IM39003- OHM Lab

Term Project

Remaining Useful Life prediction

Shivam Singh

17IM10021

Problem Statement:

Drilling process, one of the most commonly used machining processes, is selected here as a test-bed. The objective is to predict the Remaining Useful Life (RUL) of drill-bit during the machining process by utilizing **thrust-force** and **torque** signals captured by a dynamometer during the drilling cycle (constituting a logical sensor signal segment). Tests were conducted on HAAS VF-1 CNC Machining Center with Kistler 9257B piezo-dynamometer (sampled at 250Hz) to drill holes in ¼ inch stainless steel bars. High-speed twist drill-bits with two flutes were operated at feed rate of 4.5 inch/minute and spindle-speed at 800 rpm without coolant. Each drill-bit was used until it reached a state of physical failure either due to excessive wear or due to gross plastic deformation of the tool tip due to excessive temperature (resulting from excessive wear).

The objective of the project is to build a model from the data which predicts the **RUL** by optimizing the performance evaluated in the form of **Median RMSE**for testing dataset made up of the last five holes for each drill bit.

Solution:

Step 1: Converting given data in dbhole.mat file to RUL_data.csv file.

Step 2:After conversion we get 28 features to be used for predicting RUL

```
['dxhy.name',
 'force.mean',
 'torque.mean',
 'force.median',
 'torque.median',
 'force.sd',
 'torque.sd',
 'force.max',
 'torque.max',
 'force.skewness',
 'torque.skewness',
 'force.kurtosis',
 'torque.kurtosis',
 'force.IQR',
 'torque.IQR',
 'force.coefficient_of_variation',
 'torque.coefficient_of_variation',
 'spearman.correlation',
 'pearson.correlation',
 'spectral_density.df',
 'spectral_density.bandwidth',
 'dual.gini',
 'force.gini',
 'torque.gini',
 'force.autocorr.mean',
 'force.autocorr.sd',
 'torque.autocorr.mean',
 'torque.autocorr.sd',
 'RUL']
```

Step 3: Splitting the dataset

-Dataset is split using the following rule:

  -Train data- has first 10 drill bits

  -Test data- has last 4 drill bits

-Data is scaled using the MinMaxScaler() function.

Step 4: Hyperparamter are set for the Neural Networks to be used

```
params = {
    "epochs": [10, 20],
    "batch_size": [10,20,40],
    "n_layers": [1, 2],
    "n_neurons": [20, 40, 60],
    "dropout": [0.1, 0.2],
    "optimizers": ["nadam", "adam"],
    "activations": ["relu", "sigmoid"],
    "last_layer_activations": ["relu"],
    "losses": ["mean_squared_error"],
    "metrics": ["MSE"]
}
```

Step 5: Using GA for tuning Hyperparameters for the Neural Network

- Use the link https://github.com/subpath/neuro-evolution.git to get the pre-built-in package for hyperparameter tuning.

Best parameters come out to be:

"epochs": 20,
"batch_size": 40,
"n_layers": 2,
"n_neurons": 60,
"dropout": 0.2,
"optimizers": "nadam",
"activations": "relu",
"last_layer_activations": "relu",
"losses": "mean_squared_error",
"metrics": "MSE"

Step 6: The Neural Network

-The Keras package along with the values obtained above using GA is used to build the model.

- We fit the model to the training data set and predict the values of the test set.

```
import keras
from keras.models import Sequential
from keras.layers import Dense
```

```
mod = Sequential()
mod.add(Dense(60, input_dim=27, activation='relu'))
mod.add(Dense(60, activation='relu'))
mod.add(Dense(60,  activation='relu'))
```

```
mod.add(Dense(1, activation='relu'))

mod.compile(loss='mean_squared_error', optimizer='nadam', metrics=['MSE'])

history = mod.fit(X_train_scaled, Y_train, epochs=20, batch_size=40)

predicted=mod.predict(X_test_scaled)

predicted=pd.Series(predicted.reshape(-1))
```

Step 7: Performance test

-As stated in the problem statement, we analysis the accuracy of our model using the Median RMSE on test data.

```
import statistics
print("Median RMSE:")
statistics.median([RMSE_h11,RMSE_h12,RMSE_h13,RMSE_h14])
```

```
Median RMSE:

0.6324555320336759
```

Conclusion:

The performance criteria we get is 0.63245. Hence, this model can be used for predicting RUL.