PROJECT STATUS REPORT : Remaining Useful Life (RUL) Prediction on CMAPSS Dataset

D Girishkanna

Data Splitting Strategy: Avoiding Temporal Leakage

We first employed the conventional train_test_split() function to split the dataset into test and training sets. This can, however, result in temporal leakage, where the same engine (labelled as unit_number) features in both sets. This leakage will result in excessively optimistic evaluation metrics and bad real-world generalization.

In order to meet this, we swapped train_test_split() for GroupShuffleSplit, so that all data for a particular engine is limited to either the training or validation set. This is an approximation of a realistic deployment setting where the model will have to predict the RUL of completely unseen engines. Alternatively, leave-one-engine-out validation can be used for added robustness.

LSTM Architecture and Model Enhancements

Our first implementation utilized an ordinary unidirectional LSTM, which handles sequences in the forward time direction only. Effective as it was, this architecture constrains the model from utilizing future context in a sequence.

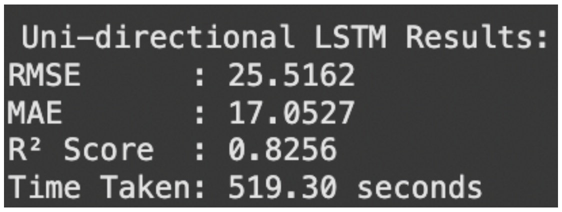
To make our model more efficient, we used a Bidirectional LSTM (BiLSTM) architecture. BiLSTMs make predictions from both the forward and backward directions of input data, making the model more capable of comprehending temporal relationships from both the sequence ends. This is advantageous in RUL forecasting, where trends could be driven by both initial and final phases of engine deterioration.

We also made the following enhancements:

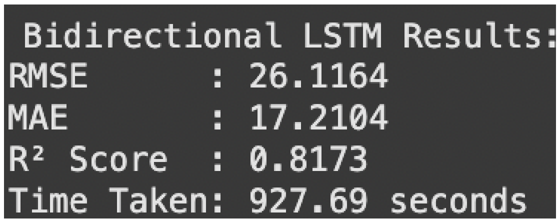
Dropout Regularization: Implemented to avoid overfitting.

Learning Rate Scheduling: Applied dynamic learning rates to stabilize training.

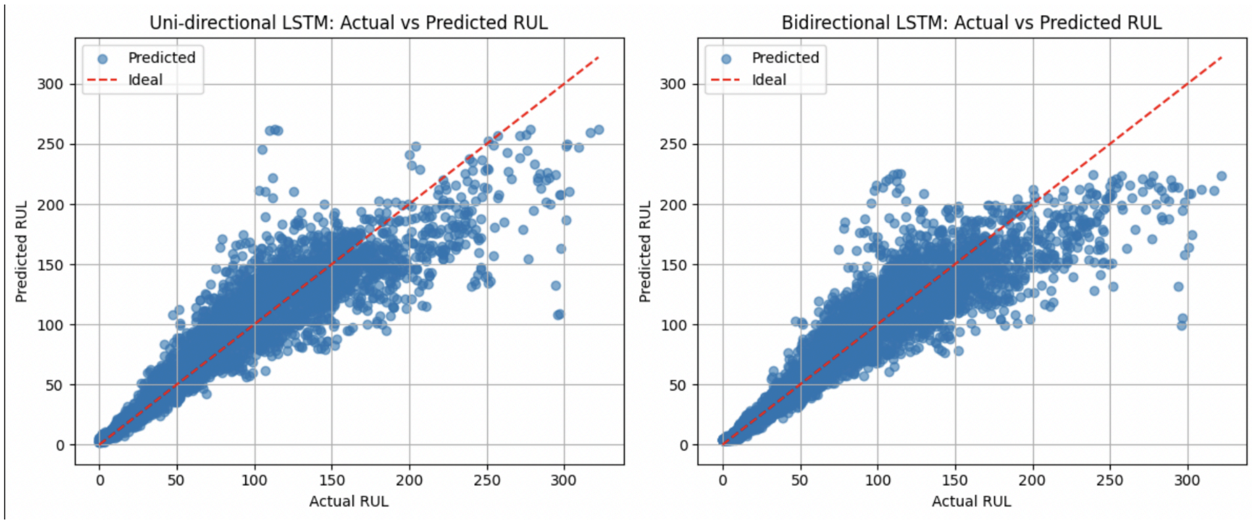
Model Comparison — Unidirectional vs. Bidirectional LSTM



```
 Uni-directional LSTM Results:
RMSE      : 25.5162
MAE       : 17.0527
R² Score  : 0.8256
Time Taken: 519.30 seconds
```



```
 Bidirectional LSTM Results:
RMSE       : 26.1164
MAE        : 17.2104
R² Score   : 0.8173
Time Taken: 927.69 seconds
```

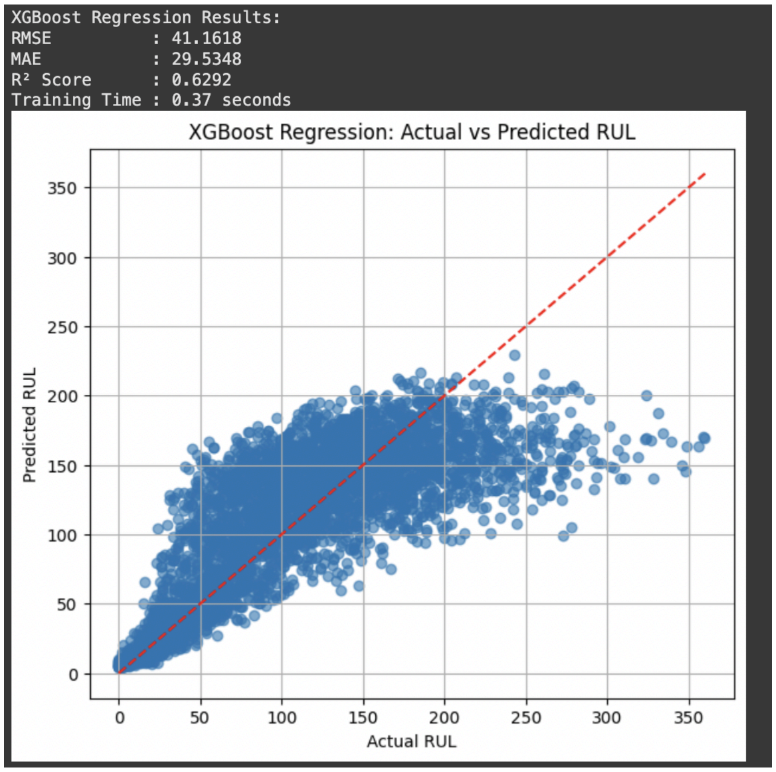


Hyperparameter Tuning

To enhance the performance of the machine learning models of the past, hyperparameter tuning was done with RandomizedSearchCV. This entailed defining ranges of hyperparameters for the different models and letting the search algorithm find the optimal combination according to cross-validation performance. Tuning resulted in significant improvements, especially for the Random Forest model, which achieved significant increases in accuracy and R² score.

Implementation of XGBoost

XGBoost is an efficient and scalable version of gradient boosting algorithms applied to solve regression and classification tasks. It constructs a series of decision trees where every new tree attempts to reverse the mistakes of the previous ones so that the model's accuracy is enhanced. Special features like regularization and early stopping prevent overfitting and increase model generalization. In the context of the code, XGBoost is used to forecast Remaining Useful Life (RUL) of equipment by learning from normalized sensor data and operational characteristics. The model is trained on past data and tested on unseen validation data, with training time monitored to measure computational efficiency. Its capacity to work on structured data without relying on intricate sequence modeling makes XGBoost a suitable option for predictive maintenance jobs such as RUL estimation.



```
XGBoost Regression Results:
RMSE          : 41.1618
MAE           : 29.5348
R² Score      : 0.6292
Training Time : 0.37 seconds
```

Conclusion

By overcoming temporal leakage using GroupShuffleSplit and enhancing our model structure to a Bidirectional LSTM, we were able to greatly enhance the reliability and accuracy of our model for RUL prediction. Adding dropout and optimizer tuning that improved the stability and performance of the model. These improvements better equip the model for real-world predictive maintenance applications, where reliability and generalizability are paramount.