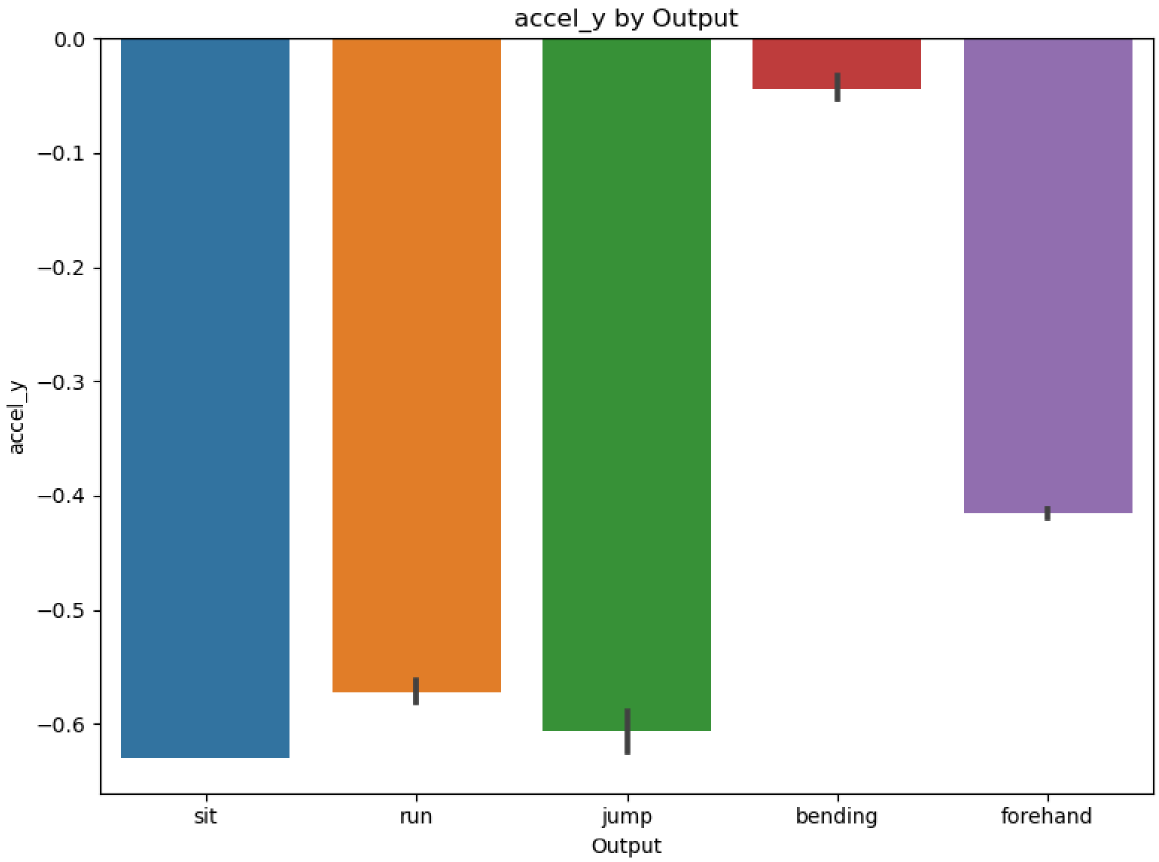
```
# gyro ve accel sütunlarının outputa göre dağılımı

input_columns = ['gyro_x', 'gyro_y', 'gyro_z', 'accel_x', 'accel_y', 'accel_z']

for column in input_columns:
    plt.figure(figsize=(8, 6))
    sns.barplot(x='Output', y=column, data=yeni_data)
    plt.xlabel('Output')
    plt.ylabel(column)
    plt.title(column + ' by Output')
    plt.tight_layout()
    plt.show()
```

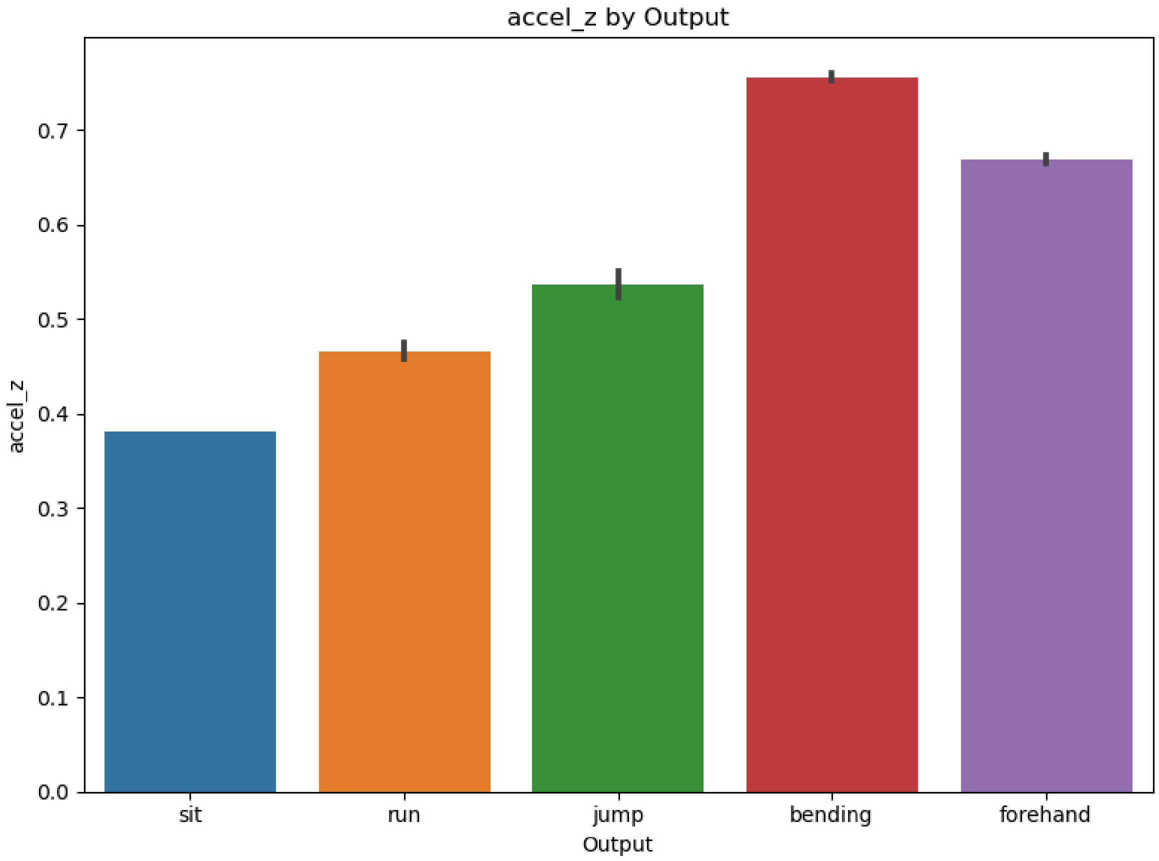

gyro_x by Output
gyro_x
0
10
20
30
40
50
sit
run
jump
bending
forehand
Output


gyro_y by Output
gyro_y
0
−10
−20
−30
−40
−50
−60
sit
run
jump
bending
forehand
Output


gyro_z by Output
gyro_z
0
20
40
60
80
sit
run
jump
bending
forehand
Output

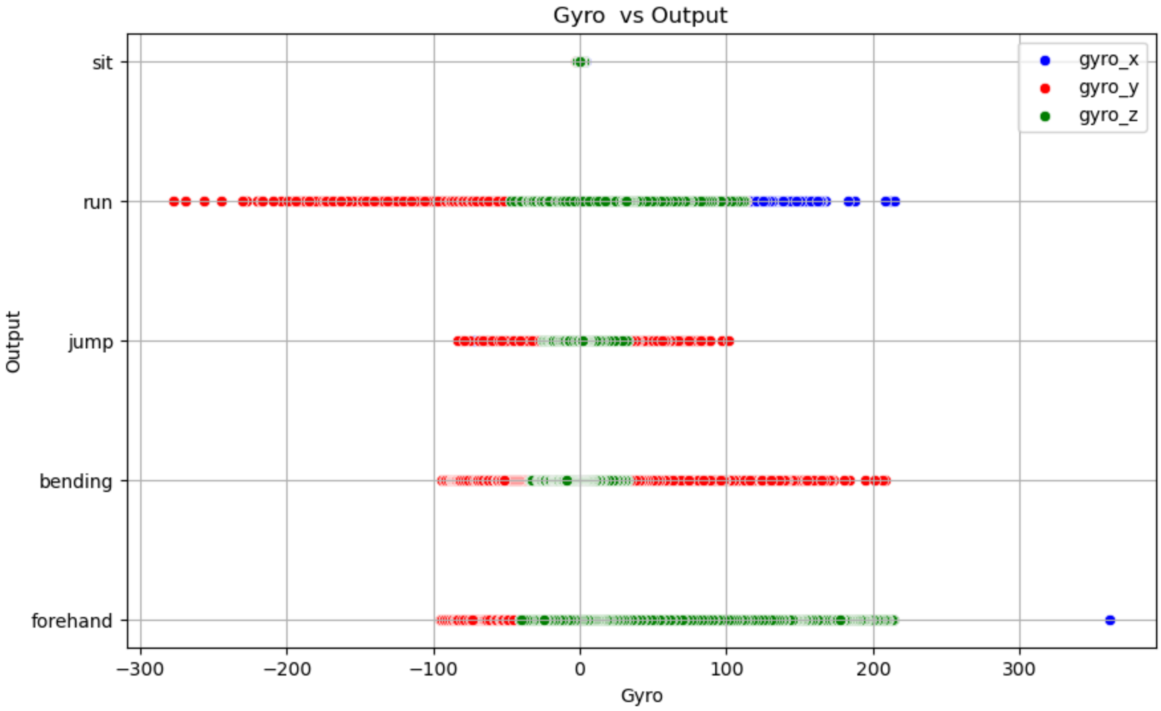

accel_x by Output
accel_x
−0.1
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
sit
run
jump
bending
forehand
Output





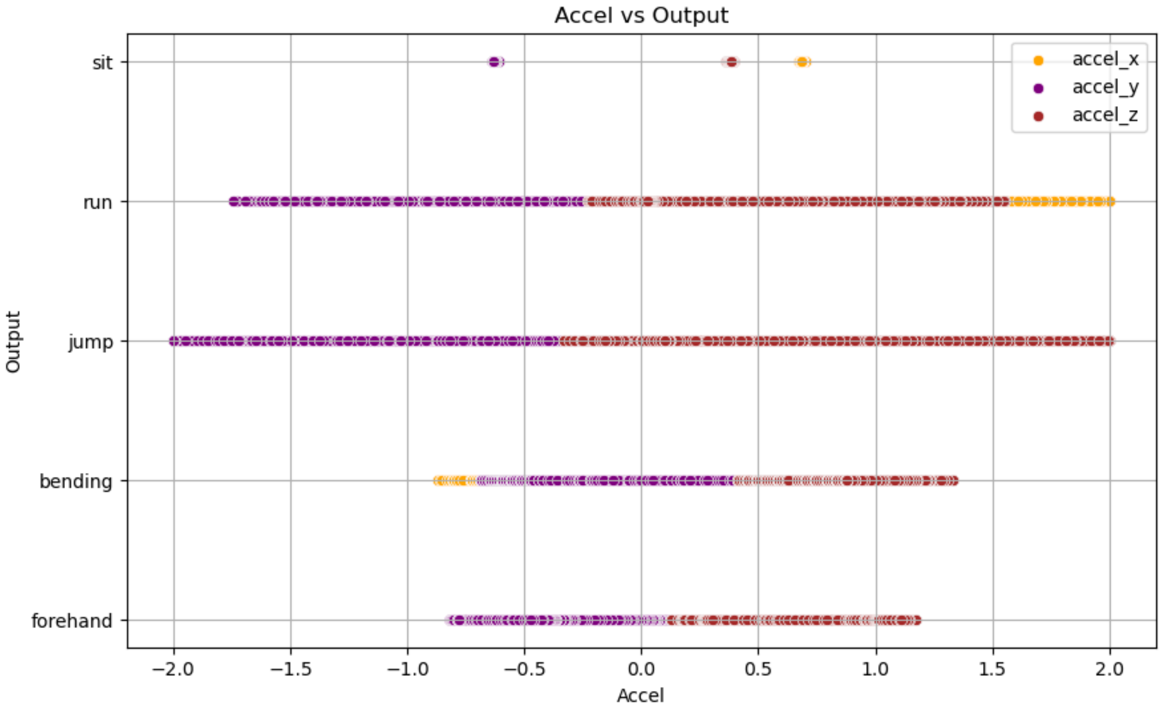
In [14]:

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

plt.figure(figsize=(10, 6))

sns.scatterplot(data=yeni_data, x='gyro_x', y='Output', color='blue', label='gyr
sns.scatterplot(data=yeni_data, x='gyro_y', y='Output', color='red', label='gyro
sns.scatterplot(data=yeni_data, x='gyro_z', y='Output', color='green', label='gy


plt.title('Gyro  vs Output')
plt.xlabel('Gyro')
plt.ylabel('Output')
plt.legend()
plt.grid(True)
plt.show()
```



In [15]:

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

plt.figure(figsize=(10, 6))


sns.scatterplot(data=yeni_data, x='accel_x', y='Output', color='orange', label='
sns.scatterplot(data=yeni_data, x='accel_y', y='Output', color='purple', label='
sns.scatterplot(data=yeni_data, x='accel_z', y='Output', color='brown', label='a

plt.title('Accel vs Output')
plt.xlabel('Accel')
plt.ylabel('Output')
plt.legend()
plt.grid(True)
plt.show()
```



2) Feature Extraction

2.1 peak to peak

In [16]:
```
#  peak to peak  - ['gyro_x', 'gyro_y', 'gyro_z', 'accel_x', 'accel_y', 'accel_z
#  yeni featurelar elde edilmesi: Her bir output değerine indexleri dikkate alın
```

In [17]:
```
# Output sütununun aldığı unique değerler ve sayıları:

output_values = yeni_data["Output"]
number_of_output_values = output_values.value_counts()

print(number_of_output_values)
```

```
Output
forehand    8125
run         7386
sit         7365
jump        7255
bending     7030
Name: count, dtype: int64
```

In [18]:
```
# Output sütunun her bir unique değerinin index bilgileri:

import pandas as pd

output_unique_values = yeni_data["Output"].unique()

for value in output_unique_values :
    start_index = yeni_data[yeni_data["Output"] == value].index.min()
    end_index = yeni_data[yeni_data["Output"] == value].index.max()
    print(f"{value} value: start index : {start_index}, end index index: {end_in
sit value: start index : 0, end index index: 7364
run value: start index : 7365, end index index: 14750
jump value: start index : 14751, end index index: 22005
bending value: start index : 22006, end index index: 29035
forehand value: start index : 29036, end index index: 37160
```

In [19]:

```
# sit value: start index : 0, end index index: 7364. window sizedan dolayı işlem

import pandas as pd

gyro_x = yeni_data['gyro_x']
gyro_y = yeni_data['gyro_y']
gyro_z = yeni_data['gyro_z']

accel_x = yeni_data['accel_x']
accel_y = yeni_data['accel_y']
accel_z = yeni_data['accel_z']

window_size = 5

for i in range(7361):  #7364'te duracak
    accel_x_window = accel_x[i:i+window_size]
    accel_y_window = accel_y[i:i+window_size]
    accel_z_window = accel_z[i:i+window_size]

    gyro_x_window = gyro_x[i:i+window_size]
    gyro_y_window = gyro_y[i:i+window_size]
    gyro_z_window = gyro_z[i:i+window_size]

    peak_to_peak_accel_x = accel_x_window.max() - accel_x_window.min()
    peak_to_peak_accel_y = accel_y_window.max() - accel_y_window.min()
    peak_to_peak_accel_z = accel_z_window.max() - accel_z_window.min()

    peak_to_peak_gyro_x = gyro_x_window.max() - gyro_x_window.min()
    peak_to_peak_gyro_y = gyro_y_window.max() - gyro_y_window.min()
    peak_to_peak_gyro_z = gyro_z_window.max() - gyro_z_window.min()

    yeni_data.at[i + window_size - 1, 'peak_to_peak_acc_x'] = peak_to_peak_accel
    yeni_data.at[i + window_size - 1, 'peak_to_peak_acc_y'] = peak_to_peak_accel
    yeni_data.at[i + window_size - 1, 'peak_to_peak_acc_z'] = peak_to_peak_accel

    yeni_data.at[i + window_size - 1, 'peak_to_peak_gyro_x'] = peak_to_peak_gyrc
    yeni_data.at[i + window_size - 1, 'peak_to_peak_gyro_y'] = peak_to_peak_gyrc
    yeni_data.at[i + window_size - 1, 'peak_to_peak_gyro_z'] = peak_to_peak_gyrc
```

In [20]:

```
# Eklenen yeni sütunlar:
print(yeni_data.head(10))
```

```
    gyro_x   gyro_y   gyro_z   accel_x   accel_y   accel_z  std_acc_30  \
0  0.49875 -0.64750  0.13125  0.685396 -0.630008  0.383141         0.0   
1  0.47250 -0.72625  0.12250  0.684420 -0.630191  0.383690         0.0   
2  0.39375 -0.63875  0.12250  0.687531 -0.629764  0.383507         0.0   
3  0.35875 -0.65625  0.09625  0.686616 -0.628971  0.384056         0.0   
4  0.29750 -0.60375  0.14000  0.685640 -0.631594  0.382714         0.0   
5  0.14875 -0.65625  0.14875  0.685640 -0.630374  0.380152         0.0   
6  0.24500 -0.64750  0.12250  0.685701 -0.630862  0.382348         0.0   
7  0.26250 -0.53375  0.25375  0.688141 -0.631167  0.381189         0.0   
8  0.24500 -0.71750  0.16625  0.686250 -0.631533  0.381982         0.0   
9  0.28875 -0.61250  0.21875  0.686067 -0.630008  0.385032         0.0   

   std_gyro_10  mean_acc_20  mean_gyro_20  max_acc_15  min_acc_20 Output  \
0     0.000000          0.0           0.0         0.0         0.0    sit   
1     0.000000          0.0           0.0         0.0         0.0    sit   
2     0.000000          0.0           0.0         0.0         0.0    sit   
3     0.000000          0.0           0.0         0.0         0.0    sit   
4     0.000000          0.0           0.0         0.0         0.0    sit   
5     0.000000          0.0           0.0         0.0         0.0    sit   
6     0.000000          0.0           0.0         0.0         0.0    sit   
7     0.000000          0.0           0.0         0.0         0.0    sit   
8     0.000000          0.0           0.0         0.0         0.0    sit   
9     0.426983          0.0           0.0         0.0         0.0    sit   

   peak_to_peak_acc_x  peak_to_peak_acc_y  peak_to_peak_acc_z  \
0                 NaN                 NaN                 NaN   
1                 NaN                 NaN                 NaN   
2                 NaN                 NaN                 NaN   
3                 NaN                 NaN                 NaN   
4            0.003111            0.002623            0.001342   
5            0.003111            0.002623            0.003904   
6            0.001891            0.002623            0.003904   
7            0.002501            0.002623            0.003904   
8            0.002501            0.001220            0.002562   
9            0.002501            0.001525            0.004880   

   peak_to_peak_gyro_x  peak_to_peak_gyro_y  peak_to_peak_gyro_z  
0                  NaN                  NaN                  NaN  
1                  NaN                  NaN                  NaN  
2                  NaN                  NaN                  NaN  
3                  NaN                  NaN                  NaN  
4              0.20125              0.12250              0.04375  
5              0.32375              0.12250              0.05250  
6              0.24500              0.05250              0.05250  
7              0.21000              0.12250              0.15750  
8              0.14875              0.18375              0.13125  
9              0.14000              0.18375              0.13125  
```

In [21]:

```
# İşlemin durması gereken yerde durduğunu kontrol ediyoruz:

print(yeni_data.loc[7360:7370])
```

```
        gyro_x     gyro_y   gyro_z   accel_x   accel_y   accel_z  std_acc_30  \
7360  0.56875  -0.236250  0.08750  0.685823 -0.629764  0.383324    0.562036   
7361  0.49875  -0.210000  0.13125  0.684664 -0.628483  0.385947    0.561985   
7362  0.62125  -0.262500  0.14000  0.685335 -0.629642  0.385825    0.561956   
7363  0.63875  -0.131250  0.11375  0.683566 -0.630923  0.385337    0.561958   
7364  0.56875  -0.227500  0.12250  0.685091 -0.629520  0.385093    0.561925   
7365  1.44375  17.202499 -4.67250  0.559248 -0.641537  0.549427    0.000000   
7366  0.36750  16.213751 -4.80375  0.550403 -0.638182  0.542595    0.000000   
7367  0.07875  16.379999 -4.71625  0.551196 -0.620675  0.554978    0.000000   
7368 -0.02625  15.627500 -4.62875  0.550830 -0.629154  0.561566    0.000000   
7369 -0.93625  14.096250 -4.64625  0.549061 -0.641171  0.560102    0.000000   
7370 -1.53125  13.116250 -4.58500  0.544364 -0.654408  0.539606    0.000000   

      std_gyro_10  mean_acc_20  mean_gyro_20  max_acc_15  min_acc_20 Output  \
7360     0.539216     0.145744      0.172813     0.68808    -0.63196    sit   
7361     0.481757     0.145752      0.178937     0.68808    -0.63196    sit   
7362     0.437847     0.145807      0.183312     0.68808    -0.63196    sit   
7363     0.404484     0.145834      0.189875     0.68808    -0.63196    sit   
7364     0.372649     0.145953      0.192354     0.68808    -0.63196    sit   
7365     0.000000     0.000000      0.000000     0.00000     0.00000    run   
7366     0.000000     0.000000      0.000000     0.00000     0.00000    run   
7367     0.000000     0.000000      0.000000     0.00000     0.00000    run   
7368     0.000000     0.000000      0.000000     0.00000     0.00000    run   
7369     0.000000     0.000000      0.000000     0.00000     0.00000    run   
7370     0.000000     0.000000      0.000000     0.00000     0.00000    run   

      peak_to_peak_acc_x  peak_to_peak_acc_y  peak_to_peak_acc_z  \
7360            0.000793            0.002379            0.002684   
7361            0.001342            0.001952            0.005307   
7362            0.001342            0.001952            0.005307   
7363            0.002440            0.002440            0.005307   
7364            0.002257            0.002440            0.002623   
7365                 NaN                 NaN                 NaN   
7366                 NaN                 NaN                 NaN   
7367                 NaN                 NaN                 NaN   
7368                 NaN                 NaN                 NaN   
7369                 NaN                 NaN                 NaN   
7370                 NaN                 NaN                 NaN   

      peak_to_peak_gyro_x  peak_to_peak_gyro_y  peak_to_peak_gyro_z  
7360              0.12250              0.16625              0.12250  
7361              0.09625              0.18375              0.15750  
7362              0.12250              0.17500              0.13125  
7363              0.14000              0.13125              0.13125  
7364              0.14000              0.13125              0.05250  
7365                  NaN                  NaN                  NaN  
7366                  NaN                  NaN                  NaN  
7367                  NaN                  NaN                  NaN  
7368                  NaN                  NaN                  NaN  
7369                  NaN                  NaN                  NaN  
7370                  NaN                  NaN                  NaN  
```

In [22]:

```
# run value: start index : 7365, end index index: 14750

window_size = 5

for i in range(7365, 14747):  # 14750'de duracak
    accel_x_window = accel_x[i:i+window_size]
    accel_y_window = accel_y[i:i+window_size]
    accel_z_window = accel_z[i:i+window_size]
    gyro_x_window = gyro_x[i:i+window_size]
    gyro_y_window = gyro_y[i:i+window_size]
    gyro_z_window = gyro_z[i:i+window_size]

    peak_to_peak_accel_x = accel_x_window.max() - accel_x_window.min()
    peak_to_peak_accel_y = accel_y_window.max() - accel_y_window.min()
    peak_to_peak_accel_z = accel_z_window.max() - accel_z_window.min()

    peak_to_peak_gyro_x = gyro_x_window.max() - gyro_x_window.min()
    peak_to_peak_gyro_y = gyro_y_window.max() - gyro_y_window.min()
    peak_to_peak_gyro_z = gyro_z_window.max() - gyro_z_window.min()

    yeni_data.at[i + window_size - 1, 'peak_to_peak_acc_x'] = peak_to_peak_accel
    yeni_data.at[i + window_size - 1, 'peak_to_peak_acc_y'] = peak_to_peak_accel
    yeni_data.at[i + window_size - 1, 'peak_to_peak_acc_z'] = peak_to_peak_accel

    yeni_data.at[i + window_size - 1, 'peak_to_peak_gyro_x'] = peak_to_peak_gyrc
    yeni_data.at[i + window_size - 1, 'peak_to_peak_gyro_y'] = peak_to_peak_gyrc
    yeni_data.at[i + window_size - 1, 'peak_to_peak_gyro_z'] = peak_to_peak_gyrc
```

In [23]:

```
# run için başlangıç indexi kontrol

print(yeni_data.loc[7360:7375])
```

```
       gyro_x     gyro_y   gyro_z   accel_x   accel_y   accel_z  std_acc_30  \
7360  0.56875  -0.236250  0.08750  0.685823 -0.629764  0.383324    0.562036   
7361  0.49875  -0.210000  0.13125  0.684664 -0.628483  0.385947    0.561985   
7362  0.62125  -0.262500  0.14000  0.685335 -0.629642  0.385825    0.561956   
7363  0.63875  -0.131250  0.11375  0.683566 -0.630923  0.385337    0.561958   
7364  0.56875  -0.227500  0.12250  0.685091 -0.629520  0.385093    0.561925   
7365  1.44375  17.202499 -4.67250  0.559248 -0.641537  0.549427    0.000000   
7366  0.36750  16.213751 -4.80375  0.550403 -0.638182  0.542595    0.000000   
7367  0.07875  16.379999 -4.71625  0.551196 -0.620675  0.554978    0.000000   
7368 -0.02625  15.627500 -4.62875  0.550830 -0.629154  0.561566    0.000000   
7369 -0.93625  14.096250 -4.64625  0.549061 -0.641171  0.560102    0.000000   
7370 -1.53125  13.116250 -4.58500  0.544364 -0.654408  0.539606    0.000000   
7371 -2.04750  13.536250 -4.66375  0.545157 -0.638121  0.533079    0.000000   
7372 -1.45250  14.463750 -4.62875  0.540460 -0.636840  0.541802    0.000000   
7373 -0.95375  15.688750 -4.55875  0.530029 -0.647759  0.547048    0.000000   
7374 -0.84875  16.030001 -4.46250  0.520452 -0.645319  0.544486    0.000000   
7375 -0.46375  15.802500 -4.36625  0.522465 -0.649162  0.546865    0.000000   

      std_gyro_10  mean_acc_20  mean_gyro_20  max_acc_15  min_acc_20 Output  \
7360     0.539216     0.145744      0.172813     0.68808    -0.63196    sit   
7361     0.481757     0.145752      0.178937     0.68808    -0.63196    sit   
7362     0.437847     0.145807      0.183312     0.68808    -0.63196    sit   
7363     0.404484     0.145834      0.189875     0.68808    -0.63196    sit   
7364     0.372649     0.145953      0.192354     0.68808    -0.63196    sit   
7365     0.000000     0.000000      0.000000     0.00000     0.00000    run   
7366     0.000000     0.000000      0.000000     0.00000     0.00000    run   
7367     0.000000     0.000000      0.000000     0.00000     0.00000    run   
7368     0.000000     0.000000      0.000000     0.00000     0.00000    run   
7369     0.000000     0.000000      0.000000     0.00000     0.00000    run   
7370     0.000000     0.000000      0.000000     0.00000     0.00000    run   
7371     0.000000     0.000000      0.000000     0.00000     0.00000    run   
7372     0.000000     0.000000      0.000000     0.00000     0.00000    run   
7373     0.000000     0.000000      0.000000     0.00000     0.00000    run   
7374     0.000000     0.000000      0.000000     0.00000     0.00000    run   
7375     8.565508     0.000000      0.000000     0.00000     0.00000    run   

      peak_to_peak_acc_x  peak_to_peak_acc_y  peak_to_peak_acc_z  \
7360            0.000793            0.002379            0.002684   
7361            0.001342            0.001952            0.005307   
7362            0.001342            0.001952            0.005307   
7363            0.002440            0.002440            0.005307   
7364            0.002257            0.002440            0.002623   
7365                 NaN                 NaN                 NaN   
7366                 NaN                 NaN                 NaN   
7367                 NaN                 NaN                 NaN   
7368                 NaN                 NaN                 NaN   
7369            0.010187            0.020862            0.018971   
7370            0.006832            0.033733            0.021960   
7371            0.006832            0.033733            0.028487   
7372            0.010370            0.025254            0.028487   
7373            0.019032            0.017568            0.027023   
7374            0.024705            0.017568            0.013969   
7375            0.024705            0.012322            0.013969   

      peak_to_peak_gyro_x  peak_to_peak_gyro_y  peak_to_peak_gyro_z  
7360              0.12250             0.166250              0.12250  
7361              0.09625             0.183750              0.15750  
7362              0.12250             0.175000              0.13125  
7363              0.14000             0.131250              0.13125  
7364              0.14000             0.131250              0.05250  
```

```
7365          NaN          NaN          NaN
7366          NaN          NaN          NaN
7367          NaN          NaN          NaN
7368          NaN          NaN          NaN
7369      2.38000     3.106249      0.17500
7370      1.89875     3.263749      0.21875
7371      2.12625     3.263749      0.13125
7372      2.02125     2.511250      0.07875
7373      1.11125     2.572500      0.10500
7374      1.19875     2.913751      0.20125
7375      1.58375     2.493751      0.29750
```

In [24]:

```
# run için bitiş indexi kontrol
print(yeni_data.loc[14744:14755])
```

```
          gyro_x     gyro_y     gyro_z   accel_x   accel_y   accel_z  \
14744 -18.830000 -36.216251  30.021250  0.013664  0.025925  0.024217   
14745 -19.565001 -38.928749  29.671249  0.023485 -0.015921  0.017934   
14746 -18.576250 -41.133751  29.793751  0.030256 -0.037820  0.009821   
14747 -14.603750 -41.623749  30.161249  0.035197 -0.077043  0.004331   
14748 -11.112500 -41.790001  30.458750  0.040626 -0.113338  0.002989   
14749  -3.062500 -40.197498  30.721251  0.052399 -0.128771  0.010431   
14750   1.356250 -38.972500  31.465000  0.054839 -0.153110  0.028548   
14751  -1.426250  28.507500  -0.595000  0.442311 -0.383751  0.459452   
14752  -2.231250  27.282499  -0.840000  0.456768 -0.429623  0.508801   
14753  -3.648750  24.605000  -1.076250  0.491111 -0.485011  0.525820   
14754  -4.331250  23.415001  -1.513750  0.513498 -0.512034  0.564006   
14755  -4.112500  20.396250  -1.802500  0.554124 -0.564921  0.603717   

       std_acc_30  std_gyro_10  mean_acc_20  mean_gyro_20  max_acc_15  \
14744    0.196126    28.958456     0.086995     -1.705375    0.199653   
14745    0.175374    28.623732     0.083136     -1.737167    0.199653   
14746    0.155563    28.408533     0.078463     -1.977208    0.199653   
14747    0.138266    28.263083     0.072397     -2.285938    0.199653   
14748    0.124906    28.196623     0.065485     -2.678958    0.199653   
14749    0.113570    28.266397     0.058942     -2.956771    0.199653   
14750    0.105013    28.491808     0.052213     -3.144604    0.199653   
14751    0.000000     0.000000     0.000000      0.000000    0.000000   
14752    0.000000     0.000000     0.000000      0.000000    0.000000   
14753    0.000000     0.000000     0.000000      0.000000    0.000000   
14754    0.000000     0.000000     0.000000      0.000000    0.000000   
14755    0.000000     0.000000     0.000000      0.000000    0.000000   

       min_acc_20 Output  peak_to_peak_acc_x  peak_to_peak_acc_y  \
14744   -0.086010    run            0.013786            0.136457   
14745   -0.036417    run            0.011956            0.160308   
14746   -0.037820    run            0.018727            0.132370   
14747   -0.077043    run            0.022814            0.144997   
14748   -0.113338    run            0.026962            0.139263   
14749   -0.128771    run            0.028914            0.112850   
14750   -0.153110    run            0.024583            0.115290   
14751    0.000000   jump                 NaN                 NaN   
14752    0.000000   jump                 NaN                 NaN   
14753    0.000000   jump                 NaN                 NaN   
14754    0.000000   jump                 NaN                 NaN   
14755    0.000000   jump                 NaN                 NaN   

       peak_to_peak_acc_z  peak_to_peak_gyro_x  peak_to_peak_gyro_y  \
14744            0.015372            15.828750             3.780002   
14745            0.021655            11.628751             6.492500   
14746            0.025071             9.056251             8.400002   
14747            0.028975             4.961251             6.798748   
14748            0.021228             8.452501             5.573750   
14749            0.014945            16.502501             2.861252   
14750            0.025559            19.932500             2.817501   
14751                 NaN                  NaN                  NaN   
14752                 NaN                  NaN                  NaN   
14753                 NaN                  NaN                  NaN   
14754                 NaN                  NaN                  NaN   
14755                 NaN                  NaN                  NaN   

       peak_to_peak_gyro_z  
14744             1.723751  
14745             1.513750  
14746             1.023751  
```

```
14747           0.778752
14748           0.787501
14749           1.050002
14750           1.671249
14751                NaN
14752                NaN
14753                NaN
14754                NaN
14755                NaN
```

In [25]:

```
# jump value: start index : 14751, end index index: 22005

window_size = 5

for i in range(14751, 22002):
    accel_x_window = accel_x[i:i+window_size]
    accel_y_window = accel_y[i:i+window_size]
    accel_z_window = accel_z[i:i+window_size]

    gyro_x_window = gyro_x[i:i+window_size]
    gyro_y_window = gyro_y[i:i+window_size]
    gyro_z_window = gyro_z[i:i+window_size]

    peak_to_peak_accel_x = accel_x_window.max() - accel_x_window.min()
    peak_to_peak_accel_y = accel_y_window.max() - accel_y_window.min()
    peak_to_peak_accel_z = accel_z_window.max() - accel_z_window.min()

    peak_to_peak_gyro_x = gyro_x_window.max() - gyro_x_window.min()
    peak_to_peak_gyro_y = gyro_y_window.max() - gyro_y_window.min()
    peak_to_peak_gyro_z = gyro_z_window.max() - gyro_z_window.min()

    yeni_data.at[i + window_size - 1, 'peak_to_peak_acc_x'] = peak_to_peak_accel
    yeni_data.at[i + window_size - 1, 'peak_to_peak_acc_y'] = peak_to_peak_accel
    yeni_data.at[i + window_size - 1, 'peak_to_peak_acc_z'] = peak_to_peak_accel

    yeni_data.at[i + window_size - 1, 'peak_to_peak_gyro_x'] = peak_to_peak_gyrc
    yeni_data.at[i + window_size - 1, 'peak_to_peak_gyro_y'] = peak_to_peak_gyrc
    yeni_data.at[i + window_size - 1, 'peak_to_peak_gyro_z'] = peak_to_peak_gyrc
```

In [26]:

```
# jump için başlangıç indexi kontrol

print(yeni_data.loc[14745:14757])
```

```
          gyro_x     gyro_y     gyro_z   accel_x   accel_y   accel_z  \
14745 -19.565001 -38.928749  29.671249  0.023485 -0.015921  0.017934
14746 -18.576250 -41.133751  29.793751  0.030256 -0.037820  0.009821
14747 -14.603750 -41.623749  30.161249  0.035197 -0.077043  0.004331
14748 -11.112500 -41.790001  30.458750  0.040626 -0.113338  0.002989
14749  -3.062500 -40.197498  30.721251  0.052399 -0.128771  0.010431
14750   1.356250 -38.972500  31.465000  0.054839 -0.153110  0.028548
14751  -1.426250  28.507500  -0.595000  0.442311 -0.383751  0.459452
14752  -2.231250  27.282499  -0.840000  0.456768 -0.429623  0.508801
14753  -3.648750  24.605000  -1.076250  0.491111 -0.485011  0.525820
14754  -4.331250  23.415001  -1.513750  0.513498 -0.512034  0.564006
14755  -4.112500  20.396250  -1.802500  0.554124 -0.564921  0.603717
14756  -3.246250  18.514999  -1.925000  0.597129 -0.618357  0.627507
14757  -1.435000  17.990000  -2.126250  0.619699 -0.632326  0.683200

       std_acc_30  std_gyro_10  mean_acc_20  mean_gyro_20  max_acc_15  \
14745    0.175374    28.623732     0.083136     -1.737167    0.199653
14746    0.155563    28.408533     0.078463     -1.977208    0.199653
14747    0.138266    28.263083     0.072397     -2.285938    0.199653
14748    0.124906    28.196623     0.065485     -2.678958    0.199653
14749    0.113570    28.266397     0.058942     -2.956771    0.199653
14750    0.105013    28.491808     0.052213     -3.144604    0.199653
14751    0.000000     0.000000     0.000000      0.000000    0.000000
14752    0.000000     0.000000     0.000000      0.000000    0.000000
14753    0.000000     0.000000     0.000000      0.000000    0.000000
14754    0.000000     0.000000     0.000000      0.000000    0.000000
14755    0.000000     0.000000     0.000000      0.000000    0.000000
14756    0.000000     0.000000     0.000000      0.000000    0.000000
14757    0.000000     0.000000     0.000000      0.000000    0.000000

       min_acc_20 Output  peak_to_peak_acc_x  peak_to_peak_acc_y  \
14745   -0.036417    run            0.011956            0.160308
14746   -0.037820    run            0.018727            0.132370
14747   -0.077043    run            0.022814            0.144997
14748   -0.113338    run            0.026962            0.139263
14749   -0.128771    run            0.028914            0.112850
14750   -0.153110    run            0.024583            0.115290
14751    0.000000   jump                 NaN                 NaN
14752    0.000000   jump                 NaN                 NaN
14753    0.000000   jump                 NaN                 NaN
14754    0.000000   jump                 NaN                 NaN
14755    0.000000   jump            0.111813            0.181170
14756    0.000000   jump            0.140361            0.188734
14757    0.000000   jump            0.128588            0.147315

       peak_to_peak_acc_z  peak_to_peak_gyro_x  peak_to_peak_gyro_y  \
14745            0.021655            11.628751             6.492500
14746            0.025071             9.056251             8.400002
14747            0.028975             4.961251             6.798748
14748            0.021228             8.452501             5.573750
14749            0.014945            16.502501             2.861252
14750            0.025559            19.932500             2.817501
14751                 NaN                  NaN                  NaN
14752                 NaN                  NaN                  NaN
14753                 NaN                  NaN                  NaN
14754                 NaN                  NaN                  NaN
14755            0.144265             2.905000             8.111250
14756            0.118706             2.100000             8.767500
14757            0.157380             2.896250             6.615000
```

```
       peak_to_peak_gyro_z
14745             1.513750
14746             1.023751
14747             0.778752
14748             0.787501
14749             1.050002
14750             1.671249
14751                  NaN
14752                  NaN
14753                  NaN
14754                  NaN
14755             1.207500
14756             1.085000
14757             1.050000
```

In [27]:

```
# jump için bitiş indeksi kontrol

print(yeni_data.loc[22000:22010])
```

```
          gyro_x     gyro_y   gyro_z   accel_x   accel_y   accel_z  std_acc_30  \
22000   7.86625  17.876249  2.53750 -0.014762  0.043615  0.084607    0.078505   
22001   6.90375  17.254999  2.43250 -0.014823  0.038491  0.098088    0.076660   
22002   6.99125  17.228750  2.17875 -0.019093  0.031293  0.098088    0.074954   
22003   8.62750  18.313749  2.07375 -0.022997  0.025132  0.097722    0.073008   
22004  11.99625  19.101250  1.93375 -0.027328  0.020252  0.099735    0.071131   
22005  13.87750  20.571251  1.47875 -0.029341  0.020374  0.097844    0.069335   
22006   8.29000   8.660000  2.42000  0.630000 -0.620000  0.510000    0.000000   
22007   8.51000   9.000000  2.39000  0.630000 -0.620000  0.510000    0.000000   
22008   9.00000  11.870000  2.46000  0.620000 -0.630000  0.510000    0.000000   
22009   9.06000  11.780000  2.59000  0.620000 -0.630000  0.510000    0.000000   
22010   9.05000  11.740000  2.50000  0.620000 -0.630000  0.510000    0.000000   

       std_gyro_10  mean_acc_20  mean_gyro_20  max_acc_15  min_acc_20  \
22000    11.378033     0.014221     14.221521    0.119499   -0.150731   
22001    10.484486     0.018067     14.527042    0.116144   -0.150731   
22002     9.638130     0.021556     14.639479    0.098088   -0.150731   
22003     8.870547     0.024447     14.721437    0.098088   -0.143777   
22004     8.074017     0.026922     14.744625    0.099735   -0.128832   
22005     7.567599     0.028656     14.656104    0.099735   -0.104127   
22006     0.000000     0.000000      0.000000    0.000000    0.000000   
22007     0.000000     0.000000      0.000000    0.000000    0.000000   
22008     0.000000     0.000000      0.000000    0.000000    0.000000   
22009     0.000000     0.000000      0.000000    0.000000    0.000000   
22010     0.000000     0.000000      0.000000    0.000000    0.000000   

        Output  peak_to_peak_acc_x  peak_to_peak_acc_y  peak_to_peak_acc_z  \
22000     jump            0.027816            0.011834            0.059780   
22001     jump            0.016653            0.016958            0.052460   
22002     jump            0.010004            0.024156            0.041236   
22003     jump            0.011834            0.025193            0.023119   
22004     jump            0.012566            0.023363            0.015128   
22005     jump            0.014518            0.018239            0.002013   
22006  bending                 NaN                 NaN                 NaN   
22007  bending                 NaN                 NaN                 NaN   
22008  bending                 NaN                 NaN                 NaN   
22009  bending                 NaN                 NaN                 NaN   
22010  bending                 NaN                 NaN                 NaN   

       peak_to_peak_gyro_x  peak_to_peak_gyro_y  peak_to_peak_gyro_z  
22000            13.177501             5.512501              1.74125  
22001            11.051250             5.853750              1.33000  
22002             5.687500             3.885000              1.27750  
22003             2.126250             2.546250              0.91000  
22004             5.092500             1.872500              0.60375  
22005             6.973750             3.342501              0.95375  
22006                  NaN                  NaN                  NaN  
22007                  NaN                  NaN                  NaN  
22008                  NaN                  NaN                  NaN  
22009                  NaN                  NaN                  NaN  
22010                  NaN                  NaN                  NaN  
```

In [28]:

```
# bending value: start index : 22006, end index index: 29035

window_size = 5

for i in range(22006, 29032):
    accel_x_window = accel_x[i:i+window_size]
    accel_y_window = accel_y[i:i+window_size]
    accel_z_window = accel_z[i:i+window_size]
    gyro_x_window = gyro_x[i:i+window_size]
    gyro_y_window = gyro_y[i:i+window_size]
    gyro_z_window = gyro_z[i:i+window_size]

    peak_to_peak_accel_x = accel_x_window.max() - accel_x_window.min()
    peak_to_peak_accel_y = accel_y_window.max() - accel_y_window.min()
    peak_to_peak_accel_z = accel_z_window.max() - accel_z_window.min()

    peak_to_peak_gyro_x = gyro_x_window.max() - gyro_x_window.min()
    peak_to_peak_gyro_y = gyro_y_window.max() - gyro_y_window.min()
    peak_to_peak_gyro_z = gyro_z_window.max() - gyro_z_window.min()

    yeni_data.at[i + window_size - 1, 'peak_to_peak_acc_x'] = peak_to_peak_accel
    yeni_data.at[i + window_size - 1, 'peak_to_peak_acc_y'] = peak_to_peak_accel
    yeni_data.at[i + window_size - 1, 'peak_to_peak_acc_z'] = peak_to_peak_accel

    yeni_data.at[i + window_size - 1, 'peak_to_peak_gyro_x'] = peak_to_peak_gyro
    yeni_data.at[i + window_size - 1, 'peak_to_peak_gyro_y'] = peak_to_peak_gyro
    yeni_data.at[i + window_size - 1, 'peak_to_peak_gyro_z'] = peak_to_peak_gyro
```

In [29]:

```
# bending için başlangıç değeri kontrol
print(yeni_data.loc[22000:22015])
```

```
         gyro_x     gyro_y   gyro_z   accel_x   accel_y   accel_z  std_acc_30  \
22000   7.86625  17.876249  2.53750 -0.014762  0.043615  0.084607    0.078505
22001   6.90375  17.254999  2.43250 -0.014823  0.038491  0.098088    0.076660
22002   6.99125  17.228750  2.17875 -0.019093  0.031293  0.098088    0.074954
22003   8.62750  18.313749  2.07375 -0.022997  0.025132  0.097722    0.073008
22004  11.99625  19.101250  1.93375 -0.027328  0.020252  0.099735    0.071131
22005  13.87750  20.571251  1.47875 -0.029341  0.020374  0.097844    0.069335
22006   8.29000   8.660000  2.42000  0.630000 -0.620000  0.510000    0.000000
22007   8.51000   9.000000  2.39000  0.630000 -0.620000  0.510000    0.000000
22008   9.00000  11.870000  2.46000  0.620000 -0.630000  0.510000    0.000000
22009   9.06000  11.780000  2.59000  0.620000 -0.630000  0.510000    0.000000
22010   9.05000  11.740000  2.50000  0.620000 -0.630000  0.510000    0.000000
22011   9.43000  11.520000  2.61000  0.620000 -0.630000  0.510000    0.000000
22012   9.77000  11.600000  2.44000  0.620000 -0.640000  0.510000    0.000000
22013  10.05000  11.700000  2.43000  0.620000 -0.630000  0.510000    0.000000
22014  10.47000  12.250000  2.30000  0.620000 -0.630000  0.510000    0.000000
22015  10.98000  12.710000  2.32000  0.610000 -0.630000  0.510000    0.000000

       std_gyro_10  mean_acc_20  mean_gyro_20  max_acc_15  min_acc_20  \
22000    11.378033     0.014221     14.221521    0.119499   -0.150731
22001    10.484486     0.018067     14.527042    0.116144   -0.150731
22002     9.638130     0.021556     14.639479    0.098088   -0.150731
22003     8.870547     0.024447     14.721437    0.098088   -0.143777
22004     8.074017     0.026922     14.744625    0.099735   -0.128832
22005     7.567599     0.028656     14.656104    0.099735   -0.104127
22006     0.000000     0.000000      0.000000    0.000000    0.000000
22007     0.000000     0.000000      0.000000    0.000000    0.000000
22008     0.000000     0.000000      0.000000    0.000000    0.000000
22009     0.000000     0.000000      0.000000    0.000000    0.000000
22010     0.000000     0.000000      0.000000    0.000000    0.000000
22011     0.000000     0.000000      0.000000    0.000000    0.000000
22012     0.000000     0.000000      0.000000    0.000000    0.000000
22013     0.000000     0.000000      0.000000    0.000000    0.000000
22014     0.000000     0.000000      0.000000    0.000000    0.000000
22015     0.000000     0.000000      0.000000    0.000000    0.000000

        Output  peak_to_peak_acc_x  peak_to_peak_acc_y  peak_to_peak_acc_z  \
22000     jump            0.027816            0.011834            0.059780
22001     jump            0.016653            0.016958            0.052460
22002     jump            0.010004            0.024156            0.041236
22003     jump            0.011834            0.025193            0.023119
22004     jump            0.012566            0.023363            0.015128
22005     jump            0.014518            0.018239            0.002013
22006  bending                 NaN                 NaN                 NaN
22007  bending                 NaN                 NaN                 NaN
22008  bending                 NaN                 NaN                 NaN
22009  bending                 NaN                 NaN                 NaN
22010  bending            0.010000            0.010000            0.000000
22011  bending            0.010000            0.010000            0.000000
22012  bending            0.000000            0.010000            0.000000
22013  bending            0.000000            0.010000            0.000000
22014  bending            0.000000            0.010000            0.000000
22015  bending            0.010000            0.010000            0.000000

       peak_to_peak_gyro_x  peak_to_peak_gyro_y  peak_to_peak_gyro_z
22000            13.177501             5.512501              1.74125
22001            11.051250             5.853750              1.33000
22002             5.687500             3.885000              1.27750
22003             2.126250             2.546250              0.91000
22004             5.092500             1.872500              0.60375
```

```
22005           6.973750          3.342501           0.95375
22006                NaN               NaN               NaN
22007                NaN               NaN               NaN
22008                NaN               NaN               NaN
22009                NaN               NaN               NaN
22010           0.770000          3.210000           0.20000
22011           0.920000          2.870000           0.22000
22012           0.770000          0.350000           0.17000
22013           1.000000          0.260000           0.18000
22014           1.420000          0.730000           0.31000
22015           1.550000          1.190000           0.31000
```

In [30]:

```
# bending için bitiş indexi kontrol

print(yeni_data.loc[29030:29040])
```

```
         gyro_x    gyro_y    gyro_z   accel_x   accel_y   accel_z  std_acc_30  \
29030 -40.90000 -49.55000  -8.16000 -0.550000  0.410000  0.880000         0.6   
29031 -42.28000 -50.65000  -8.55000 -0.540000  0.400000  0.880000         0.6   
29032 -42.88000 -51.26000  -8.78000 -0.540000  0.400000  0.870000         0.6   
29033 -43.17000 -52.03000  -9.09000 -0.530000  0.390000  0.880000         0.6   
29034 -43.56000 -52.00000  -9.00000 -0.520000  0.390000  0.880000         0.6   
29035 -43.51000 -51.90000  -9.12000 -0.520000  0.390000  0.880000         0.6   
29036   2.97500  -2.65125   9.73000  0.920490 -0.153659  0.525149         0.0   
29037   3.31625  -2.36250  10.34250  0.923174 -0.151158  0.526918         0.0   
29038   3.37750  -2.07375  11.68125  0.922198 -0.146095  0.529785         0.0   
29039   3.57000  -1.27750  12.99375  0.930982 -0.143655  0.532164         0.0   
29040   3.13250  -0.53375  13.61500  0.934825 -0.141703  0.540948         0.0   

       std_gyro_10  mean_acc_20  mean_gyro_20  max_acc_15  min_acc_20  \
29030        16.76         0.24        -29.17        0.88       -0.58   
29031        17.01         0.24        -29.53        0.88       -0.58   
29032        17.27         0.24        -29.89        0.88       -0.58   
29033        17.52         0.24        -30.23        0.88       -0.58   
29034        17.75         0.24        -30.58        0.88       -0.58   
29035        17.94         0.24        -30.92        0.88       -0.58   
29036         0.00         0.00          0.00        0.00        0.00   
29037         0.00         0.00          0.00        0.00        0.00   
29038         0.00         0.00          0.00        0.00        0.00   
29039         0.00         0.00          0.00        0.00        0.00   
29040         0.00         0.00          0.00        0.00        0.00   

         Output  peak_to_peak_acc_x  peak_to_peak_acc_y  peak_to_peak_acc_z  \
29030   bending                0.02                0.00                0.01   
29031   bending                0.02                0.01                0.00   
29032   bending                0.02                0.01                0.01   
29033   bending                0.02                0.02                0.01   
29034   bending                0.03                0.02                0.01   
29035   bending                0.02                0.01                0.01   
29036  forehand                 NaN                 NaN                 NaN   
29037  forehand                 NaN                 NaN                 NaN   
29038  forehand                 NaN                 NaN                 NaN   
29039  forehand                 NaN                 NaN                 NaN   
29040  forehand                 NaN                 NaN                 NaN   

       peak_to_peak_gyro_x  peak_to_peak_gyro_y  peak_to_peak_gyro_z  
29030                 3.79                 3.26                 1.15  
29031                 4.77                 3.83                 1.16  
29032                 4.17                 3.48                 1.09  
29033                 2.98                 3.19                 1.24  
29034                 2.66                 2.48                 0.93  
29035                 1.28                 1.38                 0.57  
29036                  NaN                  NaN                  NaN  
29037                  NaN                  NaN                  NaN  
29038                  NaN                  NaN                  NaN  
29039                  NaN                  NaN                  NaN  
29040                  NaN                  NaN                  NaN  
```

In [31]:

```python
# forehand value: start index : 29036, end index index: 37160

window_size = 5

for i in range(29036, 37157):
    accel_x_window = accel_x[i:i+window_size]
    accel_y_window = accel_y[i:i+window_size]
    accel_z_window = accel_z[i:i+window_size]
    gyro_x_window = gyro_x[i:i+window_size]
    gyro_y_window = gyro_y[i:i+window_size]
    gyro_z_window = gyro_z[i:i+window_size]

    peak_to_peak_accel_x = accel_x_window.max() - accel_x_window.min()
    peak_to_peak_accel_y = accel_y_window.max() - accel_y_window.min()
    peak_to_peak_accel_z = accel_z_window.max() - accel_z_window.min()

    peak_to_peak_gyro_x = gyro_x_window.max() - gyro_x_window.min()
    peak_to_peak_gyro_y = gyro_y_window.max() - gyro_y_window.min()
    peak_to_peak_gyro_z = gyro_z_window.max() - gyro_z_window.min()

    yeni_data.at[i + window_size - 1, 'peak_to_peak_acc_x'] = peak_to_peak_accel
    yeni_data.at[i + window_size - 1, 'peak_to_peak_acc_y'] = peak_to_peak_accel
    yeni_data.at[i + window_size - 1, 'peak_to_peak_acc_z'] = peak_to_peak_accel

    yeni_data.at[i + window_size - 1, 'peak_to_peak_gyro_x'] = peak_to_peak_gyrc
    yeni_data.at[i + window_size - 1, 'peak_to_peak_gyro_y'] = peak_to_peak_gyrc
    yeni_data.at[i + window_size - 1, 'peak_to_peak_gyro_z'] = peak_to_peak_gyrc
```

In [32]:

```
# forehand için başlangıç indexi kontrol

print(yeni_data.loc[29034:29042])
```

```
          gyro_x    gyro_y     gyro_z   accel_x   accel_y   accel_z  \
29034 -43.56000 -52.00000  -9.000000 -0.520000  0.390000  0.880000
29035 -43.51000 -51.90000  -9.120000 -0.520000  0.390000  0.880000
29036   2.97500  -2.65125   9.730000  0.920490 -0.153659  0.525149
29037   3.31625  -2.36250  10.342500  0.923174 -0.151158  0.526918
29038   3.37750  -2.07375  11.681250  0.922198 -0.146095  0.529785
29039   3.57000  -1.27750  12.993750  0.930982 -0.143655  0.532164
29040   3.13250  -0.53375  13.615000  0.934825 -0.141703  0.540948
29041   2.59000   0.63875  15.365000  0.939217 -0.139995  0.551562
29042   1.82875   1.52250  16.196251  0.944097 -0.136396  0.554368

        std_acc_30  std_gyro_10  mean_acc_20  mean_gyro_20  max_acc_15  \
29034          0.6        17.75         0.24        -30.58        0.88
29035          0.6        17.94         0.24        -30.92        0.88
29036          0.0         0.00         0.00          0.00        0.00
29037          0.0         0.00         0.00          0.00        0.00
29038          0.0         0.00         0.00          0.00        0.00
29039          0.0         0.00         0.00          0.00        0.00
29040          0.0         0.00         0.00          0.00        0.00
29041          0.0         0.00         0.00          0.00        0.00
29042          0.0         0.00         0.00          0.00        0.00

        min_acc_20    Output  peak_to_peak_acc_x  peak_to_peak_acc_y  \
29034        -0.58   bending            0.030000            0.020000
29035        -0.58   bending            0.020000            0.010000
29036         0.00  forehand                 NaN                 NaN
29037         0.00  forehand                 NaN                 NaN
29038         0.00  forehand                 NaN                 NaN
29039         0.00  forehand                 NaN                 NaN
29040         0.00  forehand            0.014335            0.011956
29041         0.00  forehand            0.017019            0.011163
29042         0.00  forehand            0.021899            0.009699

        peak_to_peak_acc_z  peak_to_peak_gyro_x  peak_to_peak_gyro_y  \
29034             0.010000              2.66000              2.48000
29035             0.010000              1.28000              1.38000
29036                  NaN                  NaN                  NaN
29037                  NaN                  NaN                  NaN
29038                  NaN                  NaN                  NaN
29039                  NaN                  NaN                  NaN
29040             0.015799              0.59500              2.11750
29041             0.024644              0.98000              3.00125
29042             0.024583              1.74125              3.59625

        peak_to_peak_gyro_z
29034             0.930000
29035             0.570000
29036                  NaN
29037                  NaN
29038                  NaN
29039                  NaN
29040             3.885000
29041             5.022500
29042             4.515001
```

In [33]:

```
# windowdan dolayı oluşan Nan değerleri

number_nan = yeni_data.isna().sum()
print(number_nan)
```

```
gyro_x                  0
gyro_y                  0
gyro_z                  0
accel_x                 0
accel_y                 0
accel_z                 0
std_acc_30              0
std_gyro_10             0
mean_acc_20             0
mean_gyro_20            0
max_acc_15              0
min_acc_20              0
Output                  0
peak_to_peak_acc_x     20
peak_to_peak_acc_y     20
peak_to_peak_acc_z     20
peak_to_peak_gyro_x    20
peak_to_peak_gyro_y    20
peak_to_peak_gyro_z    20
dtype: int64
```

In [34]:

```
# 0 ile dolduruyoruz
yeni_data = yeni_data.fillna(0)
```

In [35]:

```
number_nan = yeni_data.isna().sum()
print(number_nan)
```

```
gyro_x                 0
gyro_y                 0
gyro_z                 0
accel_x                0
accel_y                0
accel_z                0
std_acc_30             0
std_gyro_10            0
mean_acc_20            0
mean_gyro_20           0
max_acc_15             0
min_acc_20             0
Output                 0
peak_to_peak_acc_x     0
peak_to_peak_acc_y     0
peak_to_peak_acc_z     0
peak_to_peak_gyro_x    0
peak_to_peak_gyro_y    0
peak_to_peak_gyro_z    0
dtype: int64
```

2.2 sum

In [36]:

```
# ['gyro_x', 'gyro_y', 'gyro_z', 'accel_x', 'accel_y', 'accel_z', 'Gyro_Total',
# sütunlarının eksenlerini toplayarak feature extraction:
```

In [37]:

```
import pandas as pd

son_data = yeni_data.copy()

son_data['Gyro_Total'] = son_data['gyro_x'] + son_data['gyro_y'] + son_data['gyr
```

```
son_data['Accel_Total'] = son_data['accel_x'] + son_data['accel_y'] + son_data['

display(son_data[['gyro_x', 'gyro_y', 'gyro_z', 'accel_x', 'accel_y', 'accel_z',
```

	gyro_x	gyro_y	gyro_z	accel_x	accel_y	accel_z	Gyro_Total	Accel_Total
0	0.49875	-0.64750	0.13125	0.685396	-0.630008	0.383141	-0.01750	0.438529
1	0.47250	-0.72625	0.12250	0.684420	-0.630191	0.383690	-0.13125	0.437919
2	0.39375	-0.63875	0.12250	0.687531	-0.629764	0.383507	-0.12250	0.441274
3	0.35875	-0.65625	0.09625	0.686616	-0.628971	0.384056	-0.20125	0.441701
4	0.29750	-0.60375	0.14000	0.685640	-0.631594	0.382714	-0.16625	0.436760

In [38]:

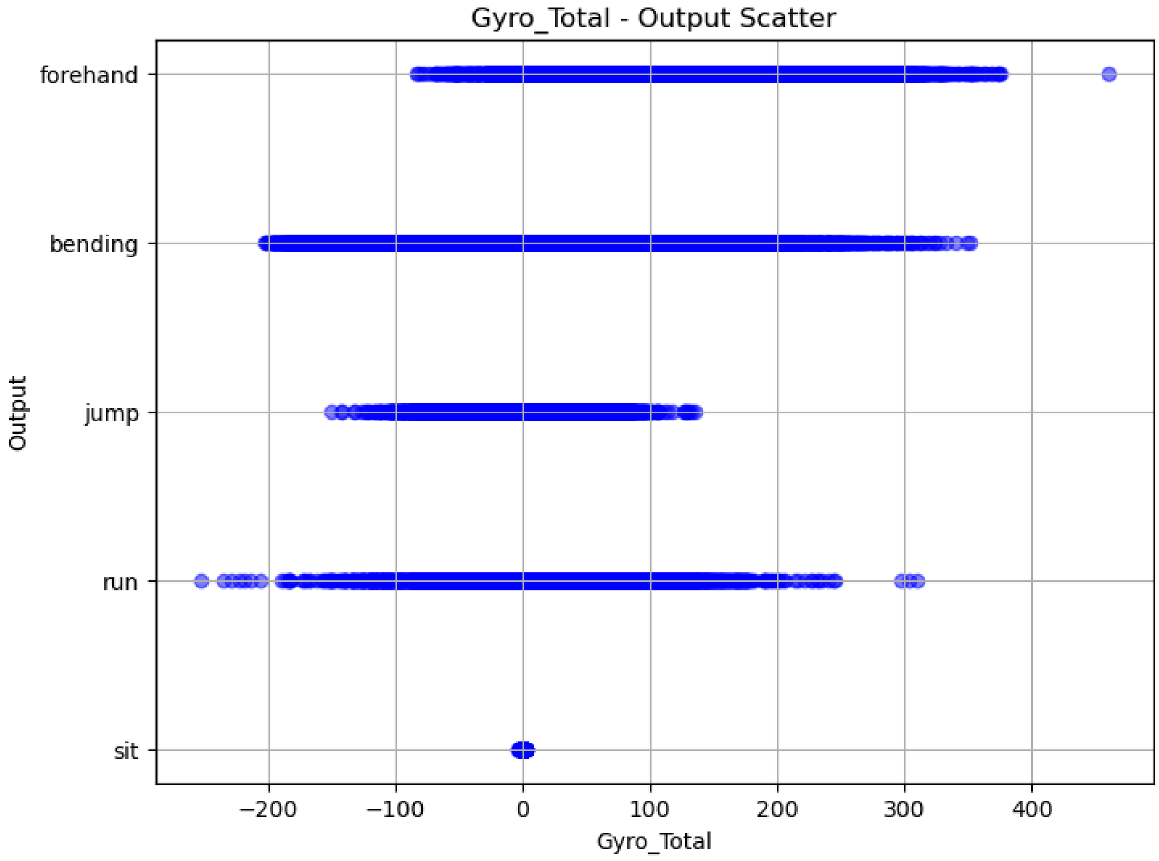
```
# feature extractionlardan sonra verisetinin son hali

print(son_data)
```

```
        gyro_x    gyro_y     gyro_z   accel_x   accel_y   accel_z  std_acc_30  \
0      0.49875  -0.64750   0.131250  0.685396 -0.630008  0.383141    0.000000   
1      0.47250  -0.72625   0.122500  0.684420 -0.630191  0.383690    0.000000   
2      0.39375  -0.63875   0.122500  0.687531 -0.629764  0.383507    0.000000   
3      0.35875  -0.65625   0.096250  0.686616 -0.628971  0.384056    0.000000   
4      0.29750  -0.60375   0.140000  0.685640 -0.631594  0.382714    0.000000   
...        ...       ...        ...       ...       ...       ...         ...   
37156  4.13000  -3.77125 -23.240000  0.301523 -0.399123  1.090009    0.587435   
37157  2.86125  -5.79250 -24.666250  0.324093 -0.418765  1.104710    0.588987   
37158  0.03500  -7.71750 -24.893749  0.349225 -0.429501  1.119045    0.590825   
37159 -1.64500 -12.98500 -24.543751  0.363621 -0.460062  1.125023    0.593140   
37160 -5.88875 -20.52750 -24.071251  0.360937 -0.474336  1.111725    0.595449   

       std_gyro_10  mean_acc_20  mean_gyro_20  ...  min_acc_20    Output  \
0         0.000000     0.000000      0.000000  ...    0.000000       sit   
1         0.000000     0.000000      0.000000  ...    0.000000       sit   
2         0.000000     0.000000      0.000000  ...    0.000000       sit   
3         0.000000     0.000000      0.000000  ...    0.000000       sit   
4         0.000000     0.000000      0.000000  ...    0.000000       sit   
...            ...          ...           ...  ...         ...       ...   
37156     8.546404     0.268468     -1.341083  ...   -0.455304  forehand   
37157     9.246883     0.273875     -2.134125  ...   -0.455304  forehand   
37158     9.796690     0.279718     -3.012333  ...   -0.455304  forehand   
37159    10.241263     0.285138     -3.975125  ...   -0.460062  forehand   
37160    10.648351     0.289882     -5.068000  ...   -0.474336  forehand   

      peak_to_peak_acc_x  peak_to_peak_acc_y  peak_to_peak_acc_z  \
0               0.000000            0.000000            0.000000   
1               0.000000            0.000000            0.000000   
2               0.000000            0.000000            0.000000   
3               0.000000            0.000000            0.000000   
4               0.003111            0.002623            0.001342   
...                  ...                 ...                 ...   
37156           0.041602            0.012810            0.064355   
37157           0.058499            0.027694            0.069174   
37158           0.065270            0.038430            0.070577   
37159           0.075152            0.068259            0.061305   
37160           0.062098            0.075213            0.035014   

       peak_to_peak_gyro_x  peak_to_peak_gyro_y  peak_to_peak_gyro_z  \
0                  0.00000              0.00000             0.000000   
1                  0.00000              0.00000             0.000000   
2                  0.00000              0.00000             0.000000   
3                  0.00000              0.00000             0.000000   
4                  0.20125              0.12250             0.043750   
...                    ...                  ...                  ...   
37156              0.80500              1.51375             6.037501   
37157              1.96000              3.53500             5.958750   
37158              4.78625              5.46000             3.841249   
37159              6.16875              9.52000             3.071249   
37160             10.01875             16.75625             1.653749   

       Gyro_Total  Accel_Total  
0       -0.017500     0.438529  
1       -0.131250     0.437919  
2       -0.122500     0.441274  
3       -0.201250     0.441701  
4       -0.166250     0.436760  
...           ...          ...  
37156  -22.881250     0.992409  
```

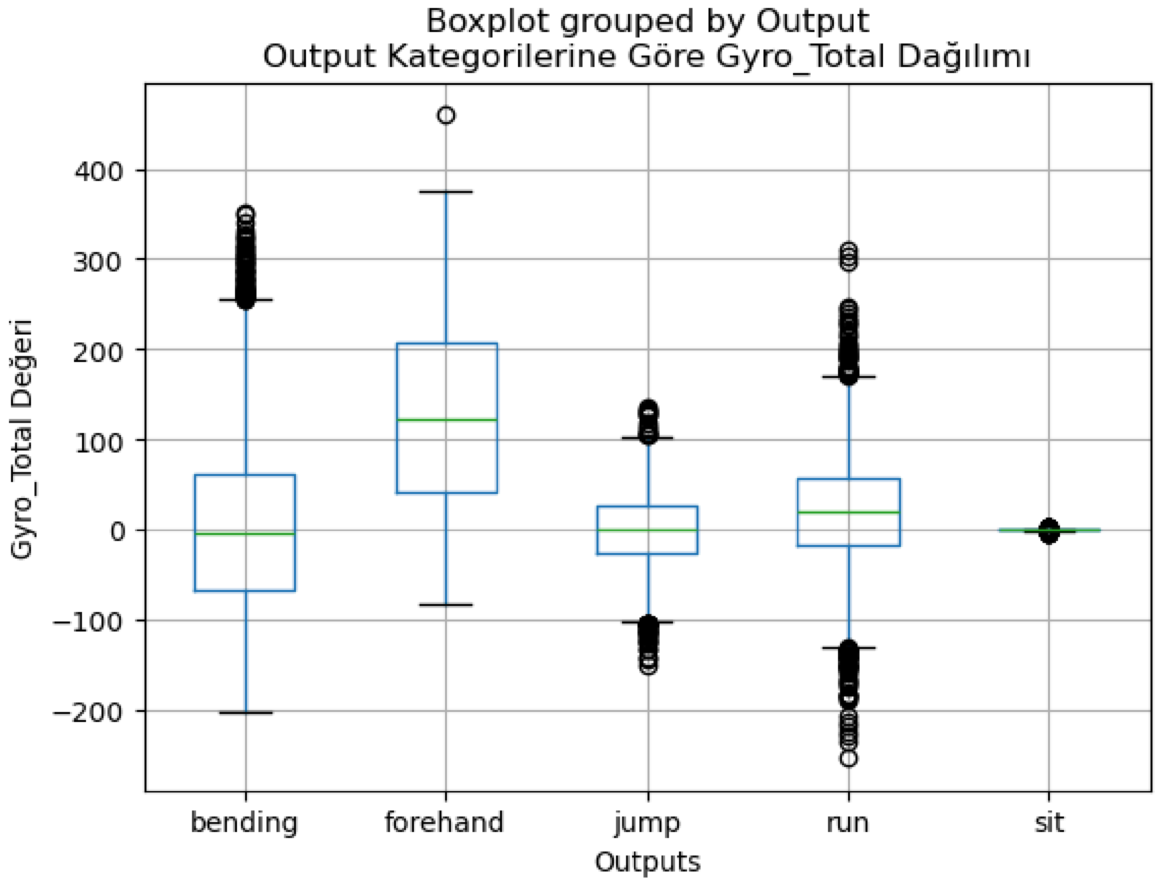
```
37157  -27.597500     1.010038
37158  -32.576249     1.038769
37159  -39.173751     1.028582
37160  -50.487501     0.998326

[37161 rows x 21 columns]
```

In [39]:

```
# Yeni Gyro_Total featureunun outputa göre dağılımı:

import matplotlib.pyplot as plt
import pandas as pd

plt.figure(figsize=(8, 6))
plt.scatter(son_data['Gyro_Total'], son_data['Output'], color='blue', alpha=0.5)
plt.title('Gyro_Total - Output Scatter')
plt.xlabel('Gyro_Total')
plt.ylabel('Output')
plt.grid(True)
plt.show()

# Yeni Gyro_Total featureunun outputa göre dağılımı:

import matplotlib.pyplot as plt
import pandas as pd

plt.figure(figsize=(8, 6))
son_data.boxplot(column='Gyro_Total', by='Output')
plt.title('Output Kategorilerine Göre Gyro_Total Dağılımı')
plt.xlabel('Outputs')
plt.ylabel('Gyro_Total Değeri')
plt.show()
```

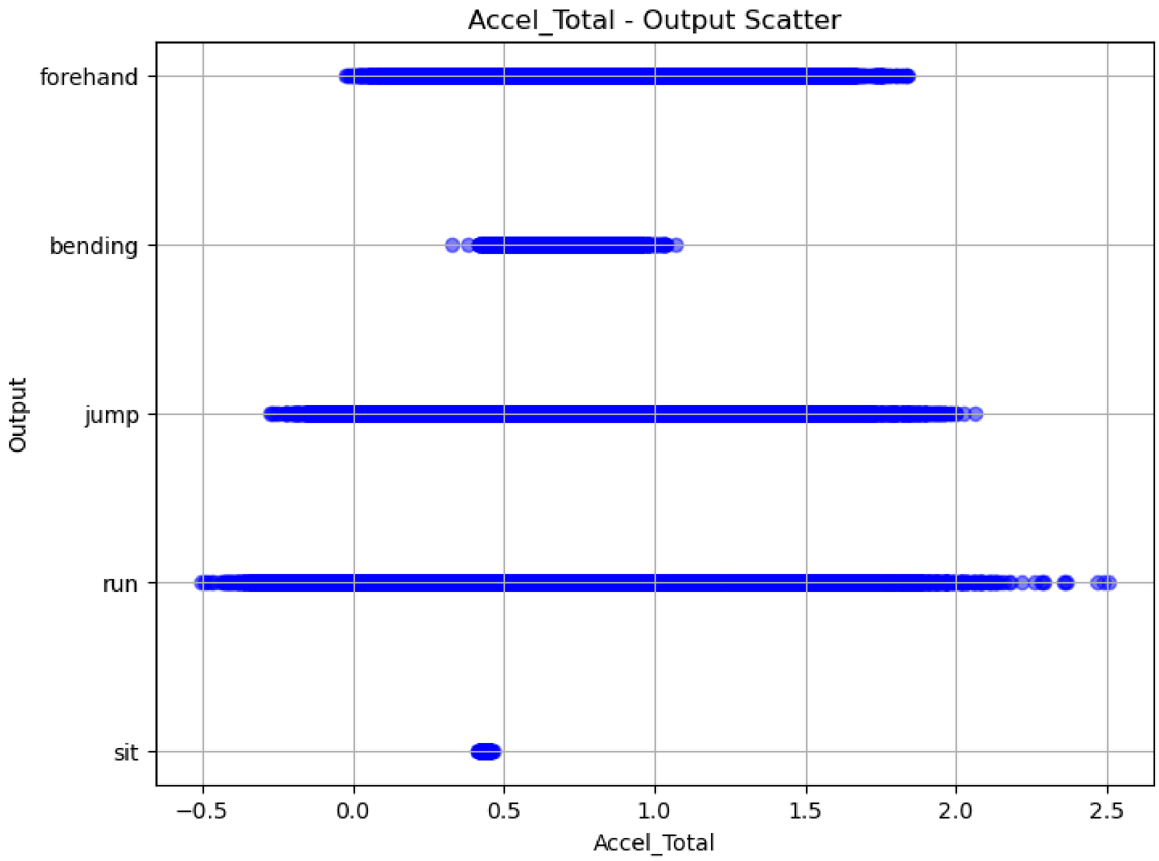


```
<Figure size 800x600 with 0 Axes>
```

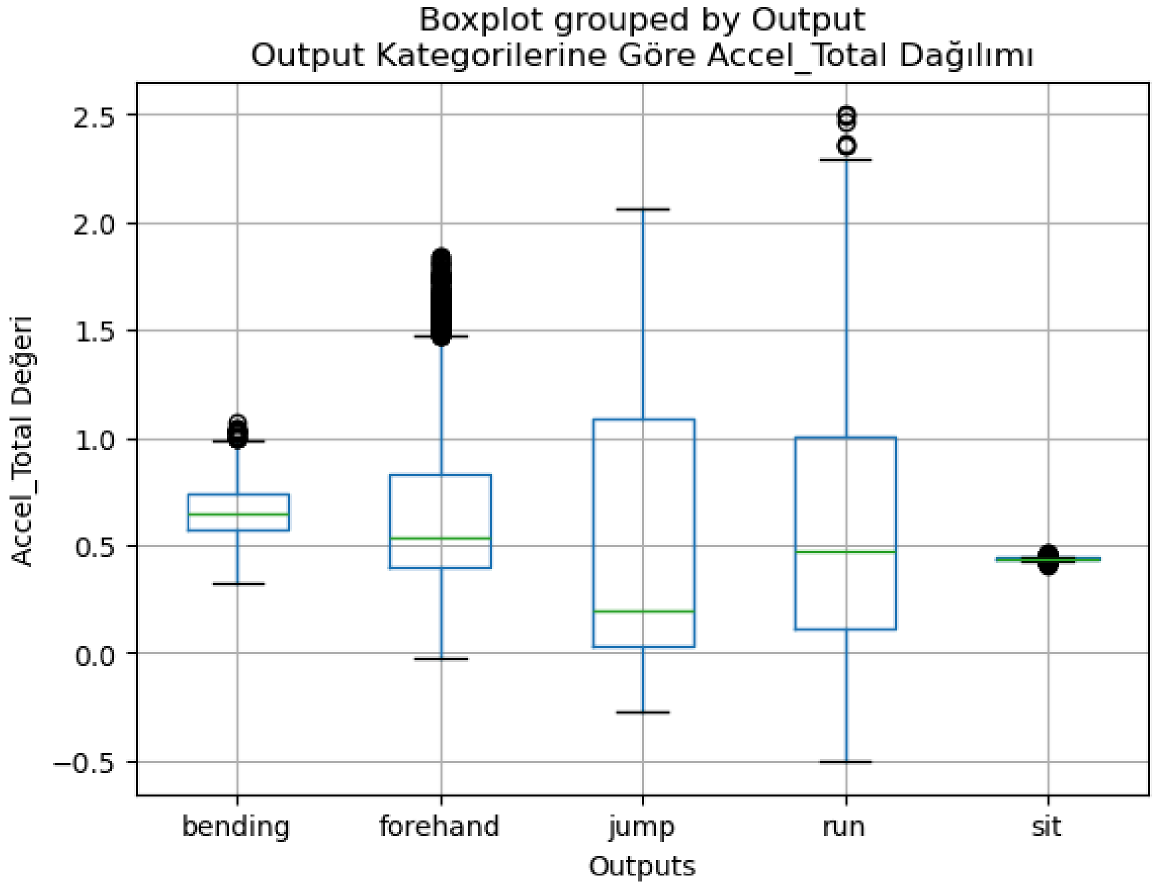


In [40]:

```
# Yeni Accel_Total featureunun outputa göre dağılımı:

import matplotlib.pyplot as plt
import pandas as pd

plt.figure(figsize=(8, 6))
plt.scatter(son_data['Accel_Total'], son_data['Output'], color='blue', alpha=0.5
plt.title('Accel_Total - Output Scatter')
plt.xlabel('Accel_Total')
plt.ylabel('Output')
plt.grid(True)
plt.show()

# Yeni Accel_Total featureunun outputa göre dağılımı:

import matplotlib.pyplot as plt
import pandas as pd

plt.figure(figsize=(8, 6))
son_data.boxplot(column='Accel_Total', by='Output')
plt.title('Output Kategorilerine Göre Accel_Total Dağılımı')
plt.xlabel('Outputs')
plt.ylabel('Accel_Total Değeri')
plt.show()
```

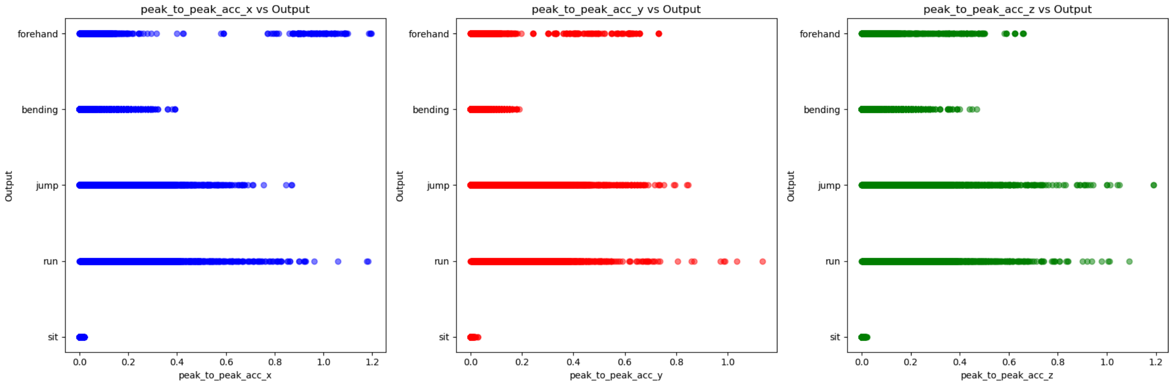


```
<Figure size 800x600 with 0 Axes>
```



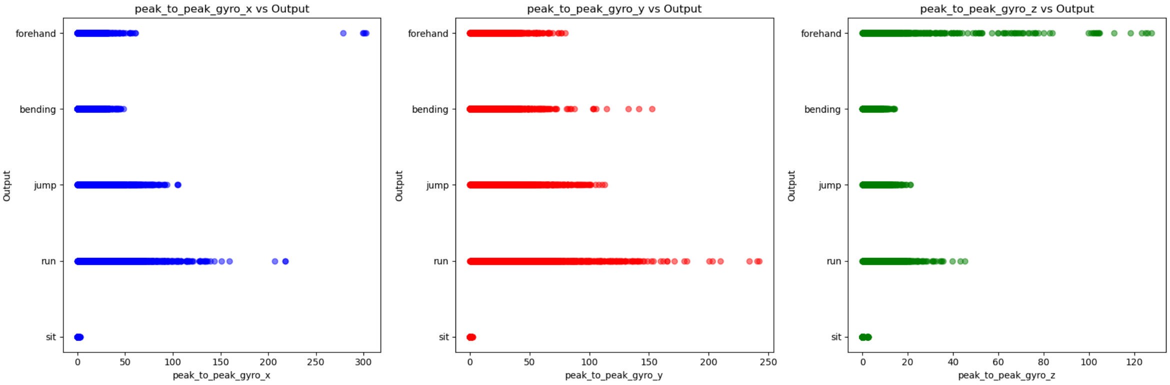
In [41]:

```
# peak_to_peak_acc değerleri ve output
plt.figure(figsize=(18, 6))

# peak_to_peak_acc_x
plt.subplot(1, 3, 1)
plt.scatter(son_data['peak_to_peak_acc_x'], son_data['Output'], color='blue', al
plt.title('peak_to_peak_acc_x vs Output')
plt.xlabel('peak_to_peak_acc_x')
plt.ylabel('Output')

# peak_to_peak_acc_y
plt.subplot(1, 3, 2)
plt.scatter(son_data['peak_to_peak_acc_y'], son_data['Output'], color='red', alp
plt.title('peak_to_peak_acc_y vs Output')
plt.xlabel('peak_to_peak_acc_y')
plt.ylabel('Output')

# peak_to_peak_acc_z
plt.subplot(1, 3, 3)
plt.scatter(son_data['peak_to_peak_acc_z'], son_data['Output'], color='green', a
plt.title('peak_to_peak_acc_z vs Output')
plt.xlabel('peak_to_peak_acc_z')
plt.ylabel('Output')

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



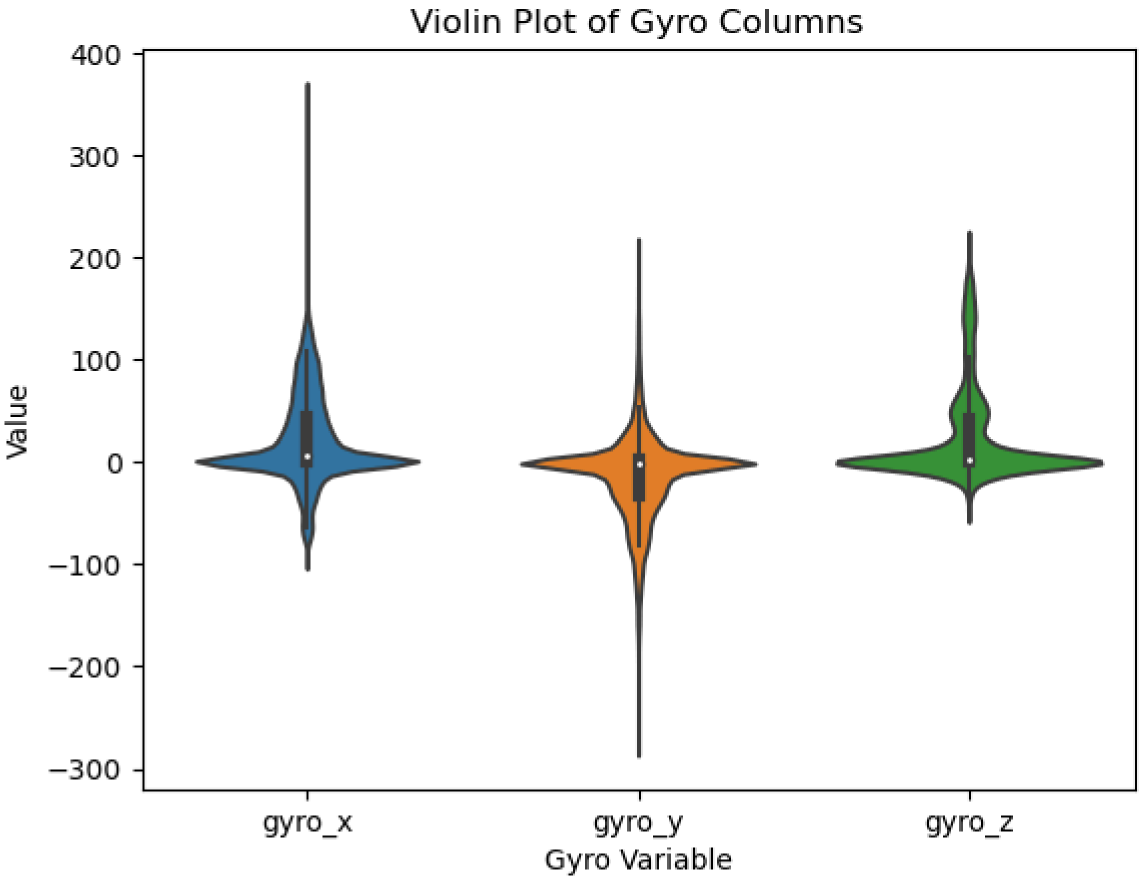
In [42]:

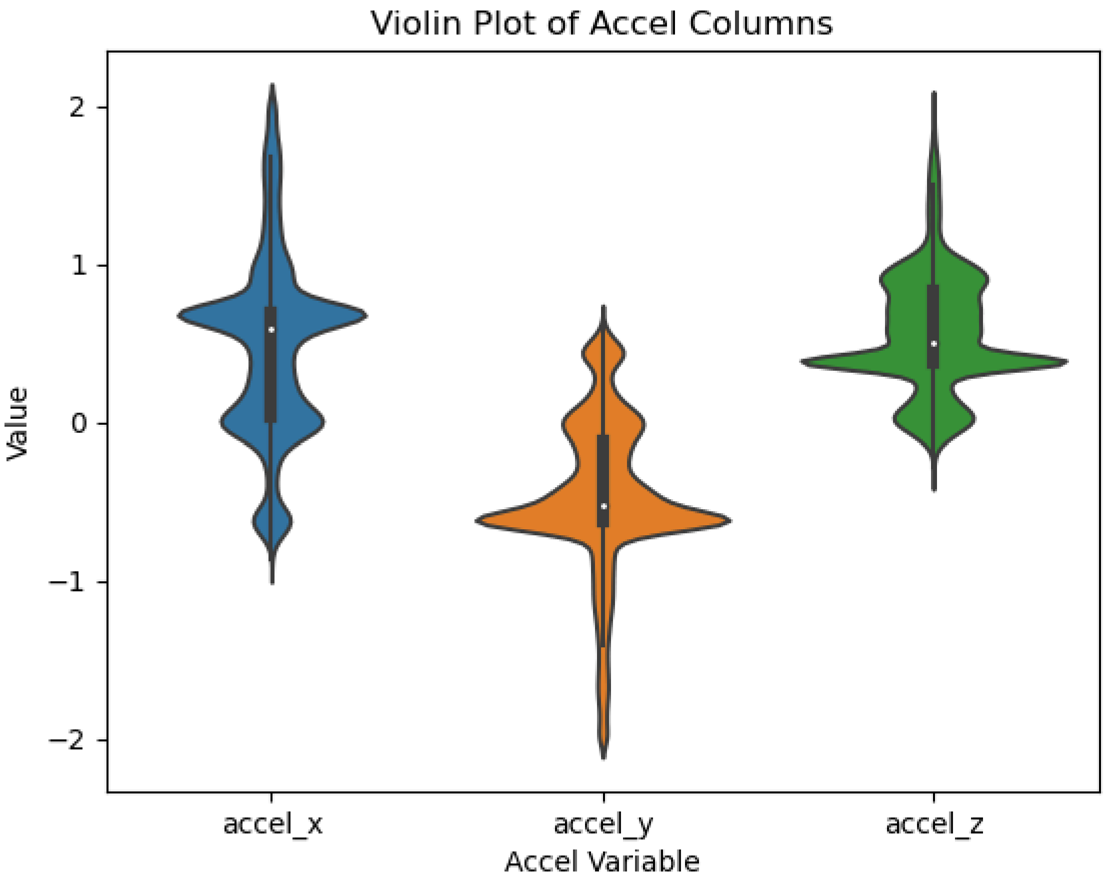
```
# peak_to_peak_gyro değerleri ve output

plt.figure(figsize=(18, 6))

# peak_to_peak_gyro_x
plt.subplot(1, 3, 1)
plt.scatter(son_data['peak_to_peak_gyro_x'], son_data['Output'], color='blue', a
plt.title('peak_to_peak_gyro_x vs Output')
plt.xlabel('peak_to_peak_gyro_x')
plt.ylabel('Output')

# peak_to_peak_gyro_y
plt.subplot(1, 3, 2)
plt.scatter(son_data['peak_to_peak_gyro_y'], son_data['Output'], color='red', al
plt.title('peak_to_peak_gyro_y vs Output')
plt.xlabel('peak_to_peak_gyro_y')
plt.ylabel('Output')

# peak_to_peak_gyro_z
plt.subplot(1, 3, 3)
plt.scatter(son_data['peak_to_peak_gyro_z'], son_data['Output'], color='green',
plt.title('peak_to_peak_gyro_z vs Output')
plt.xlabel('peak_to_peak_gyro_z')
plt.ylabel('Output')
plt.tight_layout()
plt.show()
```



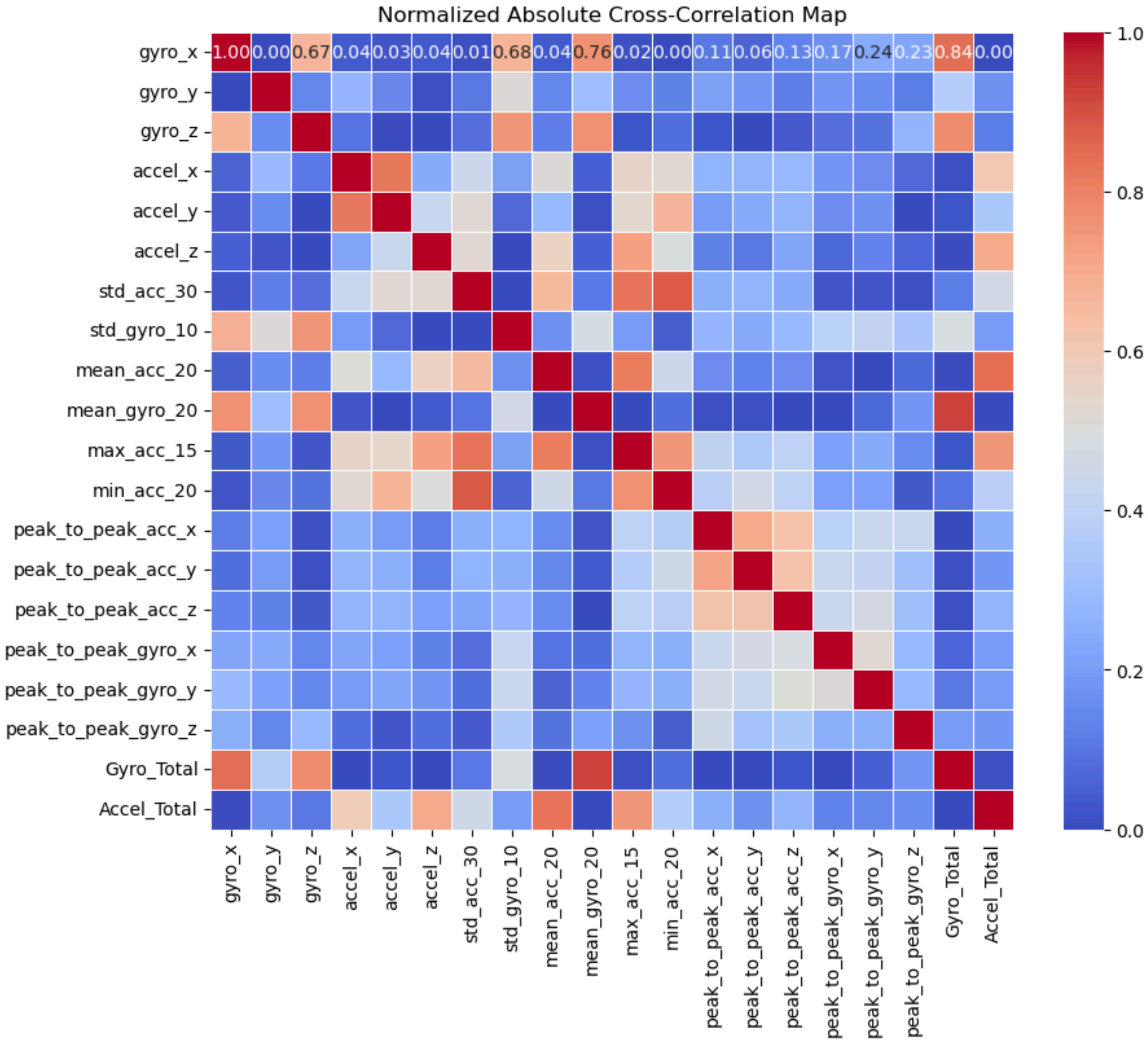
In [43]:

```
# gyro ve accel violin plot

gyro_columns = ['gyro_x', 'gyro_y', 'gyro_z']
accel_columns = ['accel_x', 'accel_y', 'accel_z']

sns.violinplot(data=son_data[gyro_columns])
plt.xlabel('Gyro Variable')
plt.ylabel('Value')
plt.title('Violin Plot of Gyro Columns')
plt.show()

sns.violinplot(data=son_data[accel_columns])
plt.xlabel('Accel Variable')
plt.ylabel('Value')
plt.title('Violin Plot of Accel Columns')
plt.show()
```





In [44]:
```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt

features = son_data.drop(columns=['Output'])


corr_matrix = features.corr().abs()

#Normalleştirilmiş mutlak çapraz korelasyon matrisi
normalized_corr_matrix = (corr_matrix - corr_matrix.min()) / (corr_matrix.max()

#Heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(normalized_corr_matrix, annot=True, cmap='coolwarm', fmt=".2f", line
plt.title('Normalized Absolute Cross-Correlation Map')
plt.show()
```



3) Statistical Information

In [45]:
```
statistical_information = son_data.describe()

print(statistical_information)
```

```
            gyro_x        gyro_y        gyro_z       accel_x       accel_y  \
count  37161.000000  37161.000000  37161.000000  37161.000000  37161.000000
mean      20.783674    -13.797805     25.534451      0.474828     -0.455960
std       41.377206     40.438210     46.621772      0.568782      0.483283
min      -94.307503   -277.156250    -47.206249     -0.870000     -1.998604
25%        0.113750    -31.631250     -0.131250      0.038369     -0.630008
50%        6.387500     -2.180000      2.563750      0.590000     -0.529663
75%       43.426250      2.310000     41.186249      0.692594     -0.112789
max      360.865002    208.390000    214.112503      1.998604      0.620000

           accel_z    std_acc_30   std_gyro_10   mean_acc_20  mean_gyro_20  \
count  37161.000000  37161.000000  37161.000000  37161.000000  37161.000000
mean       0.562330      0.583307     25.162510      0.193229     10.842123
std        0.375338      0.295393     26.719618      0.135518     27.624567
min       -0.330925      0.000000      0.000000     -0.111540    -64.270000
25%        0.378078      0.480000      3.070000      0.135489     -0.970083
50%        0.504775      0.562331     14.239748      0.159617      1.589729
75%        0.830000      0.611720     41.360000      0.250000     17.637667
max        1.998604      1.765029    125.698103      0.710211    119.218750

        max_acc_15    min_acc_20  peak_to_peak_acc_x  peak_to_peak_acc_y  \
count  37161.000000  37161.000000        37161.000000        37161.000000
mean       0.846625     -0.681360            0.072148            0.065917
std        0.429346      0.430828            0.114161            0.097158
min       -0.019581     -1.998604            0.000000            0.000000
25%        0.686128     -0.710000            0.006527            0.004575
50%        0.750000     -0.630000            0.025986            0.024705
75%        1.012600     -0.460672            0.090158            0.084912
max        1.998604      0.134261            1.195600            1.134783

       peak_to_peak_acc_z  peak_to_peak_gyro_x  peak_to_peak_gyro_y  \
count        37161.000000         37161.000000         37161.000000
mean             0.067105             9.495046            10.897254
std              0.098493            14.194731            16.079181
min              0.000000             0.000000             0.000000
25%              0.007381             0.840000             0.940000
50%              0.027267             4.260000             4.850000
75%              0.087108            12.215004            13.912503
max              1.189500           302.835003           242.086243

       peak_to_peak_gyro_z    Gyro_Total   Accel_Total
count         37161.000000  37161.000000  37161.000000
mean              3.362535     32.520320      0.581198
std               5.003731     86.756678      0.437017
min               0.000000   -252.726245     -0.504775
25%               0.300000     -6.413747      0.388753
50%               1.785004      2.620000      0.469456
75%               4.882500     61.810000      0.740000
max             127.487503    459.740004      2.502769
```

4) Standardization

In [46]:

```
# interval[-1,1]
```

In [47]:

```
numeric_columns = son_data.select_dtypes(include=['float64', 'int64']).columns

if 'Output' in numeric_columns:
    numeric_columns = numeric_columns.drop('Output')
```

```
min_val = son_data[numeric_columns].min()
max_val = son_data[numeric_columns].max()

new_min = -1
new_max = 1

normalized_data = ((son_data[numeric_columns] - min_val) / (max_val - min_val))

#Normalleştirilmiş veri seti
normalized_data = pd.concat([normalized_data, yeni_data['Output']], axis=1)
```

In [48]:

```
print(normalized_data.head(15))
```

```
      gyro_x    gyro_y    gyro_z   accel_x   accel_y   accel_z  std_acc_30  \
0  -0.583427  0.138959 -0.637703  0.084427  0.045287 -0.386944        -1.0   
1  -0.583542  0.138635 -0.637770  0.083747  0.045147 -0.386473        -1.0   
2  -0.583889  0.138996 -0.637770  0.085916  0.045473 -0.386630        -1.0   
3  -0.584042  0.138923 -0.637971  0.085278  0.046079 -0.386158        -1.0   
4  -0.584311  0.139140 -0.637636  0.084597  0.044075 -0.387310        -1.0   
5  -0.584965  0.138923 -0.637569  0.084597  0.045007 -0.389510        -1.0   
6  -0.584542  0.138959 -0.637770  0.084640  0.044634 -0.387625        -1.0   
7  -0.584465  0.139428 -0.636765  0.086341  0.044402 -0.388620        -1.0   
8  -0.584542  0.138671 -0.637435  0.085023  0.044122 -0.387939        -1.0   
9  -0.584350  0.139104 -0.637033  0.084895  0.045287 -0.385320        -1.0   
10 -0.583658  0.138671 -0.636899  0.084172  0.043749 -0.386315        -1.0   
11 -0.582850  0.138635 -0.637167  0.083959  0.044588 -0.386001        -1.0   
12 -0.582774  0.138923 -0.636498  0.085618  0.043982 -0.386211        -1.0   
13 -0.582581  0.137986 -0.636565  0.084044  0.044728 -0.384273        -1.0   
14 -0.582581  0.138563 -0.637033  0.084852  0.045473 -0.384273        -1.0   

    std_gyro_10  mean_acc_20  mean_gyro_20  ...  min_acc_20  \
0     -1.000000    -0.728532     -0.299467  ...    0.874103   
1     -1.000000    -0.728532     -0.299467  ...    0.874103   
2     -1.000000    -0.728532     -0.299467  ...    0.874103   
3     -1.000000    -0.728532     -0.299467  ...    0.874103   
4     -1.000000    -0.728532     -0.299467  ...    0.874103   
5     -1.000000    -0.728532     -0.299467  ...    0.874103   
6     -1.000000    -0.728532     -0.299467  ...    0.874103   
7     -1.000000    -0.728532     -0.299467  ...    0.874103   
8     -1.000000    -0.728532     -0.299467  ...    0.874103   
9     -0.993206    -0.728532     -0.299467  ...    0.874103   
10    -0.993155    -0.728532     -0.299467  ...    0.874103   
11    -0.993018    -0.728532     -0.299467  ...    0.874103   
12    -0.992786    -0.728532     -0.299467  ...    0.874103   
13    -0.992340    -0.728532     -0.299467  ...    0.874103   
14    -0.992006    -0.728532     -0.299467  ...    0.874103   

    peak_to_peak_acc_x  peak_to_peak_acc_y  peak_to_peak_acc_z  \
0            -1.000000           -1.000000           -1.000000   
1            -1.000000           -1.000000           -1.000000   
2            -1.000000           -1.000000           -1.000000   
3            -1.000000           -1.000000           -1.000000   
4            -0.994796           -0.995377           -0.997744   
5            -0.994796           -0.995377           -0.993436   
6            -0.996837           -0.995377           -0.993436   
7            -0.995816           -0.995377           -0.993436   
8            -0.995816           -0.997850           -0.995692   
9            -0.995816           -0.997312           -0.991795   
10           -0.994796           -0.996452           -0.993538   
11           -0.994286           -0.996452           -0.993538   
12           -0.996020           -0.996452           -0.994872   
13           -0.996020           -0.996452           -0.996000   
14           -0.996020           -0.996022           -0.996000   

    peak_to_peak_gyro_x  peak_to_peak_gyro_y  peak_to_peak_gyro_z  Gyro_Total  \
0             -1.000000            -1.000000            -1.000000   -0.290609   
1             -1.000000            -1.000000            -1.000000   -0.290928   
2             -1.000000            -1.000000            -1.000000   -0.290903   
3             -1.000000            -1.000000            -1.000000   -0.291124   
4             -0.998671            -0.998988            -0.999314   -0.291026   
5             -0.997862            -0.998988            -0.999176   -0.291566   
6             -0.998382            -0.999566            -0.999176   -0.291345   
7             -0.998613            -0.998988            -0.997529   -0.290609   
```

```
8             -0.999018            -0.998482            -0.997941   -0.291419
9             -0.999075            -0.998482            -0.997941   -0.290854
10            -0.998671            -0.998482            -0.997941   -0.290658
11            -0.997457            -0.998410            -0.998627   -0.290265
12            -0.997342            -0.999060            -0.998078   -0.289773
13            -0.997342            -0.997759            -0.998627   -0.290314
14            -0.998382            -0.998121            -0.998627   -0.290093

    Accel_Total  Output
0     -0.372708     sit
1     -0.373114     sit
2     -0.370883     sit
3     -0.370599     sit
4     -0.373884     sit
5     -0.374777     sit
6     -0.373601     sit
7     -0.372951     sit
8     -0.373925     sit
9     -0.371004     sit
10    -0.373803     sit
11    -0.373033     sit
12    -0.372140     sit
13    -0.371491     sit
14    -0.370071     sit

[15 rows x 21 columns]
```

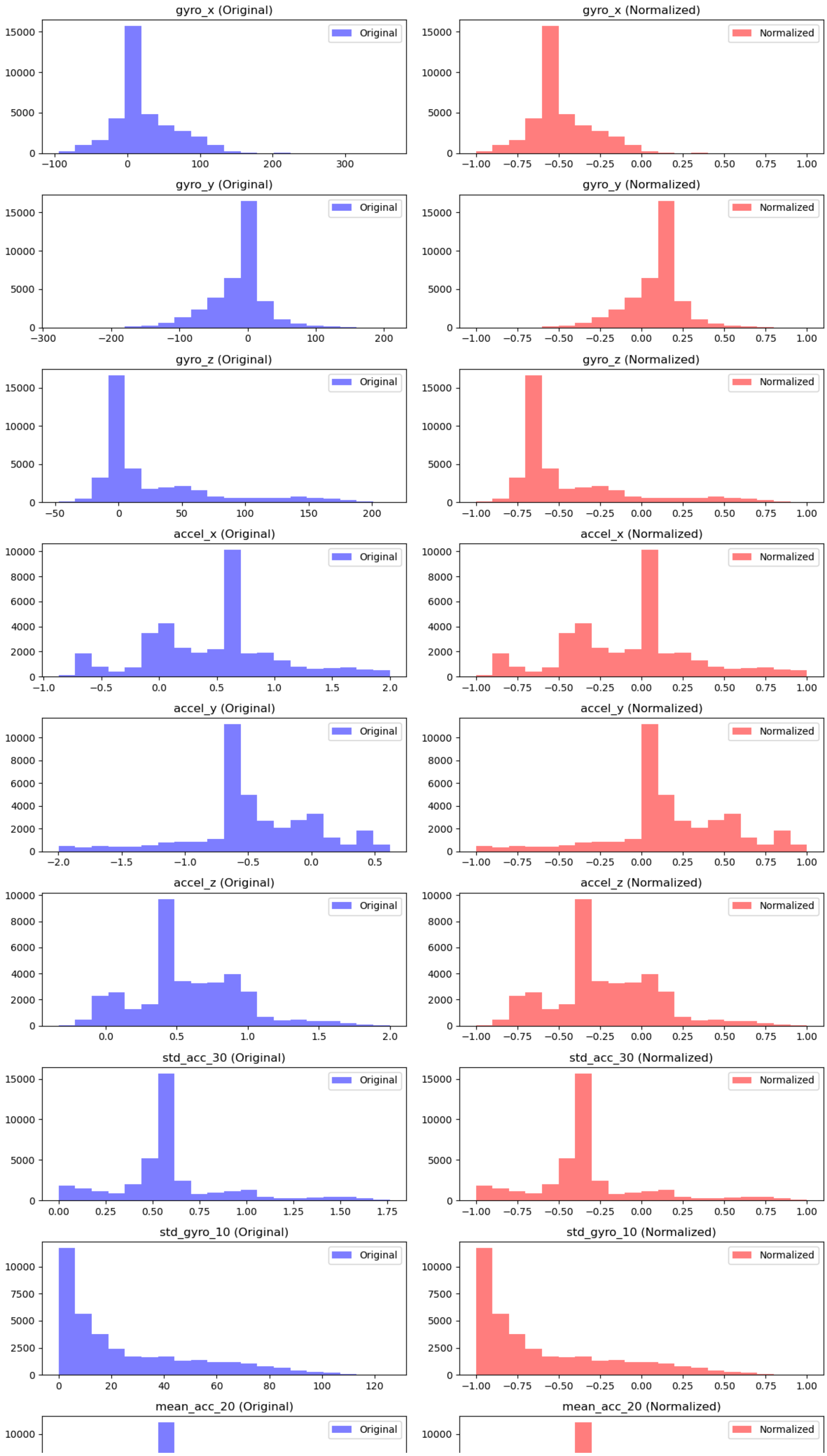
In [49]:

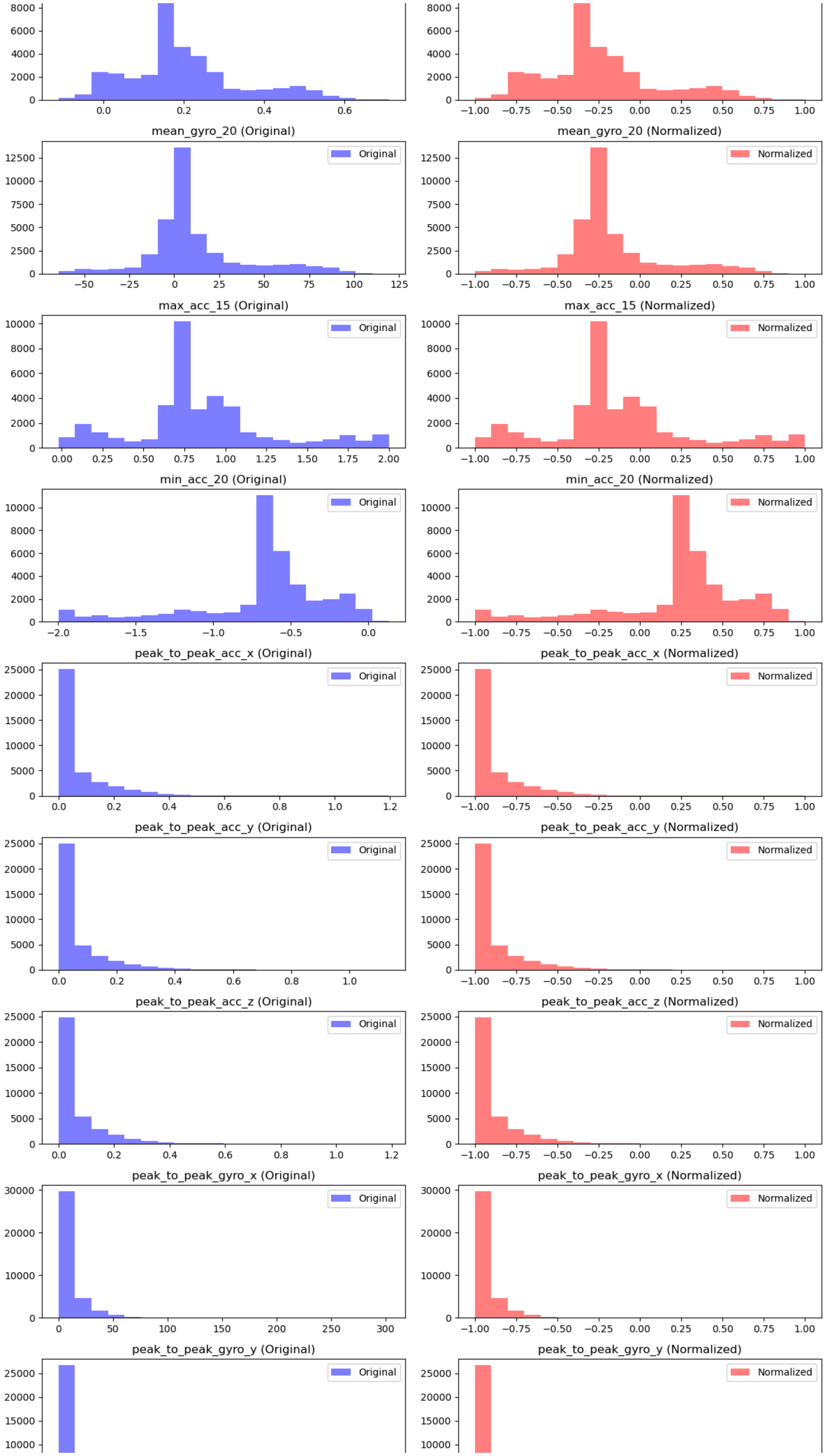
```
statistical_information = normalized_data.describe()

print(statistical_information)
```

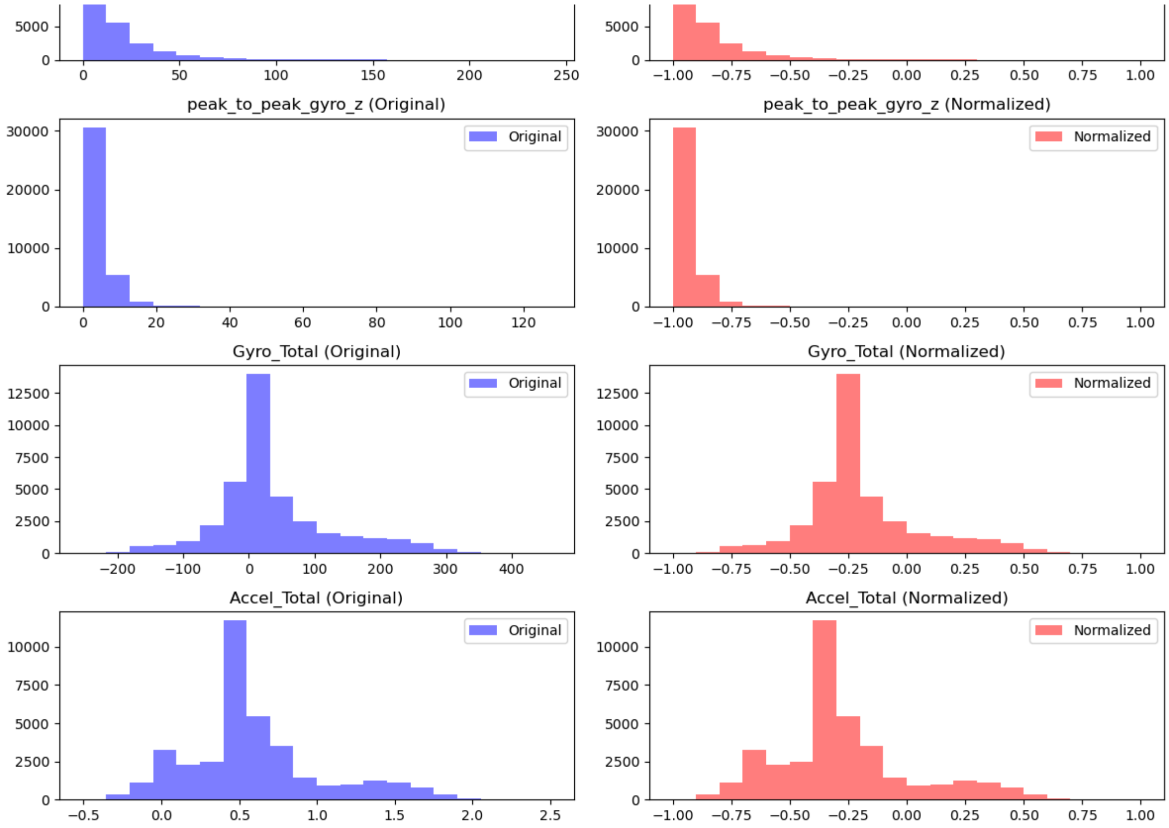
```
             gyro_x        gyro_y        gyro_z       accel_x       accel_y  \
count  37161.000000  37161.000000  37161.000000  37161.000000  37161.000000   
mean      -0.494296      0.084792     -0.443280     -0.062382      0.178219   
std        0.181809      0.166568      0.356819      0.396557      0.369115   
min       -1.000000     -1.000000     -1.000000     -1.000000     -1.000000   
25%       -0.585119      0.011335     -0.639712     -0.366682      0.045287   
50%       -0.557552      0.132647     -0.619086      0.017917      0.121927   
75%       -0.394806      0.151142     -0.323489      0.089446      0.440321   
max        1.000000      1.000000      1.000000      1.000000      1.000000   

            accel_z    std_acc_30   std_gyro_10   mean_acc_20  mean_gyro_20  \
count  37161.000000  37161.000000  37161.000000  37161.000000  37161.000000   
mean      -0.233102     -0.339039     -0.599636     -0.258245     -0.181289   
std        0.322244      0.334718      0.425140      0.329828      0.301104   
min       -1.000000     -1.000000     -1.000000     -1.000000     -1.000000   
25%       -0.391291     -0.456100     -0.951153     -0.398774     -0.310040   
50%       -0.282516     -0.362808     -0.773429     -0.340052     -0.282139   
75%       -0.003296     -0.306845     -0.341915     -0.120075     -0.107219   
max        1.000000      1.000000      1.000000      1.000000      1.000000   

         max_acc_15    min_acc_20  peak_to_peak_acc_x  peak_to_peak_acc_y  \
count  37161.000000  37161.000000        37161.000000        37161.000000   
mean      -0.141599      0.235187           -0.879310           -0.883825   
std        0.425478      0.403990            0.190969            0.171237   
min       -1.000000     -1.000000           -1.000000           -1.000000   
25%       -0.300650      0.208332           -0.989082           -0.991937   
50%       -0.237353      0.283348           -0.956531           -0.956459   
75%        0.022880      0.442128           -0.849184           -0.850347   
max        1.000000      1.000000            1.000000            1.000000   

       peak_to_peak_acc_z  peak_to_peak_gyro_x  peak_to_peak_gyro_y  \
count        37161.000000         37161.000000         37161.000000   
mean            -0.887171            -0.937292            -0.909972   
std              0.165604             0.093746             0.132838   
min             -1.000000            -1.000000            -1.000000   
25%             -0.987590            -0.994452            -0.992234   
50%             -0.954154            -0.971866            -0.959932   
75%             -0.853538            -0.919329            -0.885062   
max              1.000000             1.000000             1.000000   

       peak_to_peak_gyro_z    Gyro_Total   Accel_Total  
count         37161.000000  37161.000000  37161.000000  
mean             -0.947249     -0.199270     -0.277834  
std               0.078498      0.243539      0.290614  
min              -1.000000     -1.000000     -1.000000  
25%              -0.995294     -0.308564     -0.405809  
50%              -0.971997     -0.283205     -0.352142  
75%              -0.923404     -0.117049     -0.172232  
max               1.000000      1.000000      1.000000  
```

In [50]:

```
# etki
import matplotlib.pyplot as plt


fig, axs = plt.subplots(nrows=len(numeric_columns), ncols=2, figsize=(12, 2.5*le

for i, col in enumerate(numeric_columns):

    axs[i, 0].hist(son_data[col], bins=20, color='blue', alpha=0.5, label='Origi
    axs[i, 0].set_title(col + ' (Original)')
    axs[i, 0].legend()


    axs[i, 1].hist(normalized_data[col], bins=20, color='red', alpha=0.5, label=
    axs[i, 1].set_title(col + ' (Normalized)')
    axs[i, 1].legend()

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


gyro_x (Original)
Original
15000 10000 5000 0
−100 0 100 200 300
gyro_x (Normalized)
Normalized
15000 10000 5000 0
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00
gyro_y (Original)
Original
15000 10000 5000 0
−300 −200 −100 0 100 200
gyro_y (Normalized)
Normalized
15000 10000 5000 0
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00
gyro_z (Original)
Original
15000 10000 5000 0
−50 0 50 100 150 200
gyro_z (Normalized)
Normalized
15000 10000 5000 0
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00
accel_x (Original)
Original
10000 8000 6000 4000 2000 0
−1.0 −0.5 0.0 0.5 1.0 1.5 2.0
accel_x (Normalized)
Normalized
10000 8000 6000 4000 2000 0
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00
accel_y (Original)
Original
10000 8000 6000 4000 2000 0
−2.0 −1.5 −1.0 −0.5 0.0 0.5
accel_y (Normalized)
Normalized
10000 8000 6000 4000 2000 0
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00
accel_z (Original)
Original
10000 8000 6000 4000 2000 0
0.0 0.5 1.0 1.5 2.0
accel_z (Normalized)
Normalized
10000 8000 6000 4000 2000 0
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00
std_acc_30 (Original)
Original
15000 10000 5000 0
0.00 0.25 0.50 0.75 1.00 1.25 1.50 1.75
std_acc_30 (Normalized)
Normalized
15000 10000 5000 0
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00
std_gyro_10 (Original)
Original
10000 7500 5000 2500 0
0 20 40 60 80 100 120
std_gyro_10 (Normalized)
Normalized
10000 7500 5000 2500 0
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00
mean_acc_20 (Original)
Original
10000
mean_acc_20 (Normalized)
Normalized
10000


8000
6000
4000
2000
0
0.0
0.2
0.4
0.6
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
mean_gyro_20 (Original)
mean_gyro_20 (Normalized)
Original
Normalized
12500
10000
7500
5000
2500
0
−50
−25
0
25
50
75
100
125
max_acc_15 (Original)
max_acc_15 (Normalized)
10000
8000
6000
4000
2000
0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
min_acc_20 (Original)
min_acc_20 (Normalized)
−2.0
−1.5
−1.0
−0.5
0.0
peak_to_peak_acc_x (Original)
peak_to_peak_acc_x (Normalized)
25000
20000
15000
10000
5000
0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
peak_to_peak_acc_y (Original)
peak_to_peak_acc_y (Normalized)
peak_to_peak_acc_z (Original)
peak_to_peak_acc_z (Normalized)
peak_to_peak_gyro_x (Original)
peak_to_peak_gyro_x (Normalized)
30000
20000
10000
0
0
50
100
150
200
250
300
peak_to_peak_gyro_y (Original)
peak_to_peak_gyro_y (Normalized)
25000
20000
15000
10000



5) Classification

In [51]:

```
dataset_3 = normalized_data.copy()
```

In [52]:

```
# knn
from sklearn.model_selection import StratifiedKFold, train_test_split, cross_val
from sklearn.metrics import accuracy_score, r2_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
import numpy as np
import pandas as pd


X = dataset_3.drop(columns=['Output'])
y = dataset_3['Output']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratif

knn_model = KNeighborsClassifier()


skf = StratifiedKFold(n_splits=10, shuffle=True, random_state=47)
knn_cv_scores = cross_val_score(knn_model, X_train, y_train, cv=skf, scoring='ac

knn_avg_cv_scores = np.mean(knn_cv_scores)

print(" KNN Model - Avg Cross Validation Accuracy :" , knn_avg_cv_scores)
```

```
KNN Model - Avg Cross Validation Accuracy : 0.9992599275036568
```

In [53]:

```
# Descision Tree
```

```python
decision_tree_model = DecisionTreeClassifier()

decision_tree_cv_scores = cross_val_score(decision_tree_model , X_train, y_train

decision_tree_avg_cv_scores = np.mean(decision_tree_cv_scores)

print(" Decision Model - Avg Cross Validation Accuracy :" , decision_tree_avg_cv
```

```
 Decision Model - Avg Cross Validation Accuracy : 0.9930033378015419
```

In [54]:

```python
# Naive Bayes

naive_bayes_model =  GaussianNB()

naive_bayes_cv_scores = cross_val_score(naive_bayes_model , X_train, y_train, cv

naive_bayes_avg_cv_scores = np.mean(naive_bayes_cv_scores)

print(" Naive Bayes Model - Avg Cross Validation Accuracy :" , naive_bayes_avg_c
```

```
 Naive Bayes Model - Avg Cross Validation Accuracy : 0.9240784263395232
```

In [55]:

```python
# Random Forest
from sklearn.ensemble import RandomForestClassifier

random_forest_model =  RandomForestClassifier()

random_forest_cv_scores = cross_val_score(random_forest_model , X_train, y_train

random_forest_avg_cv_scores = np.mean(random_forest_cv_scores)

print(" Random Forest Model - Avg Cross Validation Accuracy :" , random_forest_a
```

```
 Random Forest Model - Avg Cross Validation Accuracy : 0.9996636054685079
```

In [56]:

```python
#Support Vector Machine
from sklearn.svm import SVC

svm_model =  SVC()

svm_model_cv_scores = cross_val_score(svm_model , X_train, y_train, cv=skf, scor

svm_model_avg_cv_scores = np.mean(svm_model_cv_scores)

print(" SVM Model - Avg Cross Validation Accuracy :" , svm_model_avg_cv_scores )
```

```
 SVM Model - Avg Cross Validation Accuracy : 0.9955934161151575
```

In [70]:

```python
models = ["KNN", "Decision Tree", "Naive Bayes", "Random Forest", "SVM"]
avg_cv_accuracies = [knn_avg_cv_scores, decision_tree_avg_cv_scores, naive_bayes

# En yüksek çapraz doğrulama doğruluğunu ve bu doğruluğa sahip modelin indeksini
best_accuracy = max(avg_cv_accuracies)
best_model_index = avg_cv_accuracies.index(best_accuracy)
best_model = models[best_model_index]

print("BEST MODEL:", best_model)
print("BEST MODEL AVG CROSS VALIDATION SCORE:", best_accuracy)
```

```
BEST MODEL: Random Forest
BEST MODEL AVG CROSS VALIDATION SCORE: 0.9996636054685079
```

In [57]:

```
# Cross Validation model performansları

cv_performance = {  "Model": ["KNN", "Decision Tree","Naive Bayes","Random Fores
                    "Avg Cross Validation Accuracy": [knn_avg_cv_scores , decisi
                                                       naive_bayes_avg_cv_scores
                                                       svm_model_avg_cv_scores]
}
cv_performance_df = pd.DataFrame(cv_performance)

df_sorted = cv_performance_df.sort_values(by='Avg Cross Validation Accuracy', as
df_sorted.reset_index(drop=True, inplace=True)
df_sorted.index += 1

display(df_sorted)
```

	Model	Avg Cross Validation Accuracy
1	Random Forest	0.999664
2	KNN	0.999260
3	SVM	0.995593
4	Decision Tree	0.993003
5	Naive Bayes	0.924078

In [58]:

```
# knn
from sklearn.metrics import confusion_matrix, precision_score, recall_score, f1_

knn_model.fit(X_train, y_train)

# train accuracy
knn_train_accuracy = knn_model.score(X_train, y_train)
print("Train Accuracy (KNN):", knn_train_accuracy)
print()

# test accuracy
knn_test_accuracy = knn_model.score(X_test, y_test)
print("Test Accuracy (KNN):", knn_test_accuracy)
print()

# prediction
knn_test_predictions = knn_model.predict(X_test)


# Test seti için confusion matrix
knn_test_conf_matrix = confusion_matrix(y_test, knn_test_predictions)
print("Test Confusion Matrix (KNN):\n", knn_test_conf_matrix)
print()

# Test seti için precision hesaplama
knn_test_precision = precision_score(y_test, knn_test_predictions, average='weig
print("Test Precision (KNN):", knn_test_precision)
print()

# Test seti için recall hesaplama
knn_test_recall = recall_score(y_test, knn_test_predictions, average='weighted')
print("Test  Recall (KNN):", knn_test_recall)
print()

# Test seti için F1-score hesaplama
knn_test_f1 = f1_score(y_test, knn_test_predictions, average='weighted')
print("Test  F1-score (KNN):", knn_test_f1)
```

```
Train Accuracy (KNN): 0.9999663616792249

Test Accuracy (KNN): 0.9998654648190501

Test Confusion Matrix (KNN):
 [[1406    0    0    0    0]
 [   0 1625    0    0    0]
 [   0    0 1451    0    0]
 [   1    0    0 1477    0]
 [   0    0    0    0 1473]]

Test Precision (KNN): 0.9998655604375157

Test  Recall (KNN): 0.9998654648190501

Test  F1-score (KNN): 0.9998654659681744
```

In [59]:

```
# Decision Tree
from sklearn.metrics import confusion_matrix, precision_score, recall_score, f1_

decision_tree_model.fit(X_train, y_train)

# train accuracy
dt_train_accuracy = decision_tree_model.score(X_train, y_train)
print("Train Accuracy (Decision Tree):", dt_train_accuracy)
print()

# test accuracy
dt_test_accuracy = decision_tree_model.score(X_test, y_test)
print("Test Accuracy (Decision Tree):", dt_test_accuracy)
print()

# prediction
dt_test_predictions = decision_tree_model.predict(X_test)


# Test seti için confusion matrix
dt_test_conf_matrix = confusion_matrix(y_test, dt_test_predictions)
print("Test Confusion Matrix (Decision Tree):\n", dt_test_conf_matrix)
print()

# Test seti için precision hesaplama
dt_test_precision = precision_score(y_test, dt_test_predictions, average='weight
print("Test Precision (Decision Tree):", dt_test_precision)
print()

# Test seti için recall hesaplama
dt_test_recall = recall_score(y_test, dt_test_predictions, average='weighted')
print("Test  Recall (Decision Tree):", dt_test_recall)
print()

# Test seti için F1-score hesaplama
dt_test_f1 = f1_score(y_test, dt_test_predictions, average='weighted')
print("Test  F1-score (Decision Tree)):", dt_test_f1)
```

```
Train Accuracy (Decision Tree): 1.0

Test Accuracy (Decision Tree): 0.9934077761334589

Test Confusion Matrix (Decision Tree):
 [[1400    2    1    3    0]
 [   1 1615    1    8    0]
 [   7    6 1431    7    0]
 [   2    2    9 1465    0]
 [   0    0    0    0 1473]]

Test Precision (Decision Tree): 0.9934105209125538

Test  Recall (Decision Tree): 0.9934077761334589

Test  F1-score (Decision Tree)): 0.9934063389459669
```

In [60]:

```python
# Naive Bayes
from sklearn.metrics import confusion_matrix, precision_score, recall_score, f1_

naive_bayes_model.fit(X_train, y_train)

# train accuracy
nb_train_accuracy = naive_bayes_model.score(X_train, y_train)
print("Train Accuracy (Naive Bayes):", nb_train_accuracy)
print()

# test accuracy
nb_test_accuracy = naive_bayes_model.score(X_test, y_test)
print("Test Accuracy (Naive Bayes):", nb_test_accuracy)
print()

# prediction
nb_test_predictions = naive_bayes_model.predict(X_test)

# Test seti için confusion matrix
nb_test_conf_matrix = confusion_matrix(y_test, nb_test_predictions)
print("Test Confusion Matrix (Naive Bayes):\n", nb_test_conf_matrix)
print()

# Test seti için precision hesaplama
nb_test_precision = precision_score(y_test, nb_test_predictions, average='weight
print("Test Precision (Naive Bayes):", nb_test_precision)
print()

# Test seti için recall hesaplama
nb_test_recall = recall_score(y_test, nb_test_predictions, average='weighted')
print("Test  Recall (Naive Bayes):", nb_test_recall)
print()

# Test seti için F1-score hesaplama
nb_test_f1 = f1_score(y_test, nb_test_predictions, average='weighted')
print("Test  F1-score (Naive Bayes)):", nb_test_f1)
```

```
Train Accuracy (Naive Bayes): 0.9241455866523143

Test Accuracy (Naive Bayes): 0.9269473967442486

Test Confusion Matrix (Naive Bayes):
 [[1325   60    9   12    0]
 [ 225 1304   69   27    0]
 [   0    0 1376   75    0]
 [   9   13   25 1431    0]
 [  12    0    7    0 1454]]

Test Precision (Naive Bayes): 0.9296682580074981

Test  Recall (Naive Bayes): 0.9269473967442486

Test  F1-score (Naive Bayes)): 0.926357095967973
```

In [61]:

```
# Random Forest
from sklearn.metrics import confusion_matrix, precision_score, recall_score, f1_

random_forest_model.fit(X_train, y_train)

# train accuracy
rf_train_accuracy = random_forest_model.score(X_train, y_train)
print("Train Accuracy (Random Forest)):", rf_train_accuracy)
print()

# test accuracy
rf_test_accuracy = random_forest_model.score(X_test, y_test)
print("Test Accuracy (Random Forest)):", rf_test_accuracy)
print()

# prediction
rf_test_predictions = random_forest_model.predict(X_test)


# Test seti için confusion matrix
rf_test_conf_matrix = confusion_matrix(y_test, rf_test_predictions)
print("Test Confusion Matrix (Random Forest)):\n", rf_test_conf_matrix)
print()

# Test seti için precision hesaplama
rf_test_precision = precision_score(y_test, rf_test_predictions, average='weight
print("Test Precision (Random Forest)):", rf_test_precision)
print()

# Test seti için recall hesaplama
rf_test_recall = recall_score(y_test, rf_test_predictions, average='weighted')
print("Test  Recall (Random Forest)):", rf_test_recall)
print()

# Test seti için F1-score hesaplama
rf_test_f1 = f1_score(y_test, rf_test_predictions, average='weighted')
print("Test  F1-score (Random Forest)):", rf_test_f1)
```

```
Train Accuracy (Random Forest)): 1.0

Test Accuracy (Random Forest)): 0.9998654648190501

Test Confusion Matrix (Random Forest)):
 [[1406    0    0    0    0]
 [   0 1625    0    0    0]
 [   0    0 1451    0    0]
 [   0    0    1 1477    0]
 [   0    0    0    0 1473]]

Test Precision (Random Forest)): 0.9998655574741335

Test  Recall (Random Forest)): 0.9998654648190501

Test  F1-score (Random Forest)): 0.9998654652268102
```

In [62]:

```
from sklearn.svm import SVC

svm_model = SVC()
svm_model.fit(X_train, y_train)

svm_train_accuracy = svm_model.score(X_train, y_train)
print("Train Accuracy (SVM):", svm_train_accuracy)
print()

svm_test_accuracy = svm_model.score(X_test, y_test)
print("Test Accuracy (SVM):", svm_test_accuracy)
print()

svm_test_predictions = svm_model.predict(X_test)

svm_test_conf_matrix = confusion_matrix(y_test, svm_test_predictions)
print("Test Confusion Matrix (SVM):\n", svm_test_conf_matrix)
print()

svm_test_precision = precision_score(y_test, svm_test_predictions, average='weig
print("Test Precision (SVM):", svm_test_precision)
print()

svm_test_recall = recall_score(y_test, svm_test_predictions, average='weighted')
print("Test  Recall (SVM):", svm_test_recall)
print()

svm_test_f1 = f1_score(y_test, svm_test_predictions, average='weighted')
print("Test  F1-score (SVM):", svm_test_f1)
print()
```

```
Train Accuracy (SVM): 0.996232508073197

Test Accuracy (SVM): 0.9952912686667563

Test Confusion Matrix (SVM):
 [[1401    0    0    0    5]
 [   0 1625    0    0    0]
 [   0    0 1450    1    0]
 [  19    0    6 1453    0]
 [   0    0    0    4 1469]]

Test Precision (SVM): 0.9953104713256962

Test  Recall (SVM): 0.9952912686667563

Test  F1-score (SVM): 0.9952865350564953
```

In [68]:

```
import pandas as pd

def format_float(val):
    return "{:.15f}".format(val)

model_performanslar = {
    "Model": ["KNN", "Decision Tree", "Naive Bayes", "Random Forest", "SVM"],
    "Avg Cross Validation Score": [knn_avg_cv_scores, decision_tree_avg_cv_score
                                   random_forest_avg_cv_scores, svm_model_avg
    "Test Accuracy": [knn_test_accuracy, dt_test_accuracy, nb_test_accuracy, rf_
    "Train Accuracy": [knn_train_accuracy, dt_train_accuracy, nb_train_accuracy,
    "Precision": [knn_test_precision, dt_test_precision, nb_test_precision, rf_t
    "Recall": [knn_test_recall, dt_test_recall, nb_test_recall, rf_test_recall,
    "F1-score": [knn_test_f1, dt_test_f1, nb_test_f1, rf_test_f1, svm_test_f1]
}


model_performanslar_df = pd.DataFrame(model_performanslar)


model_performanslar_df['Avg Cross Validation Score'] = model_performanslar_df['A
model_performanslar_df['Test Accuracy'] = model_performanslar_df['Test Accuracy'
model_performanslar_df['Train Accuracy'] = model_performanslar_df['Train Accurac
model_performanslar_df['Precision'] = model_performanslar_df['Precision'].apply(
model_performanslar_df['Recall'] = model_performanslar_df['Recall'].apply(format
model_performanslar_df['F1-score'] = model_performanslar_df['F1-score'].apply(fo

# Accuracy'ye göre sıralama
model_performanslar_df_sorted = model_performanslar_df.sort_values(by='Test Accu
model_performanslar_df_sorted.reset_index(drop=True, inplace=True)
model_performanslar_df_sorted.index += 1

print(model_performanslar_df_sorted)
```

```
           Model  Avg Cross Validation Score     Test Accuracy  \
1            KNN           0.999259927503657  0.999865464819050
2  Random Forest           0.999663605468508  0.999865464819050
3            SVM           0.995593416115158  0.995291268666756
4  Decision Tree           0.993003337801542  0.993407776133459
5    Naive Bayes           0.924078426339523  0.926947396744249

      Train Accuracy          Precision             Recall           F1-score
1  0.999966361679225  0.999865560437516  0.999865464819050  0.999865465968174
2  1.000000000000000  0.999865557474134  0.999865464819050  0.999865465226810
3  0.996232508073197  0.995310471325696  0.995291268666756  0.995286535056495
4  1.000000000000000  0.993410520912554  0.993407776133459  0.993406338945967
5  0.924145586652314  0.929668258007498  0.926947396744249  0.926357095967973
```

In [67]:

```
# KNN (K-Nearest Neighbors): Cross Validation Score'u oldukça yüksek (%99.93) ve
# Modelin eğitim verilerine göre de oldukça yüksek accuracy göstermesi, genel ol

#Random Forest: Random Forest modeli, Cross Validation Score ve test accuracy aç
# train accuracy 1 olması , modelin eğitim verilerine aşırı uyum sağlamış olabil
# ancak test verileri üzerinde de yüksek doğruluk elde etmesi olumlu bir işaret.

# SVM (Support Vector Machine): SVM modeli diğerlerine kıyasla biraz daha düşük
# Diğer modellere kıyasla train accuracy ve test accuracy arasında bir fark var,
# bu da modelin eğitim verilerine belirli bir oranda uyum sağladığını ancak gene

#Decision Tree: Karar Ağaçları modeli, diğerlerine göre Cross Validation Score'v
# ancak hala oldukça yüksek bir accuracy'ye sahip.

#Naive Bayes: Naive Bayes modeli diğerlerine kıyasla daha düşük bir performans g
# Cross Validation Score ve test accuracy diğer modellere göre daha düşük, ancak
```

In [ ]: