Project Title: Remaining Usable Life Estimation(Bearing Dataset)
Domain: Machine Learning
Phase4: Deployment

Submitted by:
Akanksha Singh
Thapar Institute Of Engineering & Technology
Patiala, Punjab - 147004

Under Mentorship of:
Devesh Tarasia

IITD-AIA Foundation for Smart Manufacturing

[1-June-2022 to 31-July-2022]
Remaining Usable Life Estimation (Bearing Dataset)

DATA PREPROCESSING includes:

METHODOLOGY ADOPTED:

1. INPUT: folder containing files containing CSV files
2. OUTPUT :
   ABOUT DATA: result predicted in file for download, Graphs

The internal process to be performed:

DATA ACQUISITION

Helper function:

1. Save the files to the server
2. Check if all files contain 2560 datasets or not
3. Merge all files into a single CSV file
4. Convert merged_csv file to pkz file along with header
5. Check if the pkz file formed has the same data as the merged CVV file
6. Deleted all the input files

DATASET PREPARATION

Helper Function:

- load_file() : to load data frame from pickle file
- df_row_ind_to_data_range() : get range of values (values present between a lower limit (inclusive) & and an upper limit (exclusive) from data frame given file index
- extract_feature_image: performs continuous wavelet transform (CWT) on 1D signals & return 2D feature image (Extracting 2D image features)
- signal_processing(): to extract time-frequency domain feature image
- extract_2d_feature(): to extract feature images for each data file and converting into a NumPy array and also store the probability of failure
- save_file() : to store data in pickle file

1. load the pickle test file
2. calculate: no_of_rows, ,no_of_files
3. plot the vertical & horizontal accel vibration graphs and display it to the users
4. call signal_processing() & extract the time-frequency domain feature image and display it to the user
5. call data_2d_feature() and extract the 2d feature image and convert it into NumPy as well as the probability of failure and save it in a new pickle file using save_file()

Data Testing

Helper Function:

- load_file(): to load dataframe from pickle file
- PHMDataset_Sequential(): Converting numpy array (nd array) into tensor

1. Load the pickle file containing time-freqency feature in numpy array form & probability of failure in 'DATA' variable
2. SEQ_LEN = 5
3. Converting the Data into sequence
   for loop (0, len(DATA))
   no_of_files, no_of_seqs, perm,train_indices
   train_dataset = call PHMDataset_Sequential()
   append train_datset in datasets list

4. Declare batch size as 16
5. Read the datasets list in for loop and apply DataLoader on the each data and append the result in dataloaders. The DataLoader will load 16 samples(time data sequence) at a time
6. Declare CNN_LSTM Model function:
    a. conv_bn_relu()
    b. CNN_CWT_Encoder()
    c. CNN_LSTM_FP()
7. Switch from CPU to GPU for fast processing
8. Connect the model to the device
9. Load the weights of the pre-trained model using load_state_dict(torch.load())
10. Apply inference : model_inference_helper(), sort_result
11. Plot the result graph and display it on the screen.
12. And send the pkz file of the result.

FLASK

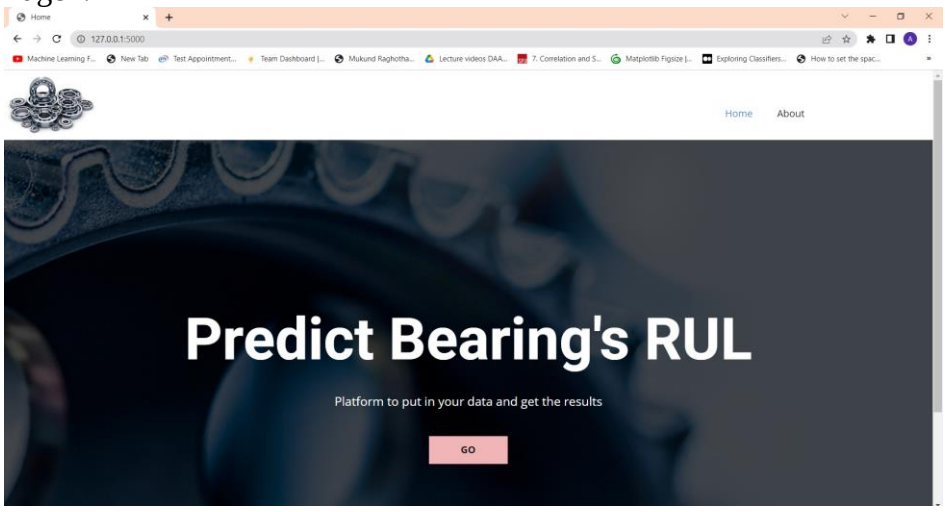
Flask is a microframework for Python.
Here we are going to build a mid-sized application of our own.

IMPLEMENTATION

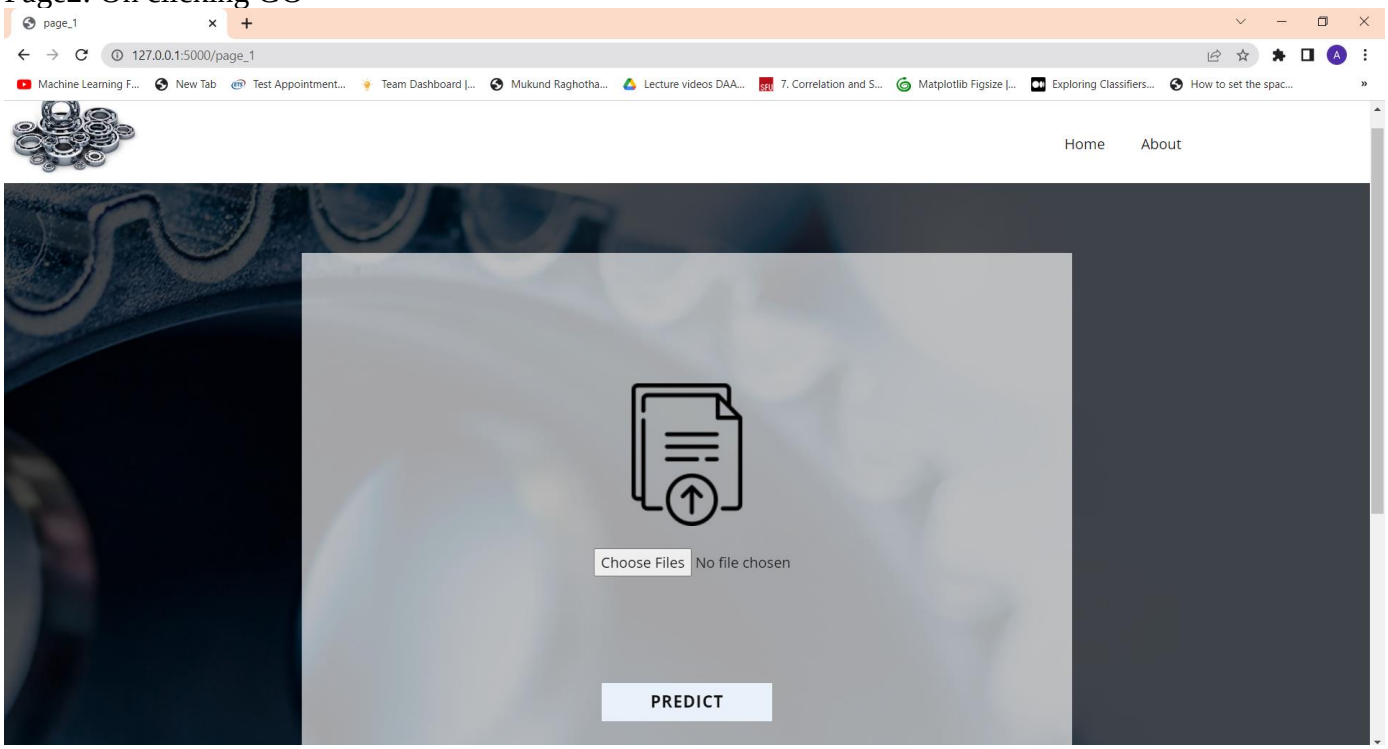
The frontend design in this project is created using the NamePage app. Later changes were made in Html and CSS files as per required for wrapping it with Flask .py file. The text editor used in the development phase is Sublime and the Git Bash was used to locally deploy the model as well as to type the command lines. Efforts were made to deploy the model globally using different hosting sites such as Heroku, digital ocean, and Azure. But due to memory limit exceeding 1 GB in Heroku, Memory Error while installing torch library (due to less RAM) in digital ocean, and .net file not accessible in case of Azure the model was unsuccessful in getting globally deployed. Further efforts were made to deploy the model by connecting these sites with GitHub but due to uploading a file limit of 100MB in GitHub again, the attempt was unsuccessful. Later GitHub desktop was used for uploading the 100 MB exceeding file but again no positive response was received. In the end, an attempt was made to decrease the size of the file using Large Storage File (LFS) but that was also unsuccessful. The last option left is using the docker container. It is still in progress.

OUTPUT

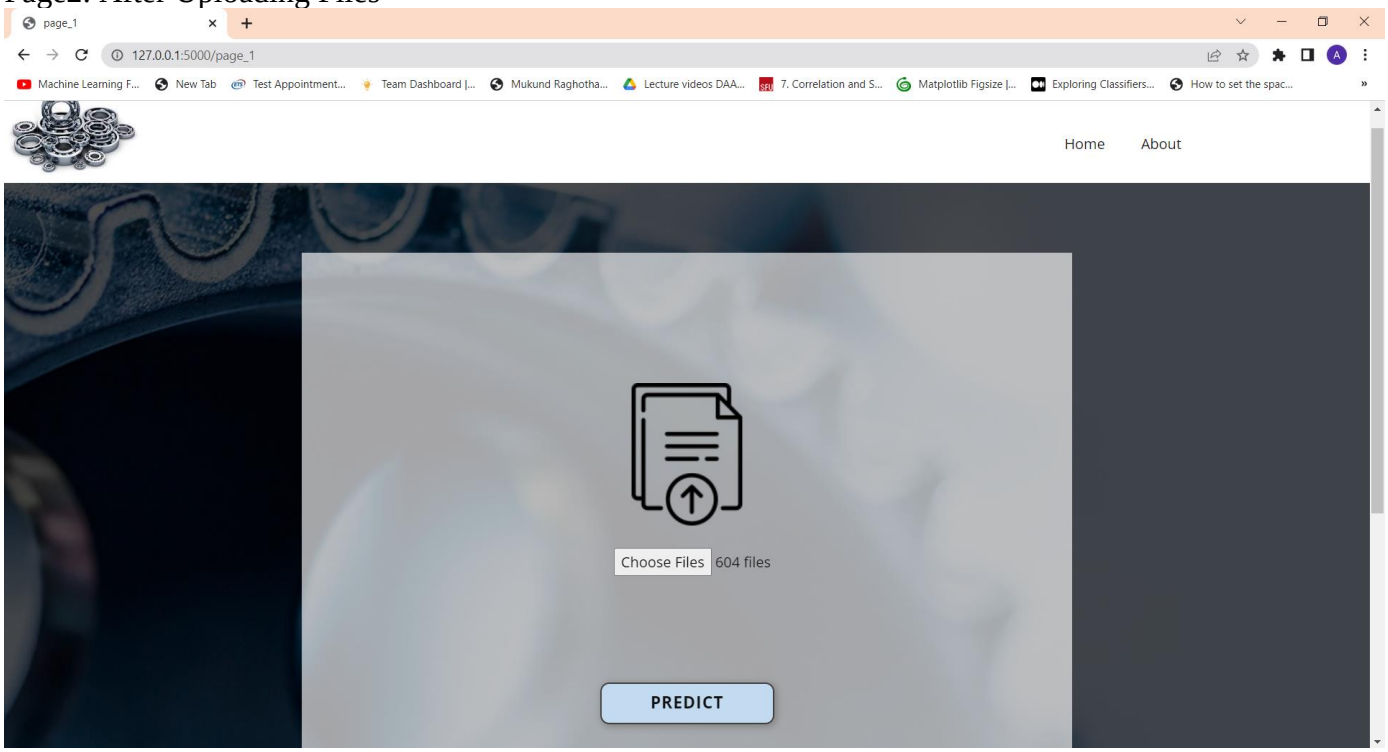
Page1:



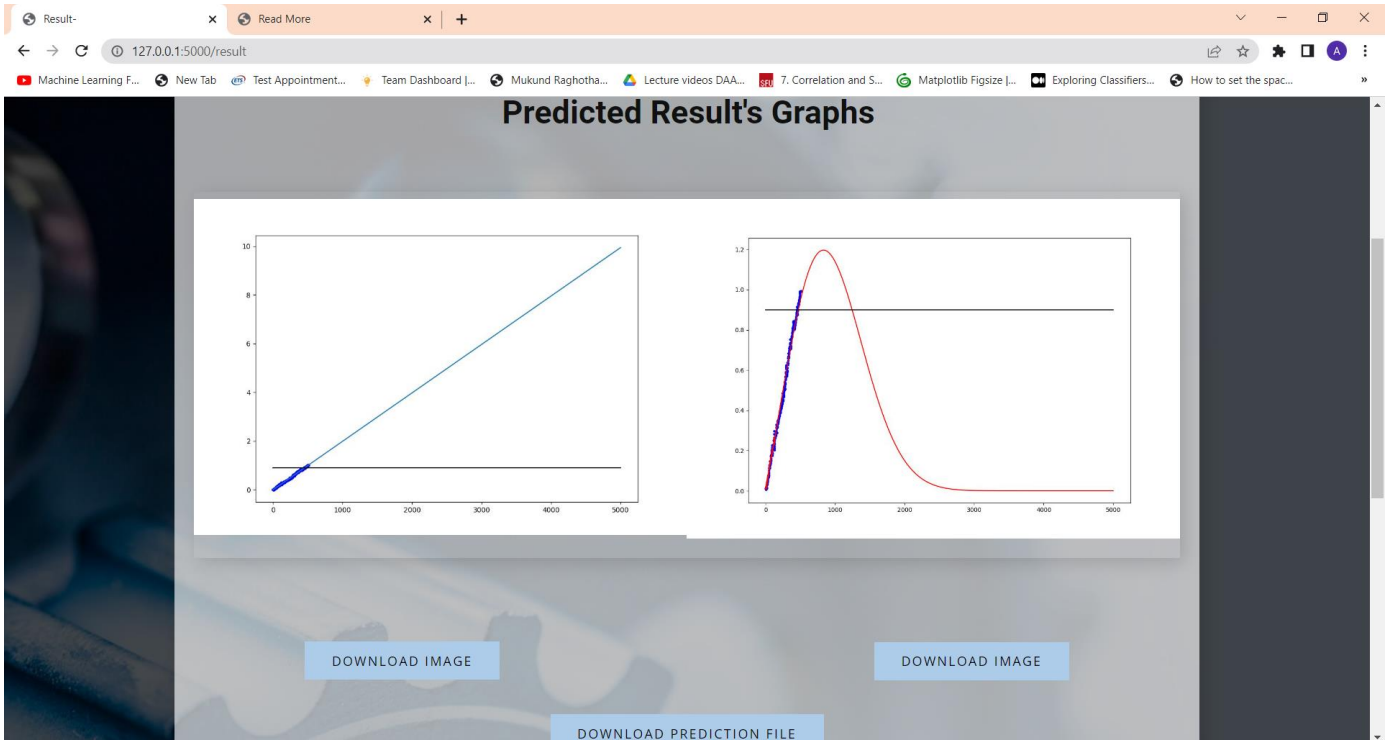
Page2: On clicking GO



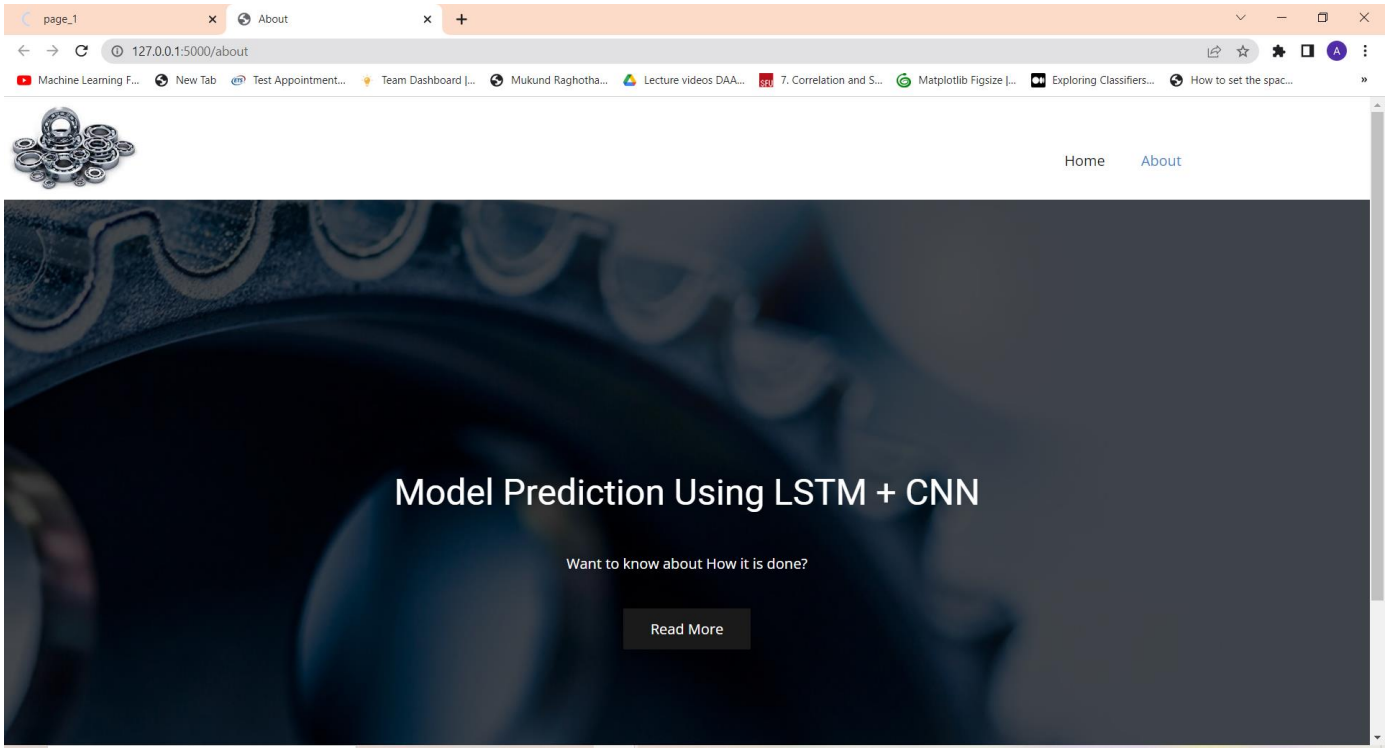
Page2: After Uploading Files



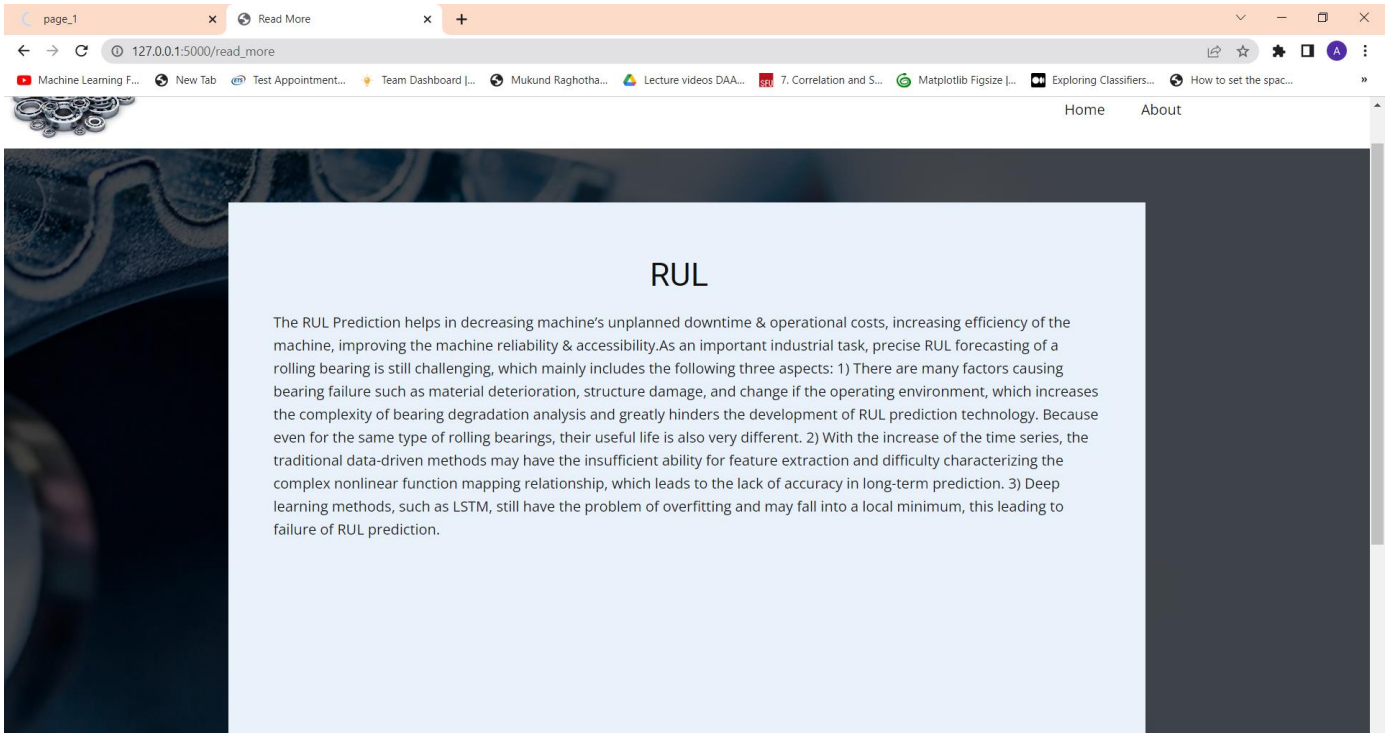
Result



About Page



On clicking Read More



About the Website:

5 web page

Home -> Page1 -> Result
About -> Read More