In [1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```python
train=pd.read_csv("train.csv")
test=pd.read_csv("test.csv")
```

In [3]:
```python
train.head(10)
```

Out[3]:

	tBodyAcc-mean()-X	tBodyAcc-mean()-Y	tBodyAcc-mean()-Z	tBodyAcc-std()-X	tBodyAcc-std()-Y	tBodyAcc-std()-Z	tBodyAcc-mad()-X	tBodyAcc-mad()-Y	tBodyAcc-mad()-Z	tBodyAcc-max()-X	...	fBodyBodyGyroJerkMag-kurtosis()	angle(tBodyAccMean,gravity)	angle(tBodyAccJerkMean),gravityMean)	angle(tBodyGyroMean,gravityMean)
0	0.288585	-0.020294	-0.132905	-0.995279	-0.983111	-0.913526	-0.995112	-0.983185	-0.923527	-0.934724	...	-0.710304	-0.112754	0.030400	-0.464761
1	0.278419	-0.016411	-0.123520	-0.998245	-0.975300	-0.960322	-0.998807	-0.974914	-0.957686	-0.943068	...	-0.861499	0.053477	-0.007435	-0.732626
2	0.279653	-0.019467	-0.113462	-0.995380	-0.967187	-0.978944	-0.996520	-0.963668	-0.977469	-0.938692	...	-0.760104	-0.118559	0.177899	0.100699
3	0.279174	-0.026201	-0.123283	-0.996091	-0.983403	-0.990675	-0.997099	-0.982750	-0.989302	-0.938692	...	-0.482845	-0.036788	-0.012892	0.640011
4	0.276629	-0.016570	-0.115362	-0.998139	-0.980817	-0.990482	-0.998321	-0.979672	-0.990441	-0.942469	...	-0.699205	0.123320	0.122542	0.693578
5	0.277199	-0.010098	-0.105137	-0.997335	-0.990487	-0.995420	-0.997627	-0.990218	-0.995549	-0.942469	...	-0.844619	0.082632	-0.143439	0.275041
6	0.279454	-0.019641	-0.110022	-0.996921	-0.967186	-0.983118	-0.997003	-0.966097	-0.983116	-0.940987	...	-0.564430	-0.212754	-0.230622	0.014637
7	0.277432	-0.030488	-0.125360	-0.996559	-0.966728	-0.981585	-0.996485	-0.966313	-0.982982	-0.940987	...	-0.421715	-0.020888	0.593996	-0.561871
8	0.277293	-0.021751	-0.120751	-0.997328	-0.961245	-0.983672	-0.997596	-0.957236	-0.984379	-0.940598	...	-0.572995	0.012954	0.080936	-0.234313
9	0.280586	-0.009960	-0.106065	-0.994803	-0.972758	-0.986244	-0.995405	-0.973663	-0.985642	-0.940028	...	0.140452	-0.020590	-0.127730	-0.482871

10 rows × 563 columns

In [48]:
```python
test.head(10)
```

Out[48]:

	tBodyAcc-mean()-X	tBodyAcc-mean()-Y	tBodyAcc-mean()-Z	tBodyAcc-std()-X	tBodyAcc-std()-Y	tBodyAcc-std()-Z	tBodyAcc-mad()-X	tBodyAcc-mad()-Y	tBodyAcc-mad()-Z	tBodyAcc-max()-X	...	fBodyBodyGyroJerkMag-kurtosis()	angle(tBodyAccMean,gravity)	angle(tBodyAccJerkMean),gravityMean)	angle(tBodyGyroMean,gravityMean)
0	0.257178	-0.023285	-0.014654	-0.938404	-0.920091	-0.667683	-0.952501	-0.925249	-0.674302	-0.894088	...	-0.705974	0.006462	0.162920	-0.825886
1	0.286027	-0.013163	-0.119083	-0.975415	-0.967458	-0.944958	-0.986799	-0.968401	-0.945823	-0.894088	...	-0.594944	-0.083495	0.017500	-0.434375
2	0.275485	-0.026050	-0.118152	-0.993819	-0.969926	-0.962748	-0.994403	-0.970735	-0.963483	-0.939260	...	-0.640736	-0.034956	0.202302	0.064103
3	0.270298	-0.032614	-0.117520	-0.994743	-0.973268	-0.967091	-0.995274	-0.974471	-0.968897	-0.938610	...	-0.736124	-0.017067	0.154438	0.340134
4	0.274833	-0.027848	-0.129527	-0.993852	-0.967445	-0.978295	-0.994111	-0.965953	-0.977346	-0.938610	...	-0.846595	-0.002223	-0.040046	0.736715
5	0.279220	-0.018620	-0.113902	-0.994455	-0.970417	-0.965316	-0.994585	-0.969481	-0.965897	-0.937856	...	-0.857565	-0.095681	0.048849	0.760684
6	0.279746	-0.018271	-0.104000	-0.995819	-0.976354	-0.977725	-0.995996	-0.973665	-0.979253	-0.937856	...	-0.391286	-0.309083	-0.000857	0.728514
7	0.274601	-0.025035	-0.116831	-0.995594	-0.982069	-0.985262	-0.995341	-0.981485	-0.984610	-0.941263	...	-0.672358	-0.025526	-0.214033	0.437640
8	0.272529	-0.020954	-0.114472	-0.996784	-0.975906	-0.986597	-0.997029	-0.973735	-0.985556	-0.941263	...	-0.853248	0.144626	-0.035564	-0.181800
9	0.275746	-0.010372	-0.099776	-0.998373	-0.986933	-0.991022	-0.998663	-0.987140	-0.991084	-0.943761	...	-0.741435	0.157584	-0.033487	-0.350893

10 rows × 563 columns

In [49]:
```python
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7352 entries, 0 to 7351
Columns: 563 entries, tBodyAcc-mean()-X to Activity
dtypes: float64(561), int64(1), object(1)
memory usage: 31.6+ MB
```

In [50]:
```python
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2947 entries, 0 to 2946
Columns: 563 entries, tBodyAcc-mean()-X to Activity
dtypes: float64(561), int64(1), object(1)
memory usage: 12.7+ MB
```

In [51]:
```python
train.isnull().sum()
```

Out[51]:
```
tBodyAcc-mean()-X       0
tBodyAcc-mean()-Y       0
tBodyAcc-mean()-Z       0
tBodyAcc-std()-X        0
tBodyAcc-std()-Y        0
                       ..
angle(X,gravityMean)    0
angle(Y,gravityMean)    0
angle(Z,gravityMean)    0
subject                 0
Activity                0
Length: 563, dtype: int64
```

In [52]:
```python
test.isnull().sum()
```

Out[52]:
```
tBodyAcc-mean()-X       0
tBodyAcc-mean()-Y       0
tBodyAcc-mean()-Z       0
tBodyAcc-std()-X        0
tBodyAcc-std()-Y        0
                       ..
angle(X,gravityMean)    0
angle(Y,gravityMean)    0
angle(Z,gravityMean)    0
subject                 0
Activity                0
Length: 563, dtype: int64
```

In [53]:
```python
train.describe()
```

Out[53]:

	tBodyAcc-mean()-X	tBodyAcc-mean()-Y	tBodyAcc-mean()-Z	tBodyAcc-std()-X	tBodyAcc-std()-Y	tBodyAcc-std()-Z	tBodyAcc-mad()-X	tBodyAcc-mad()-Y	tBodyAcc-mad()-Z	tBodyAcc-max()-X	...	fBodyBodyGyroJerkMag-skewness()	fBodyBodyGyroJerkMag-kurtosis()	angle(tBodyAccMean,gravity)
count	7352.000000	7352.000000	7352.000000	7352.000000	7352.000000	7352.000000	7352.000000	7352.000000	7352.000000	7352.000000	...	7352.000000	7352.000000	7352.000000
mean	0.274488	-0.017695	-0.109141	-0.605438	-0.510938	-0.604754	-0.630512	-0.526907	-0.606150	-0.468604	...	-0.307009	-0.625294	0.008684
std	0.070261	0.040811	0.056635	0.448734	0.502645	0.418687	0.424073	0.485942	0.414122	0.544547	...	0.321011	0.307584	0.336787
min	-1.000000	-1.000000	-1.000000	-1.000000	-0.999873	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	...	-0.995357	-0.999765	-0.976580
25%	0.262975	-0.024863	-0.120993	-0.992754	-0.978129	-0.980233	-0.993591	-0.978162	-0.980251	-0.936219	...	-0.542602	-0.845573	-0.121527
50%	0.277193	-0.017219	-0.108676	-0.946196	-0.851897	-0.859365	-0.950709	-0.857328	-0.857143	-0.881637	...	-0.343685	-0.711692	0.009509
75%	0.288461	-0.010783	-0.097794	-0.242813	-0.034231	-0.262415	-0.292680	-0.066701	-0.265671	-0.017129	...	-0.126979	-0.503878	0.150865
max	1.000000	1.000000	1.000000	1.000000	0.916238	1.000000	1.000000	0.967664	1.000000	1.000000	...	0.989538	0.956845	1.000000

8 rows × 562 columns

In [54]:
```python
test.describe()
```

Out[54]:

	tBodyAcc-mean()-X	tBodyAcc-mean()-Y	tBodyAcc-mean()-Z	tBodyAcc-std()-X	tBodyAcc-std()-Y	tBodyAcc-std()-Z	tBodyAcc-mad()-X	tBodyAcc-mad()-Y	tBodyAcc-mad()-Z	tBodyAcc-max()-X	...	fBodyBodyGyroJerkMag-skewness()	fBodyBodyGyroJerkMag-kurtosis()	angle(tBodyAccMean,gravity)
count	2947.000000	2947.000000	2947.000000	2947.000000	2947.000000	2947.000000	2947.000000	2947.000000	2947.000000	2947.000000	...	2947.000000	2947.000000	2947.000000
mean	0.273996	-0.017863	-0.108386	-0.613635	-0.508330	-0.633797	-0.641278	-0.522676	-0.637038	-0.462063	...	-0.277593	-0.598756	0.005264
std	0.060570	0.025745	0.042747	0.412597	0.494269	0.362699	0.385199	0.479899	0.357753	0.523916	...	0.317245	0.311042	0.336147
min	-0.592004	-0.362884	-0.576184	-0.999606	-1.000000	-0.998955	-0.999417	-0.999914	-0.998899	-0.952357	...	-1.000000	-1.000000	-1.000000
25%	0.262075	-0.024961	-0.121162	-0.990914	-0.973664	-0.976122	-0.992333	-0.974131	-0.975352	-0.934447	...	-0.517494	-0.829593	-0.130541
50%	0.277113	-0.016967	-0.108458	-0.931214	-0.790972	-0.827534	-0.937664	-0.799907	-0.817005	-0.852659	...	-0.311023	-0.683672	0.005188
75%	0.288097	-0.010143	-0.097123	-0.267395	-0.105919	-0.311432	-0.321719	-0.133488	-0.322771	-0.009965	...	-0.083559	-0.458332	0.146200
max	0.671887	0.246106	0.494114	0.465299	1.000000	0.489703	0.439657	1.000000	0.427958	0.786436	...	1.000000	1.000000	0.998898

8 rows × 562 columns

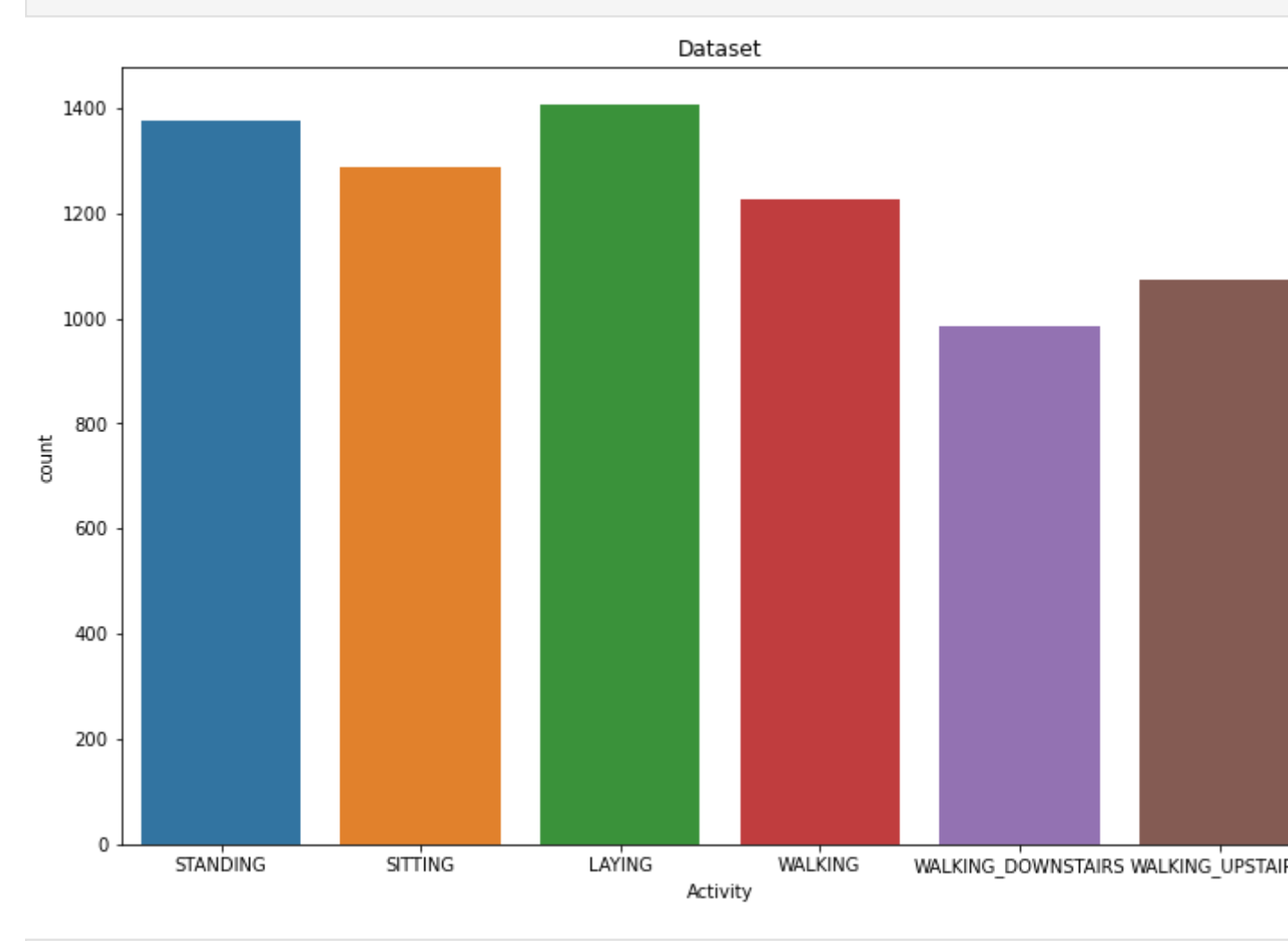
In [55]:
```python
train.shape
```

Out[55]: (7352, 563)

In [56]:
```python
test.shape
```

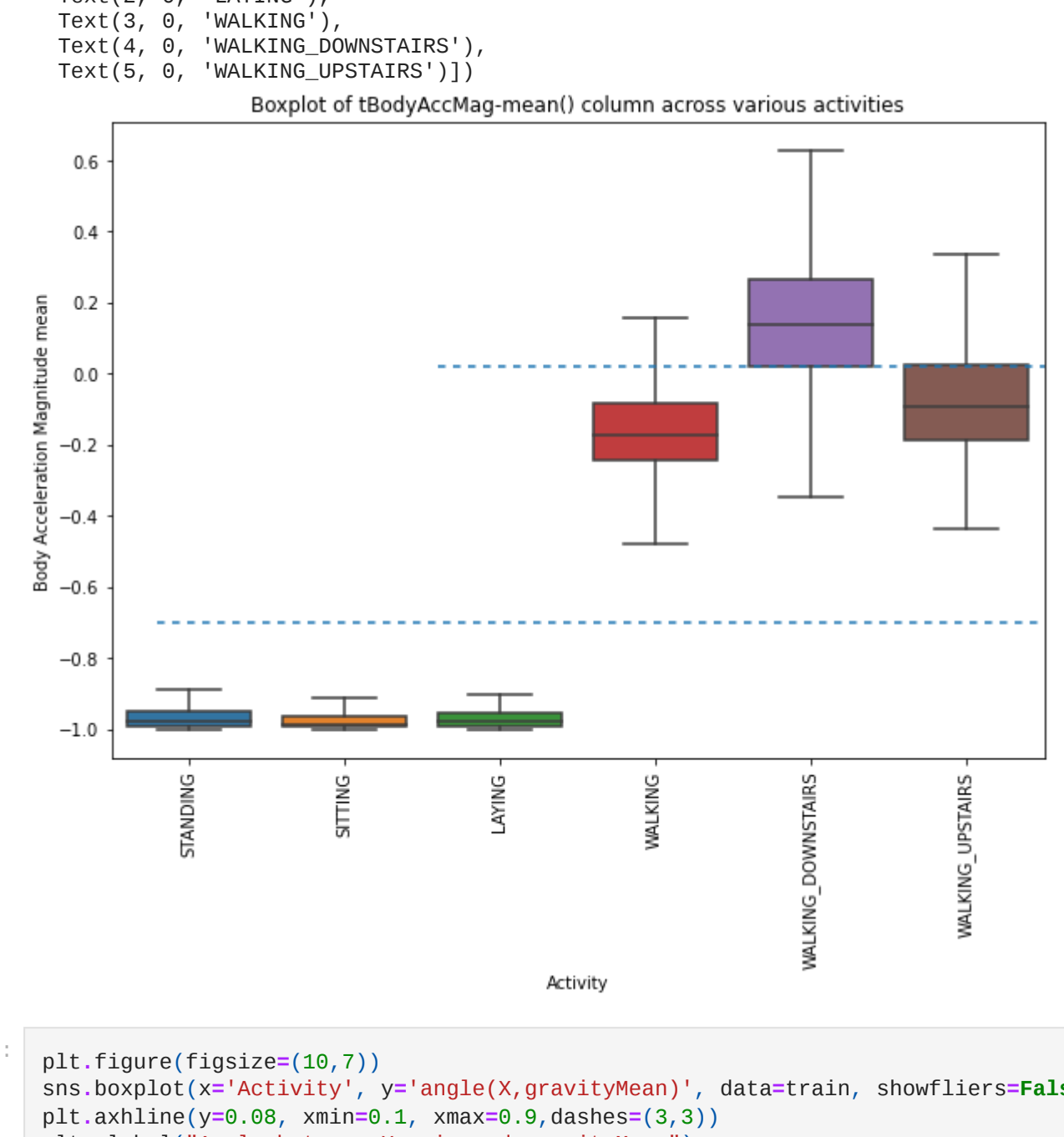
Out[56]: (2947, 563)

In [57]:
```python
#checking if there is imbalance in data
plt.figure(figsize=(12,8))
sns.countplot(train["Activity"])
plt.title("Dataset")
plt.show()
```



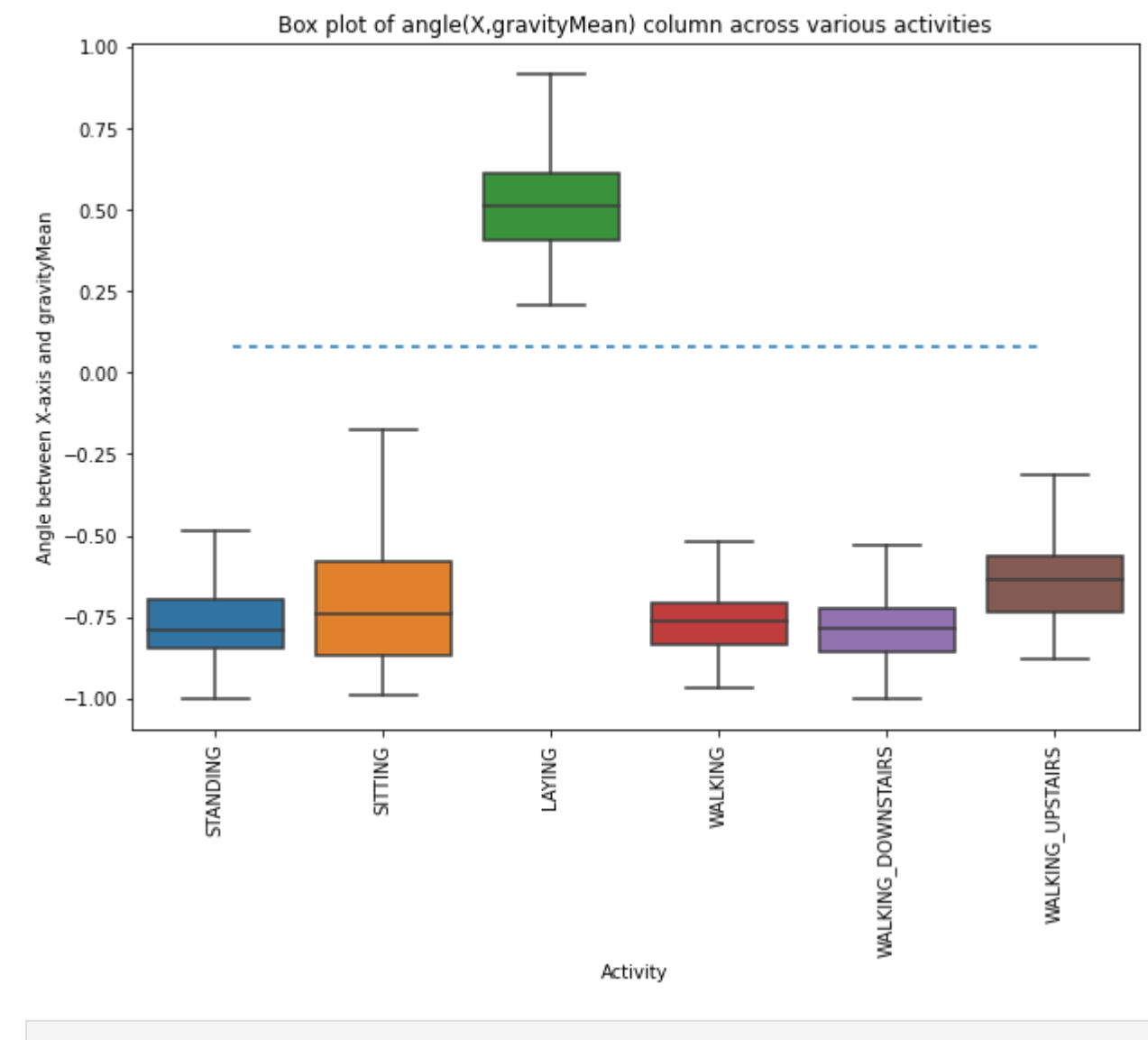
In [58]:
```python
plt.figure(figsize=(10,7))
sns.boxplot(x='Activity', y='tBodyAccMag-mean()',data=train, showfliers=False)
plt.ylabel('Body Acceleration Magnitude mean')
plt.title("Boxplot of tBodyAccMag-mean() column across various activities")
plt.axhline(y=-0.7, xmin=0.05,dashes=(3,3))
plt.axhline(y=0.020, xmin=0.35, dashes=(3,3))
plt.xticks(rotation=90)
```

Out[58]:
```
(array([0, 1, 2, 3, 4, 5]),
 [Text(0, 0, 'STANDING'),
  Text(1, 0, 'SITTING'),
  Text(2, 0, 'LAYING'),
  Text(3, 0, 'WALKING'),
  Text(4, 0, 'WALKING_DOWNSTAIRS'),
  Text(5, 0, 'WALKING_UPSTAIRS')])
```



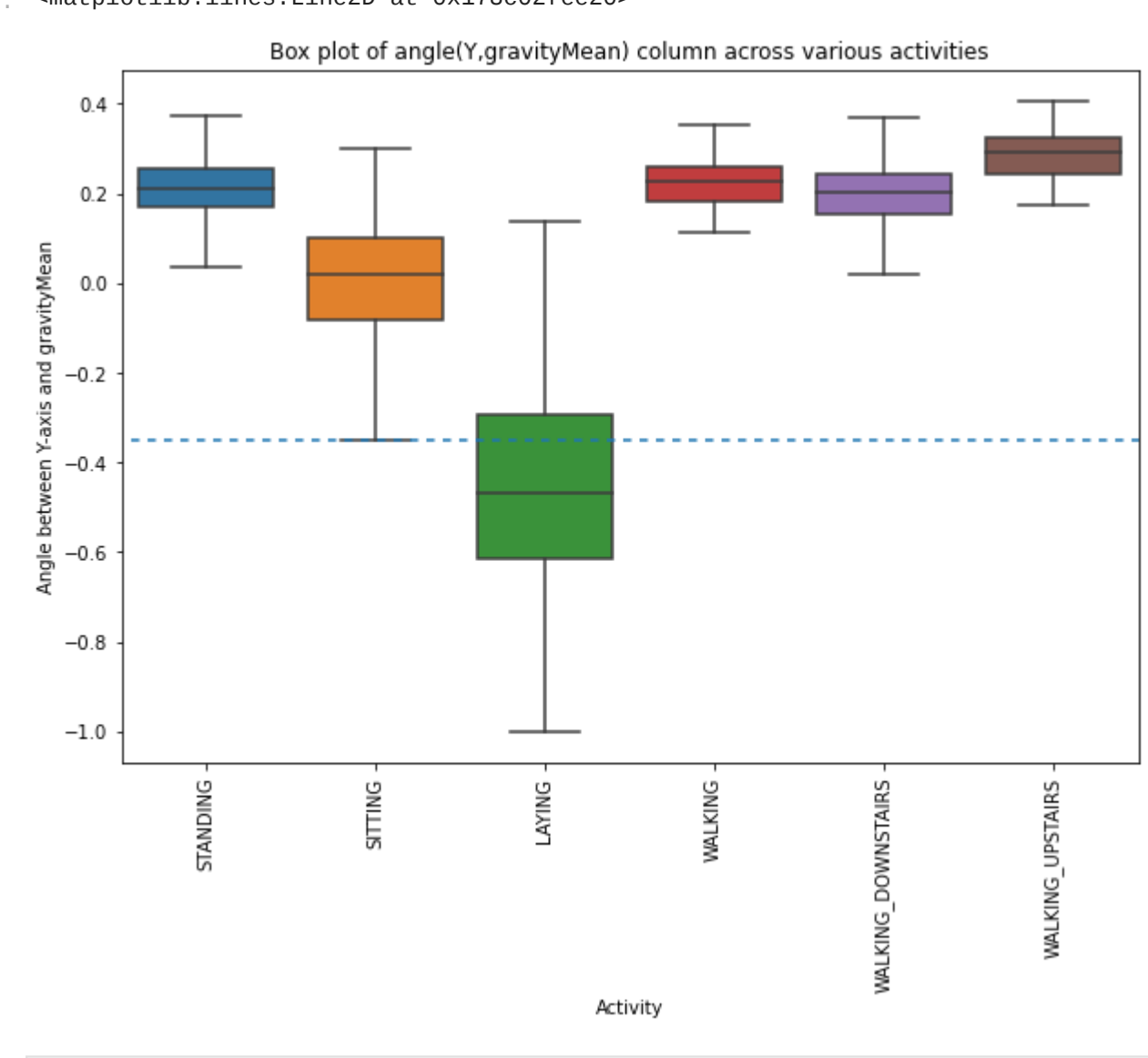
In [59]:
```python
plt.figure(figsize=(10,7))
sns.boxplot(x='Activity', y='angle(X,gravityMean)', data=train, showfliers=False)
plt.axhline(y=0.08, xmin=0.1, xmax=0.9,dashes=(3,3))
plt.ylabel("Angle between X-axis and gravityMean")
plt.title('Box plot of angle(X,gravityMean) column across various activities')
plt.xticks(rotation = 90)
```

Out[59]:
```
(array([0, 1, 2, 3, 4, 5]),
 [Text(0, 0, 'STANDING'),
  Text(1, 0, 'SITTING'),
  Text(2, 0, 'LAYING'),
  Text(3, 0, 'WALKING'),
  Text(4, 0, 'WALKING_DOWNSTAIRS'),
  Text(5, 0, 'WALKING_UPSTAIRS')])
```



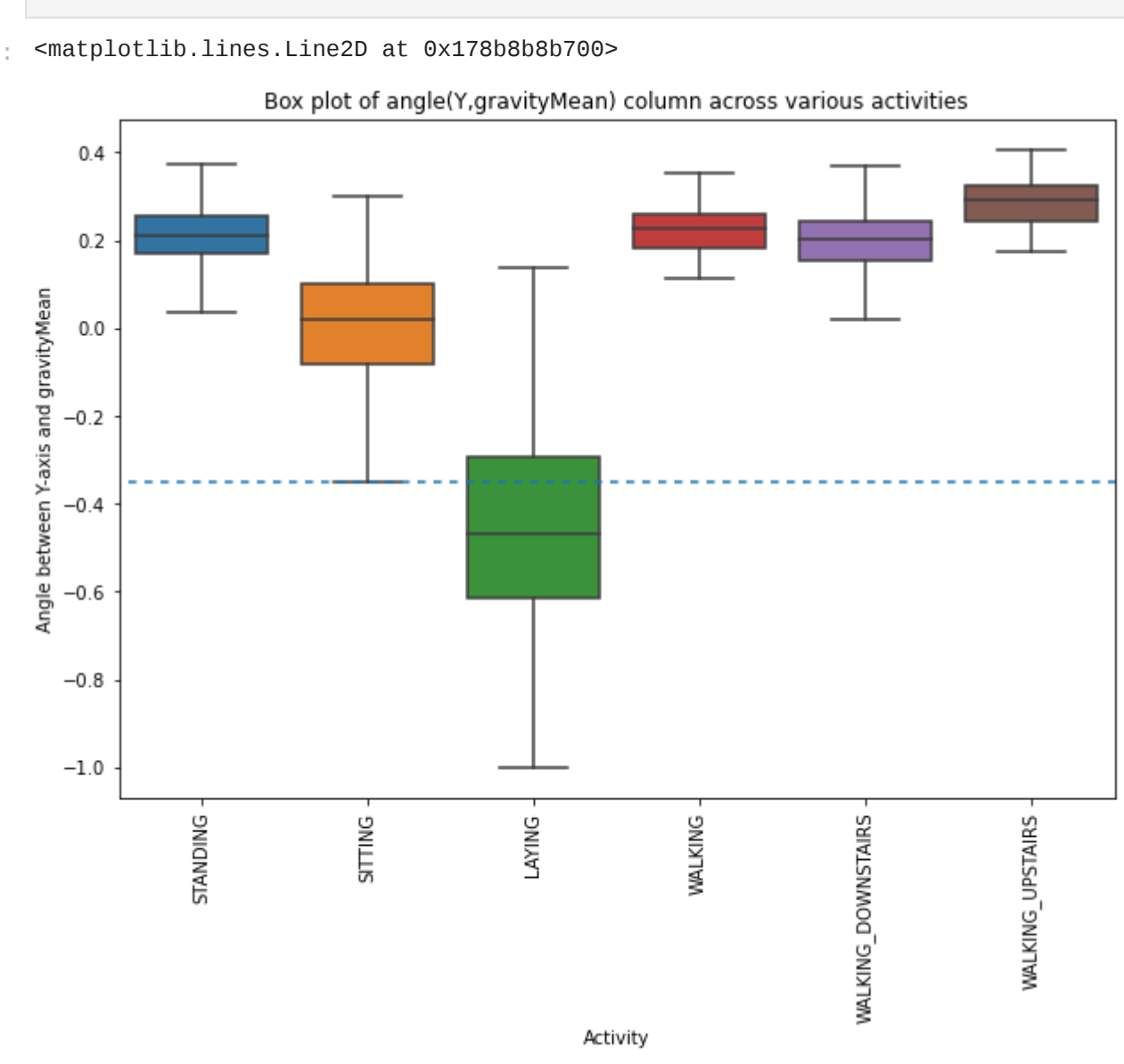
In [60]:
```python
plt.figure(figsize=(10,7))
sns.boxplot(x='Activity', y='angle(Y,gravityMean)', data = train, showfliers=False)
plt.ylabel("Angle between Y-axis and gravityMean")
plt.title('Box plot of angle(Y,gravityMean) column across various activities')
plt.xticks(rotation = 90)
plt.axhline(y=-0.35, xmin=0.01, dashes=(3,3))
```

Out[60]: <matplotlib.lines.Line2D at 0x178c02fee20>



In [61]:
```python
plt.figure(figsize=(10,7))
sns.boxplot(x='Activity', y='angle(Y,gravityMean)', data = train, showfliers=False)
plt.ylabel("Angle between Y-axis and gravityMean")
plt.title('Box plot of angle(Y,gravityMean) column across various activities')
plt.xticks(rotation = 90)
plt.axhline(y=-0.35, xmin=0.01, dashes=(3,3))
```

Out[61]: <matplotlib.lines.Line2D at 0x178b8b8b700>



In [62]:
```python
train.nunique()
```

Out[62]:
```
tBodyAcc-mean()-X       7347
tBodyAcc-mean()-Y       7352
tBodyAcc-mean()-Z       7349
tBodyAcc-std()-X        7349
tBodyAcc-std()-Y        7351
                        ...
angle(X,gravityMean)    7352
angle(Y,gravityMean)    7352
angle(Z,gravityMean)    7352
subject                   21
Activity                   6
Length: 563, dtype: int64
```

In [63]:
```python
test.nunique()
```

Out[63]:
```
tBodyAcc-mean()-X       2947
tBodyAcc-mean()-Y       2947
tBodyAcc-mean()-Z       2947
tBodyAcc-std()-X        2947
tBodyAcc-std()-Y        2947
                        ...
angle(X,gravityMean)    2947
angle(Y,gravityMean)    2947
angle(Z,gravityMean)    2947
subject                    9
Activity                   6
Length: 563, dtype: int64
```

In [64]:
```python
data=train.iloc[:,:-1].values
type(data)
```

Out[64]: numpy.ndarray

In [65]:
```python
cov=np.cov(data,rowvar=False)
```

In [66]:
```python
cov.shape
```

Out[66]: (562, 562)

In [67]:
```python
train["Activity"].unique()
```

Out[67]:
```
array(['STANDING', 'SITTING', 'LAYING', 'WALKING', 'WALKING_DOWNSTAIRS',
       'WALKING_UPSTAIRS'], dtype=object)
```

In [68]:
```python
train.groupby("Activity").count()
```

Out[68]:

Activity	tBodyAcc-mean()-X	tBodyAcc-mean()-Y	tBodyAcc-mean()-Z	tBodyAcc-std()-X	tBodyAcc-std()-Y	tBodyAcc-std()-Z	tBodyAcc-mad()-X	tBodyAcc-mad()-Y	tBodyAcc-mad()-Z	tBodyAcc-max()-X	...	fBodyBodyGyroJerkMag-skewness()	fBodyBodyGyroJerkMag-kurtosis()	angle(tBodyAccMean,gravity)
LAYING	1407	1407	1407	1407	1407	1407	1407	1407	1407	1407	...	1407	1407	1407
SITTING	1286	1286	1286	1286	1286	1286	1286	1286	1286	1286	...	1286	1286	1286
STANDING	1374	1374	1374	1374	1374	1374	1374	1374	1374	1374	...	1374	1374	1374
WALKING	1226	1226	1226	1226	1226	1226	1226	1226	1226	1226	...	1226	1226	1226
WALKING_DOWNSTAIRS	986	986	986	986	986	986	986	986	986	986	...	986	986	986
WALKING_UPSTAIRS	1073	1073	1073	1073	1073	1073	1073	1073	1073	1073	...	1073	1073	1073

6 rows × 562 columns

In [69]:
```python
from sklearn.manifold import TSNE
```

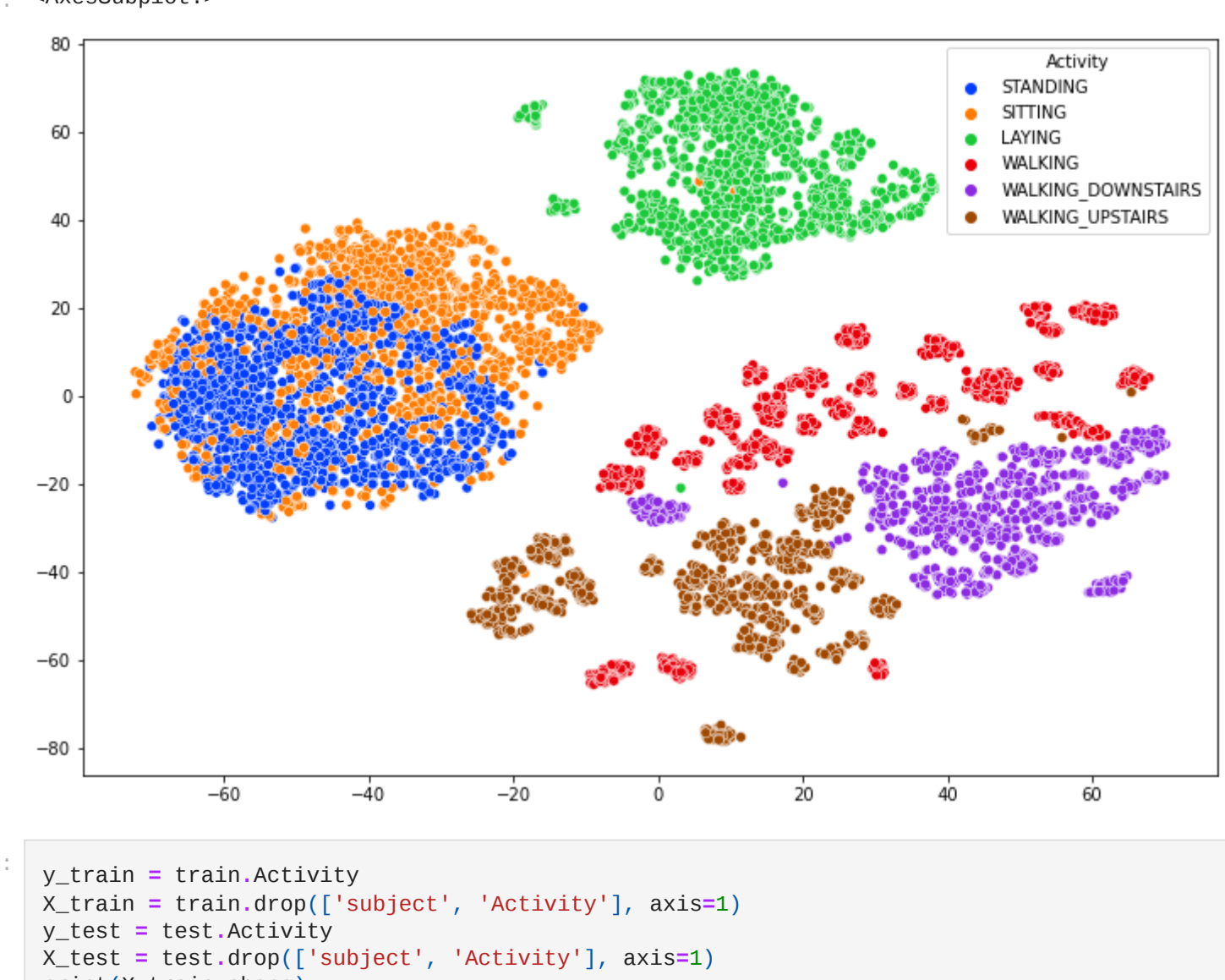
In [70]:
```python
X_for_tsne = train.drop(['subject', 'Activity'], axis=1)
```

In [71]:
```python
%time
tsne = TSNE(random_state = 42, n_components=2, verbose=1, perplexity=50, n_iter=1000).fit_transform(X_for_tsne)
```

```
Wall time: 0 ns
[t-SNE] Computing 151 nearest neighbors...
[t-SNE] Indexed 7352 samples in 0.004s...
[t-SNE] Computed neighbors for 7352 samples in 1.201s...
[t-SNE] Computed conditional probabilities for sample 1000 / 7352
[t-SNE] Computed conditional probabilities for sample 2000 / 7352
[t-SNE] Computed conditional probabilities for sample 3000 / 7352
[t-SNE] Computed conditional probabilities for sample 4000 / 7352
[t-SNE] Computed conditional probabilities for sample 5000 / 7352
[t-SNE] Computed conditional probabilities for sample 6000 / 7352
[t-SNE] Computed conditional probabilities for sample 7000 / 7352
[t-SNE] Computed conditional probabilities for sample 7352 / 7352
[t-SNE] Mean sigma: 1.437672
[t-SNE] KL divergence after 250 iterations with early exaggeration: 74.125374
[t-SNE] KL divergence after 1000 iterations: 1.282858
```

In [72]:
```python
plt.figure(figsize=(12,8))
sns.scatterplot(x =tsne[:, 0], y = tsne[:, 1], hue = train["Activity"],palette="bright")
```

Out[72]: <AxesSubplot:>



In [73]:
```python
y_train = train.Activity
X_train = train.drop(['subject', 'Activity'], axis=1)
y_test = test.Activity
X_test = test.drop(['subject', 'Activity'], axis=1)
print(X_train.shape)
print(X_test.shape)
```

```
(7352, 561)
(2947, 561)
```

In [74]:
```python
y_train
```

Out[74]:
```
0               STANDING
1               STANDING
2               STANDING
3               STANDING
4               STANDING
              ...
7347    WALKING_UPSTAIRS
7348    WALKING_UPSTAIRS
7349    WALKING_UPSTAIRS
7350    WALKING_UPSTAIRS
7351    WALKING_UPSTAIRS
Name: Activity, Length: 7352, dtype: object
```

In [75]:
```python
#lets try regression model with cross validation and hyperparameter tuning
```

In [76]:
```python
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import RandomizedSearchCV
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score,classification_report
```

In [77]:
```python
parameters = {'C':np.arange(10,30,10), 'penalty':['l2','l1']}
lr_classifier = LogisticRegression()
lr_classifier_rs = RandomizedSearchCV(lr_classifier,param_distributions=parameters,cv=5,random_state=42)
lr_classifier_rs.fit(X_train, y_train)
y_pred = lr_classifier_rs.predict(X_test)
```

In [78]:
```python
y_pred
```

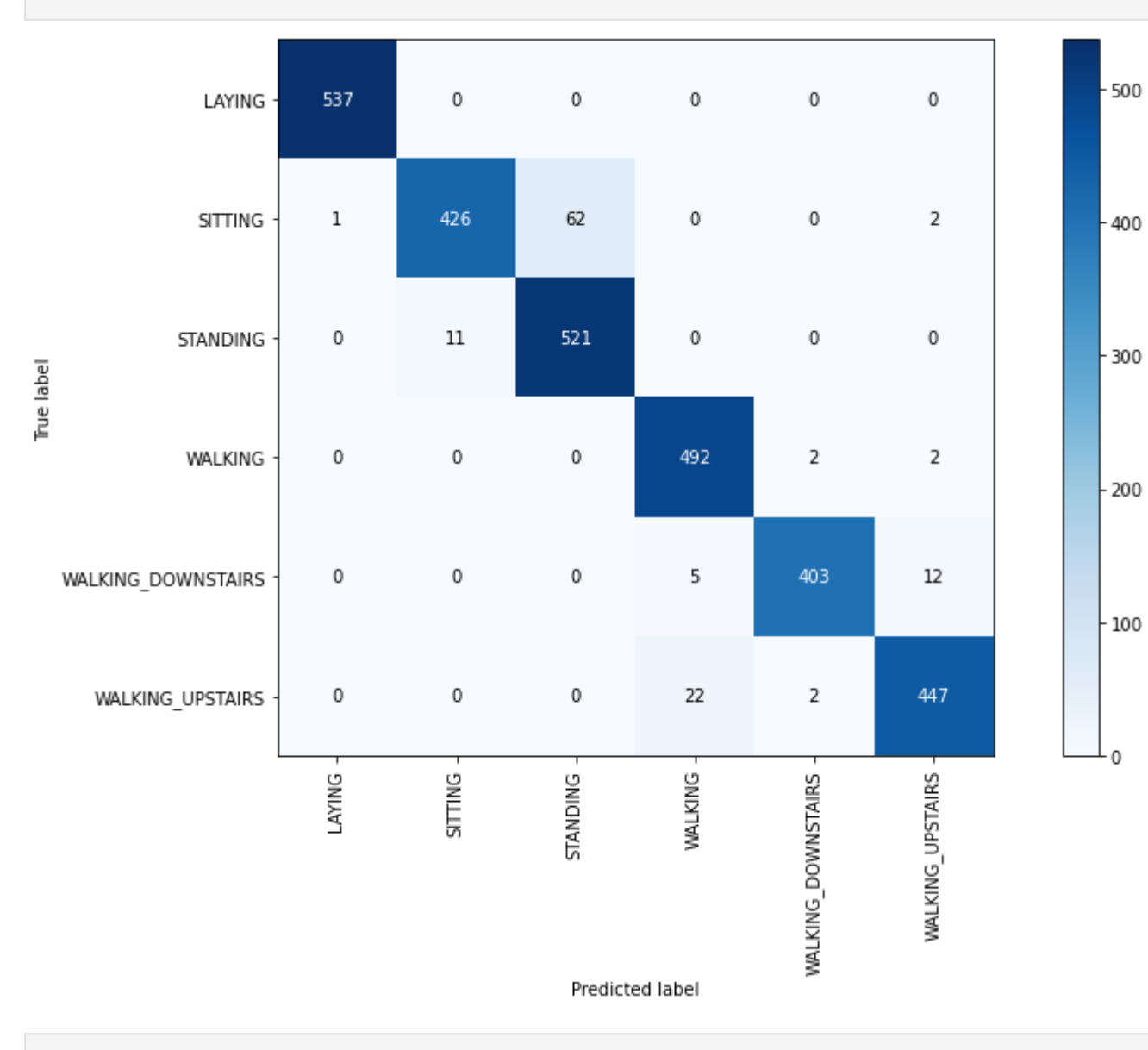
Out[78]:
```
array(['STANDING', 'STANDING', 'STANDING', ..., 'WALKING_UPSTAIRS',
       'WALKING_UPSTAIRS', 'WALKING_UPSTAIRS'], dtype=object)
```

In [79]:
```python
lr_accuracy = accuracy_score(y_true=y_test, y_pred=y_pred)
print("Accuracy using Logistic Regression : ", lr_accuracy)
```

```
Accuracy using Logistic Regression :  0.9589412962334578
```

In [80]:
```python
# function to plot confusion matrix
def plot_confusion_matrix(cm,lables):
    fig, ax = plt.subplots(figsize=(12,8)) # for plotting confusion matrix as image
    im = ax.imshow(cm, interpolation='nearest', cmap=plt.cm.Blues)
    ax.figure.colorbar(im, ax=ax)
    ax.set(xticks=np.arange(cm.shape[1]),
    yticks=np.arange(cm.shape[0]),
    xticklabels=lables, yticklabels=lables,
    ylabel='True label',
    xlabel='Predicted label')
    plt.xticks(rotation = 90)
    thresh = cm.max() / 2.
    for i in range(cm.shape[0]):
        for j in range(cm.shape[1]):
            ax.text(j, i, int(cm[i, j]),ha="center", va="center",color="white" if cm[i, j] > thresh else "black")
    fig.tight_layout()
```

In [81]:
```python
cm = confusion_matrix(y_test.values,y_pred)
plot_confusion_matrix(cm, np.unique(y_pred))
```



In [82]:
```python
#function to get best random search attributes
def get_best_randomsearch_results(model):
    print("Best estimator : ", model.best_estimator_)
    print("Best set of parameters : ", model.best_params_)
    print("Best score : ", model.best_score_)
```

In [83]:
```python
get_best_randomsearch_results(lr_classifier_rs)
```

```
Best estimator :  LogisticRegression(C=10)
Best set of parameters :  {'penalty': 'l2', 'C': 10}
Best score :  0.9302313665098942
```

In [84]:
```python
!pip install hmmlearn
```

```
Requirement already satisfied: hmmlearn in c:\users\kiratpreet kaur\anaconda3\lib\site-packages (0.2.6)
Requirement already satisfied: scipy>=0.19 in c:\users\kiratpreet kaur\anaconda3\lib\site-packages (from hmmlearn) (1.6.2)
Requirement already satisfied: numpy>=1.10 in c:\users\kiratpreet kaur\anaconda3\lib\site-packages (from hmmlearn) (1.20.1)
Requirement already satisfied: scikit-learn>=0.16 in c:\users\kiratpreet kaur\anaconda3\lib\site-packages (from hmmlearn) (0.24.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\kiratpreet kaur\anaconda3\lib\site-packages (from scikit-learn>=0.16->hmmlearn) (2.1.0)
Requirement already satisfied: joblib>=0.11 in c:\users\kiratpreet kaur\anaconda3\lib\site-packages (from scikit-learn>=0.16->hmmlearn) (1.0.1)
```

In [85]:
```python
from hmmlearn import hmm
model = hmm.GaussianHMM(n_components=2, covariance_type="full", n_iter=100)
```

In [86]:
```python
model.fit(X_train)
```

Out[86]: GaussianHMM(covariance_type='full', n_components=2, n_iter=100)

In [ ]: