Import the libraries

In [1]:

```
#importing the libraries
import tensorflow as tf
from tensorflow import keras
import numpy as np
import pandas as pd
```

```
WARNING:tensorflow:From C:\Users\Teo Boon Kean\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\s
rc\losses.py:2976: The name tf.losses.sparse_softmax_cross_entropy is deprecated. Please use tf.compat.v1.losses
.sparse_softmax_cross_entropy instead.
```

Load the data and data pre-processing

In [2]:

```
#load the datasets
baseline_df = pd.read_excel('extracted_features_baseline.xlsx')
toolwear_df = pd.read_excel('extracted_features_toolwear.xlsx')
```

In [3]:

```
#labelling the datasets. 0 for baseline, 1 for toolwear. This will be the variable the model tries to predict
baseline_df["state"] = 0
toolwear_df["state"] = 1
```

In [4]:

```
#concantanate the datasets
combined_df = pd.concat([baseline_df, toolwear_df], axis=0)
print(combined_df.shape)
```

```
(840, 67)
```

In [5]:

```
#getting the y label
state = combined_df["state"].values
print(state.shape)
```

```
(840,)
```

In [6]:

```
#getting the features to train the model
features = combined_df.drop('state', axis=1).values
print(features.shape)
```

```
(840, 66)
```

In [7]:

```
#train test split
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(features, state, test_size=0.2, random_state=40)
```

In [8]:

```
#data scalling
from sklearn.preprocessing import StandardScaler

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

Training and Evaluation of Classification Models

1) Support Vector Machine (SVM)

In [9]:

```
#Import svm model from scikit learn
from sklearn import svm

#create the svm classifier model
SVM = svm.SVC(kernel="linear")

#train the model
SVM.fit(X_train, Y_train)
```

Out[9]:

```
▾ SVC
SVC(kernel='linear')
```

In [10]:

```
#test the model
predict_SVM = SVM.predict(X_test)

#crosstabs
pd.crosstab(predict_SVM, Y_test)
```

Out[10]:

col_0	0	1
row_0		
0	81	0
1	0	87

In [11]:

```python
from sklearn.metrics import classification_report

#print detailed report for SVM
print(classification_report(Y_test, predict_SVM))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        81
           1       1.00      1.00      1.00        87

    accuracy                           1.00       168
   macro avg       1.00      1.00      1.00       168
weighted avg       1.00      1.00      1.00       168
```

Timing analysis for SVM

In [12]:

```python
#data is fed in 1 by 1 in the for loop and the time taken for each prediction is summed in "time_passed"

result = []
time_passed = 0
import time

for j in range (0, X_test.shape[0]):
    #process starts so record the start time
    start = time.time()

    #classify with SVM
    prediction = SVM.predict(np.array( [X_test[j],] ))

    #process ends so record the end time
    end = time.time()

    #storing the result
    result.append(prediction[0])
    time_passed = time_passed + (end-start)
```

In [13]:

```python
pd.crosstab(result, Y_test)
```

Out[13]:

col_0	0	1
row_0		
0	81	0
1	0	87

In [14]:

```python
print(time_passed)
```

```
0.03381824493408203
```

In [15]:

```python
#The average time is calculated by dividing the total time with the number of predictions
avg_time = time_passed / len(result)
print(avg_time)
```

```
0.00020129907698858353
```

2) Naive Bayes

In [16]:

```python
#Import NaiveBayes model from scikit learn
from sklearn.naive_bayes import GaussianNB

#create the NB classifier model
GAU_NB = GaussianNB()

#train the model
GAU_NB.fit(X_train, Y_train)
```

Out[16]:

```
▾ GaussianNB
GaussianNB()
```

In [17]:

```python
#test the model
```

```
predict_NB = GAU_NB.predict(X_test)

#crosstabs
pd.crosstab(predict_NB, Y_test)
```

Out[17]:

col_0	0	1
row_0		
0	79	0
1	2	87

In [18]:

```
from sklearn.metrics import classification_report

#print detailed report for NB
print(classification_report(Y_test, predict_NB))
```

```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99        81
           1       0.98      1.00      0.99        87

    accuracy                           0.99       168
   macro avg       0.99      0.99      0.99       168
weighted avg       0.99      0.99      0.99       168
```

3) K-Nearest Neighbor

In [19]:

```
#Import KNN model from scikit learn
from sklearn.neighbors import KNeighborsClassifier

#create the KNN classifier model
KNN = KNeighborsClassifier(n_neighbors=2)

#train the model
KNN.fit(X_train, Y_train)
```

Out[19]:

```
▾        KNeighborsClassifier
KNeighborsClassifier(n_neighbors=2)
```

In [20]:

```
#test the model
predict_KNN = KNN.predict(X_test)

#crosstabs
pd.crosstab(predict_KNN, Y_test)
```

Out[20]:

col_0	0	1
row_0		
0	81	0
1	0	87

In [21]:

```
from sklearn.metrics import classification_report

#print detailed report for KNN
print(classification_report(Y_test, predict_KNN))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        81
           1       1.00      1.00      1.00        87

    accuracy                           1.00       168
   macro avg       1.00      1.00      1.00       168
weighted avg       1.00      1.00      1.00       168
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js