Weekly Progress Report 7

Submitted by:

Sandesh Pabitwar

Modern Education Society's College of Engineering Pune

Project Title: INTP22-ML-2 Remaining Usable Life Estimation (NASA Turbine dataset)

Objectives:

1. Understand the components and working of Turbo Engine.
2. Understand the dataset and analyze dataset
3. To learn about different Python libraries and their implementation.
4. To develop model for remaining usable life estimation

Task done in this week:

1. Done hyperparameter tuning on random forest regression.
2. Referred research paper to implement CNN model.
3. Processed data to train a CNN model.
4. Successfully implemented CNN model.

Done hyperparameter tuning on Random Forest algorithm.

In previous week I have implemented random forest regression, to improve the accuracy score in this week tried to do hyperparameter tuning on Random Forest.

Using the get_params() function I got all the 17 parameters available for Random forest.

- Parameters available for Random forest are {'bootstrap', 'ccp_alpha', 'criterion', 'max_depth','max_features' , 'max_leaf_nodes', 'max_samples' , 'min_impurity_decrease', 'min_samples_leaf', 'min_samples_split' , 'min_weight_fraction_leaf' , 'n_estimators' , 'n_jobs' , 'oob_score', 'random_state', 'verbose' , 'warm_start'}

- Out of those parameters I selected 6 parameters for tuning.

```
# Create the param grid
param_grid = {'n_estimators': n_estimators,
              'max_features': max_features,
              'max_depth': max_depth,
              'min_samples_split': min_samples_split,
              'min_samples_leaf': min_samples_leaf,
              'bootstrap': bootstrap}
print(param_grid)
```

- To get the best parameters I have used Randomize search and Grid search.
- Best parameters returned using Randomize search are
    - 'bootstrap': True,
    - 'max_depth': 4,
    - 'max_features': 'sqrt',
    - 'min_samples_leaf': 1,
    - 'min_samples_split': 2,
    - 'n_estimators': 72
- Best parameters returned using Grid search are
    - 'criterion': 'gini',
    - 'max_depth': 55,
    - 'max_features': 'log2',
    - 'min_samples_leaf': 0.001,
    - 'min_samples_split': 0.001,
    - 'n_estimators': 190
- Using Randomize search accuracy was
    - Train Accuracy -: 0.714
    - Test Accuracy -: 0.558
- Using Grid search for tuning accuracy was
    - Train Accuracy - : 0.6764420746485701
    - Test Accuracy - : 0.6966319360310153

Referred research paper to implement CNN model

Referred the paper titled as “A Remaining Useful Life Prognosis of Turbofan Engine Using

Temporal and Spatial Feature Fusion” by Cheng Peng , Yufeng Chen , Qing Chen , Zhaohui Tang , Lingling Li and Weihua Gui .

This research paper explains about what FCLCNN model and the sequential layers added in model.

According to paper using this model RMSE is 11.17, 9.99 on FD001, FD003 respectively.

Processed data to train a CNN model

CNN models are mostly used for image processing to use CNN on time series data some processing is required.

To train model firstly I have used minmax scalar function to convert all data into the range of 0 to 1.

To feed the data into CNN model some processing is required. To do that I have used time series generator.

```
win_length = 25   ######### Sliding Window Length
feature_num = 13  ######### Total number of features

ts_generator = TimeseriesGenerator(features,target,length=win_length,sampling_rate=1,batch_size=1)
```

Successfully implemented CNN model.

```
model=Sequential()
# CNN
model.add(Conv2D(filters=32, kernel_size=3, activation='relu', input_shape=(win_length,feature_num,1)))
model.add(Conv2D(filters=64, kernel_size=3, activation='relu'))
model.add(Conv2D(filters=128, kernel_size=3, activation='relu'))
model.add(Flatten())
model.add(Dense(1, activation='linear'))

model.compile(loss='mean_squared_error',optimizer='adam',metrics=['mean_squared_error'])
```

In the CNN model I have added 5 layers. 1st is input layer, 2nd is 2D convolution layer, 3rd is same as 2nd layer, 4th is flatten layer and 5th is dense and last is compiling layer.

3 convolution layers convolves over the spatial and time dimensions of input data.

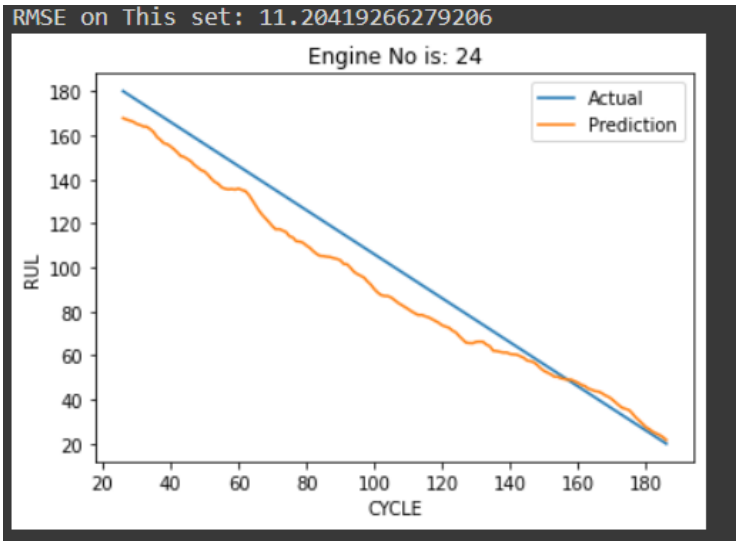
Flatten layer is used to convert 2D data into a single 1D array.

Dense layer will receive input from each neuron of previous layer.

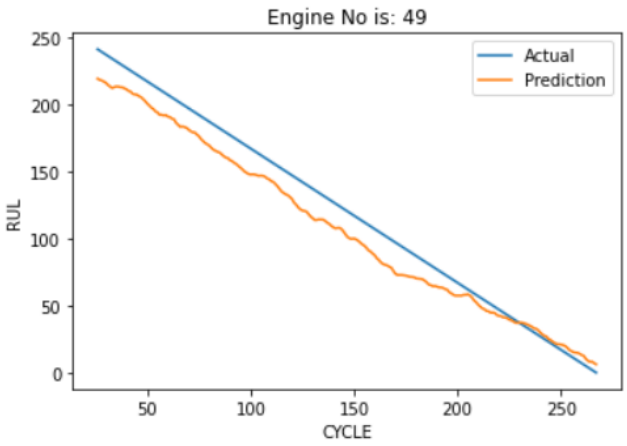
At last compile is used to compile all the layers with adam optimizer.

Results:

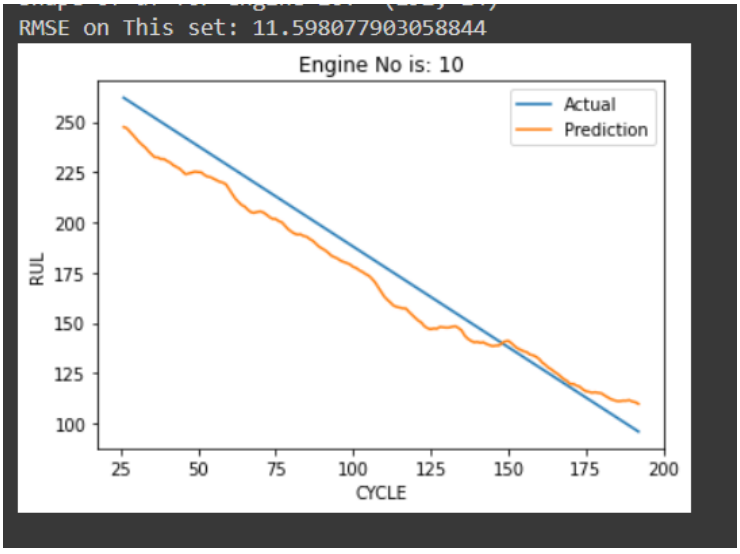
RMSE on This set: 11.20419266279206



RMSE on This set: 15.304817515075763
R2 on This set: 0.952002983623837



RMSE on This set: 11.598077903058844



RMSE on This set: 7.988997154076171