Vardan S Kamra

Wireframe Document
Predictive Maintenance

Version: 1.0

Last Date of Revision: 16/07/2024

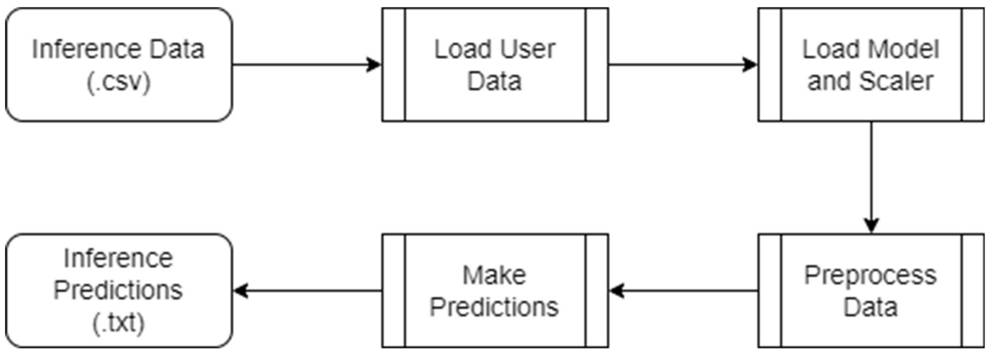
Contents

1. Introduction

The Predictive Maintenance Project is designed to analyze and predict the Remaining Useful Life (RUL) of engines based on historical operational data. This project utilizes advanced machine learning techniques to model engine degradation and provide timely predictions that can help in planning maintenance and avoiding unexpected failures. Given the critical nature of such predictions in industries like aerospace and manufacturing, the accuracy and reliability of the model are paramount.

While this project does not have a traditional graphical user interface (GUI), it plays a vital role in automating the analysis of large datasets and generating actionable insights. The core functionality is encapsulated within the main.py script, which reads an input CSV file containing engine performance data, preprocesses the data, and uses a neural network model to predict the RUL of each engine. The results are then written to an output TXT file, which can be further utilized for decision-making and maintenance planning.

2. Data Flow



The data flow diagram illustrates the process from reading the input CSV file, processing the data, running predictions, and generating the output file.

Key Components:

1. Input CSV File:
    - Contains engine operational data (sensor readings, cycle counts, etc.).
2. Data Preprocessing:
    - Standardization of data.
    - Feature engineering.
    - Data validation.
3. Model Prediction:
    - Neural network model predicts RUL based on preprocessed data.
4. Output TXT File:
    - Contains predicted RUL values for each engine.

3. Deployment Components

3.1 Input Handling

Description:

- The script starts by reading the input CSV file.
- Verifies the data format and integrity.

Pseudo-Code:

```
import pandas as pd
data = pd.read_csv('input.csv')
```

3.2 Data Preprocessing

Description:

- Processes the raw data into a format suitable for model input.
- Includes standardization, handling missing values (if any), and feature extraction.

Pseudo-Code:

```
from sklearn.preprocessing import StandardScaler
with open("app/models/scaler.pkl", 'rb') as f:
    loaded_scaler = pickle.load(f)
data_scaled = scaler.transform(data)
```

3.3 Model Prediction

Description:

- Uses a pre-trained neural network model to predict the RUL.

Pseudo-Code:

```
from tensorflow.keras.models import load_model
model = load_model('rul_model.keras')
predictions = model.predict(data_scaled)
```

3.4 Output Generation

Description:

- Writes the predicted RUL values to a TXT file.

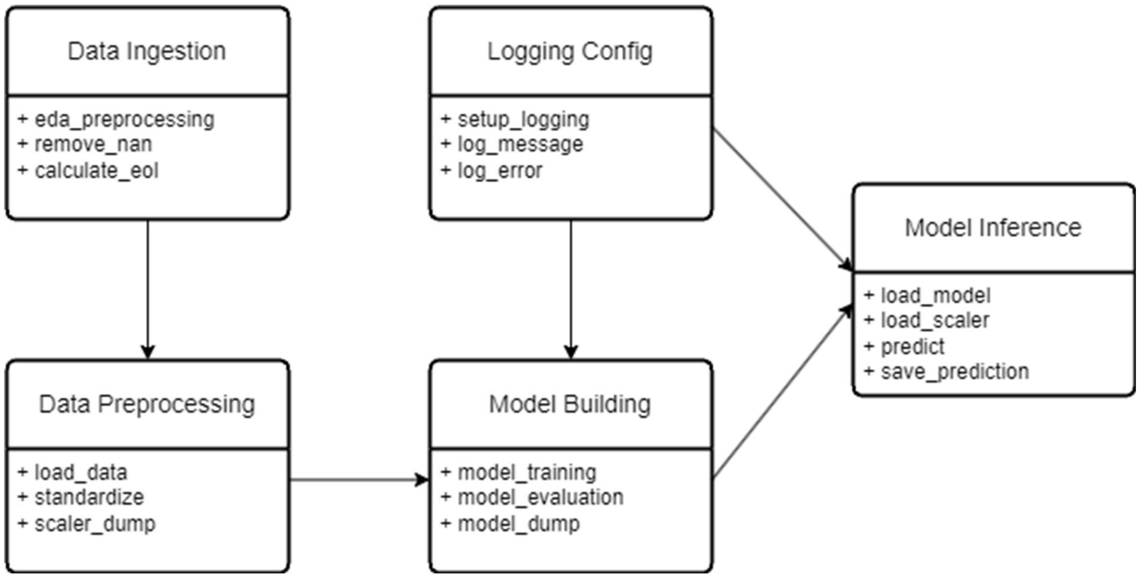
Pseudo-Code:

```
with open('output.txt', 'w') as f:
    for pred in predictions:
        f.write(f"{pred}\n")
```

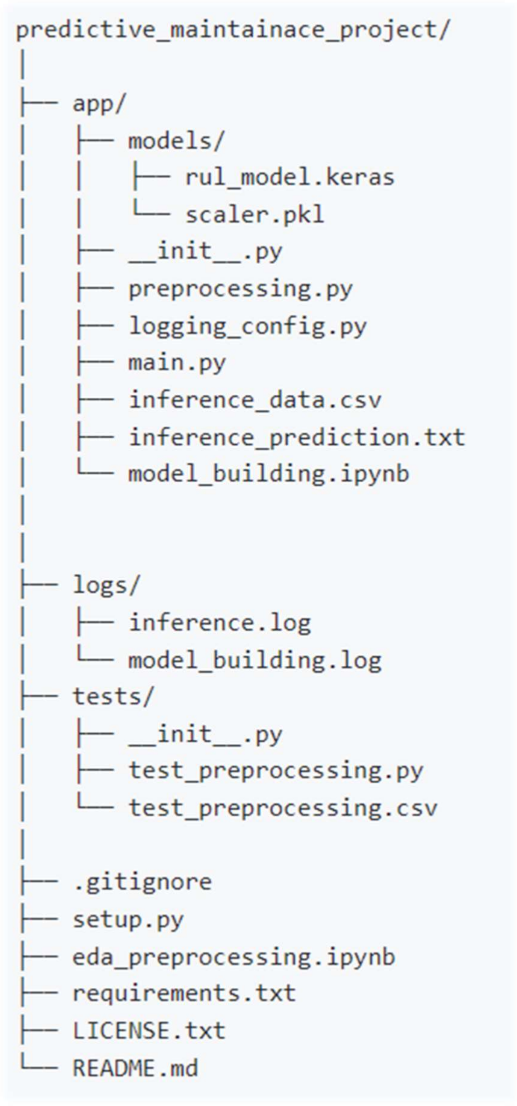
4. Wireframe Representation

Since there's no need for a user interface in such a project, this section focuses on the logical steps and file flow.

4.1 Logical Flow Diagram



4.2 File Structure Overview



```
predictive_maintainace_project/
|
├── app/
|   ├── models/
|   |   ├── rul_model.keras
|   |   └── scaler.pkl
|   ├── __init__.py
|   ├── preprocessing.py
|   ├── logging_config.py
|   ├── main.py
|   ├── inference_data.csv
|   ├── inference_prediction.txt
|   └── model_building.ipynb
|
|
├── logs/
|   ├── inference.log
|   └── model_building.log
├── tests/
|   ├── __init__.py
|   ├── test_preprocessing.py
|   └── test_preprocessing.csv
|
├── .gitignore
├── setup.py
├── eda_preprocessing.ipynb
├── requirements.txt
├── LICENSE.txt
└── README.md
```

5. Conclusion

This wireframe document provides an overview of the data processing flow and the structure of the main.py script in the Predictive Maintenance Project. It serves as a guide for understanding how the input data is transformed into predictions and saved to the output file, even in the absence of a graphical user interface.