In [38]:
```
# Import Libraries
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler,OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix,accuracy_score,recall_scor
e,f1_score
```

In [2]:
```
# import Dataset
df = pd.read_csv("C:\\Users\\91876\\Pictures\\Camera Roll\\british lib
\\Time Series\\Human activity recognition\\test.csv")
```

In [3]:
```
# Check the Header
df.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
0	0.257178	-0.023285	-0.014654	-0.938404	-0.920091	-0.667683	-0.952501	-0.925249
1	0.286027	-0.013163	-0.119083	-0.975415	-0.967458	-0.944958	-0.986799	-0.968401
2	0.275485	-0.026050	-0.118152	-0.993819	-0.969926	-0.962748	-0.994403	-0.970735
3	0.270298	-0.032614	-0.117520	-0.994743	-0.973268	-0.967091	-0.995274	-0.974471
4	0.274833	-0.027848	-0.129527	-0.993852	-0.967445	-0.978295	-0.994111	-0.965953
5	0.279220	-0.018620	-0.113902	-0.994455	-0.970417	-0.965316	-0.994585	-0.969481
6	0.279746	-0.018271	-0.104000	-0.995819	-0.976354	-0.977725	-0.995996	-0.973665

	tBodyAcc-mean()-X	tBodyAcc-mean()-Y	tBodyAcc-mean()-Z	tBodyAcc-std()-X	tBodyAcc-std()-Y	tBodyAcc-std()-Z	tBodyAcc-mad()-X	tBodyAcc-mad()-Y
7	0.274601	-0.025035	-0.116831	-0.995594	-0.982069	-0.985262	-0.995341	-0.981485
8	0.272529	-0.020954	-0.114472	-0.996784	-0.975906	-0.986597	-0.997029	-0.973735
9	0.275746	-0.010372	-0.099776	-0.998373	-0.986933	-0.991022	-0.998663	-0.987140

10 rows × 563 columns

In [4]:
```
# Check the relationship between the variables
df.corr()
```

Out[4]:

	tBodyAcc-mean()-X	tBodyAcc-mean()-Y	tBodyAcc-mean()-Z	tBodyAcc-std()-X	tBodyAcc-std()-Y
tBodyAcc-mean()-X	1.000000	0.041274	-0.129645	0.016984	-0.001799
tBodyAcc-mean()-Y	0.041274	1.000000	0.225980	-0.054264	-0.059066
tBodyAcc-mean()-Z	-0.129645	0.225980	1.000000	-0.038578	-0.048340
tBodyAcc-std()-X	0.016984	-0.054264	-0.038578	1.000000	0.910636
tBodyAcc-std()-Y	-0.001799	-0.059066	-0.048340	0.910636	1.000000
...	...	...	...	...	...
angle(tBodyGyroJerkMean,gravityMean)	0.049701	0.092905	-0.021375	-0.033609	-0.018611
angle(X,gravityMean)	-0.058421	-0.017138	-0.013933	-0.382696	-0.383742
angle(Y,gravityMean)	0.034220	-0.030253	-0.007318	0.401433	0.467572
angle(Z,gravityMean)	0.038936	-0.027410	-0.051057	0.388747	0.405681
subject	0.005077	0.003163	0.021476	-0.068487	-0.036466

562 rows × 562 columns

In [5]:
```
# Check for any null values
```

```
df.isnull().sum()
```

Out[5]:
```
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 [6]:
```
df.info()
```

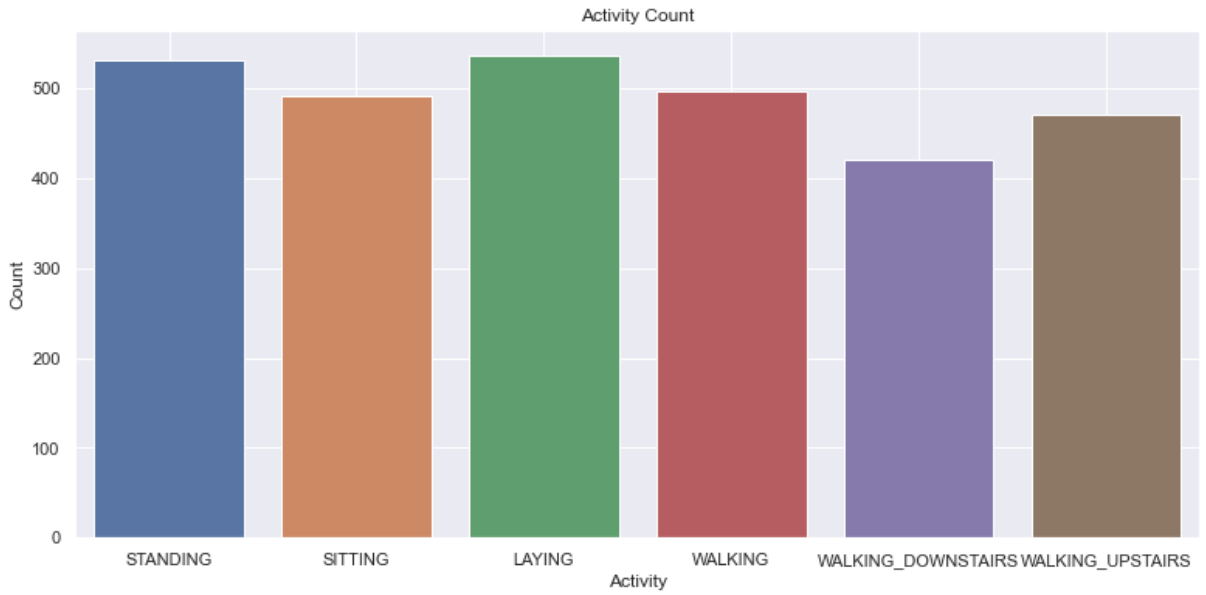
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2947 entries, 0 to 2946
Columns: 563 entries, tBodyAcc-mean()-X to Activity
dtypes: float64(561), int64(1), object(1)
memory usage: 12.7+ MB
```

In [7]:
```
# check for the categoris in the reponse variable
df["Activity"].value_counts()
```

Out[7]:
```
LAYING                537
STANDING              532
WALKING               496
SITTING               491
WALKING_UPSTAIRS      471
WALKING_DOWNSTAIRS    420
Name: Activity, dtype: int64
```

In [8]:
```
#check for any imbalance in response variable
sns.set(rc={'figure.figsize':(13,6)})
fig = sns.countplot(x = "Activity" , data = df)
plt.xlabel("Activity")
plt.ylabel("Count")
```

```
plt.title("Activity Count")
plt.grid(True)
plt.show(fig)
```



In [9]:
```
# Split the dataset into train and test
X = df.drop(["Activity"], axis= 1)
y = df["Activity"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2
,random_state=44)
```

In [10]:
```
# Transformation of the data
from sklearn import preprocessing

encoder = preprocessing.LabelEncoder()

encoder.fit(y_train)
y_train = encoder.transform(y_train)

# encoding test labels
encoder.fit(y_test)
```

```
y_test = encoder.transform(y_test)


scaler = StandardScaler()
X_train= scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [11]:
```
classifier = LogisticRegression(random_state = 0,max_iter=1000)
classifier.fit(X_train, y_train)
```

Out[11]:
```
LogisticRegression(max_iter=1000, random_state=0)
```

In [12]:
```
#After training the model, it time to use it to do prediction on testin
g data.
y_pred = classifier.predict(X_test)
```

In [13]:
```
#Let’s test the performance of our model – Confusion Matrix

cm = confusion_matrix(y_test, y_pred)
```

In [41]:
```
#Let’s test the performance of our model – Confusion Matrix
print ("confusion_matrix : \n", cm)
```

```
confusion_matrix :
 [[107   0   0   0   0   0]
 [  0  93   4   0   0   0]
 [  0   6 109   0   0   0]
 [  0   0   0 107   1   0]
 [  0   0   0   0  63   0]
 [  0   0   0   0   0 100]]
```

In [42]:
```
# check the performance matrics
from sklearn.metrics import accuracy_score
print ("Accuracy : ", accuracy_score(y_test, y_pred))
```

```
Accuracy :  0.9813559322033898
```

In [43]:
```
# check the performance matrics
accuracy_score=accuracy_score(y_test,y_pred)
recall_score=recall_score(y_test,y_pred,average='weighted')
f1_score=f1_score(y_test,y_pred,average='weighted')
print(y_pred)
print(cm)
print(accuracy_score)
print(recall_score)
print(f1_score)
```

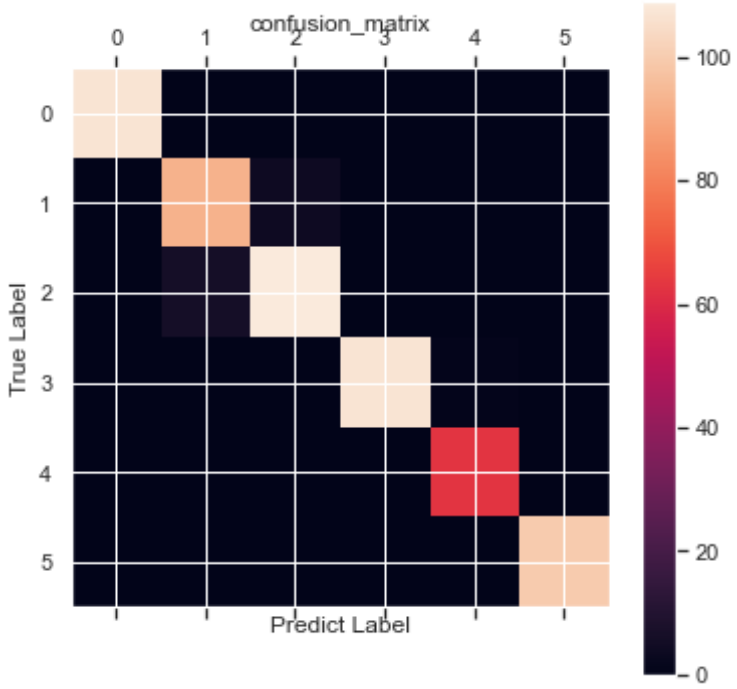
```
[3 0 5 2 2 2 3 1 1 5 5 2 0 0 0 0 3 3 3 5 4 5 0 1 2 4 3 1 5 3 3 5 4 4 5
0 4
 0 1 5 1 2 5 4 3 4 0 3 3 0 0 3 2 3 2 1 2 3 5 3 0 1 1 5 1 3 2 5 4 1 1 1
5 0
 5 0 0 5 5 1 2 4 5 1 0 4 5 5 2 5 0 1 2 5 0 1 2 1 3 5 0 0 1 5 5 3 2 5 2
5 0
 1 1 5 3 5 1 2 5 4 4 3 1 2 5 1 1 1 4 3 1 2 5 3 2 3 5 0 3 0 4 3 4 5 2 3
0 0
 2 5 1 0 2 0 3 2 3 1 1 5 0 1 5 1 3 4 2 2 0 0 4 1 1 4 3 0 0 0 2 5 3 0 1
1 2
 2 3 2 0 3 2 2 5 2 0 3 5 0 1 5 3 3 5 3 0 1 3 4 0 4 0 1 1 0 3 0 4 2 0 0
2 3
 0 1 1 0 3 3 1 0 1 0 4 4 1 0 2 2 3 3 0 2 2 0 1 3 1 2 2 4 3 2 2 2 5 1 0
1 2
 2 5 5 0 3 2 3 5 0 1 0 4 1 0 2 4 2 3 2 1 4 1 4 5 5 3 1 1 0 5 5 2 2 5 0
1 3
 0 1 3 1 1 2 5 0 2 3 0 3 2 3 5 4 0 1 1 2 2 5 4 4 0 2 4 3 2 0 2 0 2 2 1
1 4
 4 3 3 0 2 0 0 1 0 2 2 3 0 3 2 5 5 2 5 2 5 4 0 1 4 1 3 1 1 3 4 3 2 0 5
5 5
 0 4 3 0 0 2 3 3 3 5 4 3 0 1 5 2 5 4 0 0 1 2 1 0 5 1 2 1 0 3 2 5 5 3 1
4 5
 2 5 2 3 0 1 0 0 5 4 2 0 5 5 4 4 3 2 0 3 2 5 5 2 0 5 2 1 3 1 1 0 2 5 1
4 1
 5 1 1 2 4 5 3 5 5 3 3 3 0 1 1 3 3 4 5 4 2 4 2 3 3 0 2 3 2 0 2 2 3 3 0
2 3
 5 0 3 4 1 1 2 4 5 2 2 3 0 2 2 2 2 4 0 3 1 4 2 3 1 3 1 5 3 4 3 5 0 4 5
5 2
 0 1 2 4 5 0 3 3 1 0 2 0 0 1 2 2 4 2 2 0 5 5 0 0 4 5 5 3 5 3 0 3 2 3 1
5 2
```

```
 2 5 4 2 2 5 4 0 3 5 1 1 1 1 0 2 5 5 3 2 3 3 4 2 0 3 3 5 3 4 1 3 4 4 1]
[[107   0   0   0   0   0]
 [  0  93   4   0   0   0]
 [  0   6 109   0   0   0]
 [  0   0   0 107   1   0]
 [  0   0   0   0  63   0]
 [  0   0   0   0   0 100]]
0.9813559322033898
0.9813559322033898
0.9813708003144391
```

In [45]:

```
# plot the confusion_matrix
y_test= y_test
y_pred = y_pred
confusion_matrix = confusion_matrix(y_test,y_pred)
print(confusion_matrix)
plt.matshow(confusion_matrix)
plt.title("confusion_matrix")
plt.colorbar()
plt.ylabel('True Label')
plt.xlabel("Predict Label")
plt.show()
```

```
[[107   0   0   0   0   0]
 [  0  93   4   0   0   0]
 [  0   6 109   0   0   0]
 [  0   0   0 107   1   0]
 [  0   0   0   0  63   0]
 [  0   0   0   0   0 100]]
```



In [ ]: