Project 5

Arka Bera, Het Mevada, John Pope, Dongyang Zhen

Import packages and load dataset

```
import pandas as pd
import numpy as np
import glob
```

```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
# filepath = '/content/gdrive/MyDrive/ENME691_P5/'
filepath = '/content/gdrive/MyDrive/ENME691/Project/Project 5/'
```

```
colnames = ['force_x', 'force_y', 'force_z', 'vibration_x', 'vibration_y', 'vibration_z', 'AE-rms']
```

```
c1_data = []

files = glob.glob(filepath+'Training Dataset/'+'c1/'+'*.csv')
for file in files:
    c1_data.append(pd.read_csv(file, names=colnames, header=None).to_numpy())

c1_target = pd.read_csv(filepath+'Training Dataset/'+'c1_wear.csv', header = 0).drop(columns=['cut']).to_numpy()
```

```
c4_data = []

files = glob.glob(filepath+'Training Dataset/'+'c4/'+'*.csv')
for file in files:
    c4_data.append(pd.read_csv(file, names=colnames, header=None).to_numpy())

c4_target = pd.read_csv(filepath+'Training Dataset/'+'c4_wear.csv', header = 0).drop(columns=['cut']).to_numpy()
```

```
c6_data = []

files = glob.glob(filepath+'Testing Dataset/'+'c6/'+'*.csv')
for file in files:
    c6_data.append(pd.read_csv(file, names=colnames, header=None).to_numpy())

c6_target = pd.read_csv(filepath+'Testing Dataset/'+'c6_wear.csv', header = 0).drop(columns=['cut']).to_numpy()
```

```
c1_target = np.linalg.norm(c1_target, axis=1)
c4_target = np.linalg.norm(c4_target, axis=1)
c6_target = np.linalg.norm(c6_target, axis=1)
```

```
# Comment out the cells below to use full dataset
```

```
c1_data = c1_data
c1_target = c1_target
```

```
c4_data = c4_data
c4_target = c4_target
```

```
c6_data = c6_data
c6_target = c6_target
```

Visualize training and test data

```
from matplotlib import pyplot as plt
```

```
# fig, ax = plt.subplots(7, figsize = (10, 40))
force_x_c1 = a = [c1_data[i][:, 0] for i in range(len(c1_data))]
# ax[0].plot(force_x_c1[0])
force_y_c1 = a = [c1_data[i][:, 1] for i in range(len(c1_data))]
# ax[1].plot(force_y_c1[0])
force_z_c1 = a = [c1_data[i][:, 2] for i in range(len(c1_data))]
# ax[2].plot(force_z_c1[0])
vibration_x_c1 = a = [c1_data[i][:, 3] for i in range(len(c1_data))]
# ax[3].plot(vibration_x_c1[0])
vibration_y_c1 = a = [c1_data[i][:, 4] for i in range(len(c1_data))]
# ax[4].plot(vibration_y_c1[0])
vibration_z_c1 = a = [c1_data[i][:, 5] for i in range(len(c1_data))]
# ax[5].plot(vibration_z_c1[0])
AE_rms_c1 = a = [c1_data[i][:, 6] for i in range(len(c1_data))]
# ax[6].plot(AE_rms_c1[0])
# plt.suptitle("C1 Data", fontsize = 16, y = 0.9)
# for count, ele in enumerate(colnames):
#     ax[count].set_title(ele)
#     ax[count].set_xlabel('Time')
#     ax[count].set_ylabel('Amplitude')


# fig, ax = plt.subplots(7, figsize = (10, 40))
force_x_c4 = a = [c4_data[i][:, 0] for i in range(len(c4_data))]
# ax[0].plot(force_x_c4[0])
force_y_c4 = a = [c4_data[i][:, 1] for i in range(len(c4_data))]
# ax[1].plot(force_y_c4[0])
force_z_c4 = a = [c4_data[i][:, 2] for i in range(len(c4_data))]
# ax[2].plot(force_z_c4[0])
vibration_x_c4 = a = [c4_data[i][:, 3] for i in range(len(c4_data))]
# ax[3].plot(vibration_x_c4[0])
vibration_y_c4 = a = [c4_data[i][:, 4] for i in range(len(c4_data))]
# ax[4].plot(vibration_y_c4[0])
vibration_z_c4 = a = [c4_data[i][:, 5] for i in range(len(c4_data))]
# ax[5].plot(vibration_z_c4[0])
AE_rms_c4 = a = [c4_data[i][:, 6] for i in range(len(c4_data))]
# ax[6].plot(AE_rms_c4[0])
# plt.suptitle("C4 Data", fontsize = 16, y = 0.9)
# for count, ele in enumerate(colnames):
#     ax[count].set_title(ele)
#     ax[count].set_xlabel('Time')
#     ax[count].set_ylabel('Amplitude')


# fig, ax = plt.subplots(7, figsize = (10, 40))
force_x_c6 = a = [c6_data[i][:, 0] for i in range(len(c6_data))]
# ax[0].plot(force_x_c6[0])
force_y_c6 = a = [c6_data[i][:, 1] for i in range(len(c6_data))]
# ax[1].plot(force_y_c6[0])
force_z_c6 = a = [c6_data[i][:, 2] for i in range(len(c6_data))]
# ax[2].plot(force_z_c6[0])
vibration_x_c6 = a = [c6_data[i][:, 3] for i in range(len(c6_data))]
# ax[3].plot(vibration_x_c6[0])
vibration_y_c6 = a = [c6_data[i][:, 4] for i in range(len(c6_data))]
# ax[4].plot(vibration_y_c6[0])
vibration_z_c6 = a = [c6_data[i][:, 5] for i in range(len(c6_data))]
# ax[5].plot(vibration_z_c6[0])
AE_rms_c6 = a = [c6_data[i][:, 6] for i in range(len(c6_data))]
# ax[6].plot(AE_rms_c6[0])
# plt.suptitle("C6 Data", fontsize = 16, y = 0.9)
# for count, ele in enumerate(colnames):
#     ax[count].set_title(ele)
#     ax[count].set_xlabel('Time')
#     ax[count].set_ylabel('Amplitude')
```

Feature Extraction

```
from scipy.stats import skew, kurtosis
from scipy.fftpack import fft
import pywt
import seaborn as sns
```

```
def wavelet_energy(arr):
  wavelet = 'db4'
  level = 5

  coeffs = pywt.wavedec(arr, wavelet, level=level)
  wlet_energy = [np.sum(np.square(detail)) for detail in coeffs[1:]]

  return np.sum(wlet_energy)
```

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range = (0, 1))
```

```
# features for force_x_c1

force_x_c1_rms = scaler.fit_transform(np.sqrt([np.mean(array**2) for array in force_x_c1]).reshape(-1, 1)).flatten()
force_x_c1_var = scaler.fit_transform(np.array([np.var(array) for array in force_x_c1]).reshape(-1, 1)).flatten()
force_x_c1_max = scaler.fit_transform(np.array([np.max(array) for array in force_x_c1]).reshape(-1, 1)).flatten()
force_x_c1_skw = scaler.fit_transform(np.array([skew(array) for array in force_x_c1]).reshape(-1, 1)).flatten()
force_x_c1_krt = scaler.fit_transform(np.array([kurtosis(array) for array in force_x_c1]).reshape(-1, 1)).flatten()
force_x_c1_ptp = scaler.fit_transform(np.array([np.ptp(array) for array in force_x_c1]).reshape(-1, 1)).flatten()

force_x_c1_fft = np.abs([fft(array) for array in force_x_c1])
force_x_c1_sskw = scaler.fit_transform(np.array([skew(array) for array in force_x_c1_fft]).reshape(-1, 1)).flatten()
force_x_c1_skrt = scaler.fit_transform(np.array([kurtosis(array) for array in force_x_c1_fft]).reshape(-1, 1)).flatten()

force_x_c1_wave = scaler.fit_transform(np.array([wavelet_energy(array) for array in force_x_c1_fft]).reshape(-1, 1)).flatten()
```

```
# features for force_y_c1

force_y_c1_rms = scaler.fit_transform(np.sqrt([np.mean(array**2) for array in force_y_c1]).reshape(-1, 1)).flatten()
force_y_c1_var = scaler.fit_transform(np.array([np.var(array) for array in force_y_c1]).reshape(-1, 1)).flatten()
force_y_c1_max = scaler.fit_transform(np.array([np.max(array) for array in force_y_c1]).reshape(-1, 1)).flatten()
force_y_c1_skw = scaler.fit_transform(np.array([skew(array) for array in force_y_c1]).reshape(-1, 1)).flatten()
force_y_c1_krt = scaler.fit_transform(np.array([kurtosis(array) for array in force_y_c1]).reshape(-1, 1)).flatten()
force_y_c1_ptp = scaler.fit_transform(np.array([np.ptp(array) for array in force_y_c1]).reshape(-1, 1)).flatten()

force_y_c1_fft = np.abs([fft(array) for array in force_y_c1])
force_y_c1_sskw = scaler.fit_transform(np.array([skew(array) for array in force_y_c1_fft]).reshape(-1, 1)).flatten()
force_y_c1_skrt = scaler.fit_transform(np.array([kurtosis(array) for array in force_y_c1_fft]).reshape(-1, 1)).flatten()

force_y_c1_wave = scaler.fit_transform(np.array([wavelet_energy(array) for array in force_y_c1_fft]).reshape(-1, 1)).flatten()
```

```
# features for force_z_c1

force_z_c1_rms = scaler.fit_transform(np.sqrt([np.mean(array**2) for array in force_z_c1]).reshape(-1, 1)).flatten()
force_z_c1_var = scaler.fit_transform(np.array([np.var(array) for array in force_z_c1]).reshape(-1, 1)).flatten()
force_z_c1_max = scaler.fit_transform(np.array([np.max(array) for array in force_z_c1]).reshape(-1, 1)).flatten()
force_z_c1_skw = scaler.fit_transform(np.array([skew(array) for array in force_z_c1]).reshape(-1, 1)).flatten()
force_z_c1_krt = scaler.fit_transform(np.array([kurtosis(array) for array in force_z_c1]).reshape(-1, 1)).flatten()
force_z_c1_ptp = scaler.fit_transform(np.array([np.ptp(array) for array in force_z_c1]).reshape(-1, 1)).flatten()

force_z_c1_fft = np.abs([fft(array) for array in force_z_c1])
force_z_c1_sskw = scaler.fit_transform(np.array([skew(array) for array in force_z_c1_fft]).reshape(-1, 1)).flatten()
force_z_c1_skrt = scaler.fit_transform(np.array([kurtosis(array) for array in force_z_c1_fft]).reshape(-1, 1)).flatten()

force_z_c1_wave = scaler.fit_transform(np.array([wavelet_energy(array) for array in force_z_c1_fft]).reshape(-1, 1)).flatten()
```

```
# features for vibration_x_c1

vibration_x_c1_rms = scaler.fit_transform(np.sqrt([np.mean(array**2) for array in vibration_x_c1]).reshape(-1, 1)).flatten()
vibration_x_c1_var = scaler.fit_transform(np.array([np.var(array) for array in vibration_x_c1]).reshape(-1, 1)).flatten()
vibration_x_c1_max = scaler.fit_transform(np.array([np.max(array) for array in vibration_x_c1]).reshape(-1, 1)).flatten()
vibration_x_c1_skw = scaler.fit_transform(np.array([skew(array) for array in vibration_x_c1]).reshape(-1, 1)).flatten()
vibration_x_c1_krt = scaler.fit_transform(np.array([kurtosis(array) for array in vibration_x_c1]).reshape(-1, 1)).flatten()
vibration_x_c1_ptp = scaler.fit_transform(np.array([np.ptp(array) for array in vibration_x_c1]).reshape(-1, 1)).flatten()

vibration_x_c1_fft = np.abs([fft(array) for array in vibration_x_c1])
vibration_x_c1_sskw = scaler.fit_transform(np.array([skew(array) for array in vibration_x_c1_fft]).reshape(-1, 1)).flatten()
vibration_x_c1_skrt = scaler.fit_transform(np.array([kurtosis(array) for array in vibration_x_c1_fft]).reshape(-1, 1)).flatten()

vibration_x_c1_wave = scaler.fit_transform(np.array([wavelet_energy(array) for array in vibration_x_c1_fft]).reshape(-1, 1)).flatten()
```

```
# features for vibration_y_c1

vibration_y_c1_rms = scaler.fit_transform(np.sqrt([np.mean(array**2) for array in vibration_y_c1]).reshape(-1, 1)).flatten()
vibration_y_c1_var = scaler.fit_transform(np.array([np.var(array) for array in vibration_y_c1]).reshape(-1, 1)).flatten()
vibration_y_c1_max = scaler.fit_transform(np.array([np.max(array) for array in vibration_y_c1]).reshape(-1, 1)).flatten()
vibration_y_c1_skw = scaler.fit_transform(np.array([skew(array) for array in vibration_y_c1]).reshape(-1, 1)).flatten()
vibration_y_c1_krt = scaler.fit_transform(np.array([kurtosis(array) for array in vibration_y_c1]).reshape(-1, 1)).flatten()
vibration_y_c1_ptp = scaler.fit_transform(np.array([np.ptp(array) for array in vibration_y_c1]).reshape(-1, 1)).flatten()

vibration_y_c1_fft = np.abs([fft(array) for array in vibration_y_c1])
vibration_y_c1_sskw = scaler.fit_transform(np.array([skew(array) for array in vibration_y_c1_fft]).reshape(-1, 1)).flatten()
vibration_y_c1_skrt = scaler.fit_transform(np.array([kurtosis(array) for array in vibration_y_c1_fft]).reshape(-1, 1)).flatten()

vibration_y_c1_wave = scaler.fit_transform(np.array([wavelet_energy(array) for array in vibration_y_c1_fft]).reshape(-1, 1)).flatten()


# features for vibration_z_c1

vibration_z_c1_rms = scaler.fit_transform(np.sqrt([np.mean(array**2) for array in vibration_z_c1]).reshape(-1, 1)).flatten()
vibration_z_c1_var = scaler.fit_transform(np.array([np.var(array) for array in vibration_z_c1]).reshape(-1, 1)).flatten()
vibration_z_c1_max = scaler.fit_transform(np.array([np.max(array) for array in vibration_z_c1]).reshape(-1, 1)).flatten()
vibration_z_c1_skw = scaler.fit_transform(np.array([skew(array) for array in vibration_z_c1]).reshape(-1, 1)).flatten()
vibration_z_c1_krt = scaler.fit_transform(np.array([kurtosis(array) for array in vibration_z_c1]).reshape(-1, 1)).flatten()
vibration_z_c1_ptp = scaler.fit_transform(np.array([np.ptp(array) for array in vibration_z_c1]).reshape(-1, 1)).flatten()

vibration_z_c1_fft = np.abs([fft(array) for array in vibration_z_c1])
vibration_z_c1_sskw = scaler.fit_transform(np.array([skew(array) for array in vibration_z_c1_fft]).reshape(-1, 1)).flatten()
vibration_z_c1_skrt = scaler.fit_transform(np.array([kurtosis(array) for array in vibration_z_c1_fft]).reshape(-1, 1)).flatten()

vibration_z_c1_wave = scaler.fit_transform(np.array([wavelet_energy(array) for array in vibration_z_c1_fft]).reshape(-1, 1)).flatten()


c1_features = np.stack((force_x_c1_rms, force_x_c1_var, force_x_c1_max,
                        force_x_c1_skw, force_x_c1_krt, force_x_c1_ptp,
                        force_x_c1_sskw, force_x_c1_skrt, force_x_c1_wave,
                        force_y_c1_rms, force_y_c1_var, force_y_c1_max,
                        force_y_c1_skw, force_y_c1_krt, force_y_c1_ptp,
                        force_y_c1_sskw, force_y_c1_skrt, force_y_c1_wave,
                        force_z_c1_rms, force_z_c1_var, force_z_c1_max,
                        force_z_c1_skw, force_z_c1_krt, force_z_c1_ptp,
                        force_z_c1_sskw, force_z_c1_skrt, force_z_c1_wave,
                        vibration_x_c1_rms, vibration_x_c1_var, vibration_x_c1_max,
                        vibration_x_c1_skw, vibration_x_c1_krt, vibration_x_c1_ptp,
                        vibration_x_c1_sskw, vibration_x_c1_skrt, vibration_x_c1_wave,
                        vibration_y_c1_rms, vibration_y_c1_var, vibration_y_c1_max,
                        vibration_y_c1_skw, vibration_y_c1_krt, vibration_y_c1_ptp,
                        vibration_y_c1_sskw, vibration_y_c1_skrt, vibration_y_c1_wave,
                        vibration_z_c1_rms, vibration_z_c1_var, vibration_z_c1_max,
                        vibration_z_c1_skw, vibration_z_c1_krt, vibration_z_c1_ptp,
                        vibration_z_c1_sskw, vibration_z_c1_skrt, vibration_z_c1_wave
                        ),
                       axis = 1)


c1_target = scaler.fit_transform(c1_target.reshape(-1, 1)).flatten()


# sns.pairplot(pd.DataFrame(c1_features, columns = ['force_x_c1_rms', 'force_x_c1_var', 'force_x_c1_max',
#                          'force_x_c1_skw', 'force_x_c1_krt', 'force_x_c1_ptp',
#                          'force_x_c1_sskw', 'force_x_c1_skrt', 'force_x_c1_wave',
#                          'force_y_c1_rms', 'force_y_c1_var', 'force_y_c1_max',
#                          'force_y_c1_skw', 'force_y_c1_krt', 'force_y_c1_ptp',
#                          'force_y_c1_sskw', 'force_y_c1_skrt', 'force_y_c1_wave',
#                          'force_z_c1_rms', 'force_z_c1_var', 'force_z_c1_max',
#                          'force_z_c1_skw', 'force_z_c1_krt', 'force_z_c1_ptp',
#                          'force_z_c1_sskw', 'force_z_c1_skrt', 'force_z_c1_wave',
#                          'vibration_x_c1_rms', 'vibration_x_c1_var', 'vibration_x_c1_max',
#                          'vibration_x_c1_skw', 'vibration_x_c1_krt', 'vibration_x_c1_ptp',
#                          'vibration_x_c1_sskw', 'vibration_x_c1_skrt', 'vibration_x_c1_wave',
#                          'vibration_y_c1_rms', 'vibration_y_c1_var', 'vibration_y_c1_max',
#                          'vibration_y_c1_skw', 'vibration_y_c1_krt', 'vibration_y_c1_ptp',
#                          'vibration_y_c1_sskw', 'vibration_y_c1_skrt', 'vibration_y_c1_wave',
#                          'vibration_z_c1_rms', 'vibration_z_c1_var', 'vibration_z_c1_max',
#                          'vibration_z_c1_skw', 'vibration_z_c1_krt', 'vibration_z_c1_ptp',
#                          'vibration_z_c1_sskw', 'vibration_z_c1_skrt', 'vibration_z_c1_wave']))
# features for force_x_c4

force_x_c4_rms = scaler.fit_transform(np.sqrt([np.mean(array**2) for array in force_x_c4]).reshape(-1, 1)).flatten()
force_x_c4_var = scaler.fit_transform(np.array([np.var(array) for array in force_x_c4]).reshape(-1, 1)).flatten()
force_x_c4_max = scaler.fit_transform(np.array([np.max(array) for array in force_x_c4]).reshape(-1, 1)).flatten()
force_x_c4_skw = scaler.fit_transform(np.array([skew(array) for array in force_x_c4]).reshape(-1, 1)).flatten()
force_x_c4_krt = scaler.fit_transform(np.array([kurtosis(array) for array in force_x_c4]).reshape(-1, 1)).flatten()
force_x_c4_ptp = scaler.fit_transform(np.array([np.ptp(array) for array in force_x_c4]).reshape(-1, 1)).flatten()

force_x_c4_fft = np.abs([fft(array) for array in force_x_c4])
force_x_c4_sskw = scaler.fit_transform(np.array([skew(array) for array in force_x_c4_fft]).reshape(-1, 1)).flatten()
force_x_c4_skrt = scaler.fit_transform(np.array([kurtosis(array) for array in force_x_c4_fft]).reshape(-1, 1)).flatten()

force_x_c4_wave = scaler.fit_transform(np.array([wavelet_energy(array) for array in force_x_c4_fft]).reshape(-1, 1)).flatten()


# features for force_y_c4

force_y_c4_rms = scaler.fit_transform(np.sqrt([np.mean(array**2) for array in force_y_c4]).reshape(-1, 1)).flatten()
force_y_c4_var = scaler.fit_transform(np.array([np.var(array) for array in force_y_c4]).reshape(-1, 1)).flatten()
force_y_c4_max = scaler.fit_transform(np.array([np.max(array) for array in force_y_c4]).reshape(-1, 1)).flatten()
force_y_c4_skw = scaler.fit_transform(np.array([skew(array) for array in force_y_c4]).reshape(-1, 1)).flatten()
force_y_c4_krt = scaler.fit_transform(np.array([kurtosis(array) for array in force_y_c4]).reshape(-1, 1)).flatten()
force_y_c4_ptp = scaler.fit_transform(np.array([np.ptp(array) for array in force_y_c4]).reshape(-1, 1)).flatten()

force_y_c4_fft = np.abs([fft(array) for array in force_y_c4])
force_y_c4_sskw = scaler.fit_transform(np.array([skew(array) for array in force_y_c4_fft]).reshape(-1, 1)).flatten()
force_y_c4_skrt = scaler.fit_transform(np.array([kurtosis(array) for array in force_y_c4_fft]).reshape(-1, 1)).flatten()

force_y_c4_wave = scaler.fit_transform(np.array([wavelet_energy(array) for array in force_y_c4_fft]).reshape(-1, 1)).flatten()


# features for force_z_c4

force_z_c4_rms = scaler.fit_transform(np.sqrt([np.mean(array**2) for array in force_z_c4]).reshape(-1, 1)).flatten()
force_z_c4_var = scaler.fit_transform(np.array([np.var(array) for array in force_z_c4]).reshape(-1, 1)).flatten()
force_z_c4_max = scaler.fit_transform(np.array([np.max(array) for array in force_z_c4]).reshape(-1, 1)).flatten()
force_z_c4_skw = scaler.fit_transform(np.array([skew(array) for array in force_z_c4]).reshape(-1, 1)).flatten()
force_z_c4_krt = scaler.fit_transform(np.array([kurtosis(array) for array in force_z_c4]).reshape(-1, 1)).flatten()
force_z_c4_ptp = scaler.fit_transform(np.array([np.ptp(array) for array in force_z_c4]).reshape(-1, 1)).flatten()

force_z_c4_fft = np.abs([fft(array) for array in force_z_c4])
force_z_c4_sskw = scaler.fit_transform(np.array([skew(array) for array in force_z_c4_fft]).reshape(-1, 1)).flatten()
force_z_c4_skrt = scaler.fit_transform(np.array([kurtosis(array) for array in force_z_c4_fft]).reshape(-1, 1)).flatten()

force_z_c4_wave = scaler.fit_transform(np.array([wavelet_energy(array) for array in force_z_c4_fft]).reshape(-1, 1)).flatten()


# features for vibration_x_c4

vibration_x_c4_rms = scaler.fit_transform(np.sqrt([np.mean(array**2) for array in vibration_x_c4]).reshape(-1, 1)).flatten()
vibration_x_c4_var = scaler.fit_transform(np.array([np.var(array) for array in vibration_x_c4]).reshape(-1, 1)).flatten()
vibration_x_c4_max = scaler.fit_transform(np.array([np.max(array) for array in vibration_x_c4]).reshape(-1, 1)).flatten()
vibration_x_c4_skw = scaler.fit_transform(np.array([skew(array) for array in vibration_x_c4]).reshape(-1, 1)).flatten()
vibration_x_c4_krt = scaler.fit_transform(np.array([kurtosis(array) for array in vibration_x_c4]).reshape(-1, 1)).flatten()
vibration_x_c4_ptp = scaler.fit_transform(np.array([np.ptp(array) for array in vibration_x_c4]).reshape(-1, 1)).flatten()

vibration_x_c4_fft = np.abs([fft(array) for array in vibration_x_c4])
vibration_x_c4_sskw = scaler.fit_transform(np.array([skew(array) for array in vibration_x_c4_fft]).reshape(-1, 1)).flatten()
vibration_x_c4_skrt = scaler.fit_transform(np.array([kurtosis(array) for array in vibration_x_c4_fft]).reshape(-1, 1)).flatten()

vibration_x_c4_wave = scaler.fit_transform(np.array([wavelet_energy(array) for array in vibration_x_c4_fft]).reshape(-1, 1)).flatten()


# features for vibration_y_c4

vibration_y_c4_rms = scaler.fit_transform(np.sqrt([np.mean(array**2) for array in vibration_y_c4]).reshape(-1, 1)).flatten()
vibration_y_c4_var = scaler.fit_transform(np.array([np.var(array) for array in vibration_y_c4]).reshape(-1, 1)).flatten()
vibration_y_c4_max = scaler.fit_transform(np.array([np.max(array) for array in vibration_y_c4]).reshape(-1, 1)).flatten()
vibration_y_c4_skw = scaler.fit_transform(np.array([skew(array) for array in vibration_y_c4]).reshape(-1, 1)).flatten()
vibration_y_c4_krt = scaler.fit_transform(np.array([kurtosis(array) for array in vibration_y_c4]).reshape(-1, 1)).flatten()
vibration_y_c4_ptp = scaler.fit_transform(np.array([np.ptp(array) for array in vibration_y_c4]).reshape(-1, 1)).flatten()

vibration_y_c4_fft = np.abs([fft(array) for array in vibration_y_c4])
vibration_y_c4_sskw = scaler.fit_transform(np.array([skew(array) for array in vibration_y_c4_fft]).reshape(-1, 1)).flatten()
vibration_y_c4_skrt = scaler.fit_transform(np.array([kurtosis(array) for array in vibration_y_c4_fft]).reshape(-1, 1)).flatten()

vibration_y_c4_wave = scaler.fit_transform(np.array([wavelet_energy(array) for array in vibration_y_c4_fft]).reshape(-1, 1)).flatten()
```

```
# features for vibration_z_c4

vibration_z_c4_rms = scaler.fit_transform(np.sqrt([np.mean(array**2) for array in vibration_z_c4]).reshape(-1, 1)).flatten()
vibration_z_c4_var = scaler.fit_transform(np.array([np.var(array) for array in vibration_z_c4]).reshape(-1, 1)).flatten()
vibration_z_c4_max = scaler.fit_transform(np.array([np.max(array) for array in vibration_z_c4]).reshape(-1, 1)).flatten()
vibration_z_c4_skw = scaler.fit_transform(np.array([skew(array) for array in vibration_z_c4]).reshape(-1, 1)).flatten()
vibration_z_c4_krt = scaler.fit_transform(np.array([kurtosis(array) for array in vibration_z_c4]).reshape(-1, 1)).flatten()
vibration_z_c4_ptp = scaler.fit_transform(np.array([np.ptp(array) for array in vibration_z_c4]).reshape(-1, 1)).flatten()

vibration_z_c4_fft = np.abs([fft(array) for array in vibration_z_c4])
vibration_z_c4_sskw = scaler.fit_transform(np.array([skew(array) for array in vibration_z_c4_fft]).reshape(-1, 1)).flatten()
vibration_z_c4_skrt = scaler.fit_transform(np.array([kurtosis(array) for array in vibration_z_c4_fft]).reshape(-1, 1)).flatten()

vibration_z_c4_wave = scaler.fit_transform(np.array([wavelet_energy(array) for array in vibration_z_c4_fft]).reshape(-1, 1)).flatten()


c4_features = np.stack((force_x_c4_rms, force_x_c4_var, force_x_c4_max,
                        force_x_c4_skw, force_x_c4_krt, force_x_c4_ptp,
                        force_x_c4_sskw, force_x_c4_skrt, force_x_c4_wave,
                        force_y_c4_rms, force_y_c4_var, force_y_c4_max,
                        force_y_c4_skw, force_y_c4_krt, force_y_c4_ptp,
                        force_y_c4_sskw, force_y_c4_skrt, force_y_c4_wave,
                        force_z_c4_rms, force_z_c4_var, force_z_c4_max,
                        force_z_c4_skw, force_z_c4_krt, force_z_c4_ptp,
                        force_z_c4_sskw, force_z_c4_skrt, force_z_c4_wave,
                        vibration_x_c4_rms, vibration_x_c4_var, vibration_x_c4_max,
                        vibration_x_c4_skw, vibration_x_c4_krt, vibration_x_c4_ptp,
                        vibration_x_c4_sskw, vibration_x_c4_skrt, vibration_x_c4_wave,
                        vibration_y_c4_rms, vibration_y_c4_var, vibration_y_c4_max,
                        vibration_y_c4_skw, vibration_y_c4_krt, vibration_y_c4_ptp,
                        vibration_y_c4_sskw, vibration_y_c4_skrt, vibration_y_c4_wave,
                        vibration_z_c4_rms, vibration_z_c4_var, vibration_z_c4_max,
                        vibration_z_c4_skw, vibration_z_c4_krt, vibration_z_c4_ptp,
                        vibration_z_c4_sskw, vibration_z_c4_skrt, vibration_z_c4_wave
                        ),
                       axis = 1)


c4_target = scaler.fit_transform(c4_target.reshape(-1, 1)).flatten()


# features for force_x_c6

force_x_c6_rms = scaler.fit_transform(np.sqrt([np.mean(array**2) for array in force_x_c6]).reshape(-1, 1)).flatten()
force_x_c6_var = scaler.fit_transform(np.array([np.var(array) for array in force_x_c6]).reshape(-1, 1)).flatten()
force_x_c6_max = scaler.fit_transform(np.array([np.max(array) for array in force_x_c6]).reshape(-1, 1)).flatten()
force_x_c6_skw = scaler.fit_transform(np.array([skew(array) for array in force_x_c6]).reshape(-1, 1)).flatten()
force_x_c6_krt = scaler.fit_transform(np.array([kurtosis(array) for array in force_x_c6]).reshape(-1, 1)).flatten()
force_x_c6_ptp = scaler.fit_transform(np.array([np.ptp(array) for array in force_x_c6]).reshape(-1, 1)).flatten()

force_x_c6_fft = np.abs([fft(array) for array in force_x_c6])
force_x_c6_sskw = scaler.fit_transform(np.array([skew(array) for array in force_x_c6_fft]).reshape(-1, 1)).flatten()
force_x_c6_skrt = scaler.fit_transform(np.array([kurtosis(array) for array in force_x_c6_fft]).reshape(-1, 1)).flatten()

force_x_c6_wave = scaler.fit_transform(np.array([wavelet_energy(array) for array in force_x_c6_fft]).reshape(-1, 1)).flatten()


# features for force_y_c6

force_y_c6_rms = scaler.fit_transform(np.sqrt([np.mean(array**2) for array in force_y_c6]).reshape(-1, 1)).flatten()
force_y_c6_var = scaler.fit_transform(np.array([np.var(array) for array in force_y_c6]).reshape(-1, 1)).flatten()
force_y_c6_max = scaler.fit_transform(np.array([np.max(array) for array in force_y_c6]).reshape(-1, 1)).flatten()
force_y_c6_skw = scaler.fit_transform(np.array([skew(array) for array in force_y_c6]).reshape(-1, 1)).flatten()
force_y_c6_krt = scaler.fit_transform(np.array([kurtosis(array) for array in force_y_c6]).reshape(-1, 1)).flatten()
force_y_c6_ptp = scaler.fit_transform(np.array([np.ptp(array) for array in force_y_c6]).reshape(-1, 1)).flatten()

force_y_c6_fft = np.abs([fft(array) for array in force_y_c6])
force_y_c6_sskw = scaler.fit_transform(np.array([skew(array) for array in force_y_c6_fft]).reshape(-1, 1)).flatten()
force_y_c6_skrt = scaler.fit_transform(np.array([kurtosis(array) for array in force_y_c6_fft]).reshape(-1, 1)).flatten()

force_y_c6_wave = scaler.fit_transform(np.array([wavelet_energy(array) for array in force_y_c6_fft]).reshape(-1, 1)).flatten()
```

```
# features for force_z_c6

force_z_c6_rms = scaler.fit_transform(np.sqrt([np.mean(array**2) for array in force_z_c6]).reshape(-1, 1)).flatten()
force_z_c6_var = scaler.fit_transform(np.array([np.var(array) for array in force_z_c6]).reshape(-1, 1)).flatten()
force_z_c6_max = scaler.fit_transform(np.array([np.max(array) for array in force_z_c6]).reshape(-1, 1)).flatten()
force_z_c6_skw = scaler.fit_transform(np.array([skew(array) for array in force_z_c6]).reshape(-1, 1)).flatten()
force_z_c6_krt = scaler.fit_transform(np.array([kurtosis(array) for array in force_z_c6]).reshape(-1, 1)).flatten()
force_z_c6_ptp = scaler.fit_transform(np.array([np.ptp(array) for array in force_z_c6]).reshape(-1, 1)).flatten()

force_z_c6_fft = np.abs([fft(array) for array in force_z_c6])
force_z_c6_sskw = scaler.fit_transform(np.array([skew(array) for array in force_z_c6_fft]).reshape(-1, 1)).flatten()
force_z_c6_skrt = scaler.fit_transform(np.array([kurtosis(array) for array in force_z_c6_fft]).reshape(-1, 1)).flatten()

force_z_c6_wave = scaler.fit_transform(np.array([wavelet_energy(array) for array in force_z_c6_fft]).reshape(-1, 1)).flatten()


# features for vibration_x_c6

vibration_x_c6_rms = scaler.fit_transform(np.sqrt([np.mean(array**2) for array in vibration_x_c6]).reshape(-1, 1)).flatten()
vibration_x_c6_var = scaler.fit_transform(np.array([np.var(array) for array in vibration_x_c6]).reshape(-1, 1)).flatten()
vibration_x_c6_max = scaler.fit_transform(np.array([np.max(array) for array in vibration_x_c6]).reshape(-1, 1)).flatten()
vibration_x_c6_skw = scaler.fit_transform(np.array([skew(array) for array in vibration_x_c6]).reshape(-1, 1)).flatten()
vibration_x_c6_krt = scaler.fit_transform(np.array([kurtosis(array) for array in vibration_x_c6]).reshape(-1, 1)).flatten()
vibration_x_c6_ptp = scaler.fit_transform(np.array([np.ptp(array) for array in vibration_x_c6]).reshape(-1, 1)).flatten()

vibration_x_c6_fft = np.abs([fft(array) for array in vibration_x_c6])
vibration_x_c6_sskw = scaler.fit_transform(np.array([skew(array) for array in vibration_x_c6_fft]).reshape(-1, 1)).flatten()
vibration_x_c6_skrt = scaler.fit_transform(np.array([kurtosis(array) for array in vibration_x_c6_fft]).reshape(-1, 1)).flatten()

vibration_x_c6_wave = scaler.fit_transform(np.array([wavelet_energy(array) for array in vibration_x_c6_fft]).reshape(-1, 1)).flatten()


# features for vibration_y_c6

vibration_y_c6_rms = scaler.fit_transform(np.sqrt([np.mean(array**2) for array in vibration_y_c6]).reshape(-1, 1)).flatten()
vibration_y_c6_var = scaler.fit_transform(np.array([np.var(array) for array in vibration_y_c6]).reshape(-1, 1)).flatten()
vibration_y_c6_max = scaler.fit_transform(np.array([np.max(array) for array in vibration_y_c6]).reshape(-1, 1)).flatten()
vibration_y_c6_skw = scaler.fit_transform(np.array([skew(array) for array in vibration_y_c6]).reshape(-1, 1)).flatten()
vibration_y_c6_krt = scaler.fit_transform(np.array([kurtosis(array) for array in vibration_y_c6]).reshape(-1, 1)).flatten()
vibration_y_c6_ptp = scaler.fit_transform(np.array([np.ptp(array) for array in vibration_y_c6]).reshape(-1, 1)).flatten()

vibration_y_c6_fft = np.abs([fft(array) for array in vibration_y_c6])
vibration_y_c6_sskw = scaler.fit_transform(np.array([skew(array) for array in vibration_y_c6_fft]).reshape(-1, 1)).flatten()
vibration_y_c6_skrt = scaler.fit_transform(np.array([kurtosis(array) for array in vibration_y_c6_fft]).reshape(-1, 1)).flatten()

vibration_y_c6_wave = scaler.fit_transform(np.array([wavelet_energy(array) for array in vibration_y_c6_fft]).reshape(-1, 1)).flatten()


# features for vibration_z_c6

vibration_z_c6_rms = scaler.fit_transform(np.sqrt([np.mean(array**2) for array in vibration_z_c6]).reshape(-1, 1)).flatten()
vibration_z_c6_var = scaler.fit_transform(np.array([np.var(array) for array in vibration_z_c6]).reshape(-1, 1)).flatten()
vibration_z_c6_max = scaler.fit_transform(np.array([np.max(array) for array in vibration_z_c6]).reshape(-1, 1)).flatten()
vibration_z_c6_skw = scaler.fit_transform(np.array([skew(array) for array in vibration_z_c6]).reshape(-1, 1)).flatten()
vibration_z_c6_krt = scaler.fit_transform(np.array([kurtosis(array) for array in vibration_z_c6]).reshape(-1, 1)).flatten()
vibration_z_c6_ptp = scaler.fit_transform(np.array([np.ptp(array) for array in vibration_z_c6]).reshape(-1, 1)).flatten()

vibration_z_c6_fft = np.abs([fft(array) for array in vibration_z_c6])
vibration_z_c6_sskw = scaler.fit_transform(np.array([skew(array) for array in vibration_z_c6_fft]).reshape(-1, 1)).flatten()
vibration_z_c6_skrt = scaler.fit_transform(np.array([kurtosis(array) for array in vibration_z_c6_fft]).reshape(-1, 1)).flatten()

vibration_z_c6_wave = scaler.fit_transform(np.array([wavelet_energy(array) for array in vibration_z_c6_fft]).reshape(-1, 1)).flatten()
```

```
c6_features = np.stack((force_x_c6_rms, force_x_c6_var, force_x_c6_max,
                        force_x_c6_skw, force_x_c6_krt, force_x_c6_ptp,
                        force_x_c6_sskw, force_x_c6_skrt, force_x_c6_wave,
                        force_y_c6_rms, force_y_c6_var, force_y_c6_max,
                        force_y_c6_skw, force_y_c6_krt, force_y_c6_ptp,
                        force_y_c6_sskw, force_y_c6_skrt, force_y_c6_wave,
                        force_z_c6_rms, force_z_c6_var, force_z_c6_max,
                        force_z_c6_skw, force_z_c6_krt, force_z_c6_ptp,
                        force_z_c6_sskw, force_z_c6_skrt, force_z_c6_wave,
                        vibration_x_c6_rms, vibration_x_c6_var, vibration_x_c6_max,
                        vibration_x_c6_skw, vibration_x_c6_krt, vibration_x_c6_ptp,
                        vibration_x_c6_sskw, vibration_x_c6_skrt, vibration_x_c6_wave,
                        vibration_y_c6_rms, vibration_y_c6_var, vibration_y_c6_max,
                        vibration_y_c6_skw, vibration_y_c6_krt, vibration_y_c6_ptp,
                        vibration_y_c6_sskw, vibration_y_c6_skrt, vibration_y_c6_wave,
                        vibration_z_c6_rms, vibration_z_c6_var, vibration_z_c6_max,
                        vibration_z_c6_skw, vibration_z_c6_krt, vibration_z_c6_ptp,
                        vibration_z_c6_sskw, vibration_z_c6_skrt, vibration_z_c6_wave
                        ),
                       axis = 1)
```

```
c6_target = scaler.fit_transform(c6_target.reshape(-1, 1)).flatten()
```

```
fig, ax = plt.subplots(3,3)
ax[0,0].plot(force_x_c1_rms)
ax[0,0].set_title("RMS")
ax[0,1].plot(force_x_c1_var)
ax[0,1].set_title("Variance")
ax[0,2].plot(force_x_c1_max)
ax[0,2].set_title("Max")
ax[1,0].plot(force_x_c1_skw)
ax[1,0].set_title("Skewness")
ax[1,1].plot(force_x_c1_krt)
ax[1,1].set_title("Kurtosis")
ax[1,2].plot(force_x_c1_ptp)
ax[1,2].set_title("Peak to Peak")
ax[2,0].plot(force_x_c1_sskw)
ax[2,0].set_title("Spectral skewness")
ax[2,1].plot(force_x_c1_skrt)
ax[2,1].set_title("Spectral Kurtosis")
ax[2,2].plot(force_x_c1_wave)
ax[2,2].set_title("Wavelet energy")

plt.subplots_adjust(left=0.1,
                    bottom=0.2,
                    right=0.9,
                    top=1.5,
                    wspace=0.6,
                    hspace=0.4)
```

Models

```
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_squared_error, mean_absolute_error, mean_absolute_percentage_error
from keras import Input
from keras.models import Sequential
from keras.layers import Dense, Activation
```

```
X_train = np.vstack((c1_features, c4_features))
y_train = np.hstack((c1_target, c4_target))
X_test = c6_features
y_test = c6_target
```

Linear Regression without PCA

```
lrr = LinearRegression().fit(X_train, y_train)
pred = lrr.predict(X_test)
plt.plot(pred, label = 'Pred')
plt.plot(y_test, label = 'True')
plt.legend()
plt.title('Linear Regression without PCA')
plt.ylabel('Wear')
rmse = np.sqrt(mean_squared_error(y_test, pred))
mae = mean_absolute_error(y_test, pred)
mape = mean_absolute_percentage_error(y_test, pred)
print('Test RMSE: %.3f' % rmse)
print('Test MAE: %.3f' % mae)
print('Test MAPE: %.3f' % mape)
```

SVR without PCA

```
# SVR?
```

```
svr = SVR(kernel = 'poly', gamma = 10, C = 10).fit(X_train, y_train)
pred = svr.predict(X_test)
plt.plot(pred, label = 'Pred')
plt.plot(y_test, label = 'True')
plt.legend()
plt.title('SVR (poly) without PCA')
rmse = np.sqrt(mean_squared_error(y_test, pred))
mae = mean_absolute_error(y_test, pred)
mape = mean_absolute_percentage_error(y_test, pred)
print('Test RMSE: %.3f' % rmse)
print('Test MAE: %.3f' % mae)
print('Test MAPE: %.3f' % mape)
```

```
svr = SVR(kernel = 'sigmoid', gamma = 10, C = 10).fit(X_train, y_train)
pred = svr.predict(X_test)
plt.plot(pred, label = 'Pred')
plt.plot(y_test, label = 'True')
plt.legend()
plt.title('SCR (sigmoid) without PCA')
rmse = np.sqrt(mean_squared_error(y_test, pred))
mae = mean_absolute_error(y_test, pred)
mape = mean_absolute_percentage_error(y_test, pred)
print('Test RMSE: %.3f' % rmse)
print('Test MAE: %.3f' % mae)
print('Test MAPE: %.3f' % mape)
```

```
svr = SVR(kernel = 'rbf', gamma = 10, C = 10).fit(X_train, y_train)
pred = svr.predict(X_test)
plt.plot(pred, label = 'Pred')
plt.plot(y_test, label = 'True')
plt.legend()
plt.title('SVR (rbf) without PCA')
rmse = np.sqrt(mean_squared_error(y_test, pred))
mae = mean_absolute_error(y_test, pred)
mape = mean_absolute_percentage_error(y_test, pred)
print('Test RMSE: %.3f' % rmse)
print('Test MAE: %.3f' % mae)
print('Test MAPE: %.3f' % mape)
```

Neural Network without PCA

```
model = Sequential()
model.add(Input(shape = (54, )))
model.add(Dense(64, activation = 'relu'))
model.add(Dense(32, activation = 'relu'))
model.add(Dense(16, activation = 'sigmoid'))
model.add(Dense(1))
model.compile(optimizer='sgd', loss='mse')

model.fit(X_train, y_train, batch_size = 16, epochs=20)

pred = model.predict(X_test)

plt.plot(pred, label = 'Pred')
plt.plot(y_test, label = 'True')
plt.legend()
plt.title('Neural Network without PCA')
rmse = np.sqrt(mean_squared_error(y_test, pred))
```

```
mae = mean_absolute_error(y_test, pred)
mape = mean_absolute_percentage_error(y_test, pred)
print('Test RMSE: %.3f' % rmse)
print('Test MAE: %.3f' % mae)
print('Test MAPE: %.3f' % mape)
```

PCA

```
from sklearn.decomposition import PCA
```

```
pca = PCA(n_components = 12)
pca.fit(X_train)
a = pca.explained_variance_ratio_
np.cumsum(a)
```

```
print(abs( pca.components_ ))
```

```
X_train_decomposed = pca.transform(X_train)
X_test_decomposed = pca.transform(X_test)
```

```
# pca.get_feature_names_out(input_features=('force_x_c1_rms', 'force_x_c1_var', 'force_x_c1_max',
#                          'force_x_c1_skw', 'force_x_c1_krt', 'force_x_c1_ptp',
#                          'force_x_c1_sskw', 'force_x_c1_skrt', 'force_x_c1_wave',
#                          'force_y_c1_rms', 'force_y_c1_var', 'force_y_c1_max',
#                          'force_y_c1_skw', 'force_y_c1_krt', 'force_y_c1_ptp',
#                          'force_y_c1_sskw', 'force_y_c1_skrt', 'force_y_c1_wave',
#                          'force_z_c1_rms', 'force_z_c1_var', 'force_z_c1_max',
#                          'force_z_c1_skw', 'force_z_c1_krt', 'force_z_c1_ptp',
#                          'force_z_c1_sskw', 'force_z_c1_skrt', 'force_z_c1_wave',
#                          'vibration_x_c1_rms', 'vibration_x_c1_var', 'vibration_x_c1_max',
#                          'vibration_x_c1_skw', 'vibration_x_c1_krt', 'vibration_x_c1_ptp',
#                          'vibration_x_c1_sskw', 'vibration_x_c1_skrt', 'vibration_x_c1_wave',
#                          'vibration_y_c1_rms', 'vibration_y_c1_var', 'vibration_y_c1_max',
#                          'vibration_y_c1_skw', 'vibration_y_c1_krt', 'vibration_y_c1_ptp',
#                          'vibration_y_c1_sskw', 'vibration_y_c1_skrt', 'vibration_y_c1_wave',
#                          'vibration_z_c1_rms', 'vibration_z_c1_var', 'vibration_z_c1_max',
#                          'vibration_z_c1_skw', 'vibration_z_c1_krt', 'vibration_z_c1_ptp',
#                          'vibration_z_c1_sskw', 'vibration_z_c1_skrt', 'vibration_z_c1_wave'
#                          ))
```

Linear Regression with PCA

```
lrr = LinearRegression().fit(X_train_decomposed, y_train)
pred = lrr.predict(X_test_decomposed)
plt.plot(pred, label = 'Pred')
plt.plot(y_test, label = 'True')
plt.legend()
plt.title('Linear Regression with PCA')
rmse = np.sqrt(mean_squared_error(y_test, pred))
mae = mean_absolute_error(y_test, pred)
mape = mean_absolute_percentage_error(y_test, pred)
print('Test RMSE: %.3f' % rmse)
print('Test MAE: %.3f' % mae)
print('Test MAPE: %.3f' % mape)
```

SVR with PCA

```
model = SVR()
model.fit(X_train_decomposed, y_train)

# print prediction results
predictions = model.predict(X_test_decomposed)
```

```
from sklearn.model_selection import GridSearchCV

param_grid = {'C': [0.1, 1, 10, 100, 1000],
              'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
              'kernel': ['rbf','poly']}
grid = GridSearchCV(SVR(), param_grid, refit = True, verbose = 3)
grid.fit(X_train_decomposed, y_train)
```

```
print(grid.best_params_)  #Best parameters based on hyper-parameter tuninig
print(grid.best_estimator_)
```

```
svr = SVR(kernel = 'rbf', gamma = 0.0001, C = 10).fit(X_train_decomposed, y_train)
pred = svr.predict(X_test_decomposed)
plt.plot(pred, label = 'Pred')
plt.plot(y_test, label = 'True')
plt.legend()
plt.title('SVR with PCA')
rmse = np.sqrt(mean_squared_error(y_test, pred))
mae = mean_absolute_error(y_test, pred)
mape = mean_absolute_percentage_error(y_test, pred)
print('Test RMSE: %.3f' % rmse)
print('Test MAE: %.3f' % mae)
print('Test MAPE: %.3f' % mape)
```

```
svr = SVR(kernel = 'sigmoid', gamma = 10, C = 10).fit(X_train_decomposed, y_train)
pred = svr.predict(X_test_decomposed)
plt.plot(pred, label = 'Pred')
plt.plot(y_test, label = 'True')
plt.legend()
rmse = np.sqrt(mean_squared_error(y_test, pred))
mae = mean_absolute_error(y_test, pred)
mape = mean_absolute_percentage_error(y_test, pred)
print('Test RMSE: %.3f' % rmse)
print('Test MAE: %.3f' % mae)
print('Test MAPE: %.3f' % mape)
```

```
svr = SVR(kernel = 'rbf', gamma = 10, C = 10).fit(X_train_decomposed, y_train)
pred = svr.predict(X_test_decomposed)
plt.plot(pred, label = 'Pred')
plt.plot(c6_target, label = 'True')
plt.legend()
rmse = np.sqrt(mean_squared_error(y_test, pred))
mae = mean_absolute_error(y_test, pred)
mape = mean_absolute_percentage_error(y_test, pred)
print('Test RMSE: %.3f' % rmse)
print('Test MAE: %.3f' % mae)
print('Test MAPE: %.3f' % mape)
```

Neural Network with PCA

```
model = Sequential()
model.add(Input(shape = (9, )))
model.add(Dense(64, activation = 'relu'))
model.add(Dense(32, activation = 'relu'))
model.add(Dense(16, activation = 'sigmoid'))
model.add(Dense(1))
model.compile(optimizer='rmsprop', loss='mse')

model.fit(X_train_decomposed, y_train, batch_size = 128, epochs=15)

pred = model.predict(X_test_decomposed)

plt.plot(pred, label = 'Pred')
plt.plot(y_test, label = 'True')
plt.legend()
plt.title('Neural Network with PCA')
rmse = np.sqrt(mean_squared_error(y_test, pred))
mae = mean_absolute_error(y_test, pred)
mape = mean_absolute_percentage_error(y_test, pred)
print('Test RMSE: %.3f' % rmse)
print('Test MAE: %.3f' % mae)
print('Test MAPE: %.3f' % mape)
```

ARMA/ARIMA

```
import warnings
from math import sqrt
from statsmodels.tsa.arima.model import ARIMA
```

```
c6_target = pd.read_csv(filepath+'Testing Dataset/'+'c6_wear.csv', header = 0).drop(columns=['cut']).to_numpy()
c6_target = np.linalg.norm(c6_target, axis=1)
```

```
x = np.arange(1, 316, 1)
train_size = int(len(c6_target) * 0.66)
x_train, x_test = x[0:train_size], x[train_size:]
y_train, y_test = c6_target[0:train_size], c6_target[train_size:]


# evaluate an ARIMA model for a given order (p,d,q)
def evaluate_arima_model(X, arima_order):
    # prepare training dataset
    train_size = int(len(X) * 0.66)
    train, test = X[0:train_size], X[train_size:]
    history = [x for x in train]
    # make predictions
    predictions = list()
    for t in range(len(test)):
      model = ARIMA(history, order=arima_order)
      model_fit = model.fit()
      yhat = model_fit.forecast()[0]
      predictions.append(yhat)
      history.append(test[t])
    # calculate out of sample error
    rmse = sqrt(mean_squared_error(test, predictions))
    return rmse


# evaluate combinations of p, d and q values for an ARIMA model
def evaluate_models(dataset, p_values, d_values, q_values):
 dataset = dataset.astype('float32')
 best_score, best_cfg = float("inf"), None
 for p in p_values:
  for d in d_values:
    for q in q_values:
      order = (p,d,q)
      try:
        rmse = evaluate_arima_model(dataset, order)
        if rmse < best_score:
          best_score, best_cfg = rmse, order
          print('ARIMA%s RMSE=%.3f' % (order,rmse))
      except:
        continue
 print('Best ARIMA%s RMSE=%.3f' % (best_cfg, best_score))


# grid search ARMA parameters for time series (d = 0)

p_values = [1, 2, 4, 6, 8, 10]
d_values = [0]
q_values = range(0, 3)
evaluate_models(c6_target, p_values, d_values, q_values)


y_test = c6_target


from matplotlib import pyplot as plt
```

```
warnings.filterwarnings("ignore")

X = c6_target
train_size = int(len(X) * 0.66)
train, test = X[0:train_size], X[train_size:]

history = [x for x in train]
pred = list()
```

```
# grid search ARIMA parameters for time series (d > 0)

p_values = [1, 2, 4, 6, 8, 10]
d_values = range(1, 3)
q_values = range(0, 3)
evaluate_models(c6_target, p_values, d_values, q_values)
```

```
warnings.filterwarnings("ignore")

X = c6_target
train_size = int(len(X) * 0.66)
train, test = X[0:train_size], X[train_size:]

history = [x for x in train]
pred = list()

# walk-forward prediction
for t in range(len(test)):
  model = ARIMA(history, order=(1, 2, 0))   # replace with best values obtained from the grid search
  model_fit = model.fit()
  output = model_fit.forecast()
  yhat = output[0]
  pred.append(yhat)
  obs = test[t]
  history.append(obs)
#  print('predicted=%f, expected=%f' % (yhat, obs))

# evaluate predictions
rmse = np.sqrt(mean_squared_error(y_test, pred))
mae = mean_absolute_error(y_test, pred)
mape = mean_absolute_percentage_error(y_test, pred)
print('Test RMSE: %.3f' % rmse)
print('Test MAE: %.3f' % mae)
print('Test MAPE: %.3f' % mape)

# plot predictions against actual outcomes
plt.plot(np.arange(0, train_size, 1), train, label = 'train')
plt.plot(np.arange(train_size, len(X), 1), test, label = 'test')
plt.plot(np.arange(train_size, len(X), 1), pred, color='red', label = 'pred')
plt.legend()
```