```
# EMG Data Analysis
# Source: https://archive.ics.uci.edu/ml/datasets/EMG+Physical+Action+Data+Set


#How was the data collected ?
## Person/Subject was asked to perform specific physical actions
## Signals produced due to that movement were recorded over time.
## 8 channels were used to record the signals
## Channels here correspond to muscles\ For eg: Right-hand bicep
## Frequency : 10 per ms
```

```
!wget "https://drive.google.com/uc?export=download&id=1ZgZYFdMlQyccRtHs16AhJUEnLPhg
```

```
--2022-05-18 15:47:15--  https://drive.google.com/uc?export=download&id=1ZgZYFd
Resolving drive.google.com (drive.google.com)... 142.250.125.138, 142.250.125.1
Connecting to drive.google.com (drive.google.com)|142.250.125.138|:443... conne
HTTP request sent, awaiting response... 303 See Other
Location: https://doc-0o-14-docs.googleusercontent.com/docs/securesc/ha0ro937gc
Warning: wildcards not supported in HTTP.
--2022-05-18 15:47:21--  https://doc-0o-14-docs.googleusercontent.com/docs/secu
Resolving doc-0o-14-docs.googleusercontent.com (doc-0o-14-docs.googleusercontent
Connecting to doc-0o-14-docs.googleusercontent.com (doc-0o-14-docs.googleuserco
HTTP request sent, awaiting response... 200 OK
Length: 18602479 (18M) [application/vnd.rar]
Saving to: ‘EMG_data.rar’

EMG_data.rar        100%[===================>]  17.74M  35.3MB/s    in 0.5s

2022-05-18 15:47:22 (35.3 MB/s) - ‘EMG_data.rar’ saved [18602479/18602479]
```

```
!unrar x "./EMG_data.rar" "./"
```

```
Extracting  ./EMG Physical Action Data Set/sub3/Normal/log/Jumping.log  OK
Extracting  ./EMG Physical Action Data Set/sub3/Normal/log/Running.log  OK
Extracting  ./EMG Physical Action Data Set/sub3/Normal/log/Seating.log  OK
Extracting  ./EMG Physical Action Data Set/sub3/Normal/log/Standing.log  OK
Extracting  ./EMG Physical Action Data Set/sub3/Normal/log/Walking.log  OK
Extracting  ./EMG Physical Action Data Set/sub3/Normal/log/Waving.log  OK
Extracting  ./EMG Physical Action Data Set/sub3/Normal/txt/Bowing.txt  OK
Extracting  ./EMG Physical Action Data Set/sub3/Normal/txt/Clapping.txt  OK
Extracting  ./EMG Physical Action Data Set/sub3/Normal/txt/Handshaking.txt
Extracting  ./EMG Physical Action Data Set/sub3/Normal/txt/Hugging.txt  OK
Extracting  ./EMG Physical Action Data Set/sub3/Normal/txt/Jumping.txt  OK
Extracting  ./EMG Physical Action Data Set/sub3/Normal/txt/Running.txt  OK
Extracting  ./EMG Physical Action Data Set/sub3/Normal/txt/Seating.txt  OK
Extracting  ./EMG Physical Action Data Set/sub3/Normal/txt/Standing.txt  OK
Extracting  ./EMG Physical Action Data Set/sub3/Normal/txt/Walking.txt  OK
Extracting  ./EMG Physical Action Data Set/sub3/Normal/txt/Waving.txt  OK
Extracting  ./EMG Physical Action Data Set/sub4/Aggressive/log/Elbowing.log
Extracting  ./EMG Physical Action Data Set/sub4/Aggressive/log/FrontKicking.
Extracting  ./EMG Physical Action Data Set/sub4/Aggressive/log/Hamering.log
Extracting  ./EMG Physical Action Data Set/sub4/Aggressive/log/Headering.log
Extracting  ./EMG Physical Action Data Set/sub4/Aggressive/log/Kneeing.log
Extracting  ./EMG Physical Action Data Set/sub4/Aggressive/log/Pulling.log
Extracting  ./EMG Physical Action Data Set/sub4/Aggressive/log/Pulling.log
Extracting  ./EMG Physical Action Data Set/sub4/Aggressive/log/Punching.log
Extracting  ./EMG Physical Action Data Set/sub4/Aggressive/log/Pushing.log
Extracting  ./EMG Physical Action Data Set/sub4/Aggressive/log/SideKicking.l
Extracting  ./EMG Physical Action Data Set/sub4/Aggressive/log/Slapping.log
Extracting  ./EMG Physical Action Data Set/sub4/Aggressive/txt/Elbowing.txt
Extracting  ./EMG Physical Action Data Set/sub4/Aggressive/txt/Frontkicking.
Extracting  ./EMG Physical Action Data Set/sub4/Aggressive/txt/Hamering.txt
Extracting  ./EMG Physical Action Data Set/sub4/Aggressive/txt/Headering.txt
Extracting  ./EMG Physical Action Data Set/sub4/Aggressive/txt/Kneeing.txt
Extracting  ./EMG Physical Action Data Set/sub4/Aggressive/txt/Pulling.txt
Extracting  ./EMG Physical Action Data Set/sub4/Aggressive/txt/Punching.txt
Extracting  ./EMG Physical Action Data Set/sub4/Aggressive/txt/Pushing.txt
Extracting  ./EMG Physical Action Data Set/sub4/Aggressive/txt/Sidekicking.t
Extracting  ./EMG Physical Action Data Set/sub4/Aggressive/txt/Slapping.txt
Extracting  ./EMG Physical Action Data Set/sub4/Normal/log/Bowing.log  OK
Extracting  ./EMG Physical Action Data Set/sub4/Normal/log/Clapping.log  OK
Extracting  ./EMG Physical Action Data Set/sub4/Normal/log/Handshaking.log
Extracting  ./EMG Physical Action Data Set/sub4/Normal/log/Hugging.log  OK
Extracting  ./EMG Physical Action Data Set/sub4/Normal/log/Jumping.log  OK
Extracting  ./EMG Physical Action Data Set/sub4/Normal/log/Running.log  OK
Extracting  ./EMG Physical Action Data Set/sub4/Normal/log/Seating.log  OK
Extracting  ./EMG Physical Action Data Set/sub4/Normal/log/Standing.log  OK
Extracting  ./EMG Physical Action Data Set/sub4/Normal/log/Walking.log  OK
Extracting  ./EMG Physical Action Data Set/sub4/Normal/log/Waving.log  OK
Extracting  ./EMG Physical Action Data Set/sub4/Normal/txt/Bowing.txt  OK
Extracting  ./EMG Physical Action Data Set/sub4/Normal/txt/Clapping.txt  OK
Extracting  ./EMG Physical Action Data Set/sub4/Normal/txt/Handshaking.txt
Extracting  ./EMG Physical Action Data Set/sub4/Normal/txt/Hugging.txt  OK
Extracting  ./EMG Physical Action Data Set/sub4/Normal/txt/Jumping.txt  OK
Extracting  ./EMG Physical Action Data Set/sub4/Normal/txt/Running.txt  OK
Extracting  ./EMG Physical Action Data Set/sub4/Normal/txt/Seating.txt  OK
Extracting  ./EMG Physical Action Data Set/sub4/Normal/txt/Standing.txt  OK
Extracting  ./EMG Physical Action Data Set/sub4/Normal/txt/Walking.txt  OK
Extracting  ./EMG Physical Action Data Set/sub4/Normal/txt/Waving.txt  OK
All OK
```

```
!sudo apt install tree
```

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
tree is already the newest version (1.7.0-5).
The following packages were automatically installed and are no longer requirec
  libnvidia-common-460 nsight-compute-2020.2.0
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 67 not upgraded.
```

```
!tree "./EMG Physical Action Data Set/sub1"
```

```
./EMG Physical Action Data Set/sub1
├── Aggressive
│   ├── log
│   │   ├── Elbowing.log
│   │   ├── FrontKicking.log
│   │   ├── Hamering.log
│   │   ├── Headering.log
│   │   ├── Kneeing.log
│   │   ├── Pulling.log
│   │   ├── Punching.log
│   │   ├── Pushing.log
│   │   ├── SideKicking.log
│   │   └── Slapping.log
│   └── txt
│       ├── Elbowing.txt
│       ├── Frontkicking.txt
│       ├── Hamering.txt
│       ├── Headering.txt
│       ├── Kneeing.txt
│       ├── Pulling.txt
│       ├── Punching.txt
│       ├── Pushing.txt
│       ├── Sidekicking.txt
│       └── Slapping.txt
└── Normal
    ├── log
    │   ├── Bowing.log
    │   ├── Clapping.log
    │   ├── Handshaking.log
    │   ├── Hugging.log
    │   ├── Jumping.log
    │   ├── Running.log
    │   ├── Seating.log
    │   ├── Standing.log
    │   ├── Walking.log
    │   └── Waving.log
    └── txt
        ├── Bowing.txt
        ├── Clapping.txt
        ├── Handshaking.txt
        ├── Hugging.txt
        ├── Jumping.txt
        ├── Running.txt
        ├── Seating.txt
        ├── Standing.txt
        ├── Walking.txt
        └── Waving.txt

6 directories, 40 files
```

```
!ls -lrt ./EMG\ Physical\ Action\ Data\ Set/sub1/Aggressive/txt/
```

```
total 3768
-rw-r--r-- 1 root root 361096 Feb  7  2010 Slapping.txt
-rw-r--r-- 1 root root 388912 Feb  7  2010 Sidekicking.txt
-rw-r--r-- 1 root root 379428 Feb  7  2010 Pushing.txt
-rw-r--r-- 1 root root 379597 Feb  7  2010 Punching.txt
-rw-r--r-- 1 root root 387656 Feb  7  2010 Pulling.txt
-rw-r--r-- 1 root root 398523 Feb  7  2010 Kneeing.txt
-rw-r--r-- 1 root root 350285 Feb  7  2010 Headering.txt
-rw-r--r-- 1 root root 402363 Feb  7  2010 Hamering.txt
-rw-r--r-- 1 root root 390158 Feb  7  2010 Frontkicking.txt
-rw-r--r-- 1 root root 398095 Feb  7  2010 Elbowing.txt
```

```
!cat ./EMG\ Physical\ Action\ Data\ Set/sub1/Aggressive/txt/Slapping.txt
```

```
    -999    392     38      -51     1409    -90     628     -348
    -979    335     30      -95     1995    -71     653     -297
    -858    213     -9      -98     2516    -153    419     -305
    -649    93      -63     -62     2822    -11     384     -297
    -417    -15     -52     -90     2968    -87     102     -294
    -245    -91     -76     -54     2947    -216    -379    -319
    -93     -192    -59     -48     2822    -251    -236    -288
    -42     -215    -66     -12     2374    -137    329     -276
    -58     -136    -37     -5      1771    -140    411     -278
    -39     -161    62      -4      1409    -32     329     -314
    6       -173    37      18      1017    180     702     -333
    38      -167    7       1       519     426     1119    -380
    73      -210    19      23      -147    766     1520    -314
    12      -161    9       10      -1007   365     1591    -195
    -104    43      2       23      -1980   -6      1451    -179
    -84     87      21      2       -2395   101     1402    -154
    41      -53     8       16      -1945   -113    1320    -105
    137     -155    7       59      -1583   -722    1446    -101
    56      -214    12      49      -1729   -1237   1540    -98
    -48     -230    -15     68      -2134   -1005   1906    -56
    -106    -91     15      46      -2574   -327    2118    -137
    -89     113     14      63      -3508   -15     1858    -230
    -70     9       18      77      -4000   -54     2309    -155
    -75     -98     77      37      -4000   26      3205    -81
    -13     -8      131     20      -3852   185     3144    -158
    44      103     149     44      -4000   392     2452    -151
    55      187     46      63      -4000   376     1591    -151
    141     174     -14     61      -4000   246     535     -195
    167     233     -55     49      -4000   336     -110    -287
    126     239     -66     -10     -4000   464     260     -350
    146     99      -36     -20     -4000   685     1467    -301
    212     -37     7       -2      -4000   611     2309    -294
    117     -128    88      16      -4000   658     2444    -325
    22      -143    140     42      -4000   744     2322    -256
    15      -179    130     9       -4000   433     2375    -166
    59      -220    41      19      -4000   144     2444    -200
    133     -216    10      34      -4000   55      2189    -258
    77      -192    -38     -25     -4000   -3      1866    -297
    32      8       -99     -65     -3855   -535    1778    -248
    31      146     -197    -45     -3172   -1062   1864    -216
    89      30      -277    -46     -2550   -419    1772    -246
    130     16      -255    -69     -2126   229     2118    -234
    32      -1      -269    -74     -1593   156     2450    -158
    43      3       -238    -81     -1685   534     1599    -112
    -9      4       -192    -64     -2224   498     337     -132
    -172    -18     -148    -23     -2484   429     -350    -93
    -447    38      -127    10      -2344   319     -1142   -43
    -665    122     -39     -3      -1729   -193    -2015   37
    -453    282     158     -19     -1685   -347    -2549   296
    -307    212     299     -17     -1837   -34     -3455   485
    -187    -38     272     -1      -1398   167     -4000   387
    -191    -275    140     4       -864    107     -4000   339
    -482    -318    50      -5      -658    283     -4000   308
    -596    -162    8       -12     -406    115     -4000   202
    -361    -149    -55     4       625     85      -4000   155
    -26     -224    -91     -16     976     -37     -4000   -5
    186     -155    -87     -32     635     27      -4000   -338
    408     72      -97     22      635     27      -4000   -338
```

```
!wc -l ./EMG\ Physical\ Action\ Data\ Set/sub1/Aggressive/txt/Slapping.txt
```

```
    9788 ./EMG Physical Action Data Set/sub1/Aggressive/txt/Slapping.txt
```

```
import os
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import sklearn

actions = {}

data_dirs = ["./EMG Physical Action Data Set/sub1/Aggressive/txt",
             "./EMG Physical Action Data Set/sub1/Normal/txt"]

ind = 0
data = pd.DataFrame()

for dirs in data_dirs :

  for files in os.listdir(dirs):

    with open(os.path.join(dirs, files), "r") as f:

      temp = pd.read_csv(f.name,
                        sep = "\t",
                        header = None,
                        names = ["ch" + str(i) for i in range(1, 9)] # 8 input chan
                        )

      # chunking using Max of every 10 sequential values.
      temp_chunked = pd.DataFrame()

      for i in range(0, len(temp), 10):
        temp_chunked = temp_chunked.append(temp.iloc[i:i+10].max(), ignore_index =

      labels = [files[:-4] for i in range(len(temp_chunked))] # remove the last 4 c
      actions[files[:-4]] = ind

      temp_chunked["Action"] = labels

      data = pd.concat([data, temp_chunked])

      ind+=1

print(actions)
```

```
    {'Frontkicking': 0, 'Slapping': 1, 'Elbowing': 2, 'Pulling': 3, 'Kneeing': 4,
```

```
data.head()
```

	ch1	ch2	ch3	ch4	ch5	ch6	ch7	ch8	Action
0	328.0	433.0	533.0	210.0	3918.0	961.0	-3193.0	4000.0	Frontkicking
1	577.0	309.0	1550.0	-21.0	-4000.0	707.0	2297.0	3719.0	Frontkicking
2	712.0	160.0	632.0	149.0	-3231.0	1515.0	3294.0	4000.0	Frontkicking
3	234.0	68.0	125.0	402.0	2972.0	2428.0	3009.0	1895.0	Frontkicking
4	2471.0	118.0	-263.0	-94.0	-4000.0	4000.0	-3592.0	613.0	Frontkicking

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19711 entries, 0 to 999
Data columns (total 9 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   ch1     19711 non-null  float64
 1   ch2     19711 non-null  float64
 2   ch3     19711 non-null  float64
 3   ch4     19711 non-null  float64
 4   ch5     19711 non-null  float64
 5   ch6     19711 non-null  float64
 6   ch7     19711 non-null  float64
 7   ch8     19711 non-null  float64
 8   Action  19711 non-null  object 
dtypes: float64(8), object(1)
memory usage: 1.5+ MB
```

```
data["Action"].value_counts()
```

```
Seating         1000
Jumping         1000
Waving          1000
Kneeing         1000
Hamering        1000
Clapping        1000
Headering       1000
Walking         1000
Running          997
Bowing           983
Sidekicking      983
Frontkicking     982
Slapping         979
Elbowing         978
Hugging          976
Standing         973
Pushing          968
Pulling          966
Punching         964
Handshaking      962
Name: Action, dtype: int64
```

```
Y = data["Action"]
X = data.drop(columns = ["Action"])
```

```
# Label encoding
Y = Y.map(actions)
Y.head()
print(Y.value_counts())
```

```
19    1000
11    1000
18    1000
4     1000
5     1000
14    1000
9     1000
12    1000
13     997
15     983
6      983
0      982
1      979
2      978
10     976
16     973
7      968
3      966
8      964
17     962
Name: Action, dtype: int64
```

```
# Domain specific pre-processing
X = abs(X)
X.head()
```

	ch1	ch2	ch3	ch4	ch5	ch6	ch7	ch8
0	328.0	433.0	533.0	210.0	3918.0	961.0	3193.0	4000.0
1	577.0	309.0	1550.0	21.0	4000.0	707.0	2297.0	3719.0
2	712.0	160.0	632.0	149.0	3231.0	1515.0	3294.0	4000.0
3	234.0	68.0	125.0	402.0	2972.0	2428.0	3009.0	1895.0
4	2471.0	118.0	263.0	94.0	4000.0	4000.0	3592.0	613.0

```
# Train, test split
# no CV?
from sklearn.model_selection import train_test_split

X = np.array(X.values.tolist())
Y = np.array(Y.values.tolist())

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, shuffle
print(f"Sizes of the sets created are:\nTraining set:{X_train.shape[0]}\nTest set:{
```

```
Sizes of the sets created are:
Training set:15768
Test set:3943
```

```
# Simple DT
# 5-fold CV
# Grid Search for best hyper-param
from sklearn.tree import DecisionTreeClassifier as DTC
from sklearn import tree
from sklearn.model_selection import GridSearchCV

params = {
    "max_depth" : [3, 5, 7],
    "max_leaf_nodes" : [15, 20, 25]
}

model1 = DTC()
clf = GridSearchCV(model1, params, scoring = "accuracy", cv=5)

clf.fit(X_train, Y_train)
```

```
    ---------------------------------------------------------------------------
    NameError                                 Traceback (most recent call last)
    <ipython-input-2-9851e8f8f4ca> in <module>()
         14 clf = GridSearchCV(model1, params, scoring = "accuracy", cv=5)
         15
    ---> 16 clf.fit(X_train, Y_train)

    NameError: name 'X_train' is not defined
```

SEARCH STACK OVERFLOW

```
res = clf.cv_results_

for i in range(len(res["params"])):
  print(f"Parameters:{res['params'][i]} Mean_score: {res['mean_test_score'][i]} Ran
```

```
    Parameters:{'max_depth': 3, 'max_leaf_nodes': 15} Mean_score: 0.31925426177643
    Parameters:{'max_depth': 3, 'max_leaf_nodes': 20} Mean_score: 0.31925426177643
    Parameters:{'max_depth': 3, 'max_leaf_nodes': 25} Mean_score: 0.31925426177643
    Parameters:{'max_depth': 5, 'max_leaf_nodes': 15} Mean_score: 0.40544144628994
    Parameters:{'max_depth': 5, 'max_leaf_nodes': 20} Mean_score: 0.43600910728898
    Parameters:{'max_depth': 5, 'max_leaf_nodes': 25} Mean_score: 0.44482467905574
    Parameters:{'max_depth': 7, 'max_leaf_nodes': 15} Mean_score: 0.40544144628994
    Parameters:{'max_depth': 7, 'max_leaf_nodes': 20} Mean_score: 0.42288167140996
    Parameters:{'max_depth': 7, 'max_leaf_nodes': 25} Mean_score: 0.44552240712059
```

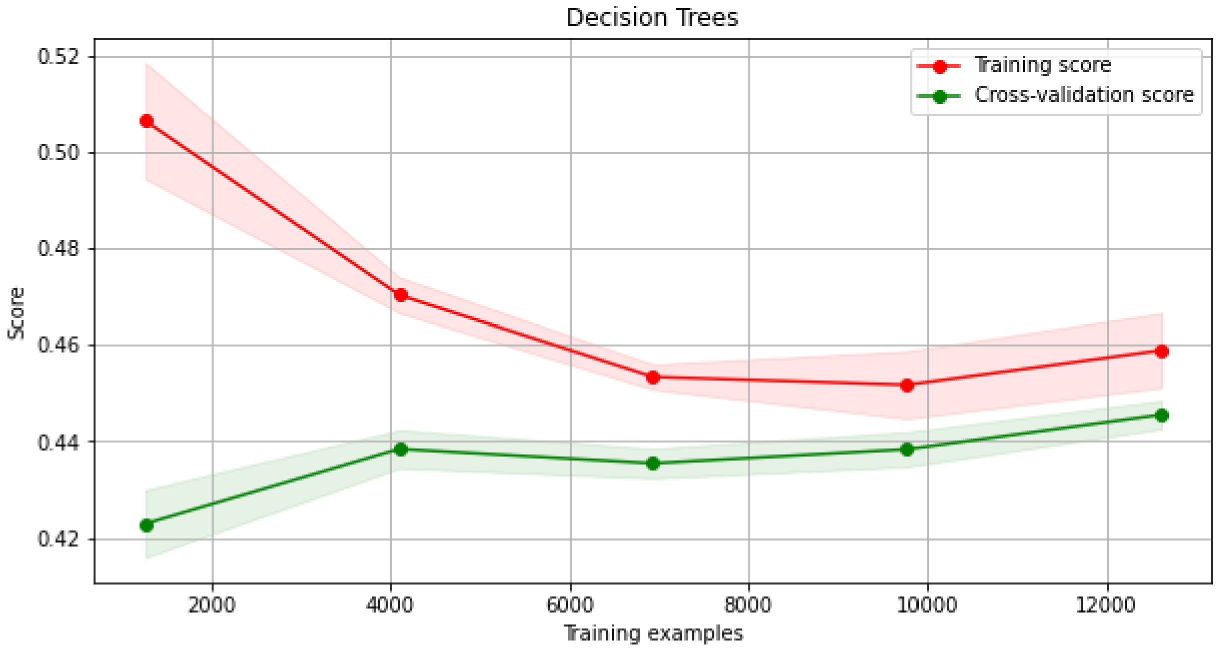
```
print(clf.best_estimator_)
```

```
    DecisionTreeClassifier(max_depth=7, max_leaf_nodes=25)
```

```
# Learning Curves
from sklearn.model_selection import learning_curve
```

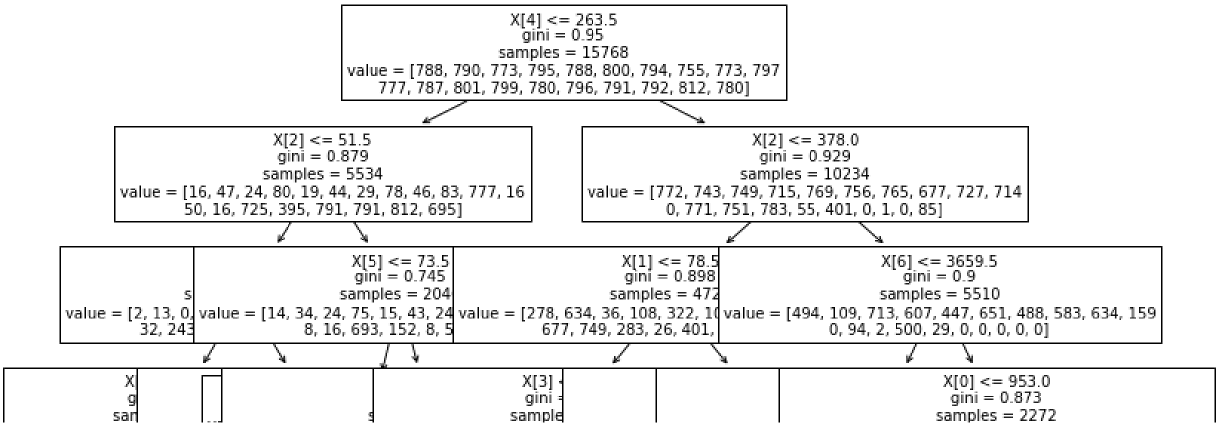
```
def plot_learning_curve(estimator, X, Y, title):

  train_sizes, train_scores, test_scores, _, _ = learning_curve(estimator,
                                                                        X,
                                                                        Y,
                                                                        return_time
                                                                        )

  fig, axes = plt.subplots(1, 1, figsize = (10, 5))

  axes.set_title(title)
  axes.plot
  axes.set_xlabel("Training examples")
  axes.set_ylabel("Score")


  train_scores_mean = np.mean(train_scores, axis=1)
  train_scores_std = np.std(train_scores, axis=1)
  test_scores_mean = np.mean(test_scores, axis=1)
  test_scores_std = np.std(test_scores, axis=1)

  # Plot learning curve
  axes.grid()
  axes.fill_between(
      train_sizes,
      train_scores_mean - train_scores_std,
      train_scores_mean + train_scores_std,
      alpha=0.1,
      color="r",
  )
  axes.fill_between(
      train_sizes,
      test_scores_mean - test_scores_std,
      test_scores_mean + test_scores_std,
      alpha=0.1,
      color="g",
  )
  axes.plot(
      train_sizes, train_scores_mean, "o-", color="r", label="Training score"
  )
  axes.plot(
      train_sizes, test_scores_mean, "o-", color="g", label="Cross-validation score
  )
  axes.legend(loc="best")

  plt.show()


model1 = clf.best_estimator_

model1.fit(X_train, Y_train)

plot_learning_curve(model1, X_train, Y_train, "Decision Trees")
```

```
print(model1.score(X_train, Y_train))

# more data could help as CV-score is improving as datset size increases.
```



```
0.4644216133942161
```

```
# plot the decision tree
from sklearn import tree

plt.figure(figsize=(12,12))  # set plot size (denoted in inches)
tree.plot_tree(model1, fontsize=10)
plt.show()
```



```
# Xgboost
from xgboost import XGBClassifier
from sklearn.model_selection import RandomizedSearchCV, GridSearchCV
from sklearn.model_selection import StratifiedKFold

import datetime as dt

params = {
        'learning_rate': [0.1, 0.5, 0.8],
        'subsample': [0.6, 0.8, 1.0],
        'colsample_bytree': [0.6, 0.8, 1.0],
        'max_depth': [3, 4, 5]
        }
xgb = XGBClassifier(n_estimators=100, objective='multi:softmax', num_class=20, sile




folds = 3

skf = StratifiedKFold(n_splits=folds, shuffle = True, random_state = 1001)

random_search = RandomizedSearchCV(xgb, param_distributions=params, n_iter=10, scor


start = dt.datetime.now()
random_search.fit(X_train, Y_train)
end = dt.datetime.now()
```

```
     Fitting 3 folds for each of 10 candidates, totalling 30 fits
```

```
print('\n Best hyperparameters:')
print(random_search.best_params_)
```

```
      Best hyperparameters:
     {'subsample': 0.6, 'max_depth': 3, 'learning_rate': 0.1, 'colsample_bytree': 1
```

```
best_xgb = XGBClassifier(n_estimators=100, objective='multi:softmax', num_class=20,
best_xgb.fit(X_train, Y_train)
```

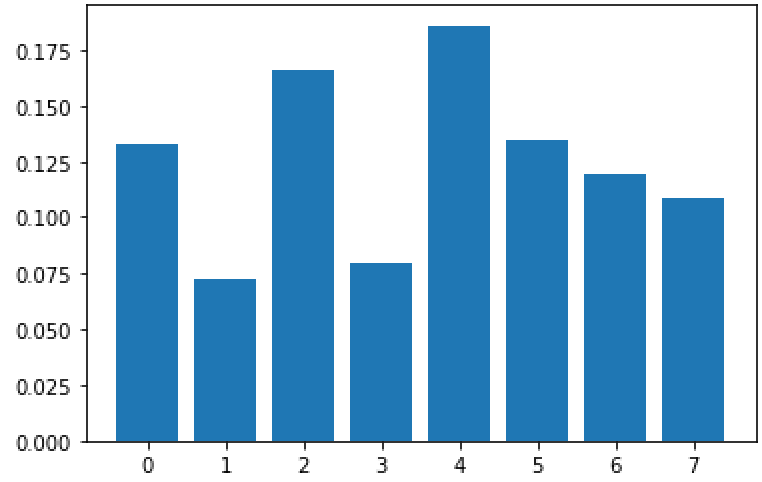
```
    XGBClassifier(colsample_bytree=1.0, num_class=20, objective='multi:softprob',
                  silent=True, subsample=0.6)
```

```
print(f"Time taken for training : {end - start}\nTraining accuracy:{best_xgb.score(
```

```
    Time taken for training : 0:05:07.294350
    Training accuracy:0.6747209538305429
    Test Accuracy: 0.6114633527770733
```

```
print(best_xgb.feature_importances_)

plt.bar(range(len(best_xgb.feature_importances_)), best_xgb.feature_importances_)
plt.show()
```

```
    [0.13315983 0.07298602 0.16566639 0.07955533 0.18605755 0.13500534
     0.11930533 0.10826417]
```



```
# Domain specific ideas to improve the results: Average data across time.
## Source: https://www.researchgate.net/figure/EMG-signal-process-recommended-Green
```

0s completed at 22:24