APRIL 26TH, 2019

LANL-Earthquake Prediction

Sumanth Varambally (2017MT60855)
Saksham Jain (2017MT10747)
Indian Institute of Technology, Delhi

PROBLEM STATEMENT:

Given acoustic data and time to failure for lab earthquakes, predict the time to failure

Deconstructing the problem

INTRODUCTION

- Forecasting earthquakes is one of the most important problems in Earth science because of their devastating consequences.
- This competition is aimed at predicting when a laboratory earthquake would occur, given acoustic data leading upto the point of earthquake

DATA-SET

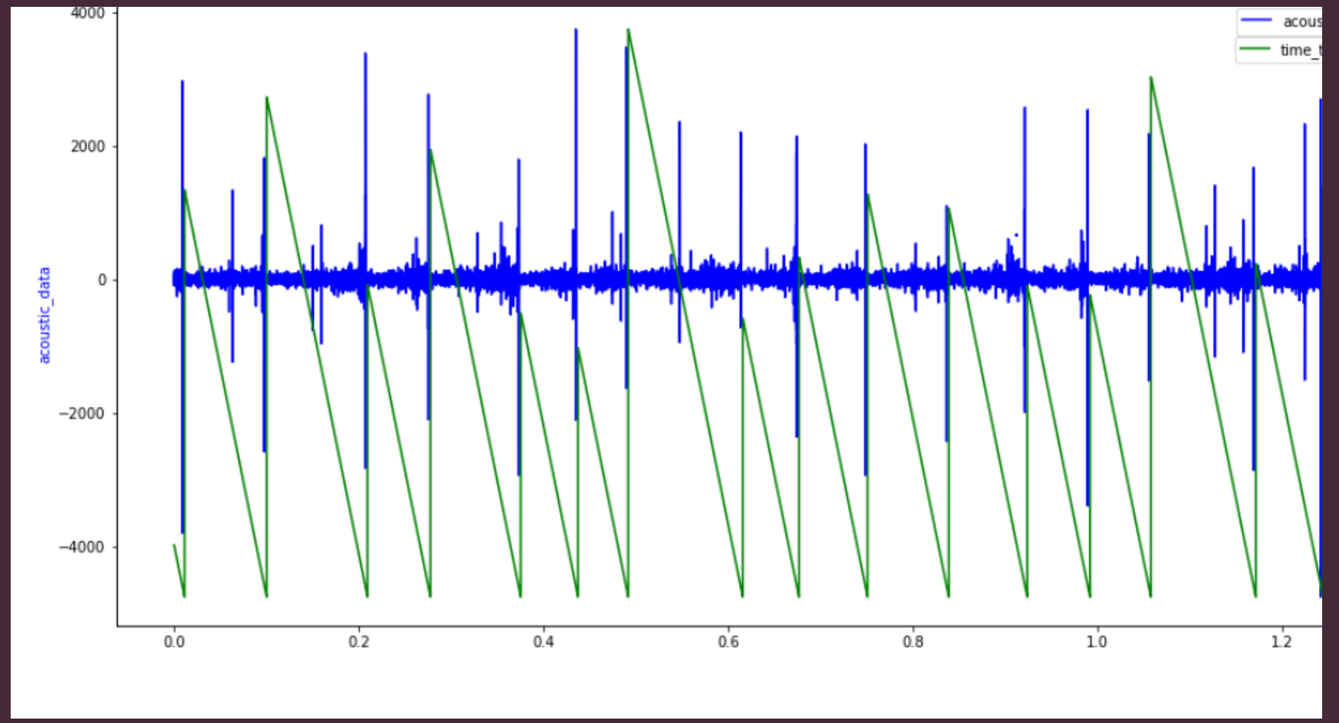
- The training data-set consists of two features, acoustic data (integer) and time to failure (float).
- At each line, the acoustic data and the corresponding time left for the laboratory earthquake was presented.
- Task: To detect the time to failure at the end of a given test segment of acoustic data values.

Deconstructing the problem

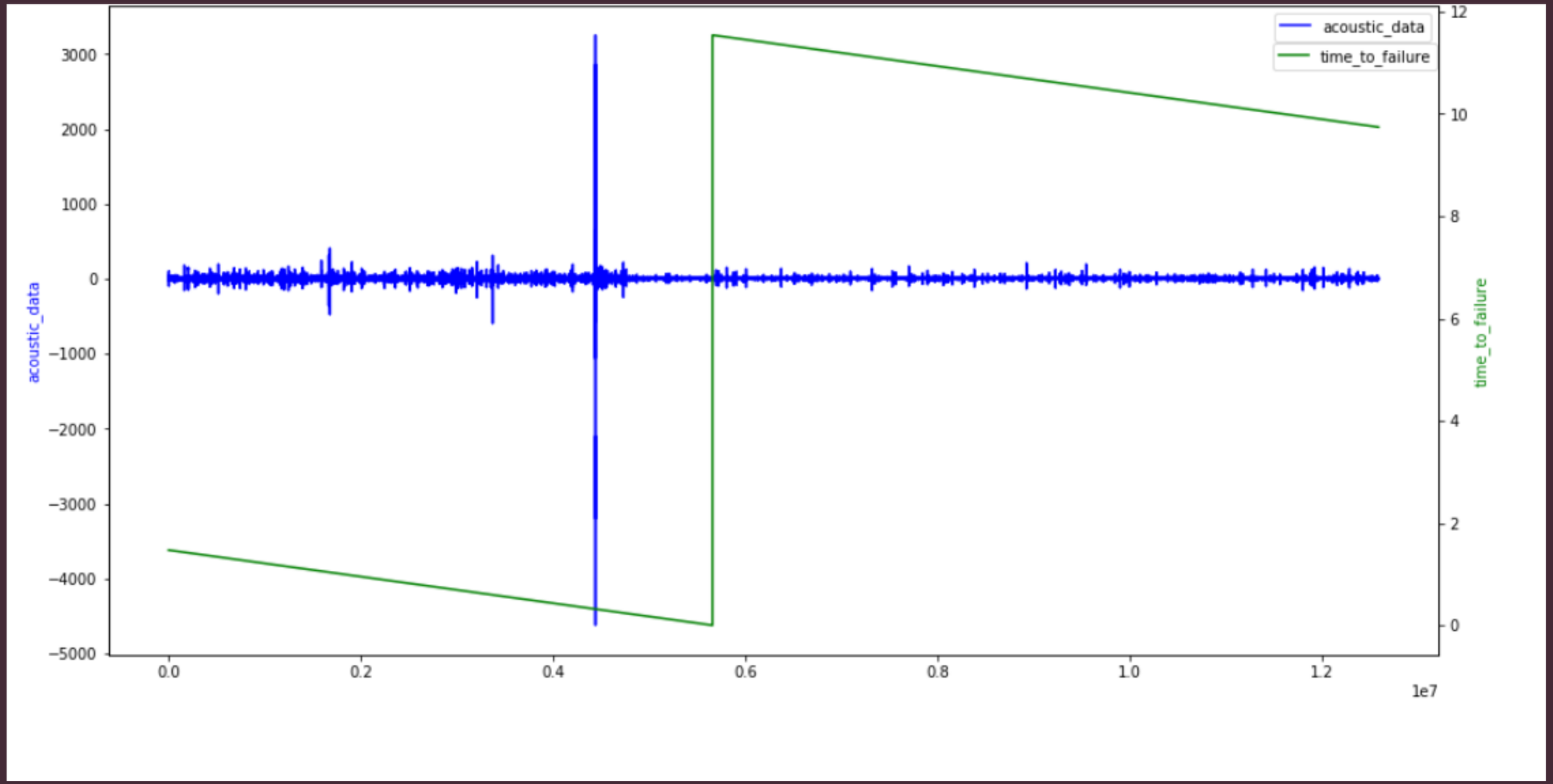
DATASET

- The test data consisted of segments of continuous data, and we had to predict the time to failure for each segment. The training file looks like this:

	acoustic_data	time_to_failure
0	12	1.4690999984741210
1	6	1.4690999984741210
2	8	1.4690999984741210
3	5	1.4690999984741210
4	8	1.4690999984741210



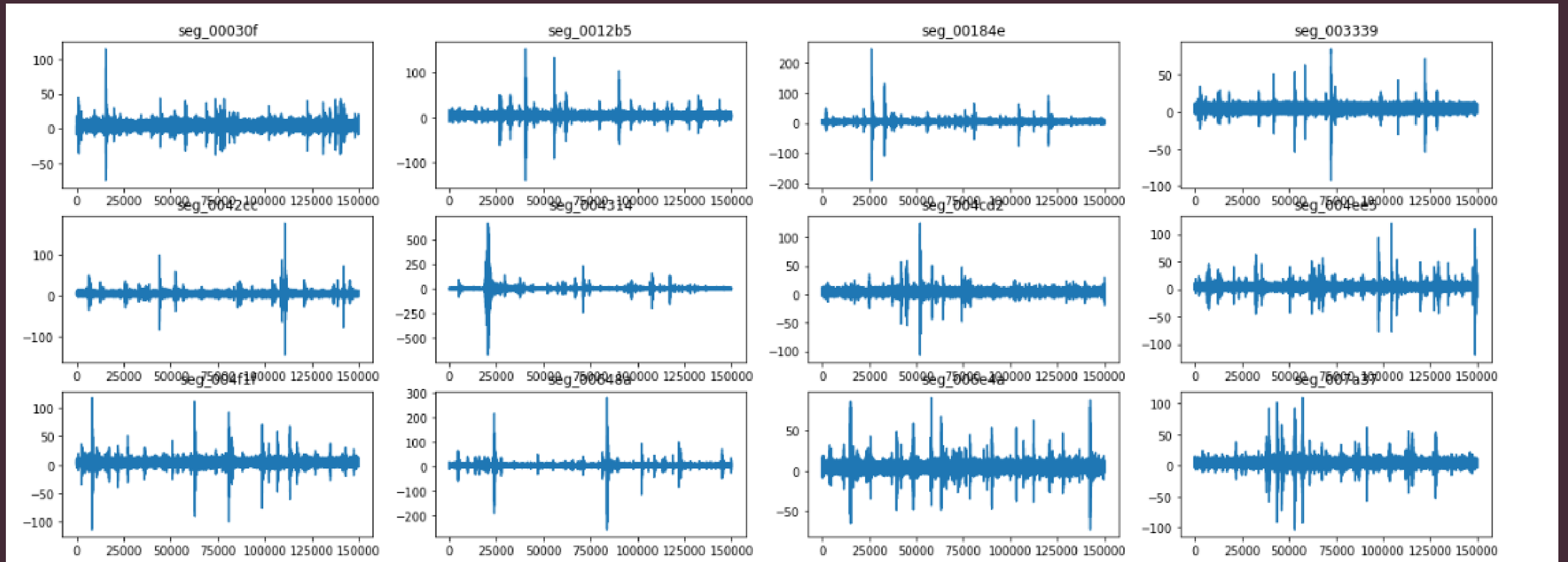
A closer look at a failure



Main challenge

- The biggest challenge in the problem is the huge size. The training data consists of 629145480 rows (around 600 million). Time-series are usually dealt with methods like subsequence generation, however these methods would be infeasible because of the large size of the data involved.
- Therefore, we need to use some method of aggregate extraction from the data to better condense and represent it, in order to make it more suitable for training.
- Our approach: The test segments were each of length 150000. So we split our training data into 4194 training segments, each of length 150000

Test Segments



General Approach

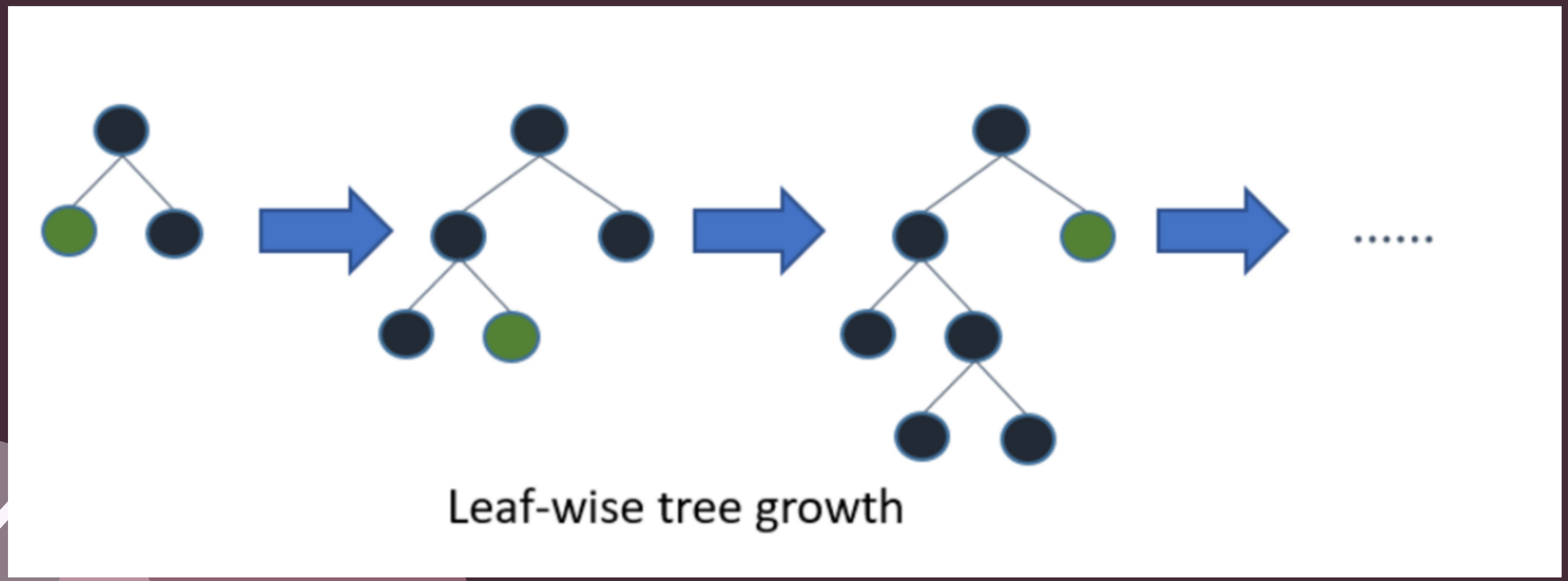
- Here the green spikes correspond to a potential failure of the rock.
- The data has continuous **variation across a set of contiguous rows.**
- Hence, we split our training data into **segments** and did extensive **feature extraction.**

mean	count_big	classic_sta_lta9_mean	ave_roll_mean_10	q01_roll_mean_100	av_change_abs_roll_mean_1000
std	sum	exp_Moving_average_300_mean	std_roll_mean_10	q05_roll_mean_100	av_change_rate_roll_mean_1000
max	q95	exp_Moving_average_3000_mea	max_roll_mean_10	q95_roll_mean_100	abs_max_roll_mean_1000
min	q99	exp_Moving_average_30000_me	min_roll_mean_10	q99_roll_mean_100	ave_roll_std_10000
mean_change_abs	q05	MA_700MA_std_mean	q01_roll_mean_10	av_change_abs_roll_mean_100	std_roll_std_10000
abs_max	q01	MA_700MA_BB_high_mean	q05_roll_mean_10	av_change_rate_roll_mean_10	max_roll_std_10000
abs_min	abs_q95	MA_700MA_BB_low_mean	q95_roll_mean_10	abs_max_roll_mean_100	min_roll_std_10000
std_first_50000	abs_q99	MA_400MA_std_mean	q99_roll_mean_10	ave_roll_std_1000	q01_roll_std_10000
std_last_50000	abs_q05	MA_400MA_BB_high_mean	av_change_abs_roll_mean_10	std_roll_std_1000	q05_roll_std_10000
std_first_10000	abs_q01	MA_400MA_BB_low_mean	av_change_rate_roll_mean_10	max_roll_std_1000	q95_roll_std_10000
std_last_10000	abs_mean	MA_1000MA_std_mean	abs_max_roll_mean_10	min_roll_std_1000	q99_roll_std_10000
avg_first_50000	abs_std	iqr	ave_roll_std_100	q01_roll_std_1000	av_change_abs_roll_std_10000
avg_last_50000	mad	q999	std_roll_std_100	q05_roll_std_1000	av_change_rate_roll_std_10000
avg_first_10000	kurt	q001	max_roll_std_100	q95_roll_std_1000	abs_max_roll_std_10000
avg_last_10000	skew	ave10	min_roll_std_100	q99_roll_std_1000	ave_roll_mean_10000
min_first_50000	med	ave_roll_std_10	q01_roll_std_100	av_change_abs_roll_std_1000	std_roll_mean_10000
min_last_50000	Hilbert_mean	std_roll_std_10	q05_roll_std_100	av_change_rate_roll_std_1000	max_roll_mean_10000
min_first_10000	Hann_window_mean	max_roll_std_10	q95_roll_std_100	abs_max_roll_std_1000	min_roll_mean_10000
min_last_10000	classic_sta_lta1_mean	min_roll_std_10	q99_roll_std_100	ave_roll_mean_1000	q01_roll_mean_10000
max_first_50000	classic_sta_lta2_mean	q01_roll_std_10	av_change_abs_roll_std_100	std_roll_mean_1000	q05_roll_mean_10000
max_last_50000	classic_sta_lta3_mean	q05_roll_std_10	av_change_rate_roll_std_100	max_roll_mean_1000	q95_roll_mean_10000
max_first_10000	classic_sta_lta4_mean	q95_roll_std_10	abs_max_roll_std_100	min_roll_mean_1000	q99_roll_mean_10000
max_last_10000	classic_sta_lta5_mean	q99_roll_std_10	ave_roll_mean_100	q01_roll_mean_1000	av_change_abs_roll_mean_10000
max_to_min	classic_sta_lta6_mean	av_change_abs_roll_std_10	std_roll_mean_100	q05_roll_mean_1000	av_change_rate_roll_mean_10000
max_to_min_abs_diff	classic_sta_lta7_mean	av_change_rate_roll_std_10	max_roll_mean_100	q95_roll_mean_1000	abs_max_roll_mean_10000
max_to_min_diff	classic_sta_lta8_mean	abs_max_roll_std_10	min_roll_mean_100	q99_roll_mean_1000	

First Approach...

LGBM REGRESSOR

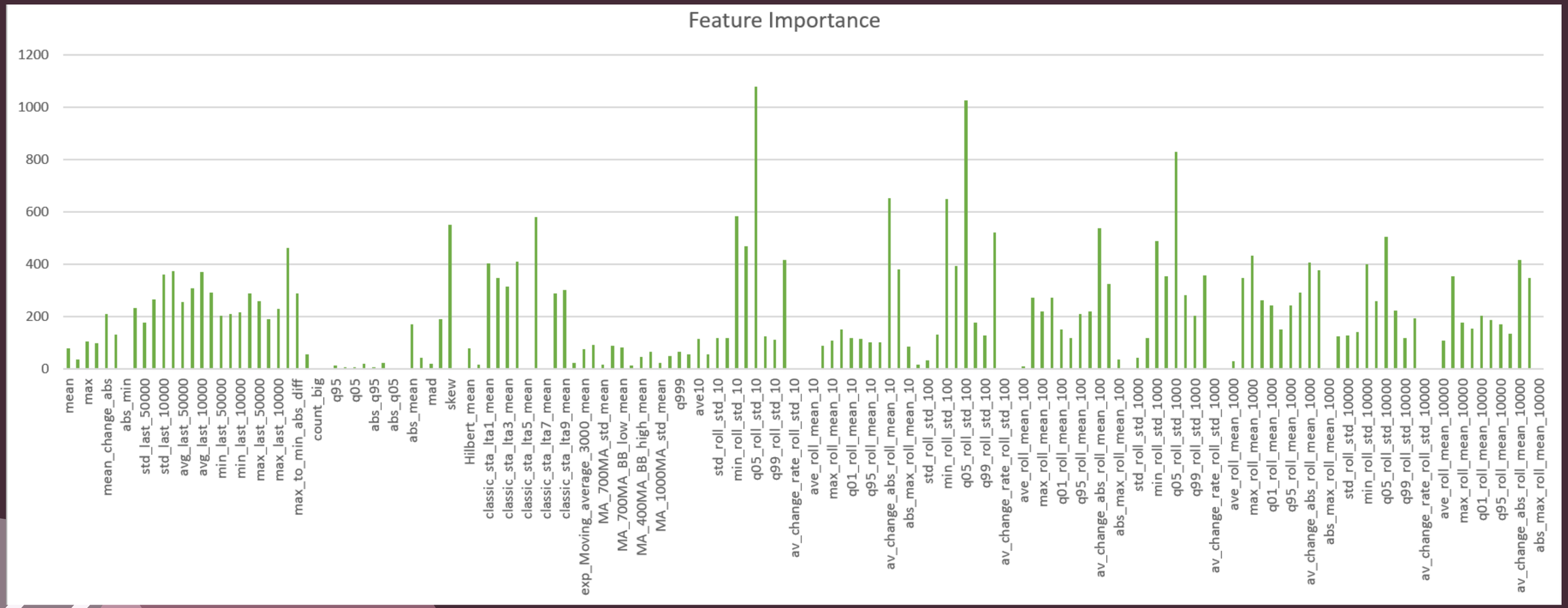
Light GBM is a gradient boosting framework that uses tree based learning algorithm.
We have used it here because firstly - it is fast, and can handle large data with ease. Secondly, it grows vertically leaf by leaf, making the decision tree learning more accurate.



Work Done

1. Splitting data into segments
2. Scaling and Normalisation of the data
3. Statistical Feature Exrtaction (total 154 in number)
4. Hyperparameter tuning using GridSearch Algorithm
5. LGB regression to learn the data, alongwith K-fold validation
6. Feature Imporance extraction, and comparison

Feature Importance



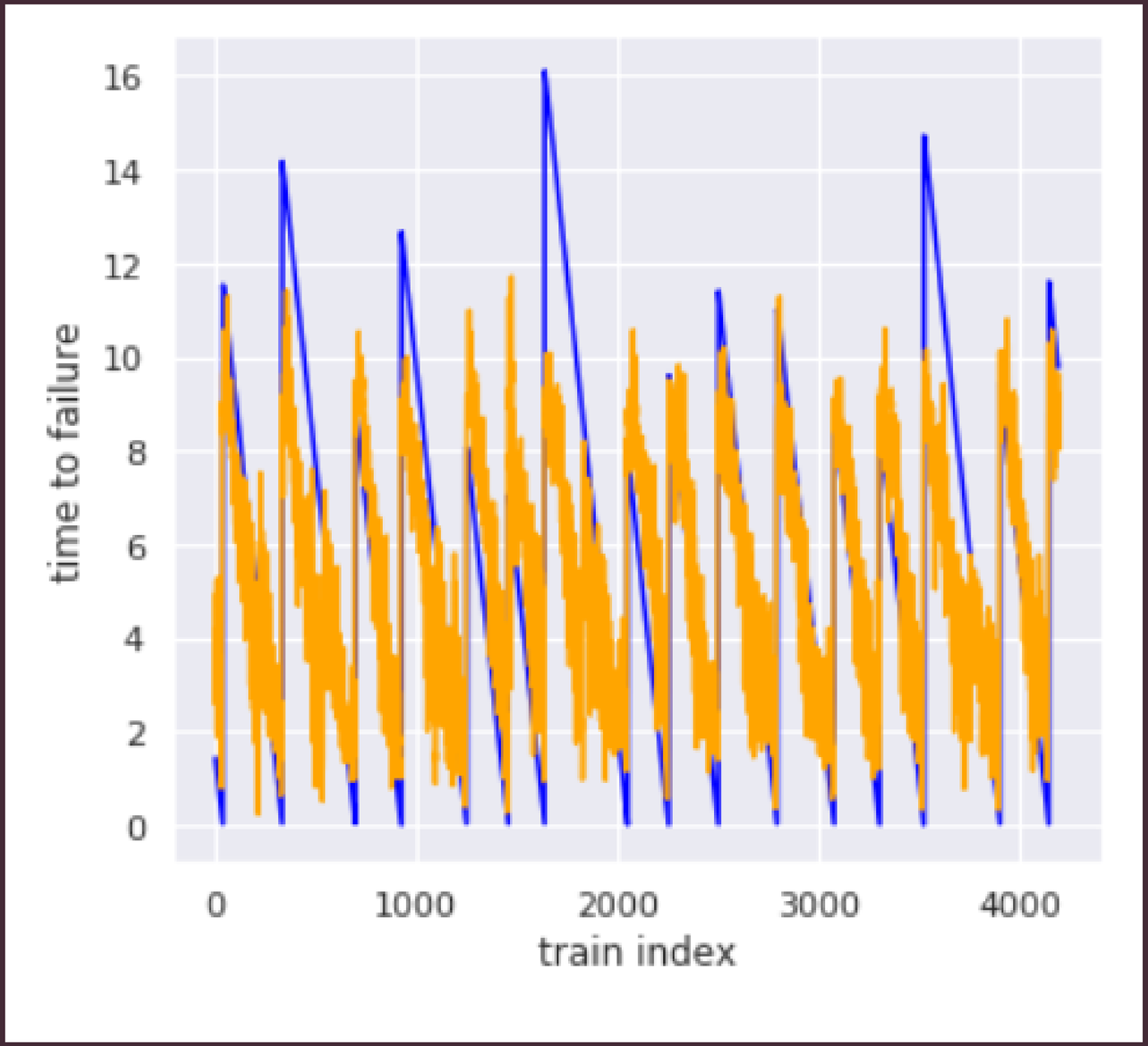
Grid-Search

We have used GridSearch (Exhaustive search over specified parameter values for an estimator) to find out the optimal hyper-parameters for the models

```
grid = [{'alpha': np.concatenate([np.linspace(0.001, 1, 100), np.l
inspace(1, 200, 1000)])}]
params = grid_search_cv(Ridge(), grid, X_train_scaled, target)
ridge_oof = make_predictions(Ridge(**params), X_train_scaled, targ
et)
```

Model Performance (Training)

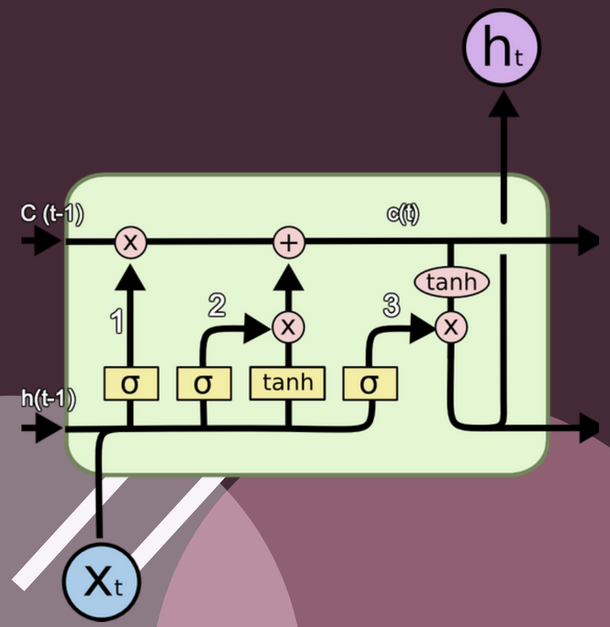
The LGB regressor performs well in recognising trends in data, but is not able to capture certain sharp peaks in the data



Performance achieved

With this approach, we achieved a mean absolute error of 1.673.
The highest score achieved on the leaderboard was 1.285.

Second Approach...

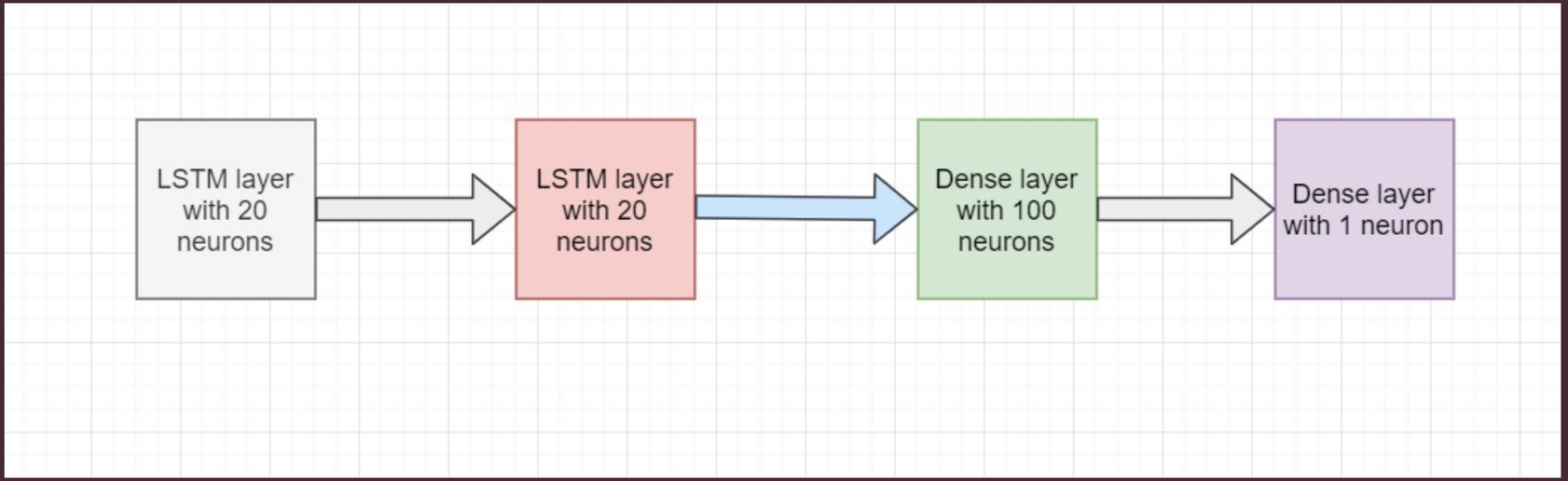


LSTM

- The Long Short-Term Memory network, or LSTM network, is a recurrent neural network that mimicks memory in the sense that it not only keeps track of it's current input, but also it's previous input and solves the issue of Vanishing Gradients with RNN.
- Here a more complex neuron, called a 'cell' is used, which can reuse old input or the input stored in the memory cell, or the current output.
- Since LSTM take the temporal variation of data into account, hence they are a suitable choice for learning time-series data.

Idea:

- The data consists of 600 million rows, varying with time. So, what we do is feed the data as a time-series with a time-step of 2280.
- We train the network with the time to failure values taken at a step of 2280 values apart, discarding the remaining values.

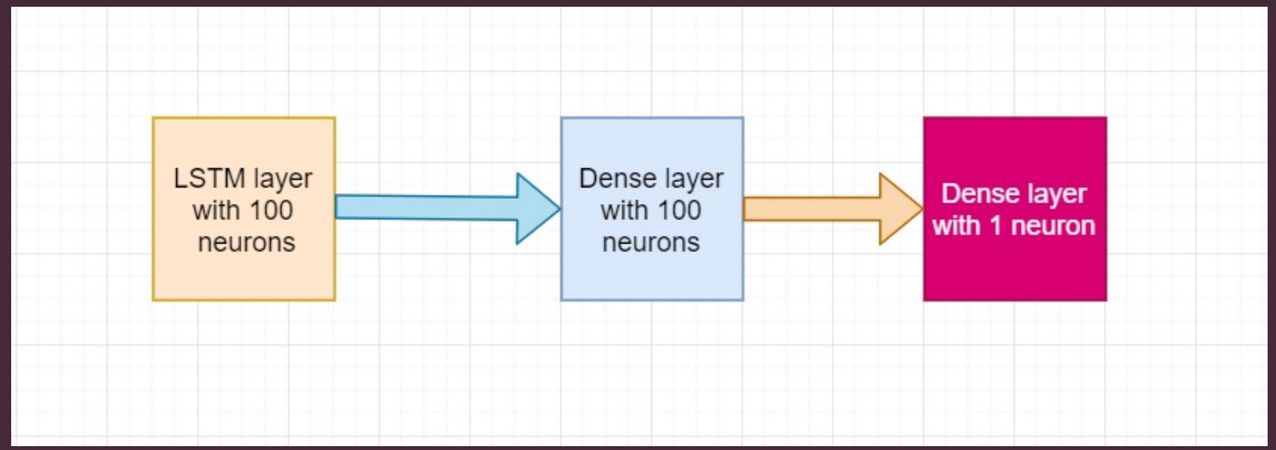


Drawback/Alternate Approach

- This resulted in a network with 275941 input segments, a size that was not able to be processed efficiently.
- So, we used knowledge from approach 1 : extract statistically significant features from the data and use them to train our LSTM

LSTM with segments

- We divided the training data into segments of length 150,000 lines, the size of every test segment and performed feature extraction as earlier.
- We train the LSTM on the statistical features so obtained with time-step 1.
- Why would this work? Because the training data is presented as data from one contiguous experiment session.
- Using this model, the network trained very quickly with reasonable results.



Performance achieved

- Using the second approach, we achieved a mean absolute error of 1.761.

Ideas to try out

- CNN LSTM - Idea is to first run the time-series through a few convolutional and pooling layers, and then run LSTM on these layers. Might be slow, but better at capturing time-series trend.
- Try to find out more relevant statistical features.
- Use sliding window approach to better capture all the failure times that we have currently discarded.