NAME	**:** Harini sri.S
EMAIL	**:** harinisrisanjeevi@gmail.com
EDUCATION	**:** Integrated MSc AIML (currently 3rd year)
COLLEGE	**:** Coimbatore Institute of Technology

AIM :

To develop a model to classify 9 activities of cow using 9 axis IMU dataset.

1. EATING
2. DRINKING
3. WALKING
4. STANDING
5. LYING
6. RUMINATING STANDING
7. RUMINATING LYING
8. GROOMING
9. IDLE/OTHER

STEP 1 :

Importing all the necessary libraries.

IMPORTING THE NECESSARY LIBRARIES

```
[8] import numpy as np
    import pandas as pd
    import seaborn as sns
    import sklearn as sk
    import matplotlib.pyplot as plt
    from sklearn.metrics import confusion_matrix
    from sklearn. preprocessing import StandardScaler
    from sklearn.model_selection import train_test_split
```

STEP 2 :

Importing the datasets.

IMPORTING THE DATASETS

```
[9] d1 = pd.read_csv('RL7_train.csv')
    d2 = pd.read_csv('D2_train.csv')
    d3 = pd.read_csv('E1_train.csv')
    d4 = pd.read_csv('G8_train.csv')
    d5 = pd.read_csv('I9_train.csv')
    d6 = pd.read_csv('L5_train.csv')
    d7 = pd.read_csv('RS6_train.csv')
    d8 = pd.read_csv('S4_train.csv')
    d9= pd.read_csv('W3_train.csv')
```

STEP 3 :

Merging all the 9 datasets into single dataset.

MERGING ALL 9 DATASETS

```
[10] dataset = pd.concat([d1,d2,d3,d4,d5,d6,d7,d8,d9], axis=0)
```

```
dataset.head()
```

	time	acc_x	acc_y	acc_z	gyr_x	gyr_y	gyr_z	mag_x	mag_y	mag_z	label
0	1628327640	-0.292480	0.950684	-0.017578	-3.112793	-0.732422	-2.441406	-198.0	1359.0	579.0	7.0
1	1628327640	-0.288086	0.929199	-0.014160	-2.807617	-0.061035	-1.953125	-207.0	1341.0	568.5	7.0
2	1628327640	-0.294434	0.923340	-0.006348	-2.197266	1.708984	-0.610352	-211.5	1344.0	571.5	7.0
3	1628327640	-0.306641	0.922852	-0.010742	-0.549316	0.610352	-0.854492	-201.0	1351.5	564.0	7.0
4	1628327640	-0.315918	0.925293	0.003906	1.220703	15.380859	-0.305176	-198.0	1369.5	580.5	7.0

```
dataset=dataset.drop(columns=['time'])
```

```
dataset.head()
```

	acc_x	acc_y	acc_z	gyr_x	gyr_y	gyr_z	mag_x	mag_y	mag_z	label
0	-0.292480	0.950684	-0.017578	-3.112793	-0.732422	-2.441406	-198.0	1359.0	579.0	7.0
1	-0.288086	0.929199	-0.014160	-2.807617	-0.061035	-1.953125	-207.0	1341.0	568.5	7.0
2	-0.294434	0.923340	-0.006348	-2.197266	1.708984	-0.610352	-211.5	1344.0	571.5	7.0
3	-0.306641	0.922852	-0.010742	-0.549316	0.610352	-0.854492	-201.0	1351.5	564.0	7.0
4	-0.315918	0.925293	0.003906	1.220703	15.380859	-0.305176	-198.0	1369.5	580.5	7.0

STEP 4 :

Total number of rows = 5548032

Total number of culumns = 10

SHAPE OF DATSET

```
[43] dataset.shape
```

```
(5548032, 10)
```

STEP 5 :

Checking for NULL values in the dataset.

CHECKING FOR NULL VALUES

```
[14] dataset.isnull().sum()
```

```
time      0
acc_x     1
acc_y     2
acc_z     3
gyr_x     3
gyr_y     3
gyr_z     4
mag_x     5
mag_y     5
mag_z     6
label     7
dtype: int64
```

STEP 6 :

Replacing the Null values and dropping nan values

REPLACING NULL VALUES AND DROPPING NAN VALUES

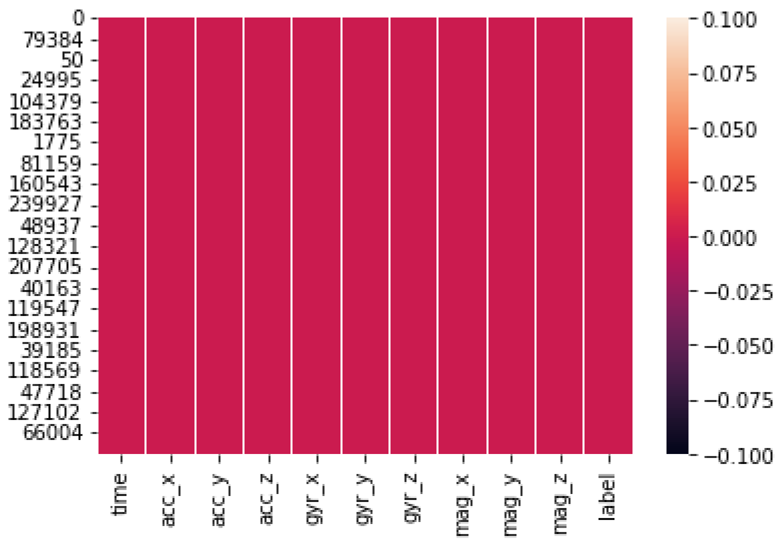
```
[15] dataset.replace([np.inf, -np.inf], np.nan, inplace=True)
     dataset = dataset.dropna()
     dataset.isnull().sum()
```

```
time      0
acc_x     0
acc_y     0
acc_z     0
gyr_x     0
gyr_y     0
gyr_z     0
mag_x     0
mag_y     0
mag_z     0
label     0
dtype: int64
```

BEFORE REMOVING NULL VALUES	AFTER REMOVING NULL VALUES
time 0 acc_x 1 acc_y 2 acc_z 3 gyr_x 3 gyr_y 3 gyr_z 4 mag_x 5 mag_y 5 mag_z 6 label 7 dtype: int64	time 0 acc_x 0 acc_y 0 acc_z 0 gyr_x 0 gyr_y 0 gyr_z 0 mag_x 0 mag_y 0 mag_z 0 label 0 dtype: int64

```
sns.heatmap(dataset.isnull())
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc8d66eae50>
```



COLUMNS

```
[17] dataset.columns

     Index(['time', 'acc_x', 'acc_y', 'acc_z', 'gyr_x', 'gyr_y', 'gyr_z', 'mag_x',
            'mag_y', 'mag_z', 'label'],
           dtype='object')
```

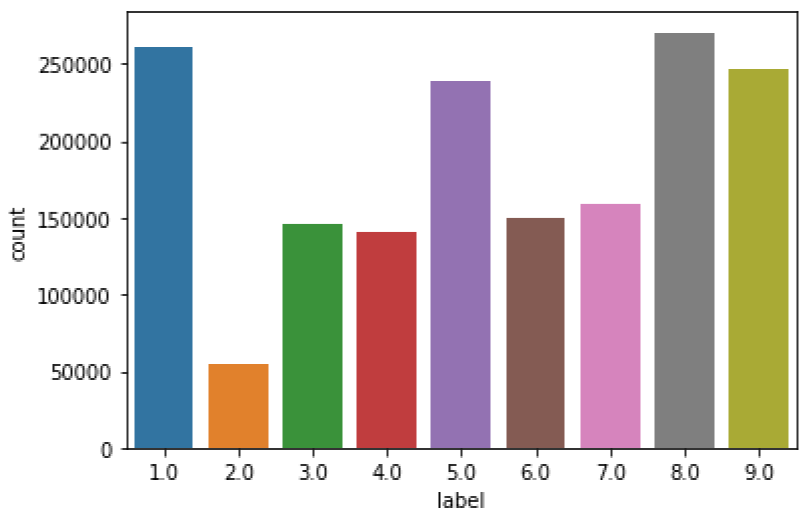
STEP 7 :

Visualizing the count of each values in “label” attribute.

VISUALIZING THE LABEL ATTRIBUTE

```
[18] sns.countplot(x="label",data=dataset)

     <matplotlib.axes._subplots.AxesSubplot at 0x7fc8cd2ab410>
```



CHECKING FOR UNIQUE VALUES IN LABEL COLUMN

```
[20] print('values in Label column:',dataset['label'].nunique())
     print('values:',dataset['label'].unique())
```

```
values in Label column: 9
values: [7. 2. 1. 8. 9. 5. 6. 4. 3.]
```

STEP 8 :

Finding out the numerical and categorical data in dataset.

There are no categorical data. So there is no need for encoding.

CHECKING FOR CATEGORICAL COLUMN

```
[19] def value_type(dataset):
       categorical=[]
       numerical=[]
       for i in dataset.columns:
         if dataset[i].dtype == 'object':
           categorical.append(i)
         else:
           numerical.append(i)
       return categorical,numerical

     category,numerical=value_type(dataset)
     print('columns with categorical values:',category)
     print('columns with numerical values:',numerical)
```

```
columns with categorical values: []
columns with numerical values: ['time', 'acc_x', 'acc_y', 'acc_z', 'gyr_x', 'gyr_y', 'gyr_z', 'mag_x', 'mag_y', 'mag_z', ']
```

STEP 9 :

Splitting the dependant variable and independent variable as X and y

SPLITTING THE DEPENDANT AND INDEPENDANT VARIABLE

```
[21] X = dataset.iloc[:, :-1].values
     y = dataset.iloc[:, -1].values
```

STEP 10 :

Splitting the dataset into training set and test set .

30 percent of dataset is used as test set and the remaining 70 percent is used as training set.

SPLITTING THE TRAINING AND TEST DATA

```
[22] from sklearn.model_selection import train_test_split
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30, random_state = 0)
```

STEP 11 :

Performing feature scaling to scale the values in particular range.

FEATURE SCALING

```
[23] from sklearn.preprocessing import StandardScaler
     sc = StandardScaler()
     X_train = sc.fit_transform(X_train)
     X_test = sc.transform(X_test)
```

STEP 12 :

Applying Random forest classifier with n_estimaters = 4 and entropy as criterion

APPLYING RANDOM FOREST CLASSIFIER

```
[24] from sklearn.ensemble import RandomForestClassifier
     classifier = RandomForestClassifier(n_estimators = 4, criterion = 'entropy', random_state = 0)
     classifier.fit(X_train, y_train)
```

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='entropy', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=4,
                       n_jobs=None, oob_score=False, random_state=0, verbose=0,
                       warm_start=False)
```

```
[25] y_pred = classifier.predict(X_test)
```

CONFUSION MATRIX AND ACCURACY SCORE

```
[51] from sklearn.metrics import confusion_matrix, accuracy_score
     cm = confusion_matrix(y_test, y_pred)
     print(cm)
     accuracy_score(y_test, y_pred)
```

```
[[283219    135    230   4924     10   1434     52   4161      2]
 [   415  15092    234    242     15    132      9     74      4]
 [  1186    350  35881   2230    312   1782    293   1674     34]
 [ 12490    233   1645 220630    609   9702   1404   6121     17]
 [    74     43    528   1454 269061    274   9687    322    200]
 [  3150    175   1581  19706    202 238905    450   4715     21]
 [   141     27    372   2029  15095    803 255301    516    147]
 [  9299     94   2203  12987    581  12412    842  72265     61]
 [    14      4    143     68    588    108    459    145 120181]]
0.9075498224596104
```

CLASSIFICATION REPORT

```
[52] from sklearn.metrics import classification_report
     print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

         1.0       0.91      0.96      0.94    294167
         2.0       0.93      0.93      0.93     16217
         3.0       0.84      0.82      0.83     43742
         4.0       0.83      0.87      0.85    252851
         5.0       0.94      0.96      0.95    281643
         6.0       0.90      0.89      0.89    268905
         7.0       0.95      0.93      0.94    274431
         8.0       0.80      0.65      0.72    110744
         9.0       1.00      0.99      0.99    121710

    accuracy                           0.91   1664410
   macro avg       0.90      0.89      0.89   1664410
weighted avg       0.91      0.91      0.91   1664410
```

ACTUAL VS PREDICTED

```
[28] df=pd.DataFrame({'Actual':y_test, 'Predicted':y_pred})
     df.head(5)
```

	Actual	Predicted
0	6.0	6.0
1	7.0	7.0
2	3.0	3.0
3	7.0	7.0
4	1.0	1.0

The training score and test score lies in same range .

Hence the model is not overfitted.

TRAINING SCORE AND TEST SCORE

```
[62] print(classifier.score(X_train,y_train))
     print(classifier.score(X_test,y_test))

     0.9836508290456693
     0.9075498224596104
```