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.
```

Select the number of encoded features here

In [2]:
```
#specify the number of condensed features. This will be the number of neurons in the hidden layer. Comment the
#condensed_f = 5
#condensed_f = 10
#condensed_f = 20
#condensed_f = 30
condensed_f = 50
```

Load the data and data pre-processing

In [3]:
```
#load the datasets
baseline_df = pd.read_excel('extracted_features_baseline.xlsx')
toolwear_df = pd.read_excel('extracted_features_toolwear.xlsx')
```

In [4]:
```
#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 [5]:
```
#concantanate the datasets
combined_df = pd.concat([baseline_df, toolwear_df], axis=0)
print(combined_df.shape)
```

```
(840, 67)
```

In [6]:
```
#getting the y label
state = combined_df["state"].values
print(state.shape)
```

```
(840,)
```

In [7]:
```
#getting the features to train the model
features = combined_df.drop('state', axis=1).values
print(features.shape)
```

```
(840, 66)
```

In [8]:
```
#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=5)
```

In [9]:
```
#data scalling
from sklearn.preprocessing import StandardScaler

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

Construct and train the Autoencoder based on the number of encoded features specified

In [10]:
```
#constructing the model

#input layer which number of neurons equals the number of original features
l_in = keras.Input(features.shape[1])

#hidden layer which condenses the feature into the specified number of condensed features
l_condensed = keras.layers.Dense(condensed_f)(l_in)

#output layer which is the same as the input
l_out = keras.layers.Dense(features.shape[1])(l_condensed)
```

```
WARNING:tensorflow:From C:\Users\Teo Boon Kean\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\s
rc\backend.py:1398: The name tf.executing_eagerly_outside_functions is deprecated. Please use tf.compat.v1.execu
ting_eagerly_outside_functions instead.
```

In [11]:
```python
#defining the autoencode
autoencoder = keras.Model(l_in, l_out)
```

In [12]:
```python
autoencoder.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 66)]              0

 dense (Dense)               (None, 50)                3350

 dense_1 (Dense)             (None, 66)                3366

=================================================================
Total params: 6716 (26.23 KB)
Trainable params: 6716 (26.23 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [13]:
```python
#compile the model
autoencoder.compile(optimizer='adam', loss='mse')
#train the model
autoencoder.fit(X_train, X_train, epochs = 40, batch_size = 8, validation_split = 0.1)
```

```
WARNING:tensorflow:From C:\Users\Teo Boon Kean\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\s
rc\optimizers\__init__.py:309: The name tf.train.Optimizer is deprecated. Please use tf.compat.v1.train.Optimize
r instead.

Epoch 1/40
WARNING:tensorflow:From C:\Users\Teo Boon Kean\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\s
rc\utils\tf_utils.py:492: The name tf.ragged.RaggedTensorValue is deprecated. Please use tf.compat.v1.ragged.Rag
gedTensorValue instead.

76/76 [==============================] - 1s 5ms/step - loss: 0.9114 - val_loss: 0.5070
Epoch 2/40
76/76 [==============================] - 0s 2ms/step - loss: 0.3114 - val_loss: 0.2560
Epoch 3/40
76/76 [==============================] - 0s 3ms/step - loss: 0.1723 - val_loss: 0.1558
Epoch 4/40
76/76 [==============================] - 0s 4ms/step - loss: 0.1162 - val_loss: 0.1120
Epoch 5/40
76/76 [==============================] - 0s 3ms/step - loss: 0.0863 - val_loss: 0.0866
Epoch 6/40
76/76 [==============================] - 0s 2ms/step - loss: 0.0682 - val_loss: 0.0685
Epoch 7/40
76/76 [==============================] - 0s 2ms/step - loss: 0.0558 - val_loss: 0.0586
Epoch 8/40
76/76 [==============================] - 0s 2ms/step - loss: 0.0474 - val_loss: 0.0500
Epoch 9/40
76/76 [==============================] - 0s 2ms/step - loss: 0.0407 - val_loss: 0.0441
Epoch 10/40
76/76 [==============================] - 0s 2ms/step - loss: 0.0361 - val_loss: 0.0389
Epoch 11/40
76/76 [==============================] - 0s 2ms/step - loss: 0.0323 - val_loss: 0.0356
Epoch 12/40
76/76 [==============================] - 0s 3ms/step - loss: 0.0290 - val_loss: 0.0328
Epoch 13/40
76/76 [==============================] - 0s 2ms/step - loss: 0.0264 - val_loss: 0.0299
Epoch 14/40
76/76 [==============================] - 0s 2ms/step - loss: 0.0239 - val_loss: 0.0273
Epoch 15/40
76/76 [==============================] - 0s 2ms/step - loss: 0.0219 - val_loss: 0.0252
Epoch 16/40
76/76 [==============================] - 0s 2ms/step - loss: 0.0202 - val_loss: 0.0244
Epoch 17/40
76/76 [==============================] - 0s 2ms/step - loss: 0.0186 - val_loss: 0.0226
Epoch 18/40
76/76 [==============================] - 0s 2ms/step - loss: 0.0172 - val_loss: 0.0208
Epoch 19/40
76/76 [==============================] - 0s 3ms/step - loss: 0.0160 - val_loss: 0.0198
Epoch 20/40
76/76 [==============================] - 0s 2ms/step - loss: 0.0149 - val_loss: 0.0187
Epoch 21/40
76/76 [==============================] - 0s 2ms/step - loss: 0.0140 - val_loss: 0.0175
Epoch 22/40
76/76 [==============================] - 0s 3ms/step - loss: 0.0131 - val_loss: 0.0164
Epoch 23/40
```

```
76/76 [==============================] - 0s 3ms/step - loss: 0.0123 - val_loss: 0.0153
Epoch 24/40
76/76 [==============================] - 0s 3ms/step - loss: 0.0116 - val_loss: 0.0143
Epoch 25/40
76/76 [==============================] - 0s 2ms/step - loss: 0.0108 - val_loss: 0.0137
Epoch 26/40
76/76 [==============================] - 0s 2ms/step - loss: 0.0102 - val_loss: 0.0131
Epoch 27/40
76/76 [==============================] - 0s 2ms/step - loss: 0.0097 - val_loss: 0.0120
Epoch 28/40
76/76 [==============================] - 0s 3ms/step - loss: 0.0091 - val_loss: 0.0116
Epoch 29/40
76/76 [==============================] - 0s 2ms/step - loss: 0.0086 - val_loss: 0.0109
Epoch 30/40
76/76 [==============================] - 0s 3ms/step - loss: 0.0081 - val_loss: 0.0103
Epoch 31/40
76/76 [==============================] - 0s 3ms/step - loss: 0.0077 - val_loss: 0.0099
Epoch 32/40
76/76 [==============================] - 0s 3ms/step - loss: 0.0073 - val_loss: 0.0097
Epoch 33/40
76/76 [==============================] - 0s 3ms/step - loss: 0.0069 - val_loss: 0.0092
Epoch 34/40
76/76 [==============================] - 0s 4ms/step - loss: 0.0065 - val_loss: 0.0087
Epoch 35/40
76/76 [==============================] - 0s 3ms/step - loss: 0.0062 - val_loss: 0.0084
Epoch 36/40
76/76 [==============================] - 0s 3ms/step - loss: 0.0059 - val_loss: 0.0079
Epoch 37/40
76/76 [==============================] - 0s 3ms/step - loss: 0.0056 - val_loss: 0.0073
Epoch 38/40
76/76 [==============================] - 0s 3ms/step - loss: 0.0053 - val_loss: 0.0070
Epoch 39/40
76/76 [==============================] - 0s 3ms/step - loss: 0.0050 - val_loss: 0.0070
Epoch 40/40
76/76 [==============================] - 0s 3ms/step - loss: 0.0048 - val_loss: 0.0064
```

Out[13]: <keras.src.callbacks.History at 0x1f93109c450>

In [14]:
```
from sklearn.metrics import mean_absolute_error

#predict with the autoencoder and compute the MSE of output compared to input
pred = autoencoder.predict(X_test)
print(mean_absolute_error(X_test,pred))
```

```
6/6 [==============================] - 0s 2ms/step
0.054010149653591
```

Initialising the Encoder with the trained layers

In [15]:
```
#defining just the encoder
encoder = keras.Model(l_in, l_condensed)
```

Encoding the features in training and test datasets with the defined encoder

In [16]:
```
#using the encoder to condense the features of train and test dataset
en_train = encoder.predict(X_train)
en_test = encoder.predict(X_test)
```

```
21/21 [==============================] - 0s 2ms/step
6/6 [==============================] - 0s 2ms/step
```

Train and test the classification models using the encoded features

1) Support Vector Machine (SVM)

In [17]:
```
#defining the SVM model
#Import svm model from scikit learn
from sklearn import svm

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

In [18]:
```
#train the model
classifier.fit(en_train, Y_train)
```

Out[18]:
```
▾ SVC
SVC(kernel='linear')
```

In [19]:
```python
#test the model
predict = classifier.predict(en_test)

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

Out[19]:

col_0	0	1
row_0		
0	77	0
1	0	91

In [20]:
```python
from sklearn.metrics import classification_report

#print detailed report
print(classification_report(Y_test, predict))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        77
           1       1.00      1.00      1.00        91

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

Timing Analysis of Autoencoder + SVM

In [21]:
```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()

    #encode the feature with encoder
    en_feature = encoder.predict(np.array( [X_test[j],] ))
    #classify with SVM using encoded feature
    prediction = classifier.predict(en_feature)

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

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

```
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 24ms/step
```

```
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 60ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
```

```
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 25ms/step
```

In [22]:
```
#The average time is calculated by dividing the total time with the number of predictions
avg_time = time_passed/len(result)
print(avg_time)
```

```
0.08851170539855957
```

2) Naive Bayes

In [23]:
```
#defining the NaiveBayes model
#Import NaiveBayes model from scikit learn
from sklearn.naive_bayes import BernoulliNB

#create the svm classifier model
BER_NB = BernoulliNB(binarize=0.0)
```

In [24]:
```
#train the model
BER_NB.fit(en_train, Y_train)
```

Out[24]:
```
▾ BernoulliNB
BernoulliNB()
```

In [25]:
```
#test the model
predict = BER_NB.predict(en_test)

#crosstabs
```

```
pd.crosstab(Y_test, predict)
```

Out[25]:

col_0	0	1
row_0		
0	75	2
1	0	91

In [26]:
```
from sklearn.metrics import classification_report

#print detailed report
print(classification_report(Y_test, predict))
```

```
              precision    recall  f1-score   support

           0       1.00      0.97      0.99        77
           1       0.98      1.00      0.99        91

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

3) KNN

In [27]:
```
#defining the KNN model
#Import KNN model from scikit learn
from sklearn.neighbors import KNeighborsClassifier

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

In [28]:
```
#train the model
KNN.fit(en_train, Y_train)
```

Out[28]:
```
▾        KNeighborsClassifier
KNeighborsClassifier(n_neighbors=2)
```

In [29]:
```
#test the model
predict = KNN.predict(en_test)

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

Out[29]:

col_0	0	1
row_0		
0	77	0
1	0	91

In [30]:
```
from sklearn.metrics import classification_report

#print detailed report
print(classification_report(Y_test, predict))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        77
           1       1.00      1.00      1.00        91

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