ME 781: Course Project

Introduction:

Predictive maintenance techniques are designed to help anticipate equipment failures to allow for advance scheduling of corrective maintenance, thereby preventing unexpected equipment downtime, improving service quality for customers, and reducing the additional cost caused by over-maintenance in preventative maintenance policies. In this project we attempt to use ML and AI techniques to get insights for machine maintenance and failure prevention.

This is purely a technical report. Other project details are added in the brochure.

Dataset:

NASA Prognostic Dataset. Contains 21 Sensor data and 3 operational Settings.
The engine is operating normally at the start of each time series and starts to degrade at some point during the series. In the training set, the degradation grows in magnitude until a predefined threshold is reached beyond which it is not preferable to operate the engine. In the test set, the time series ends sometime prior to complete degradation.

Models used for Predictive Maintenance

1. Exponential Degradation model for RUL Prediction
2. Similarity-based model for RUL Prediction
3. LSTM model for RUL Prediction
4. LSTM model for binary classification
5. CNN-SVM for binary classification

1.

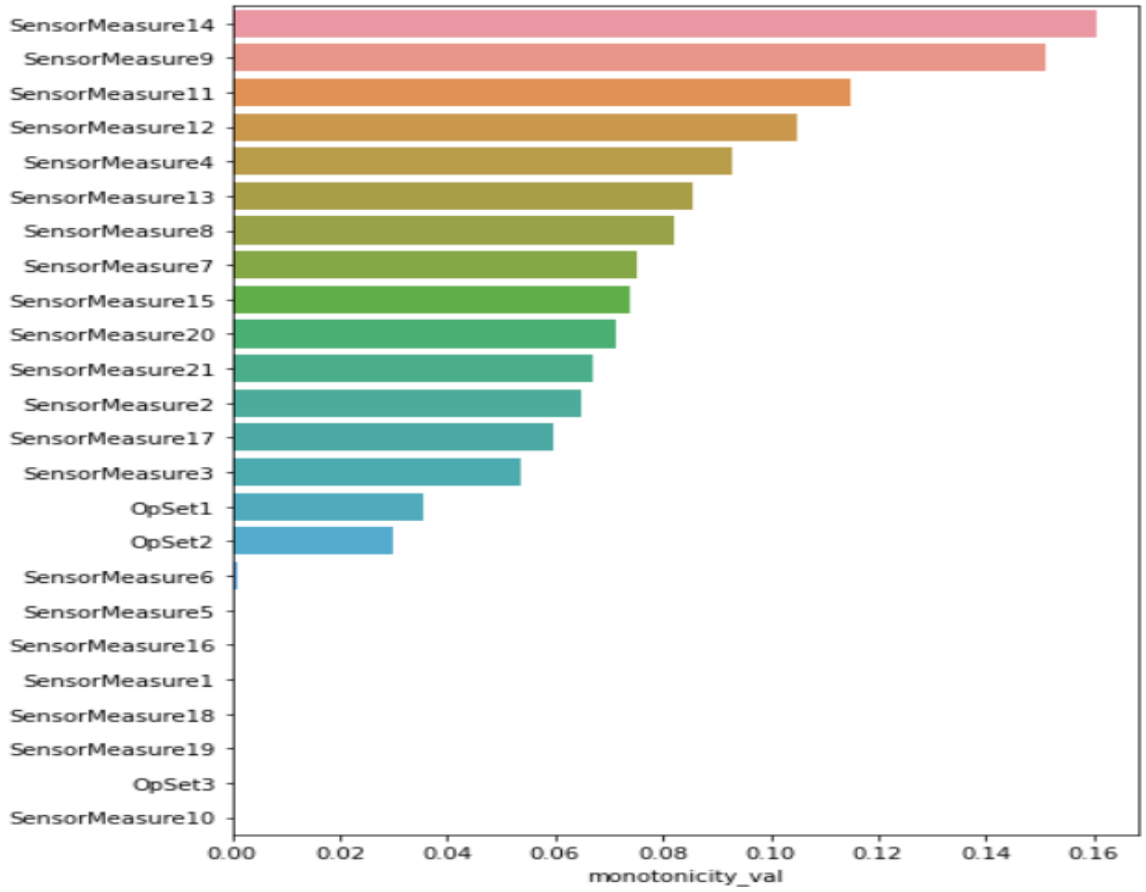
1. Exponential Degradation model for RUL Prediction:

- ## Data preprocessing

The data preprocessing starts with calculating on the basis of the ground truth given, mainly the RUL column. The data currently contains a lot of noise and sharp changes, in order to cope with this, a moving window is rolled which averages the value at any instant using its neighbouring values.

Then some of the columns are dropped out of the data as they are not relevant because of their minimal variation even if the machine fails. This is done using **Monotonicity,** which measures the variation of the sensor data in the next time step. The columns below a threshold value were dropped.

$$Monotonicity(x) = \frac{1}{m}\sum_{j=1}^{m}\frac{|\text{number of positive diff}(x_j) - \text{number of negative diff}(x_j)|}{n-1}$$



Then data is normalized using MinMax scaler. Afterwards, using Principal Component Analysis (**PCA**) the dimension of the data is reduced.

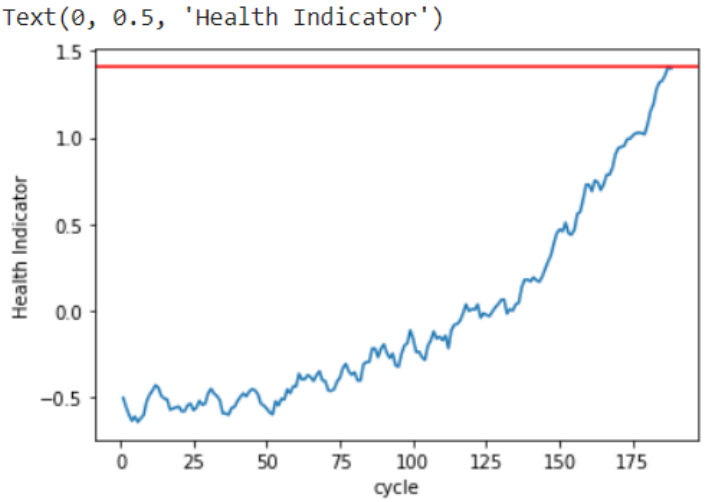
- ## Model details

In this method an exponential model is created by estimating the coefficients for each of the machine units and an average value is estimated for them. After estimating the coefficients, the model is used to predict the “Remaining Useful Life (**RUL**)”.

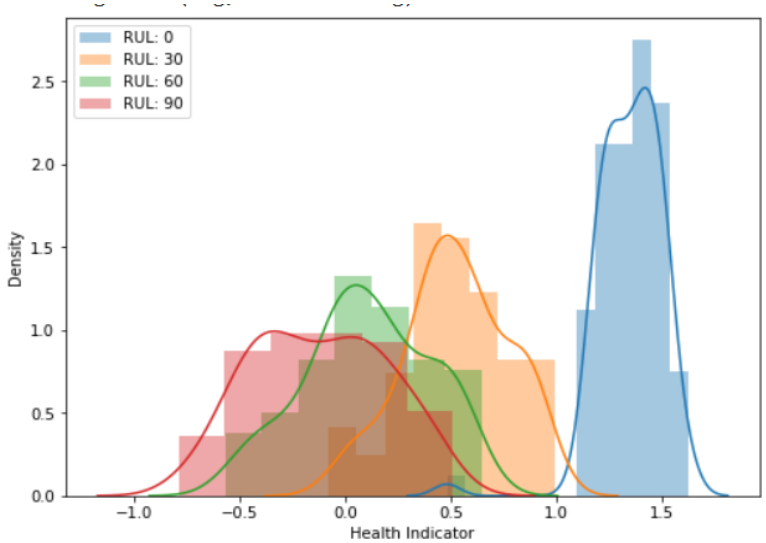
$$h(t) = \phi + \theta\, exp(\beta t)$$

- ## Results and plots:

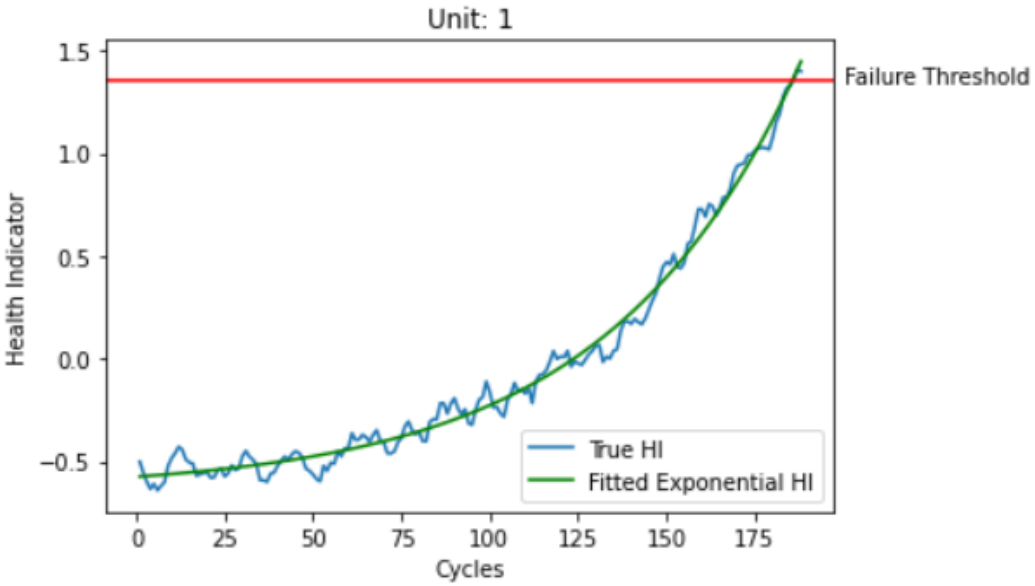
One of the PCA component is taken as the Health Indicator, as it represents the correct meaning of the component feature, as can be seen below



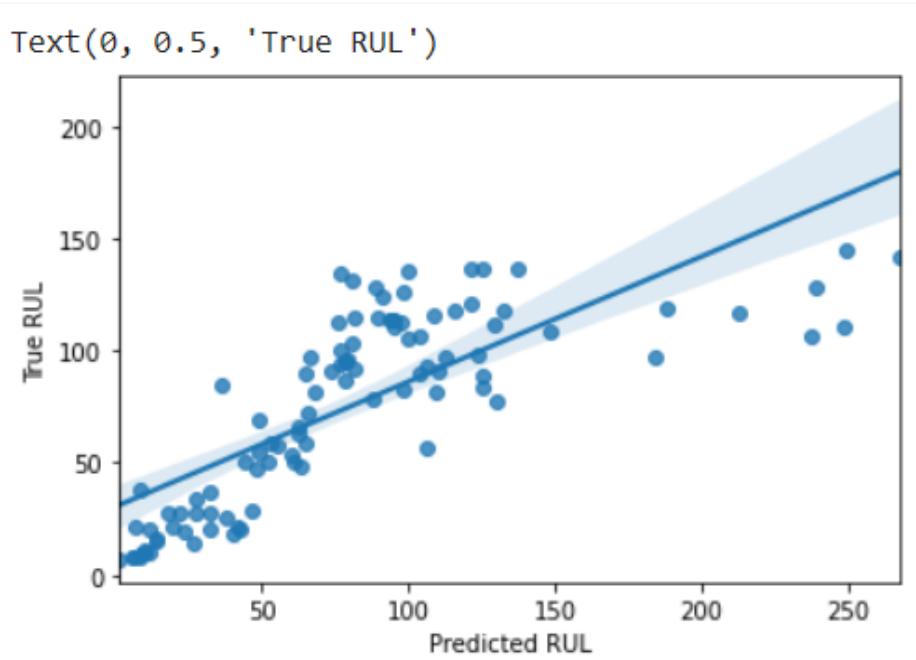
The histogram plot of the distribution of RUL can be seen below



The model was very much capable of fitting the RUL as can be seen below:



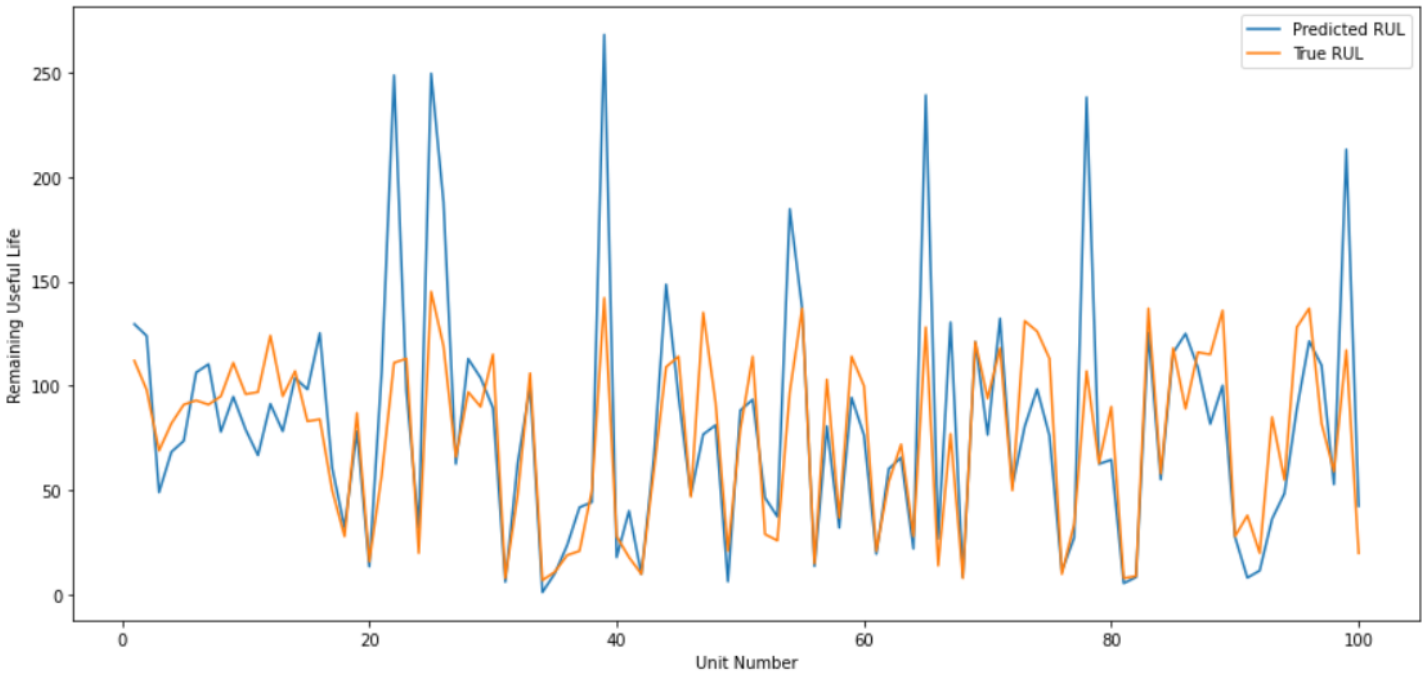
The plot of true RUL vs Predicted by this model can be seen below as:



Testing dataset:
Loss = 23.183, MSE = 1371.188

Conclusion

The model is very much capable of estimating the remaining useful life of the machine (**RUL**). However it requires a lot of feature engineering from the user side and cannot be scaled to a variety of machines.



2. LSTM model for RUL Prediction:

● Data preprocessing:

There are some columns in the dataset which have constant values, so these columns won't be useful. Hence, such columns are dropped. Extra RUL column is added in the training dataset. For the test dataset, RUL is calculated by using ground truth data. MinMax normalization is used to normalize the training data extract for 'UnitNumber', 'Cycle' and 'RUL'. Test dataset is normalized by using the parameters from the MinMax normalization applied on the training dataset.

Keras convolutional layers expect an input in the shape of a numpy array of 3 dimensions (samples, time steps, features) where samples is the number of training sequences, time steps is the look back window or sequence length and features is the number of features of each sequence at each time step. Hence X_train,X_test are converted to 3 dimensions. Sequence length is considered as 50.

Only sequences that meet the window-length are considered, no padding is used. This means for testing we need to drop those which are below the window-length. An alternative would be to pad sequences so that we can use shorter ones.

We removed the first seq_length labels because for one id the first sequence of seq_length size has as target the last label (the previous ones are discarded). All the next id's sequences will have associated step by step one label as target.

● Model details:

Here, we build a deep LSTM network with sequence length 50.
The first layer is an LSTM layer with 100 units followed by Dropout. Then another LSTM layer with 100 units followed by Dropout. Final layer is a Dense output layer with single unit and relu activation since this is a regression problem.
Dropouts are applied after each LSTM layer to control overfitting. Initially we tried to train the model for 100 epochs but the model was getting overfitted so we used early stoppings and returned the best parameters. Validation dataset is kept as 20 percent of training dataset.
Optimizer used = rmsprop, Loss = mae

● Results and plots:

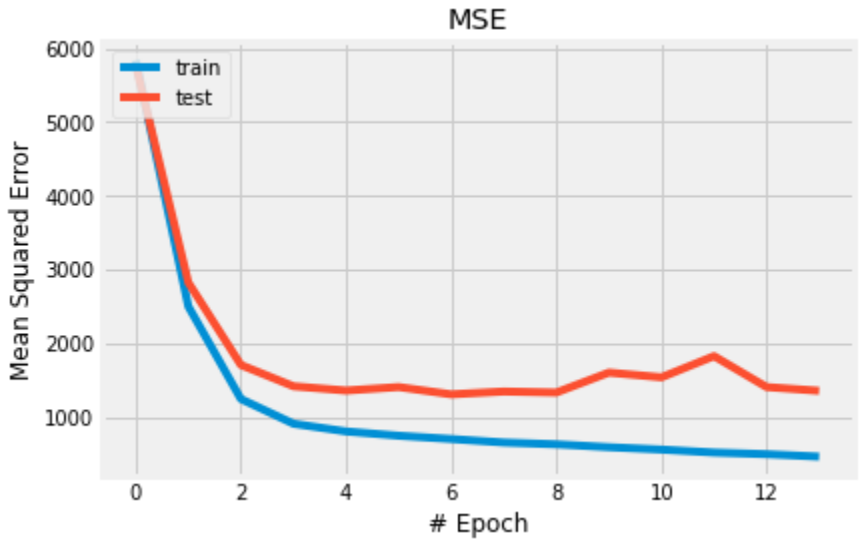
Training dataset:

Loss = 12.7656, MSE = 479.7422

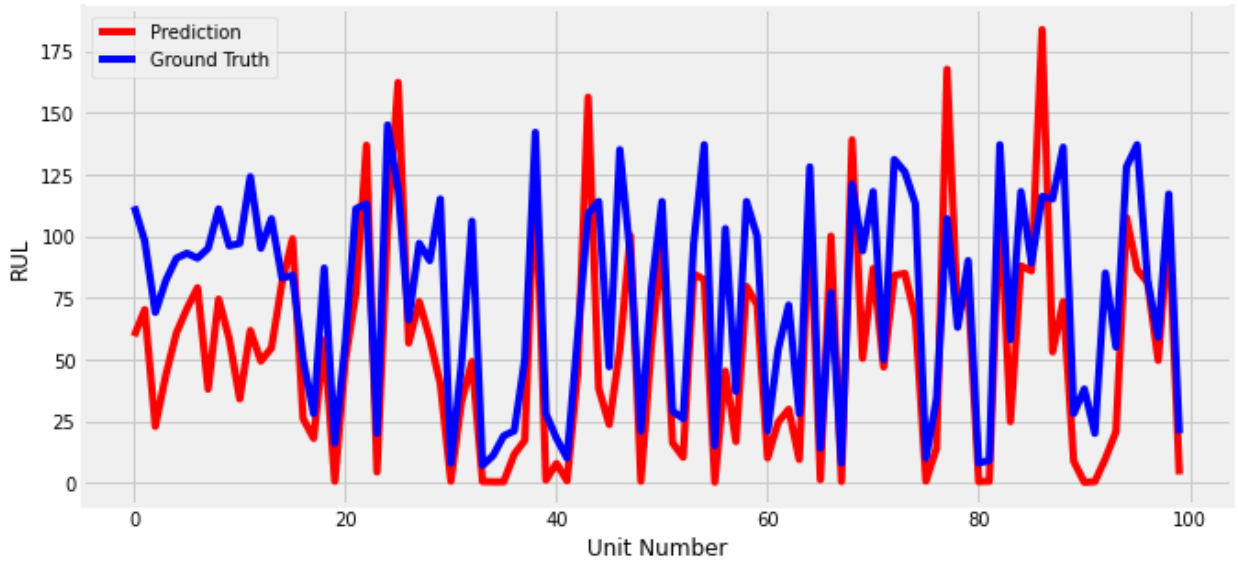
Testing dataset:

Loss = 28.5424, MSE = 1198.3757

MSE vs iterations:



Truth vs Prediction:



- ## Conclusion:

Here we have used LSTM for a basic problem of predictive maintenance but many predictive maintenance problems usually involve a variety of data sources that needs to be taken into account when applying deep learning in this domain. Additionally, it is important to tune the models for the right parameters such as window size. Here are some suggestions on future directions on improvements:

1. Try different window sizes.
2. Try different architectures with different numbers of layers and nodes.
3. Try tuning hyperparameters of the network.
4. Try predicting RUL (regression) such as in Predictive Maintenance Template Step 2A of 3 and label2 (multi-class classification) such as in Predictive Maintenance Template Step 2C of 3.
5. Try on larger data sets with more records.
6. Try a different problem scenario where multiple other data sources are involved such as maintenance records.

3. LSTM model for binary classification:

- ## Data preprocessing:

There are some columns in the dataset which have constant values, so these columns won't be useful. Hence, such columns are dropped. Extra RUL column is added in the training dataset. For the test dataset, RUL is calculated by using ground truth data. MinMax normalization is used to normalize the training data extract for 'UnitNumber', 'Cycle' and 'RUL'. Test dataset is normalized by using the parameters from the MinMax normalization applied on the training dataset. In the binary classification model it will classify the Engine whether it has remaining life is more or less than 30 cycles based on the no. of cycle the engine is run for.

- ## Model details:

Here, we build a deep network with sequence length 50.
The first layer is an LSTM layer with 8 units followed by Dropout with probability 0.2. Then another LSTM layer with 4 units followed by Dropout with probability 0.2. Final layer is a Dense output layer with single unit and sigmoid activation since this is a classification problem. Dropouts are applied after each LSTM layer to control overfitting. Initially we tried to train the model for 100 epochs but the model was getting overfitted so we used early stoppings and returned the best parameters. Validation dataset is kept as 20 percent of training dataset. Optimizer used = adam optimizer, Loss = binary cross entropy

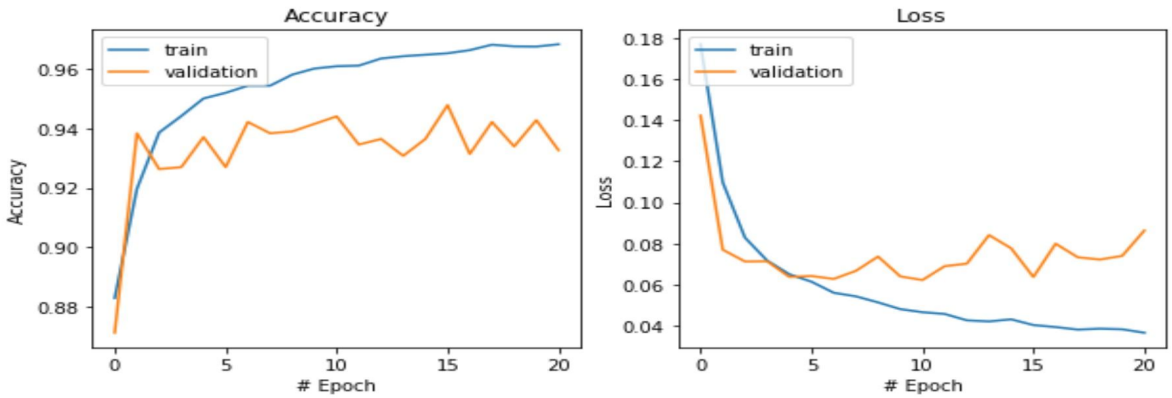
- ## Results and plots:

On training dataset:

Loss = 0.0366, Accuracy= 0.9684

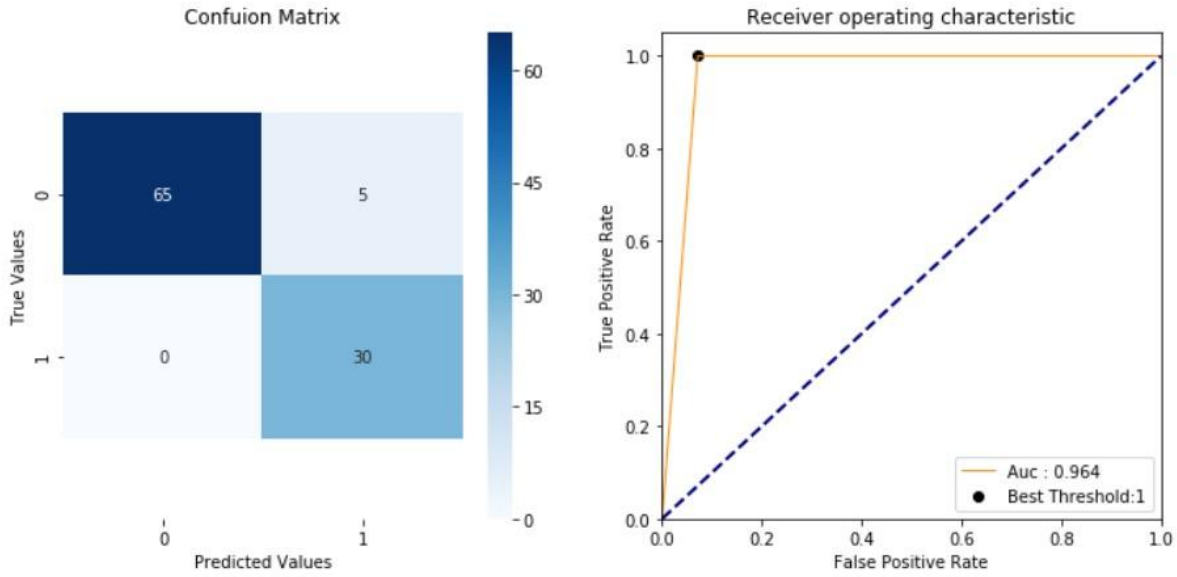
On Validation dataset:

Val_loss: 0.0863, Val_acc: 0.9327



On Testing dataset:

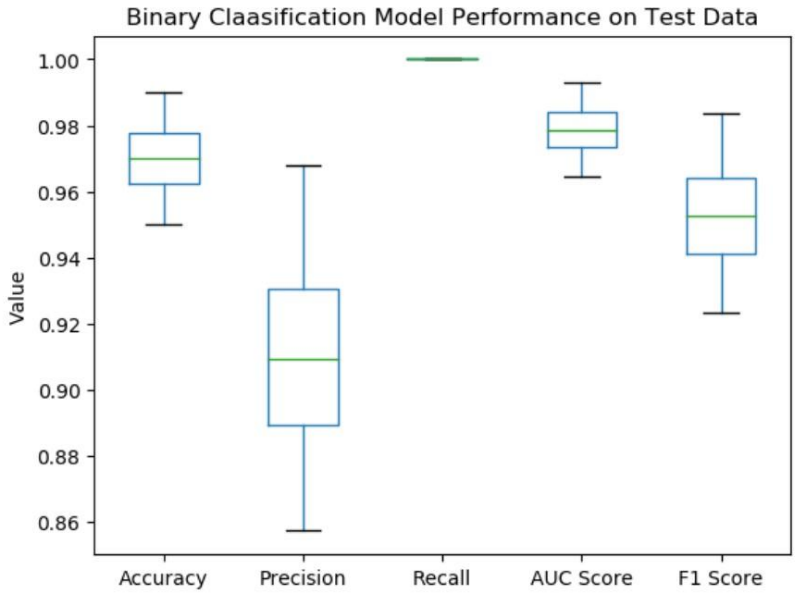
```
F1 Score:  0.923076923076923
              precision    recall  f1-score   support

           0       1.00      0.93      0.96        70
           1       0.86      1.00      0.92        30

    accuracy                           0.95       100
   macro avg       0.93      0.96      0.94       100
weighted avg       0.96      0.95      0.95       100

Area under curve :  0.9642857142857143
```



LSTM models have stochasticity and the results might be different every time.We run the model (architecture same as the final model) 10 times and check the performance of the model.

	Model	Accuracy	Precision	Recall	AUC Score	F1 Score
0	1.0	0.97	0.909091	1.0	0.978571	0.952381
1	2.0	0.97	0.909091	1.0	0.978571	0.952381
2	3.0	0.98	0.937500	1.0	0.985714	0.967742
3	4.0	0.96	0.882353	1.0	0.971429	0.937500
4	5.0	0.97	0.909091	1.0	0.978571	0.952381
5	6.0	0.96	0.882353	1.0	0.971429	0.937500
6	7.0	0.98	0.937500	1.0	0.985714	0.967742
7	8.0	0.95	0.857143	1.0	0.964286	0.923077
8	9.0	0.99	0.967742	1.0	0.992857	0.983607
9	10.0	0.97	0.909091	1.0	0.978571	0.952381

Mean Accuracy : 0.970000
Mean Precision: 0.910095
Mean Recall: 1.000000
Mean AUC Score: 0.978571
Mean F1 Score: 0.952669



- **Conclusion:**

AUC score is very much near to 1 for the test dataset so it can be said that the model is very capable of predicting whether it fails within 30 cycles or not for the unseen data and that is the reason to use machine learning so that it will predict correctly on unseen data.Future improvement can be done by

1. Try different window sizes.
2. Try different architectures with different numbers of layers and nodes.
3. Try tuning hyperparameters of the network.Try on larger data sets with more records.
4. Try a different problem scenario where multiple other data sources are involved such as maintenance records.

4. CNN-SVM model for binary classification:

● Data preprocessing:

Columns with constant values are dropped. Extra RUL column is added in the training dataset. For the test dataset, RUL is calculated by using ground truth data. MinMax normalization is used to normalize the training data extract for ‘UnitNumber’, ‘Cycle’ and ‘RUL’. Test dataset is normalized by using the parameters from the MinMax normalization applied on the training dataset.

Extra failure column is added in the same dataframe on the basis of the RUL. If the RUL value is less than 50 then it is 1 otherwise 0.

Keras convolutional layers expect an input in the shape of a numpy array of 3 dimensions (samples, time steps, features) where samples is the number of training sequences, time steps is the look back window or sequence length and features is the number of features of each sequence at each time step. Hence X_train,X_test are converted to 3 dimensions. Sequence length is considered as 50.

Only sequences that meet the window-length are considered, no padding is used. This means for testing we need to drop those which are below the window-length. An alternative would be to pad sequences so that we can use shorter ones.

We removed the first seq_length labels because for one id the first sequence of seq_length size has as target the last label (the previous ones are discarded). All the next id's sequences will have associated step by step one label as target.

● Model details:

Here, we have solved this problem with 2 approaches.

1st approach:

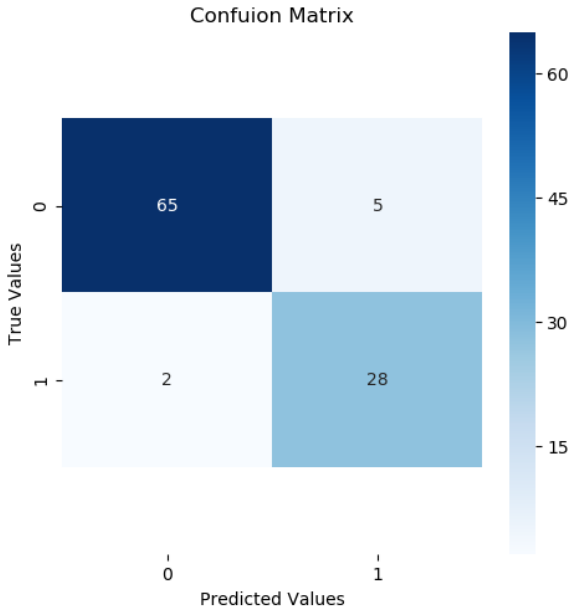
- Done one hot encoding of the target data (2 classes)
- Sequential model
- Added 4 convolutional layers with “relu” activation function, applied max pooling and avg pooling and applied dropouts
- Final dense layer with 2 neurons and “softmax” activation function
- Use hinge loss function

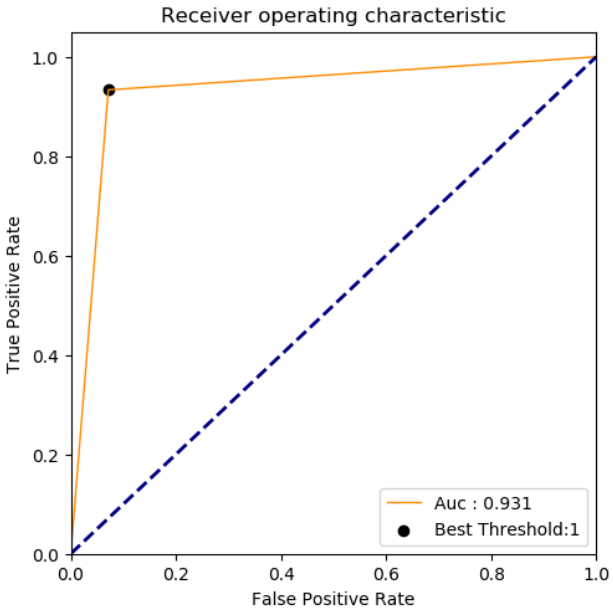
2nd approach:

- Sequential model
- Added 4 convolutional layers with “relu” activation function, applied max pooling and avg pooling and applied dropouts
- Final dense layer with 1 neurons and “sigmoid” activation function
- Use binary cross entropy as loss function
- Fit the model with X_train and y_train
- Get outputs of the 2nd last layer and feed it to SVM
- Fit the SVM model with cnn_x_train, y_train

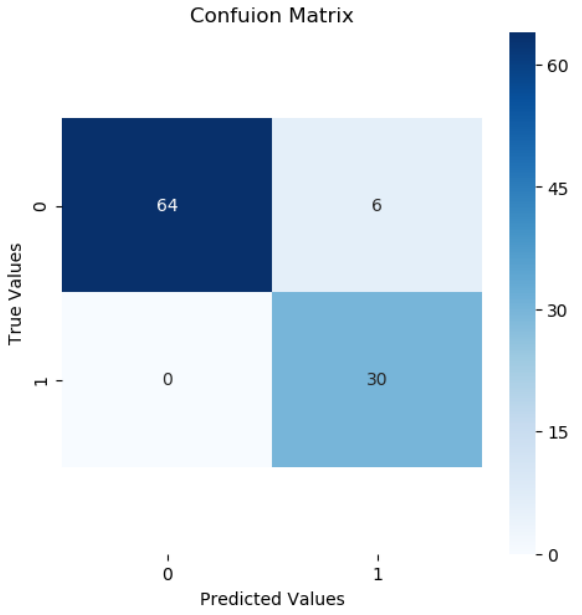
- ## Results and plots:

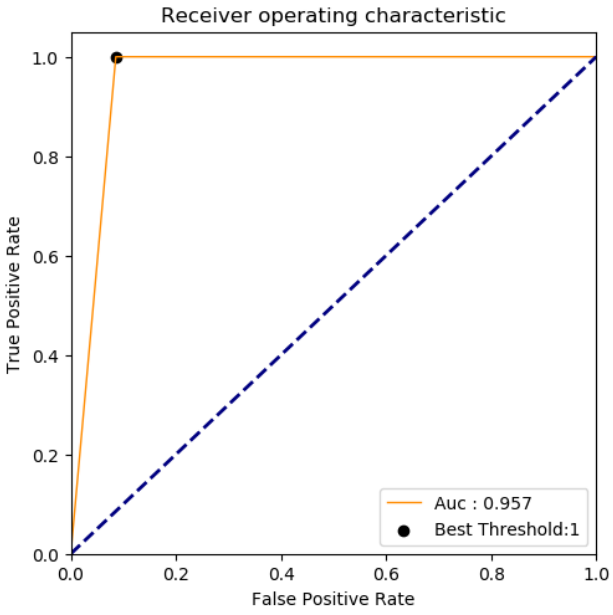
1st approach (Results on test dataset):
F1 score = 0.888
Accuracy = 0.93
Confusion matrix and Area under ROC curve:





2nd approach (Results on test dataset):
F1 score = 0.909
Accuracy = 0.94
Confusion matrix and Area under ROC curve:





- ## Conclusion:

Here, we have tried two approaches. Overall accuracy and F1 score of the 2nd approach is better than the 1st approach. So, it can be concluded that the 2nd approach is more preferable. But again we can always try changing the hyperparameters, dataset and check the results on both approaches. There can be some cases where the 1st approach will give better results.

● Final Conclusion:

In this project, we have applied 2 binary classification methods for failure prediction and 2 regression methods for RUL prediction. The **maximum accuracy** achieved in the classification task is **94%** and **minimum mse** value achieved in regression task is **1198.37**. From the results, it is concluded that for this particular dataset, **LSTM neural network has given the best results** in case of classification as well regression. But again, if the dataset is changed or better data preprocessing techniques are used or better hyperparameters are used then there are chances that other two methods can give better results.