In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import sklearn
```

In [2]:
```
import glob
path="C:\S1_Dataset"
all_files=glob.glob(path+'/*')
print(all_files)
```

```
['C:\\S1_Dataset\\d1p01M', 'C:\\S1_Dataset\\d1p02M', 'C:\\S1_Dataset\\d1p03M', 'C:\\S1_Dataset\\d1p04M', 'C:\\S1_Dataset\\d1p05M', 'C:\\S1_Dataset\\d1p06M',
'C:\\S1_Dataset\\d1p07M', 'C:\\S1_Dataset\\d1p08F', 'C:\\S1_Dataset\\d1p09F', 'C:\\S1_Dataset\\d1p10F', 'C:\\S1_Dataset\\d1p11F', 'C:\\S1_Dataset\\d1p12F',
'C:\\S1_Dataset\\d1p13F', 'C:\\S1_Dataset\\d1p14F', 'C:\\S1_Dataset\\d1p15F', 'C:\\S1_Dataset\\d1p16F', 'C:\\S1_Dataset\\d1p17F', 'C:\\S1_Dataset\\d1p18F',
'C:\\S1_Dataset\\d1p19F', 'C:\\S1_Dataset\\d1p20F', 'C:\\S1_Dataset\\d1p21F', 'C:\\S1_Dataset\\d1p22F', 'C:\\S1_Dataset\\d1p23F', 'C:\\S1_Dataset\\d1p24F',
'C:\\S1_Dataset\\d1p25F', 'C:\\S1_Dataset\\d1p26F', 'C:\\S1_Dataset\\d1p27F', 'C:\\S1_Dataset\\d1p28F', 'C:\\S1_Dataset\\d1p29F', 'C:\\S1_Dataset\\d1p30F',
'C:\\S1_Dataset\\d1p31F', 'C:\\S1_Dataset\\d1p32F', 'C:\\S1_Dataset\\d1p33F', 'C:\\S1_Dataset\\d1p34F', 'C:\\S1_Dataset\\d1p35F', 'C:\\S1_Dataset\\d1p36M',
'C:\\S1_Dataset\\d1p37M', 'C:\\S1_Dataset\\d1p38M', 'C:\\S1_Dataset\\d1p39M', 'C:\\S1_Dataset\\d1p40M', 'C:\\S1_Dataset\\d1p41M', 'C:\\S1_Dataset\\d1p42M',
'C:\\S1_Dataset\\d1p43M', 'C:\\S1_Dataset\\d1p44M', 'C:\\S1_Dataset\\d1p45M', 'C:\\S1_Dataset\\d1p46M', 'C:\\S1_Dataset\\d1p47M', 'C:\\S1_Dataset\\d1p48M',
'C:\\S1_Dataset\\d1p49F', 'C:\\S1_Dataset\\d1p50F', 'C:\\S1_Dataset\\d1p51F', 'C:\\S1_Dataset\\d1p52F', 'C:\\S1_Dataset\\d1p53F', 'C:\\S1_Dataset\\d1p54F',
'C:\\S1_Dataset\\d1p55F', 'C:\\S1_Dataset\\d1p56F', 'C:\\S1_Dataset\\d1p57F', 'C:\\S1_Dataset\\d1p58F', 'C:\\S1_Dataset\\d1p59F', 'C:\\S1_Dataset\\d1p60F']
```

In [3]:
```
li=[]
for file in all_files:
    #print(file)
    if file.endswith('.txt'):
        continue
    df=pd.read_csv(file,header=None,index_col=None)
    li.append(df)
healthy_op=pd.concat(li,axis=0,ignore_index=True)
```

In [4]:
```
healthy_op.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 52482 entries, 0 to 52481
Data columns (total 9 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   0       52482 non-null  float64
 1   1       52482 non-null  float64
 2   2       52482 non-null  float64
 3   3       52482 non-null  float64
 4   4       52482 non-null  int64  
 5   5       52482 non-null  float64
 6   6       52482 non-null  float64
 7   7       52482 non-null  float64
 8   8       52482 non-null  int64  
dtypes: float64(7), int64(2)
memory usage: 3.6 MB
```

In [5]:
```
healthy_op.columns=['Time','Acc.Front','Acc.vert','Acc.Lat','id','RSSI','Phase','Freq','Activity_Label']
healthy_op.head()
```

Out[5]:

	Time	Acc.Front	Acc.vert	Acc.Lat	id	RSSI	Phase	Freq	Activity_Label
0	0.00	0.27203	1.00820	-0.082102	1	-63.5	2.4252	924.25	1
1	0.50	0.27203	1.00820	-0.082102	1	-63.0	4.7369	921.75	1
2	1.50	0.44791	0.91636	-0.013684	1	-63.5	3.0311	923.75	1
3	1.75	0.44791	0.91636	-0.013684	1	-63.0	2.0371	921.25	1
4	2.50	0.34238	0.96229	-0.059296	1	-63.5	5.8920	920.25	1

In [6]:
```
healthy_op.isnull().values.any()
```

Out[6]:
```
False
```

In [7]:
```
cols=len(healthy_op.columns)-1
healthy_op1=healthy_op.values
X=healthy_op1[:, :8]
Y=healthy_op1[:,8]
```

In [8]:
```
from sklearn.preprocessing import Normalizer
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split,GridSearchCV
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier
```

In [9]:
```
normalize=Normalizer()
X=normalize.fit_transform(X)
print(X)
```

```
[[ 0.00000000e+00  2.93631563e-04  1.08825991e-03 ... -6.85424559e-02
   2.61778211e-03  9.97643541e-01]
 [ 5.41175984e-04  2.94432206e-04  1.09122725e-03 ... -6.81881739e-02
   5.12699303e-03  9.97657926e-01]
 [ 1.61998019e-03  4.83736884e-04  9.89656697e-04 ... -6.85791613e-02
   3.27354797e-03  9.97637799e-01]
 ...
 [ 4.64054952e-01  3.16054398e-04  9.08811718e-04 ... -5.78276511e-02
   4.78430619e-03  8.83902815e-01]
 [ 4.65829320e-01  1.14107909e-04  7.86629014e-04 ... -5.34284525e-02
   4.68329008e-03  8.83239105e-01]
 [ 4.67376010e-01  6.95187748e-05  1.26064817e-03 ... -5.40291449e-02
   9.91649969e-04  8.82396344e-01]]
```

In [10]:
```
healthy_op.shape
```

Out[10]:
```
(52482, 9)
```

In [11]:
```
train, test = train_test_split(healthy_op, test_size = 0.25)
print(train.shape)
print(test.shape)
```

```
(39361, 9)
(13121, 9)
```

In [12]:
```
X_train = train[['Time','Acc.Front','Acc.vert','Acc.Lat','id','RSSI','Phase','Freq']]
Y_train =train.Activity_Label
X_test = test[['Time','Acc.Front','Acc.vert','Acc.Lat','id','RSSI','Phase','Freq']]
Y_test =test.Activity_Label
```

In [13]:
```
X_train.head()
```

Out[13]:

	Time	Acc.Front	Acc.vert	Acc.Lat	id	RSSI	Phase	Freq
27023	13.850	0.18995	1.008200	0.054735	4	-57.0	0.62433	925.25
42145	434.750	1.16320	0.009187	-0.070699	1	-56.0	5.12500	922.25
48453	309.750	1.15140	0.020670	-0.104910	1	-56.5	5.69410	921.25
21562	76.175	1.04590	0.020670	-0.161920	3	-61.5	5.64810	923.25
24070	335.100	1.05760	0.020670	-0.150520	1	-55.5	5.12500	920.75

In [14]:
```
Y_train.head()
```

Out[14]:
```
27023    1
42145    3
48453    3
21562    3
24070    3
Name: Activity_Label, dtype: int64
```

In [15]:
```
#normalization
normalize=Normalizer()
n_X_train=normalize.fit_transform(X_train)

normalize=Normalizer()
n_X_test=normalize.fit_transform(X_test)
```

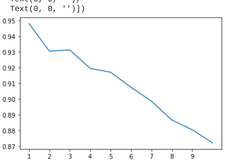
In [16]:
```
# KNN
knn=KNeighborsClassifier(n_neighbors=3)
knn.fit(n_X_train,Y_train)
y_pred_knn=knn.predict(n_X_test)
print("accuracy KNN= ",accuracy_score(Y_test,y_pred_knn))
```

```
accuracy KNN=  0.9312552396920967
```

In [17]:
```
a_index=list(range(1,11))
a=pd.Series()
x=[1,2,3,4,5,6,7,8,9]
for i in list(range(1,11)):
    model=KNeighborsClassifier(n_neighbors=i)
    model.fit(n_X_train,Y_train)
    prediction=model.predict(n_X_test)
    a=a.append(pd.Series(accuracy_score(prediction,Y_test)))
plt.plot(a_index, a)
plt.xticks(x)
```

```
C:\Users\Ganesh\AppData\Local\Temp/ipykernel_10616/3442830369.py:2: DeprecationWarning: The default dtype for empty Series will be 'object' instead of 'float6
4' in a future version. Specify a dtype explicitly to silence this warning.
  a=pd.Series()
```

Out[17]:
```
([<matplotlib.axis.XTick at 0x27e9141da60>,
  <matplotlib.axis.XTick at 0x27e9141da30>,
  <matplotlib.axis.XTick at 0x27e9140d9d0>,
  <matplotlib.axis.XTick at 0x27e92dec640>,
  <matplotlib.axis.XTick at 0x27e92decd90>,
  <matplotlib.axis.XTick at 0x27e92decdc0>,
  <matplotlib.axis.XTick at 0x27e92df8640>,
  <matplotlib.axis.XTick at 0x27e92df8d90>,
  <matplotlib.axis.XTick at 0x27e92dfd520>],
 [Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, '')])
```



In [18]:
```
lr=LogisticRegression()
estimator={'solver':('newton-cg','liblinear','lbfgs','sag')}
gsc=GridSearchCV(lr,estimator)
gsc.fit(n_X_train,Y_train)
y_gsc_pred=gsc.predict(n_X_test)
print("accuracy gsc= ",accuracy_score(Y_test,y_gsc_pred))
print(gsc.best_estimator_)
```

```
C:\Users\Ganesh\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
accuracy gsc=  0.9956065043975307
LogisticRegression(solver='newton-cg')
```

In [19]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn import svm
```

In [20]:
```
model = svm.SVC() #select the algorithm
model.fit(n_X_train,Y_train) # we train the algorithm with the training data and the training output
prediction=model.predict(n_X_test) #now we pass the testing data to the trained algorithm
print('The accuracy of the SVM is:',accuracy_score(prediction,Y_test))#now we check the accuracy of the algorithm.
#we pass the predicted output by the model and the actual output
```

```
The accuracy of the SVM is: 0.5849401722429692
```

In [21]:
```
rforest=RandomForestClassifier()
rforest.fit(n_X_train,Y_train)
y_pred_rforest=rforest.predict(n_X_test)
print("accuracy Random Forest= ",accuracy_score(Y_test,y_pred_rforest))
```

```
accuracy Random Forest=  0.9939791174453166
```

In [ ]:
```

```