Predictive Maintenance for Ship Components Using Regression AI Model

Introduction:

The aim of this project was to develop a predictive maintenance system for ship components using a regression AI model. Predictive maintenance helps ship operators determine when equipment needs servicing, which helps reduce downtime and maintenance costs. This report outlines the methodology used, including data preprocessing, model training, parameter tuning, and performance evaluation.

Data Preparation:

The dataset provided contained several features including ship sensor readings such as lever position, shaft torque, revolutions, and more. The target variable was the ship's decay state (GT_state), which indicates the health of the ship components.

Steps taken to prepare the data:

1. Data Cleaning: The dataset was cleaned by removing missing values and duplicates.

2. Outlier Handling: Winsorization was applied to manage outliers in the Injecton_control feature, capping extreme values to mitigate their influence on model performance.

3. Feature Scaling: All input features were standardized using **StandardScaler** to ensure they had a mean of 0 and a standard deviation of 1. Feature scaling helps gradient-based learning algorithms converge faster and ensures equal treatment of all features.

Model Training:

A neural network regression model was built using **PyTorch**. The model consists of two fully connected hidden layers, each followed by **ReLU activation** and **dropout** to reduce overfitting. The output layer is a single neuron for predicting the **GT_state**.

Model Architecture:

- Input Layer: 14 features
- Hidden Layer 1: 128 neurons, followed by **ReLU** and **Dropout (rate = 0.2)**
- Hidden Layer 2: 64 neurons, followed by **ReLU** and **Dropout (rate = 0.2)**
- Output Layer: 1 neuron (for regression output)

-Training Process:

- The model was trained using Adam optimizer with a learning rate of 0.001.
- Mean Squared Error (MSE) was used as the loss function.
- The model was trained for 100 epochs with a batch size of 64.

Hyperparameter Tuning:

To optimize the model’s performance, GridSearchCV was used to perform hyperparameter tuning. The following parameters were considered:

- Hidden Layer Sizes: [64, 128, 256]
- Dropout Rate: [0.1, 0.2, 0.3]

A custom wrapper was created for the PyTorch model to make it compatible with GridSearchCV from scikit-learn. The grid search used 5-fold cross-validation to find the best combination of hyperparameters that minimized the negative mean squared error.

Evaluation and Results:

After hyperparameter tuning, the best model was selected and evaluated on the test set. The performance metrics were:

- Average Test Loss (MSE): 0.0001

- RMSE: 0.0076

These values indicate that the model performed very well in predicting the **GT_state**, with an extremely low error. The low **RMSE** suggests that the model can accurately predict the decay state, indicating effective generalization to unseen data.

Discussion:

The initial model exhibited signs of overfitting, as evidenced by near-zero training loss and extremely low test error. To address this, **dropout layers** were added, and **hyperparameter tuning** was performed to find the optimal model configuration. The use of dropout and careful tuning significantly improved the model's ability to generalize.

Conclusion:

This project successfully developed a predictive maintenance model for ship components using a neural network regression approach. The model was trained on pre-processed data, with outlier handling, feature scaling, and hyperparameter tuning to ensure optimal performance. The final model demonstrated excellent prediction accuracy with low average test loss and RMSE.

Future Improvements:

- Feature Engineering: Additional feature engineering could improve model performance by adding domain-specific knowledge.

- Regularization Techniques: Further experimentation with L2 regularization (weight decay) could help improve model robustness.

- Additional Data: Incorporating more diverse training data could improve the model's generalization to different ship conditions and environments.

Appendix:

- Tools Used: Python, PyTorch, scikit-learn, Pandas, NumPy, Google Colab

- Key Libraries: PyTorch for model development, GridSearchCV for hyperparameter tuning, StandardScaler for feature scaling.