ATM | Dwight Look College of
ENGINEERING
TEXAS A&M UNIVERSITY

Project name:

monico™

SIMPLE SOLUTIONS. POWERFUL MONITORING.

Team *41* members:

Nicho Naugle, Digvijay Alluri, Michael Cubriel

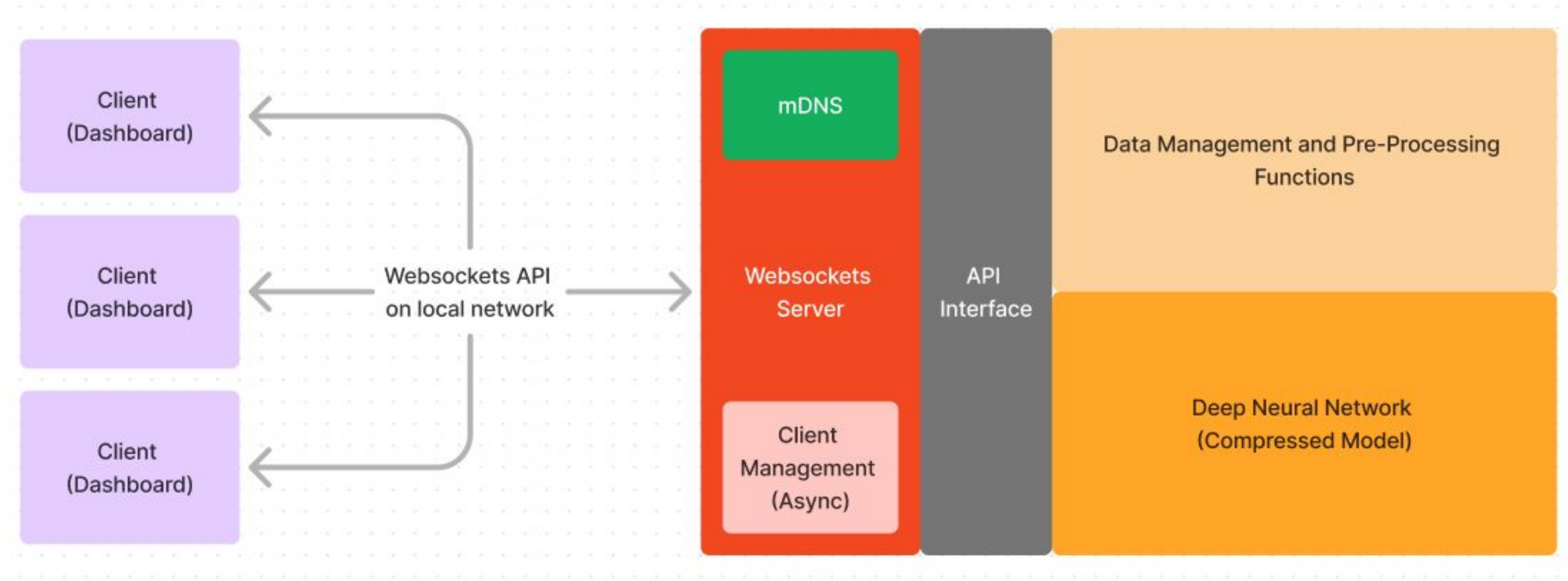
Problem Statement - Recap

“Compressor failures lead to substantial downtime in various manufacturing and mining industries, significantly impacting the bottom line and reducing the profitability of machinery undergoing maintenance.”

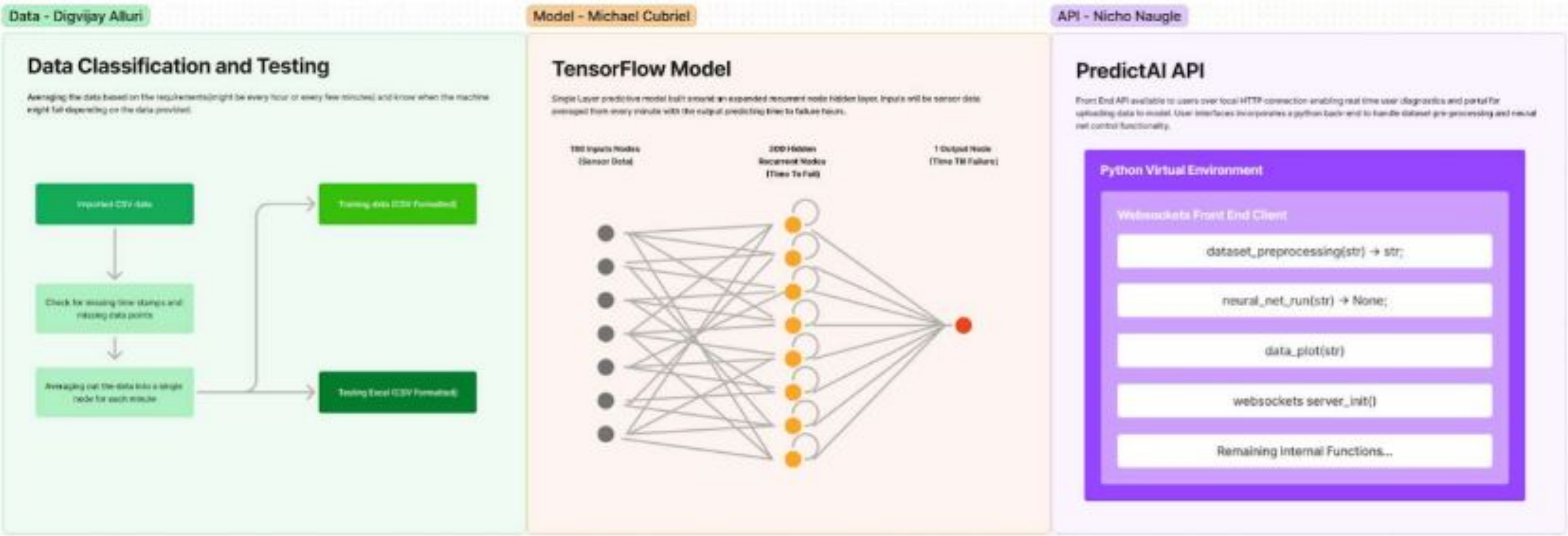
Our Solution - Recap

“Develop an edge application that utilizes local networking for real-time data streaming and incorporates a self-trainable AI with a lightweight footprint and adaptable prediction times to enable preventive maintenance.”

Block Functionality Diagram - Recap


Client (Dashboard)
Client (Dashboard)
Client (Dashboard)
Websockets API on local network
mDNS
Websockets Server
Client Management (Async)
API Interface
Data Management and Pre-Processing Functions
Deep Neural Network (Compressed Model)

Subsystem Review



Data - Digvijay Alluri

Data Classification and Testing

Averaging the data based on the requirements(might be every hour or every few minutes) and know when the machine might fail depending on the data provided.

Imported CSV data
Check for missing time stamps and missing data points
Averaging out the data into a single node for each minute
Training data (CSV Formatted)
Testing Excel (CSV Formatted)

Model - Michael Cubriel

TensorFlow Model

Single Layer predictive model built around an expanded recurrent node hidden layer. Inputs will be sensor data averaged from every minute with the output predicting time to failure hours.

100 Inputs Nodes (Sensor Data)
200 Hidden Recurrent Nodes (Time To Fail)
1 Output Node (Time Till Failure)

API - Nicho Naugle

PredictAI API

Front End API available to users over local HTTP connection enabling real time user diagnostics and portal for uploading data to model. User interfaces incorporates a python back-end to handle dataset pre-processing and neural net control functionality.

Python Virtual Environment
Websockets Front End Client
dataset_preprocessing(str) → str;
neural_net_run(str) → None;
data_plot(str)
websockets server_init()
Remaining Internal Functions...

Project Timeline

Current Date
v

Subsystem Designs and Testing (completed 1/27)	Testing API Model Inference Support and connected functions (completed 2/10)	Integration of Data Processing Functions (to complete by 2/24)	Integration of API Model Inference Support (to complete by 3/17)	Validation (to complete by 3/31)	Final Improvements and Report (to complete by 4/14)	Demo and Final Presentation (to complete by 4/28)

-Data Classification-

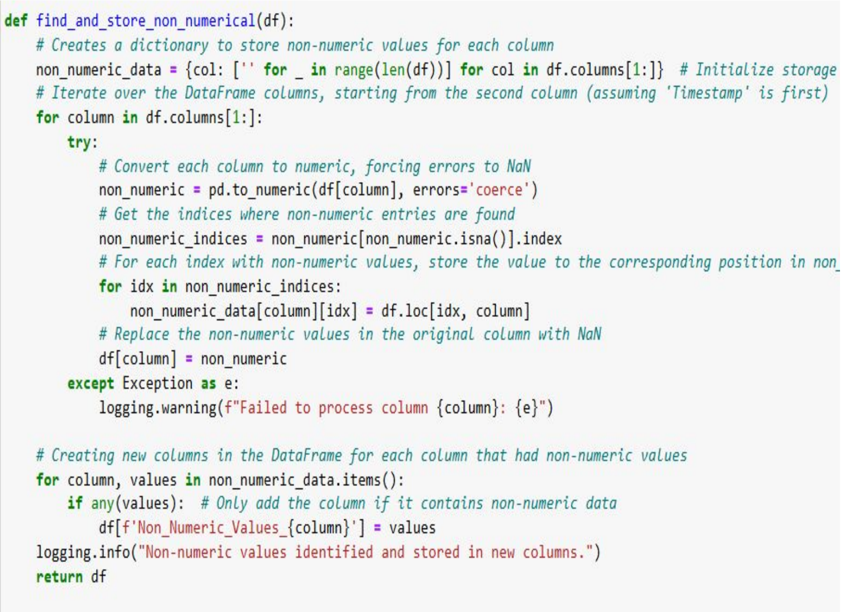
Digvijay Alluri

Current Status	Problems
Testing and debugging: Conduct testing with real data to ensure compatibility.	Encoding errors. Fixed it using python code

-Data Classification-

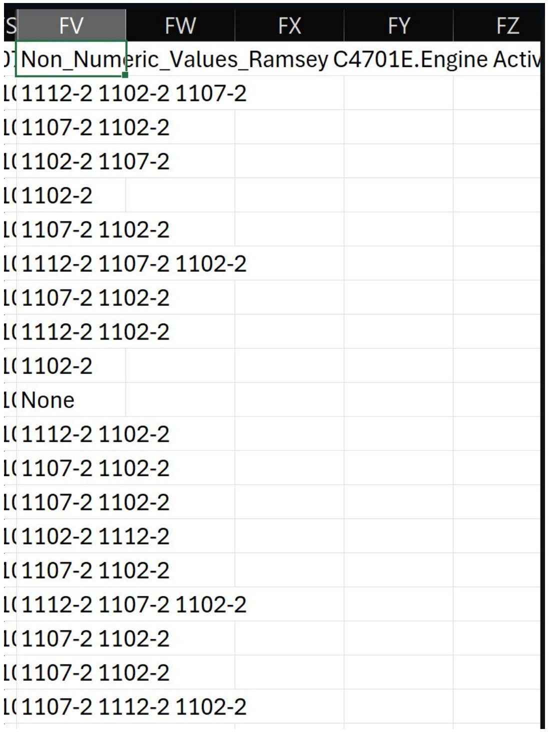
Digvijay Alluri

Step 1 : Store all the error codes in different columns with their respective columns names.



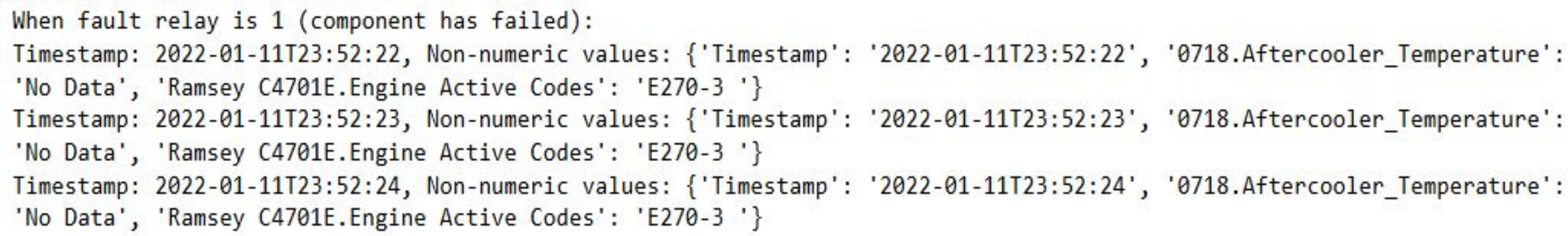
```python
def find_and_store_non_numerical(df):
    # Creates a dictionary to store non-numeric values for each column
    non_numeric_data = {col: ['' for _ in range(len(df))] for col in df.columns[1:]}  # Initialize storage
    # Iterate over the DataFrame columns, starting from the second column (assuming 'Timestamp' is first)
    for column in df.columns[1:]:
        try:
            # Convert each column to numeric, forcing errors to NaN
            non_numeric = pd.to_numeric(df[column], errors='coerce')
            # Get the indices where non-numeric entries are found
            non_numeric_indices = non_numeric[non_numeric.isna()].index
            # For each index with non-numeric values, store the value to the corresponding position in non_
            for idx in non_numeric_indices:
                non_numeric_data[column][idx] = df.loc[idx, column]
            # Replace the non-numeric values in the original column with NaN
            df[column] = non_numeric
        except Exception as e:
            logging.warning(f"Failed to process column {column}: {e}")

    # Creating new columns in the DataFrame for each column that had non-numeric values
    for column, values in non_numeric_data.items():
        if any(values):  # Only add the column if it contains non-numeric data
            df[f'Non_Numeric_Values_{column}'] = values
    logging.info("Non-numeric values identified and stored in new columns.")
    return df
```

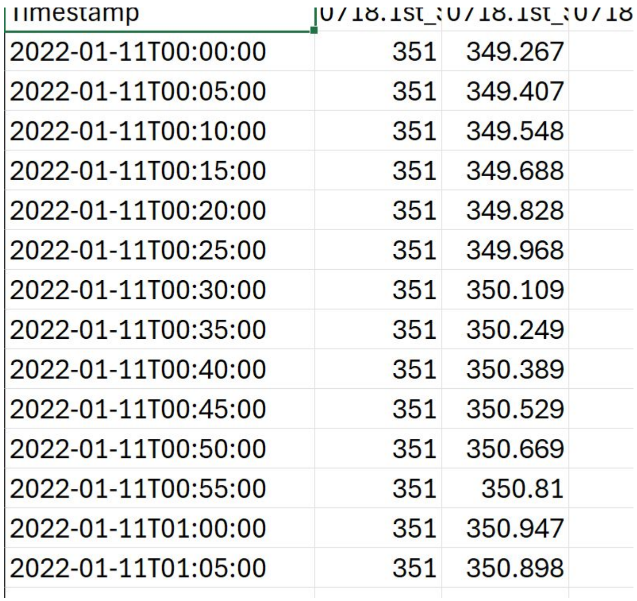


S	FV	FW	FX	FY	FZ
0	Non_Numeric_Values_Ramsey C4701E.Engine Activ				
1(	1112-2 1102-2 1107-2				
1(	1107-2 1102-2				
1(	1102-2 1107-2				
1(	1102-2				
1(	1107-2 1102-2				
1(	1112-2 1107-2 1102-2				
1(	1107-2 1102-2				
1(	1112-2 1102-2				
1(	1102-2				
1(	None				
1(	1112-2 1102-2				
1(	1107-2 1102-2				
1(	1107-2 1102-2				
1(	1102-2 1112-2				
1(	1107-2 1102-2				
1(	1112-2 1107-2 1102-2				
1(	1107-2 1102-2				
1(	1107-2 1102-2				
1(	1107-2 1112-2 1102-2				

Step 2 and 3 : Averaging the data and analyzing when the machine shuts down and why.



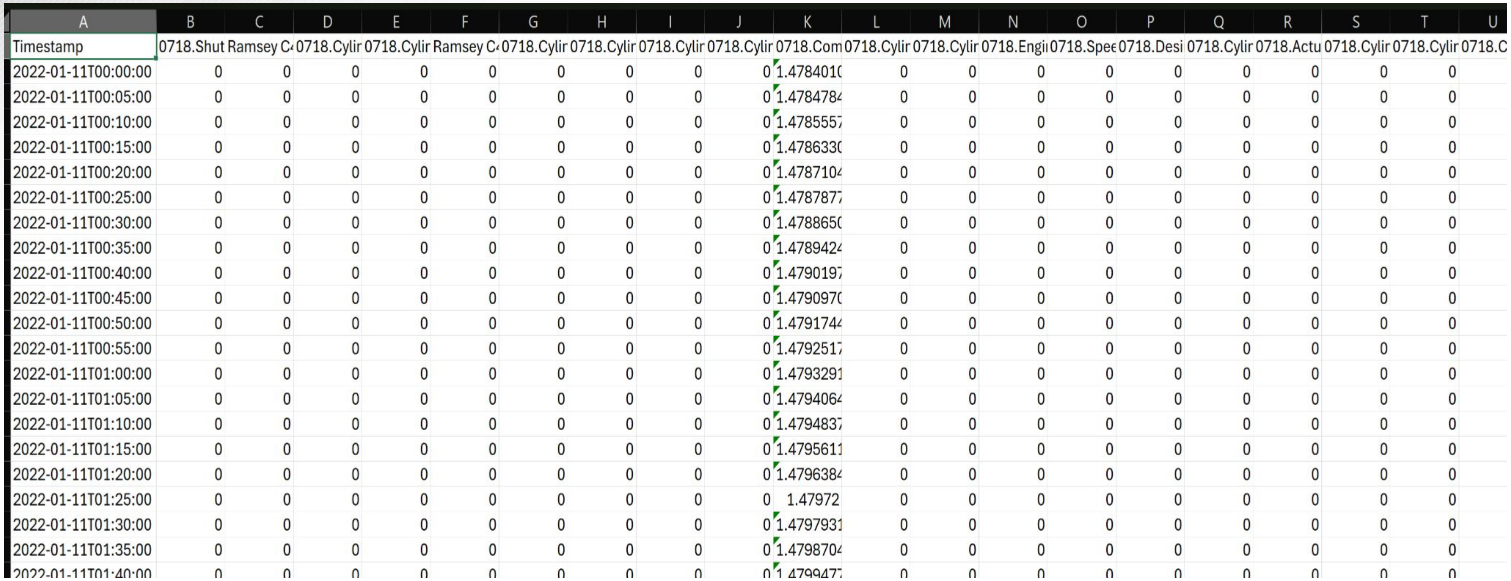
```
When fault relay is 1 (component has failed):
Timestamp: 2022-01-11T23:52:22, Non-numeric values: {'Timestamp': '2022-01-11T23:52:22', '0718.Aftercooler_Temperature':
'No Data', 'Ramsey C4701E.Engine Active Codes': 'E270-3 '}
Timestamp: 2022-01-11T23:52:23, Non-numeric values: {'Timestamp': '2022-01-11T23:52:23', '0718.Aftercooler_Temperature':
'No Data', 'Ramsey C4701E.Engine Active Codes': 'E270-3 '}
Timestamp: 2022-01-11T23:52:24, Non-numeric values: {'Timestamp': '2022-01-11T23:52:24', '0718.Aftercooler_Temperature':
'No Data', 'Ramsey C4701E.Engine Active Codes': 'E270-3 '}
```



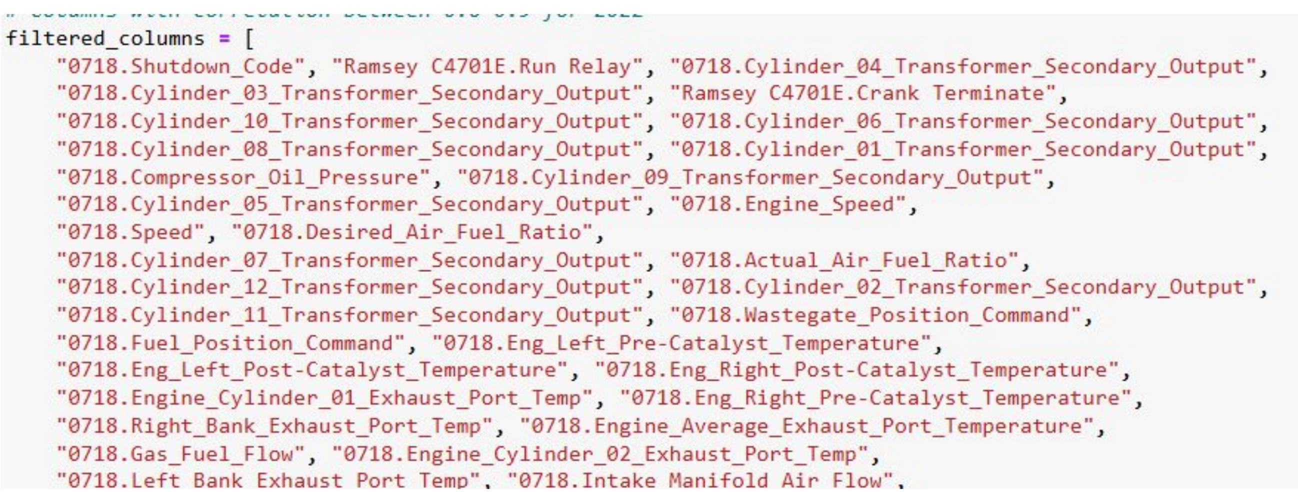
Timestamp	0718.1st_S	0718.1st_S	0718
2022-01-11T00:00:00	351	349.267	
2022-01-11T00:05:00	351	349.407	
2022-01-11T00:10:00	351	349.548	
2022-01-11T00:15:00	351	349.688	
2022-01-11T00:20:00	351	349.828	
2022-01-11T00:25:00	351	349.968	
2022-01-11T00:30:00	351	350.109	
2022-01-11T00:35:00	351	350.249	
2022-01-11T00:40:00	351	350.389	
2022-01-11T00:45:00	351	350.529	
2022-01-11T00:50:00	351	350.669	
2022-01-11T00:55:00	351	350.81	
2022-01-11T01:00:00	351	350.947	
2022-01-11T01:05:00	351	350.898	

-Data Classification-

Digvijay Alluri



A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
Timestamp	0718.Shut	Ramsey C4	0718.Cylir	0718.Cylir	Ramsey C4	0718.Cylir	0718.Cylir	0718.Cylir	0718.Cylir	0718.Com	0718.Cylir	0718.Cylir	0718.Engi	0718.Spee	0718.Desi	0718.Cylir	0718.Actu	0718.Cylir	0718.Cylir	0718.C
2022-01-11T00:00:00	0	0	0	0	0	0	0	0	0	1.4784010	0	0	0	0	0	0	0	0	0	
2022-01-11T00:05:00	0	0	0	0	0	0	0	0	0	1.4784784	0	0	0	0	0	0	0	0	0	
2022-01-11T00:10:00	0	0	0	0	0	0	0	0	0	1.4785557	0	0	0	0	0	0	0	0	0	
2022-01-11T00:15:00	0	0	0	0	0	0	0	0	0	1.4786330	0	0	0	0	0	0	0	0	0	
2022-01-11T00:20:00	0	0	0	0	0	0	0	0	0	1.4787104	0	0	0	0	0	0	0	0	0	
2022-01-11T00:25:00	0	0	0	0	0	0	0	0	0	1.4787877	0	0	0	0	0	0	0	0	0	
2022-01-11T00:30:00	0	0	0	0	0	0	0	0	0	1.4788650	0	0	0	0	0	0	0	0	0	
2022-01-11T00:35:00	0	0	0	0	0	0	0	0	0	1.4789424	0	0	0	0	0	0	0	0	0	
2022-01-11T00:40:00	0	0	0	0	0	0	0	0	0	1.4790197	0	0	0	0	0	0	0	0	0	
2022-01-11T00:45:00	0	0	0	0	0	0	0	0	0	1.4790970	0	0	0	0	0	0	0	0	0	
2022-01-11T00:50:00	0	0	0	0	0	0	0	0	0	1.4791744	0	0	0	0	0	0	0	0	0	
2022-01-11T00:55:00	0	0	0	0	0	0	0	0	0	1.4792517	0	0	0	0	0	0	0	0	0	
2022-01-11T01:00:00	0	0	0	0	0	0	0	0	0	1.4793291	0	0	0	0	0	0	0	0	0	
2022-01-11T01:05:00	0	0	0	0	0	0	0	0	0	1.4794064	0	0	0	0	0	0	0	0	0	
2022-01-11T01:10:00	0	0	0	0	0	0	0	0	0	1.4794837	0	0	0	0	0	0	0	0	0	
2022-01-11T01:15:00	0	0	0	0	0	0	0	0	0	1.4795611	0	0	0	0	0	0	0	0	0	
2022-01-11T01:20:00	0	0	0	0	0	0	0	0	0	1.4796384	0	0	0	0	0	0	0	0	0	
2022-01-11T01:25:00	0	0	0	0	0	0	0	0	0	1.47972	0	0	0	0	0	0	0	0	0	
2022-01-11T01:30:00	0	0	0	0	0	0	0	0	0	1.4797931	0	0	0	0	0	0	0	0	0	
2022-01-11T01:35:00	0	0	0	0	0	0	0	0	0	1.4798704	0	0	0	0	0	0	0	0	0	
2022-01-11T01:40:00	0	0	0	0	0	0	0	0	0	1.4799477	0	0	0	0	0	0	0	0	0	

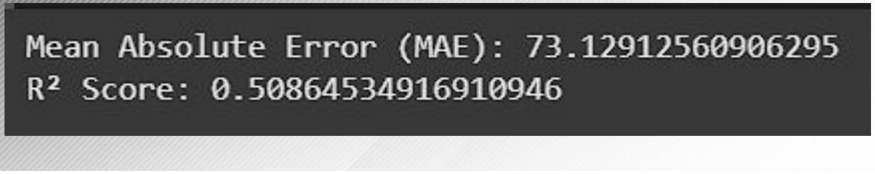


```
# columns with correlation between 0.6-0.9 for 2022
filtered_columns = [
    "0718.Shutdown_Code", "Ramsey C4701E.Run Relay", "0718.Cylinder_04_Transformer_Secondary_Output",
    "0718.Cylinder_03_Transformer_Secondary_Output", "Ramsey C4701E.Crank Terminate",
    "0718.Cylinder_10_Transformer_Secondary_Output", "0718.Cylinder_06_Transformer_Secondary_Output",
    "0718.Cylinder_08_Transformer_Secondary_Output", "0718.Cylinder_01_Transformer_Secondary_Output",
    "0718.Compressor_Oil_Pressure", "0718.Cylinder_09_Transformer_Secondary_Output",
    "0718.Cylinder_05_Transformer_Secondary_Output", "0718.Engine_Speed",
    "0718.Speed", "0718.Desired_Air_Fuel_Ratio",
    "0718.Cylinder_07_Transformer_Secondary_Output", "0718.Actual_Air_Fuel_Ratio",
    "0718.Cylinder_12_Transformer_Secondary_Output", "0718.Cylinder_02_Transformer_Secondary_Output",
    "0718.Cylinder_11_Transformer_Secondary_Output", "0718.Wastegate_Position_Command",
    "0718.Fuel_Position_Command", "0718.Eng_Left_Pre-Catalyst_Temperature",
    "0718.Eng_Left_Post-Catalyst_Temperature", "0718.Eng_Right_Post-Catalyst_Temperature",
    "0718.Engine_Cylinder_01_Exhaust_Port_Temp", "0718.Eng_Right_Pre-Catalyst_Temperature",
    "0718.Right_Bank_Exhaust_Port_Temp", "0718.Engine_Average_Exhaust_Port_Temperature",
    "0718.Gas_Fuel_Flow", "0718.Engine_Cylinder_02_Exhaust_Port_Temp",
    "0718.Left_Bank_Exhaust_Port_Temp", "0718.Intake_Manifold_Air_Flow",
```

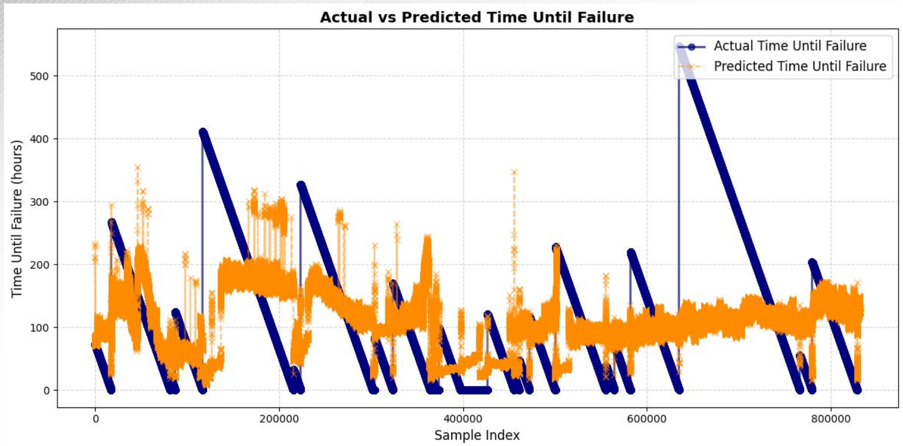
- Random Forest Model-

Michael Cubriel

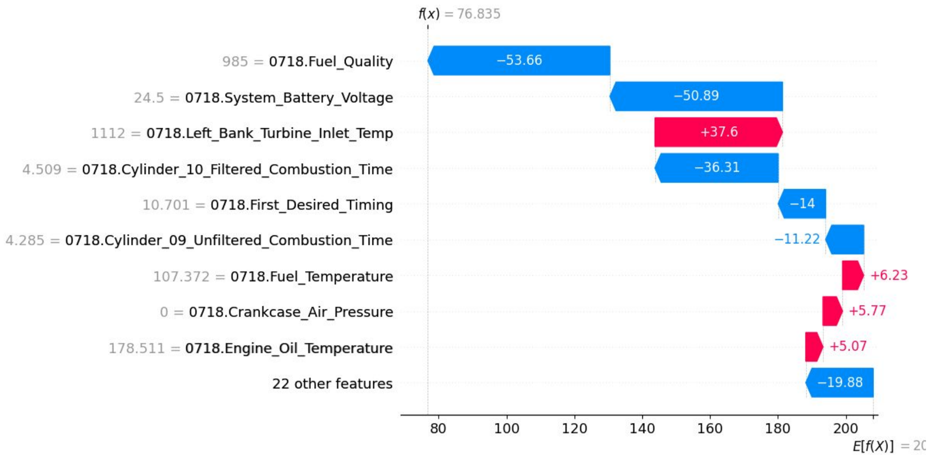
Current Status	Issues
● Conducted testing and validation. ● Made minor improvements to model.	● Desired accuracy was not achieved. ● Predicting why failure happened.

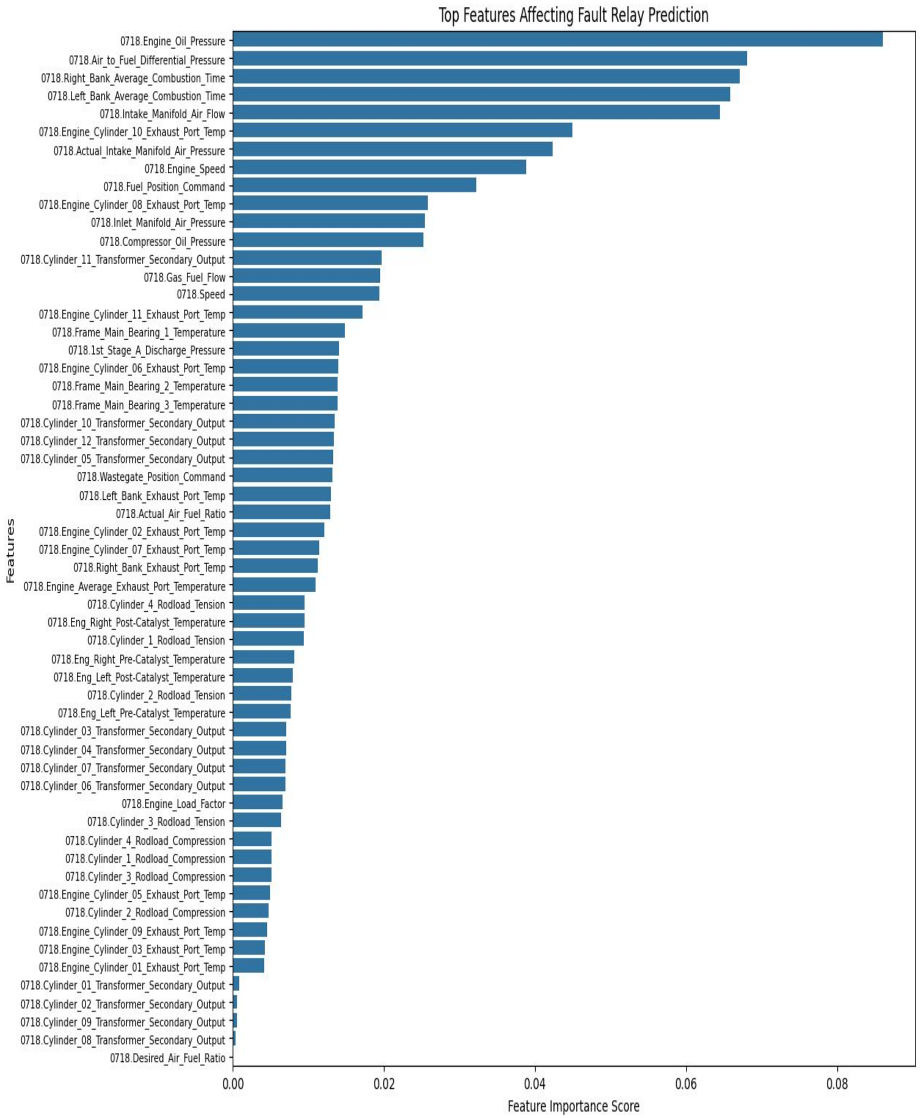


```
Mean Absolute Error (MAE): 73.12912560906295
R² Score: 0.50864534916910946
```



Actual vs Predicted Time Until Failure



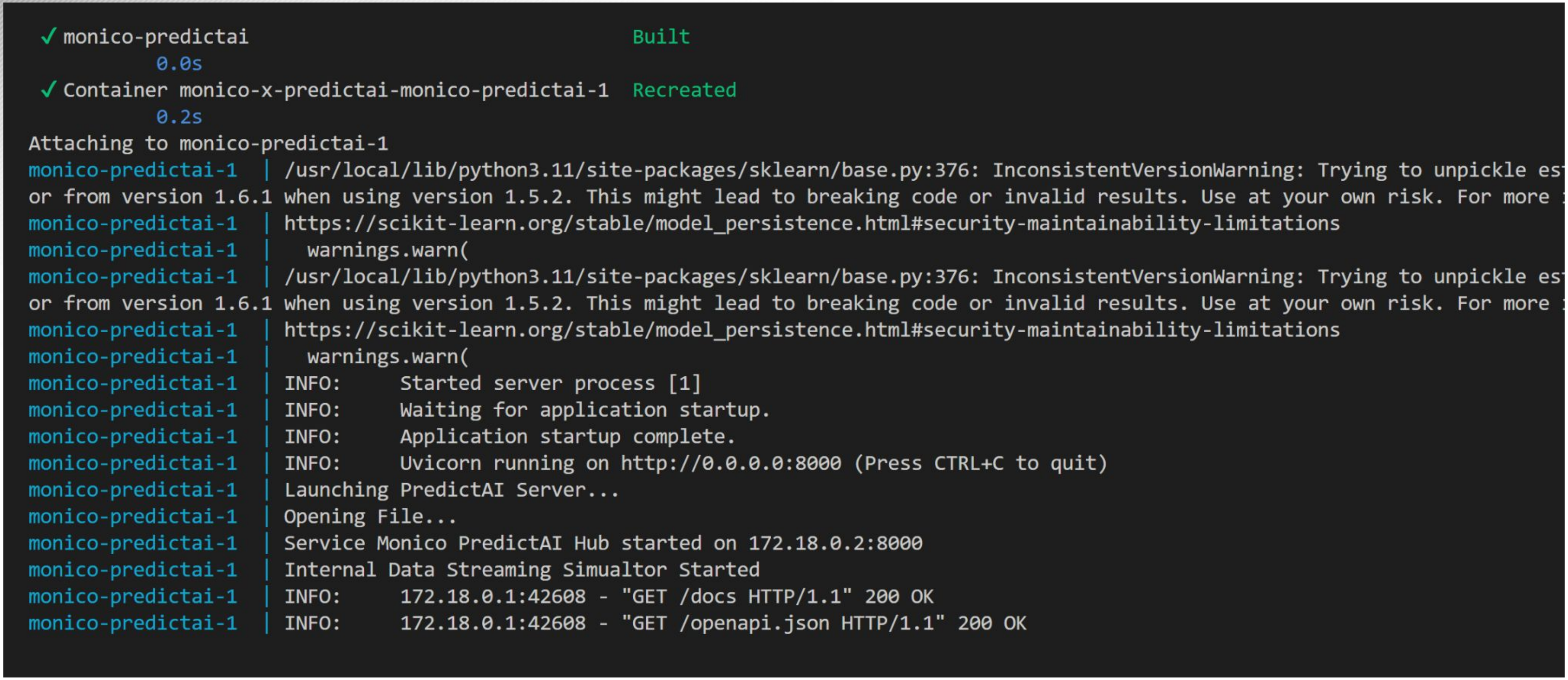


Top Features Affecting Fault Relay Prediction

- Networking & API -

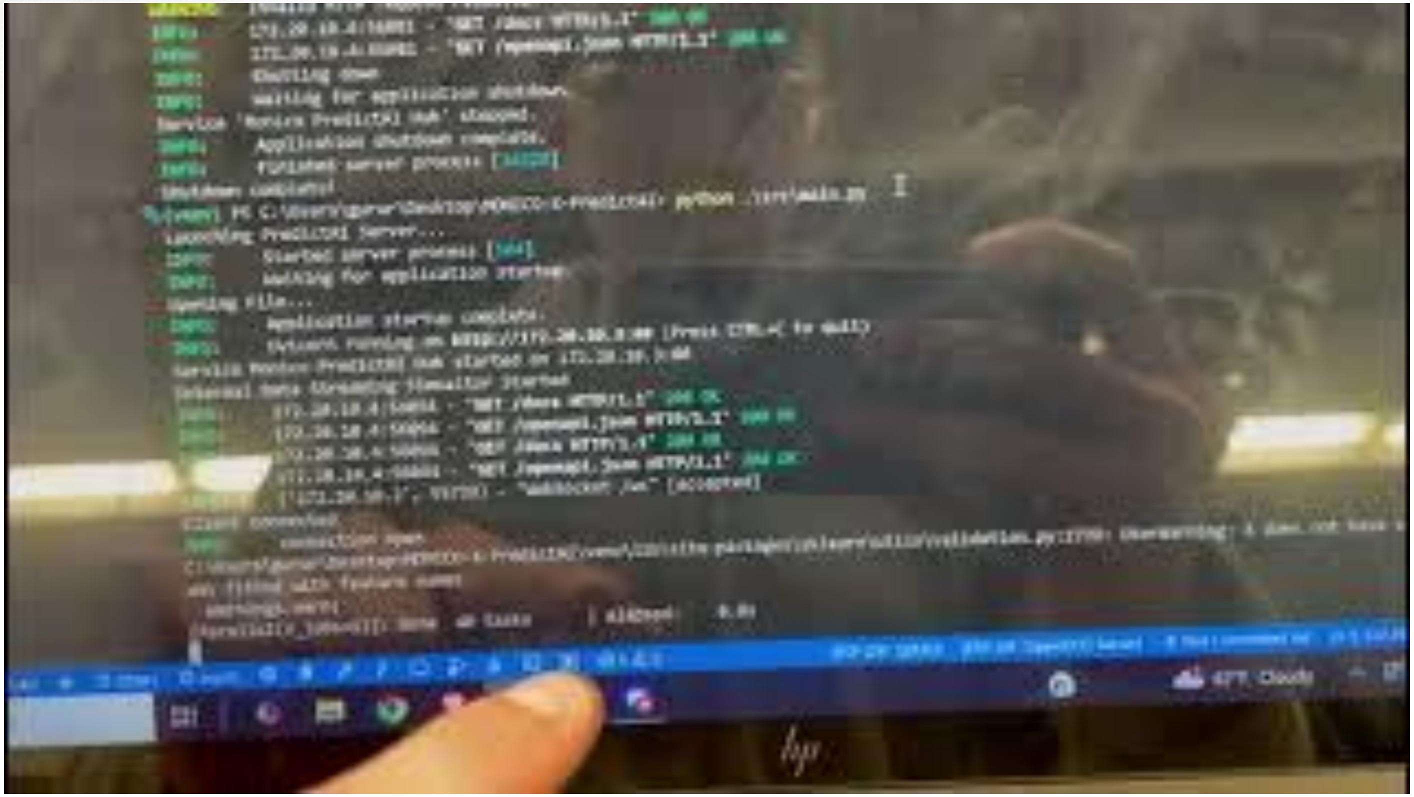
Nicho Naugle

Current Status	Issues
- Setup dockerfile - Launched service in docker	- Ready to demo (cant use the mdns on windows since wsl is blocking it however it does work on linux based machines. Will just run natively in a virtual environment)



```
 ✓ monico-predictai                                 Built
            0.0s
 ✓ Container monico-x-predictai-monico-predictai-1  Recreated
            0.2s
Attaching to monico-predictai-1
monico-predictai-1  | /usr/local/lib/python3.11/site-packages/sklearn/base.py:376: InconsistentVersionWarning: Trying to unpickle es
or from version 1.6.1 when using version 1.5.2. This might lead to breaking code or invalid results. Use at your own risk. For more
monico-predictai-1  | https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
monico-predictai-1  |   warnings.warn(
monico-predictai-1  | /usr/local/lib/python3.11/site-packages/sklearn/base.py:376: InconsistentVersionWarning: Trying to unpickle es
or from version 1.6.1 when using version 1.5.2. This might lead to breaking code or invalid results. Use at your own risk. For more
monico-predictai-1  | https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
monico-predictai-1  |   warnings.warn(
monico-predictai-1  | INFO:     Started server process [1]
monico-predictai-1  | INFO:     Waiting for application startup.
monico-predictai-1  | INFO:     Application startup complete.
monico-predictai-1  | INFO:     Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
monico-predictai-1  | Launching PredictAI Server...
monico-predictai-1  | Opening File...
monico-predictai-1  | Service Monico PredictAI Hub started on 172.18.0.2:8000
monico-predictai-1  | Internal Data Streaming Simualtor Started
monico-predictai-1  | INFO:     172.18.0.1:42608 - "GET /docs HTTP/1.1" 200 OK
monico-predictai-1  | INFO:     172.18.0.1:42608 - "GET /openapi.json HTTP/1.1" 200 OK
```

Integrated system Results - Our DEMO



```
INFO:     172.20.10.4:[illegible] - "GET /docs HTTP/1.1" 200 OK
INFO:     172.20.10.4:[illegible] - "GET /openapi.json HTTP/1.1" 200 OK
INFO:     Shutting down
INFO:     Waiting for application shutdown.
Service '[illegible] PredictAI Hub' stopped.
INFO:     Application shutdown complete.
INFO:     Finished server process [[illegible]]
Shutdown complete!
(venv) PS C:\Users\[illegible]\Desktop\[illegible]-PredictAI> python .\src\main.py
Launching PredictAI Server...
INFO:     Started server process [[illegible]]
INFO:     Waiting for application startup.
Opening file...
INFO:     Application startup complete.
INFO:     Uvicorn running on http://172.20.10.[illegible] (Press CTRL+C to quit)
Service [illegible] PredictAI Hub started on 172.20.10.[illegible]
[illegible] Data Streaming Simulator Started
INFO:     172.20.10.4:[illegible] - "GET /docs HTTP/1.1" 200 OK
INFO:     172.20.10.4:[illegible] - "GET /openapi.json HTTP/1.1" 200 OK
INFO:     172.20.10.4:[illegible] - "GET /docs HTTP/1.1" 200 OK
INFO:     172.20.10.4:[illegible] - "GET /openapi.json HTTP/1.1" 200 OK
INFO:     ('172.20.10.1', [illegible]) - "WebSocket /ws" [accepted]
Client connected
INFO:     connection open
C:\Users\[illegible]\Desktop\[illegible]-PredictAI\venv\Lib\site-packages\sklearn\utils\validation.py:[illegible]: UserWarning: X does not have v[illegible]
was fitted with feature names
  warnings.warn(
[Parallel(n_jobs=[illegible])]: Done  [illegible] tasks      | elapsed:    0.0s
```

Conclusion

Our system enables real-time monitoring and predictive maintenance for industrial machines by combining data analysis, machine learning, and edge computing. It reduces downtime, improves reliability, and sets the stage for smarter industrial operations.