Predicting Maintenance Needs for Military Assets Using Deep Learning

BUAN303

End Semester Final Assessment

Machine Learning 2: Deep Learning

Prof. Kedar Kanhere

Date: 18 April, 2025

Varsha Gunturu

Student No: 220076

Table of Contents

Introduction

In military operations, the reliability and readiness of assets such as vehicles, equipment, and infrastructure are mission-critical. Unplanned maintenance or equipment failure can lead to operational delays, increased costs, and in some cases, compromised safety. To address these challenges, predictive maintenance systems are becoming an integral part of modern defense logistics.

This project explores machine learning approaches to predict whether a military asset will require maintenance, based on historical and operational data. By identifying potential failures before they occur, the goal is to assist decision-makers in planning proactive maintenance activities, optimizing resource allocation, and improving overall asset lifecycle management.

To solve this classification problem, we evaluate two powerful machine learning models: TabNet, a deep learning model designed specifically for tabular data, and Random Forest, a widely used ensemble-based algorithm. Their performance is assessed using key evaluation metrics including AUC-ROC, Precision-Recall, and the Matthews Correlation Coefficient (MCC).

Methodology

The study follows a structured workflow to ensure the reliability and generalizability of results:

- Data Preprocessing: the data will be checked for
    - null values
    - duplicate values
    - class imbalance
    - distribution of continuous variables
    - Box Plot for outliers
    - Correlation

- Manual mapping of Categorical variables
- Feature Engineering to capture complex patterns
- SMOTENC will be applied for high class imbalanced data
- Data Splitting and Cross-Validation:

  A Stratified K-Fold Cross-Validation approach is used to ensure balanced class distribution across folds and avoid overfitting. Each model is trained and validated across five folds.
- Model Selection:
  - Two models will be built, one for detecting when an asset will fail and one when an asset requires maintenance.
  - TabNet is leveraged for its interpretability and native handling of tabular data with mixed feature types (categorical and numerical). Key techniques include the use of categorical embeddings, early stopping, learning rate scheduling, and evaluation on holdout data (X_val_final).
  - Random Forest, a bagging-based ensemble method, known for robustness to noisy data and interpretability through feature importance.
- Evaluation Metrics:

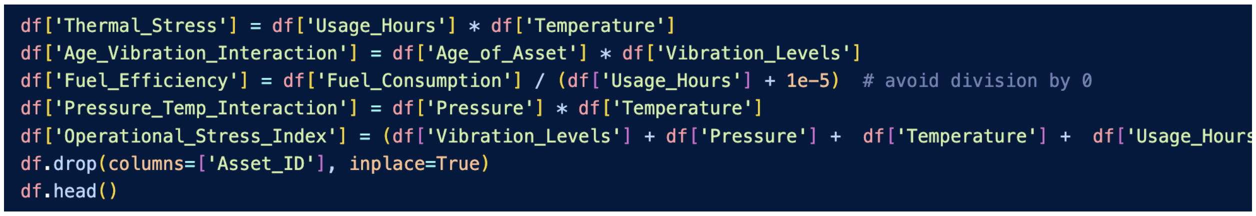
  Performance is measured using:
  - AUC-ROC to capture the trade-off between true positive and false positive rates,
  - Precision-Recall Curves to assess performance on imbalanced datasets,
  - Confusion matrix
  - Matthews Correlation Coefficient (MCC) as a balanced metric that considers all confusion matrix categories,

- Classification Reports

Data Preprocessing

The dataset contains no null values and duplicates, continuous variables in normal distribution, with no outliers and no multicollinearity. Refer code for deeper analysis.

Feature Engineering



```python
df['Thermal_Stress'] = df['Usage_Hours'] * df['Temperature']
df['Age_Vibration_Interaction'] = df['Age_of_Asset'] * df['Vibration_Levels']
df['Fuel_Efficiency'] = df['Fuel_Consumption'] / (df['Usage_Hours'] + 1e-5)  # avoid division by 0
df['Pressure_Temp_Interaction'] = df['Pressure'] * df['Temperature']
df['Operational_Stress_Index'] = (df['Vibration_Levels'] + df['Pressure'] +  df['Temperature'] +  df['Usage_Hours
df.drop(columns=['Asset_ID'], inplace=True)
df.head()
```

These features were engineered to capture interactions between variables that may not be obvious from individual inputs, reflect real-world mechanical stress patterns and improve model predictive performance by injecting domain knowledge into the data.

1. **Thermal_Stress:** Captures the cumulative thermal load an asset experiences.

   Reflects extended usage in high-temperature conditions, which can accelerate wear and tear.

2. **Age_Vibration_Interaction:** Combines mechanical wear (vibration) with asset aging.

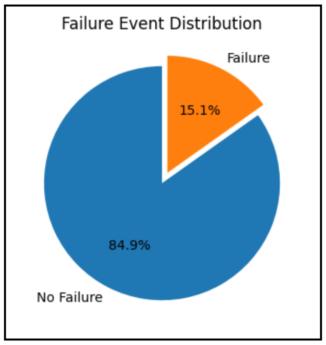
   Older assets with high vibration levels are more prone to failure, making this an important risk indicator.

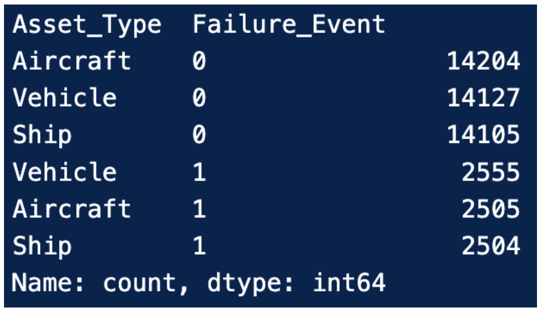
3. **Fuel_Efficiency:** Represents how efficiently an asset consumes fuel relative to its operational time. Declining efficiency may signal engine or system degradation, indicating the need for maintenance.

4. **Pressure_Temp_Interaction:** Models the combined stress of internal pressure and temperature. High values could indicate overheating or internal system strain, which may lead to failure.
5. **Operational_Stress_Index:** A composite score that reflects the overall operational stress experienced by an asset. Normalized by asset age to provide a measure of how much load the asset is handling relative to its lifecycle. Higher values may suggest overuse or abnormal stress, potentially increasing failure risk.

Class distribution

For Failure Event Model

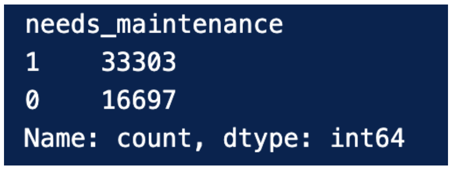




```
Asset_Type  Failure_Event
Aircraft    0                14204
Vehicle     0                14127
Ship        0                14105
Vehicle     1                 2555
Aircraft    1                 2505
Ship        1                 2504
Name: count, dtype: int64
```

The model suffers an alarming imbalance in class 1(Failure_Event) and class 0(No Failure). There are only approximately 2500 samples for Failure Event for each asset type, whereas the no failure class is 14204. Therefore, to make sure the model is not completely influenced by the majority class, SMOTE NC that can handle numeric and categorical variables is used. Further Class weights have also been used. Note that SMOTE NC has been used to generate minority class to be 50% of majority class and class weights have been used in combination.

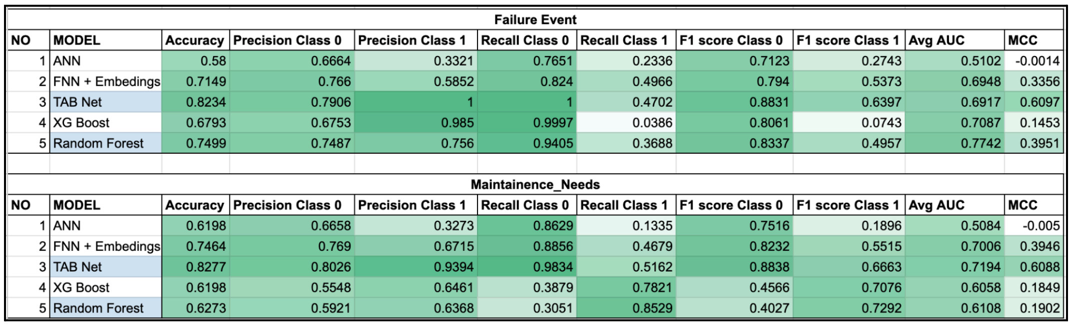
For Maintenance Needs Model



```
needs_maintenance
1    33303
0    16697
Name: count, dtype: int64
```

The above picture also shows imbalance for maintainence_needs dataset, therefore the same strategy of SMOTE NC and class weights have been used.

Model Selection



Failure Event										
NO	MODEL	Accuracy	Precision Class 0	Precision Class 1	Recall Class 0	Recall Class 1	F1 score Class 0	F1 score Class 1	Avg AUC	MCC
1	ANN	0.58	0.6664	0.3321	0.7651	0.2336	0.7123	0.2743	0.5102	-0.0014
2	FNN + Embedings	0.7149	0.766	0.5852	0.824	0.4966	0.794	0.5373	0.6948	0.3356
3	TAB Net	0.8234	0.7906	1	1	0.4702	0.8831	0.6397	0.6917	0.6097
4	XG Boost	0.6793	0.6753	0.985	0.9997	0.0386	0.8061	0.0743	0.7087	0.1453
5	Random Forest	0.7499	0.7487	0.756	0.9405	0.3688	0.8337	0.4957	0.7742	0.3951

Maintainence_Needs										
NO	MODEL	Accuracy	Precision Class 0	Precision Class 1	Recall Class 0	Recall Class 1	F1 score Class 0	F1 score Class 1	Avg AUC	MCC
1	ANN	0.6198	0.6658	0.3273	0.8629	0.1335	0.7516	0.1896	0.5084	-0.005
2	FNN + Embedings	0.7464	0.769	0.6715	0.8856	0.4679	0.8232	0.5515	0.7006	0.3946
3	TAB Net	0.8277	0.8026	0.9394	0.9834	0.5162	0.8838	0.6663	0.7194	0.6088
4	XG Boost	0.6198	0.5548	0.6461	0.3879	0.7821	0.4566	0.7076	0.6058	0.1849
5	Random Forest	0.6273	0.5921	0.6368	0.3051	0.8529	0.4027	0.7292	0.6108	0.1902

I have experimented on a variety of models whose evaluation data has been displayed above. Two ML models— XGBoost and Random Forest, followed by three deep learning models— Artificial Neural Network, Feed Forward Neural Networks with embeddings and TabNet. Based on overall performance in most metrics, TabNet from Deep learning and Random Forest from ML have been chosen. Despite FNN with embeddings showing promising results only one deep learning model and machine learning model was chosen.

All the models and training can be found in exploring.ipynb.

Final Model Architecture

Random Forest

- A RandomForestClassifier was used as one of the baseline models for failure event prediction.
- The model was initialized with:
    - class_weight='balanced' to handle class imbalance by assigning higher penalty to the majority class.
    - random_state=42 for reproducibility.
    - n_jobs=-1 to enable parallel computation and speed up training.

Hyperparameter Tuning:

- **GridSearchCV** was used to perform hyperparameter tuning on key pruning-related parameters to control the complexity of the trees.
- The following parameters were included in the search:
    - n_estimators: Fixed at 100 trees.
    - max_depth: Limited the maximum depth of each tree (values: 3, 5, 7, None).
    - min_samples_split: Minimum samples required to split a node (values: 2, 5, 10).
    - min_samples_leaf: Minimum samples required to be at a leaf node (values: 1, 2, 4).
    - max_features: Number of features considered for splitting a node (sqrt, log2).
- **3-fold cross-validation** was used during the grid search.
- The **ROC AUC score** was used as the evaluation metric for model selection.

- The input data was not scaled, as Random Forests are **scale-invariant** and do not require feature normalization.

TabNet

Tab net is a NN that is specifically designed for tabular data. It uses decision trees and attention mechanisms to process and model data effectively.

- Traditional deep learning models (like MLPs) struggle with tabular data. TabNet, however, was specifically designed for structured datasets — which have usually been better handled by models like XGBoost or Random Forests.
- TabNet processes data in steps, and at each step, it uses an attention mechanism to decide
  - Which features to focus on
  - How much each feature should influence the prediction
  - This is very different from tree-based models that use greedy splits or MLPs that treat all input features equally at all times.
- TabNet promotes sparsity in feature usage
  - Each decision step uses only a small subset of features.
  - This makes the model interpretable and efficient
  - This is controlled by the gamma parameter and the entmax activation — which encourages attention to only a few important inputs.
- Handles Categorical Variables Natively
  - Uses embeddings for categorical variables (like NLP models)
  - Learns better representations for categories during training.
- TabNet is trained end-to-end using gradient descent, which makes it

    - More flexible and scalable
    - Capable of benefiting from powerful optimization tools like learning rate schedulers, early stopping, etc.

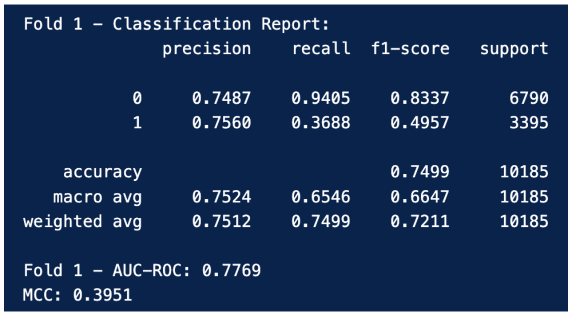
TabNet Model Configuration

- **n_d = 32:** Dimensionality of the decision step output. Controls the number of internal features used to represent data during the decision-making process, affecting the model's capacity for learning complex patterns.
- **n_a = 32**: Dimensionality of the attention step, determines how attention is distributed across features.
- **n_steps = 5:** Number of sequential decision steps. More steps = deeper feature reasoning.
- **gamma = 1.5:** Controls attention sparsity. Higher values lead to sparser (more selective) feature usage, improving interpretability and regularization.
- **cat_idxs, cat_dims, cat_emb_dim:** Used to embed categorical features properly. cat_idxs defines which features are categorical, cat_dims tells how many unique values each has, and cat_emb_dim sets the embedding size.
- **Optimizer_fn = Adam, with lr = 0.01:** Efficient optimizer with adaptive learning rate.
- **scheduler_fn = StepLR, with step_size = 10 and gamma = 0.9:** Gradually reduces the learning rate every 10 epochs to stabilize training.
- **mask_type = 'entmax':** Creates sparse feature selection masks, focusing only on the most relevant features — key for model interpretability

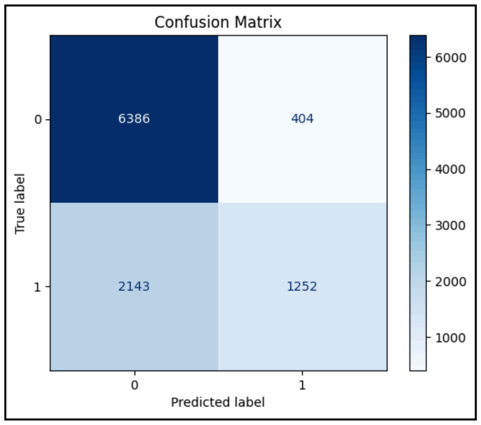
Evaluations

Failure Event Prediction

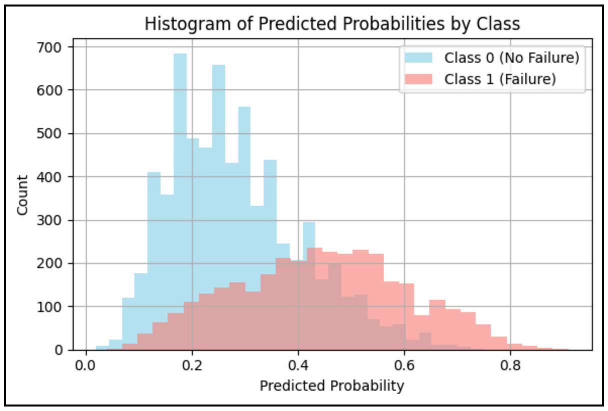
Random Forest

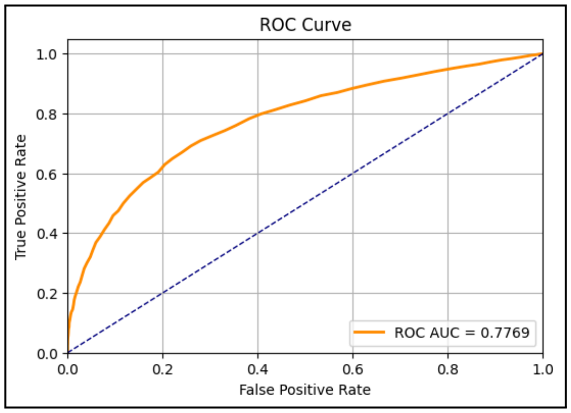


```
Fold 1 - Classification Report:
              precision    recall  f1-score   support

           0     0.7487    0.9405    0.8337      6790
           1     0.7560    0.3688    0.4957      3395

    accuracy                         0.7499     10185
   macro avg     0.7524    0.6546    0.6647     10185
weighted avg     0.7512    0.7499    0.7211     10185

Fold 1 - AUC-ROC: 0.7769
MCC: 0.3951
```



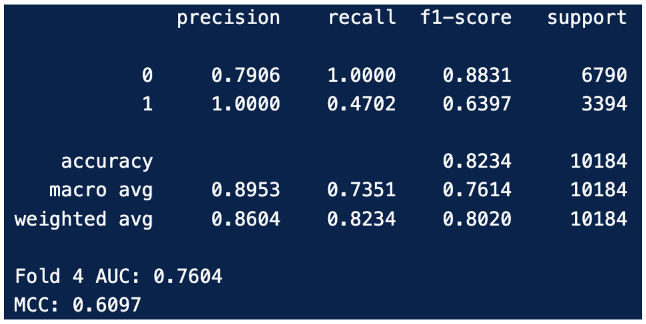
The model is highly effective at identifying non-failure cases (Class 0), with strong precision and recall. However, its ability to detect failures (Class 1) is limited, particularly in recall (only ~37%), meaning it misses a significant number of failure events. The AUC-ROC of 0.7769 suggests the model has a good ability to discriminate between the classes, but the low recall for failures suggests room for improvement in capturing true positives.



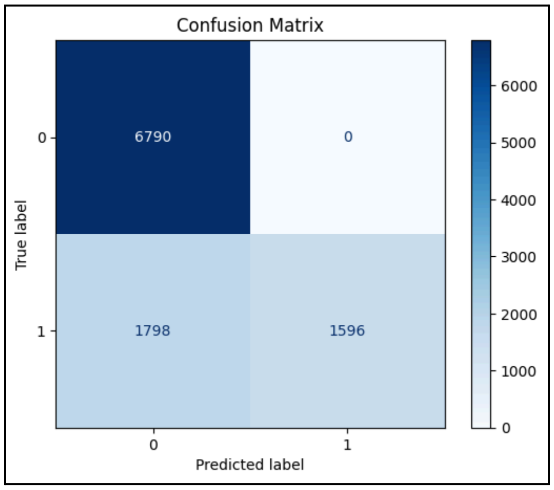


There is some overlap between the two distributions, which may contribute to false positives and false negatives. This suggests that although the model distinguishes between classes reasonably well, there is room for improving separation. An AUC of 0.7769 demonstrates that the model is effective in ranking positive (failure) examples higher than negative (no failure) ones.

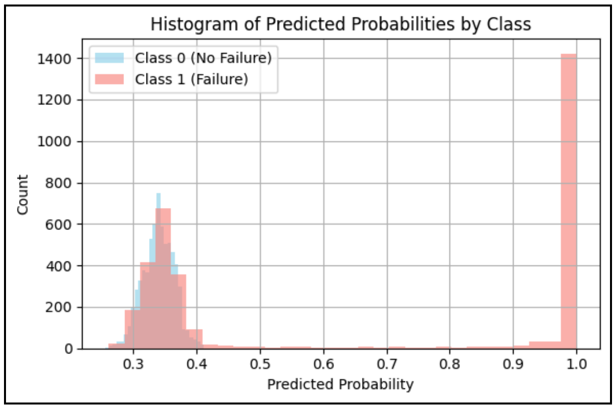
TabNet

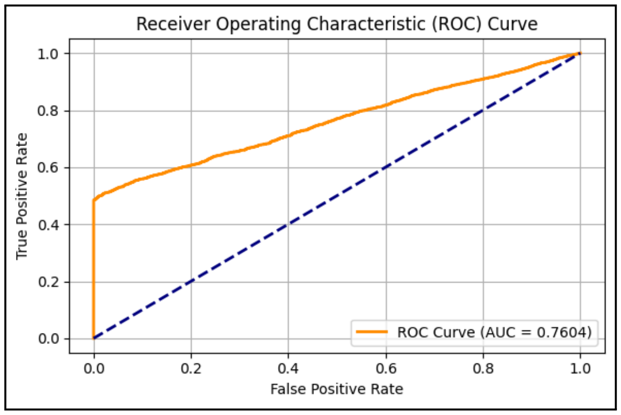


```
              precision    recall  f1-score   support

           0     0.7906    1.0000    0.8831      6790
           1     1.0000    0.4702    0.6397      3394

    accuracy                         0.8234     10184
   macro avg     0.8953    0.7351    0.7614     10184
weighted avg     0.8604    0.8234    0.8020     10184

Fold 4 AUC: 0.7604
MCC: 0.6097
```



The TabNet model demonstrates strong overall performance with an accuracy of 82.34% and an AUC of 0.76, indicating good discriminatory power. It shows perfect precision (100%) for predicting failures (Class 1), meaning it makes no false positives for that class. However, the recall for Class 1 is relatively low at 47%, suggesting that the model is missing a significant number of actual failure cases. In contrast, Class 0 (No Failure) is predicted with high reliability, showing perfect recall and decent precision (79%). The Matthews Correlation Coefficient (MCC) of 0.6097 further supports the model's balanced performance across classes. Overall, while the model is highly confident when predicting failures, its conservative approach leads to many missed failure instances, which could be critical depending on the application. However, it has fairley performed well as compared to Random Forest.

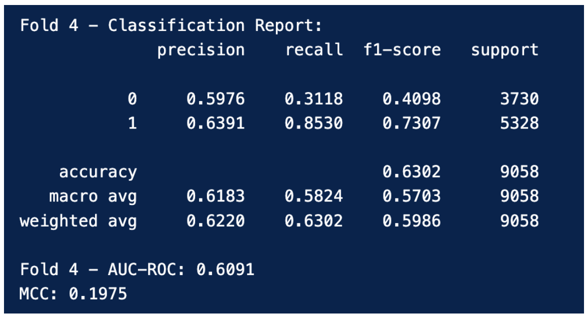




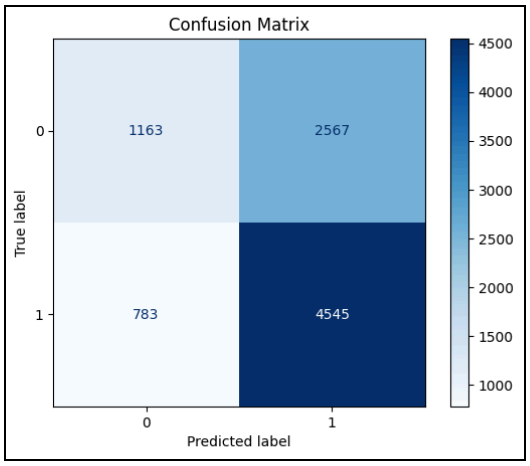
The visualizations for the TabNet model indicate that it demonstrates strong discriminative capabilities, especially in identifying failure cases (Class 1). The histogram of predicted probabilities reveals a clear peak near 1.0 for Class 1, suggesting that the model is highly confident in predicting many of the actual failures. In contrast, the probabilities for Class 0 (No Failure) are tightly clustered between 0.3 and 0.4, implying lower confidence in distinguishing negatives, which may be due to overlapping feature distributions due to class imbalance. The ROC curve further supports this observation, with an AUC of 0.7604, highlighting TabNet's ability to effectively separate the two classes. Its curve rises well above the diagonal, especially at lower false positive rates, indicating that the model is particularly good at correctly identifying true positives while minimizing false alarms in that range. Overall, TabNet shows a promising balance of precision and recall, particularly favoring confident detection of failures.

Maintenance Needs

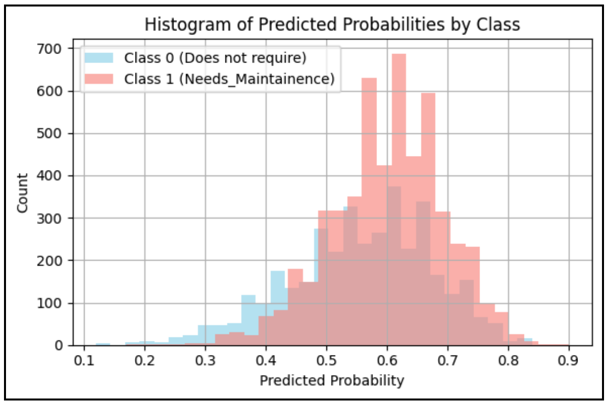
Random Forest

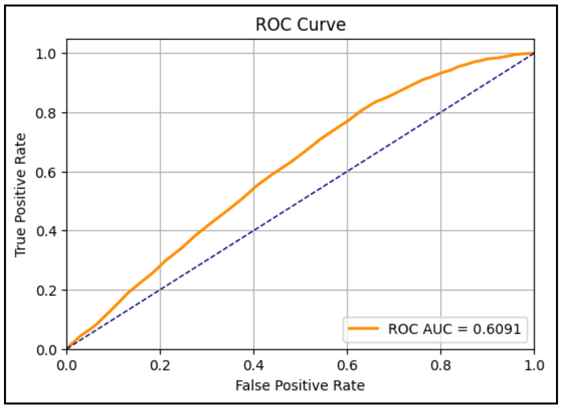


```
Fold 4 – Classification Report:
              precision    recall  f1-score   support

           0     0.5976    0.3118    0.4098      3730
           1     0.6391    0.8530    0.7307      5328

    accuracy                         0.6302      9058
   macro avg     0.6183    0.5824    0.5703      9058
weighted avg     0.6220    0.6302    0.5986      9058

Fold 4 – AUC-ROC: 0.6091
MCC: 0.1975
```



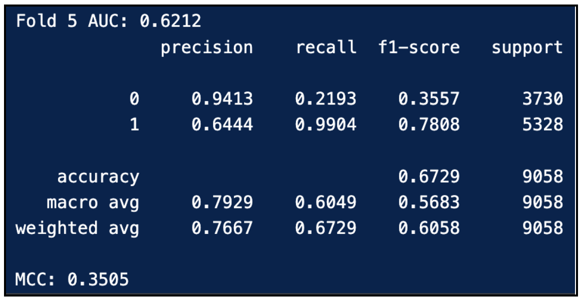
The Random Forest model’s performance in predicting the need for maintenance shows a significant class imbalance in predictive power. While it achieves a high recall of 0.8530 for the positive class (maintenance needed), its precision is relatively low at 0.6391, indicating many false positives. For the negative class (no maintenance needed), the recall drops to just 0.3118, with a precision of 0.5976, suggesting that the model struggles to correctly identify and classify non-maintenance cases. The overall accuracy stands at 63.02%, with an AUC-ROC of 0.6091 and MCC of 0.1975, which reflect moderate discrimination and limited overall reliability. The confusion matrix reinforces this observation: the model correctly predicts most maintenance-needed cases (4545 true positives) but misclassifies a large number of no-maintenance cases (2567 false positives). This imbalance suggests the model prioritizes catching potential failures at the expense of specificity, which might be acceptable depending on the operational cost of false positives in a maintenance context.



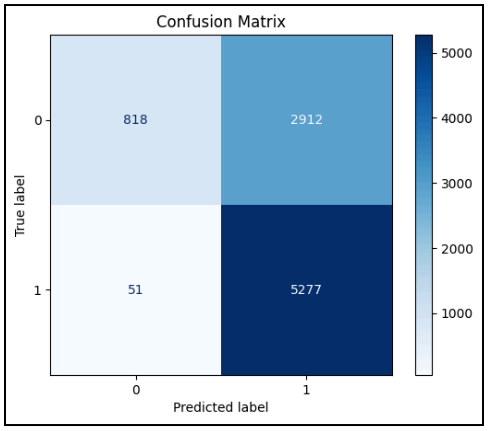


The predicted probability histogram and ROC curve further reinforce the limited discriminatory ability of the Random Forest model for predicting maintenance needs. The histogram shows a significant overlap between the predicted probabilities of both classes, indicating that the model often assigns similar probability values to both "Needs Maintenance" and "Does Not Require" classes. Ideally, we would expect more separation between the two classes. The ROC curve has an AUC of 0.6091, which is only slightly better than random guessing (0.5), suggesting the model struggles to differentiate between the two classes effectively. This poor separation and moderate AUC align with earlier classification metrics and highlight the need for improvement.

TabNet



```
Fold 5 AUC: 0.6212
              precision    recall  f1-score   support

           0     0.9413    0.2193    0.3557      3730
           1     0.6444    0.9904    0.7808      5328

    accuracy                         0.6729      9058
   macro avg     0.7929    0.6049    0.5683      9058
weighted avg     0.7667    0.6729    0.6058      9058

MCC: 0.3505
```

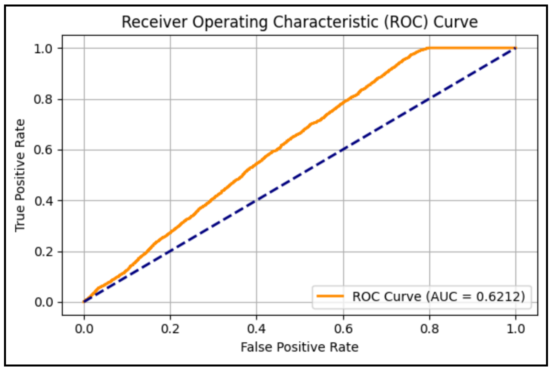


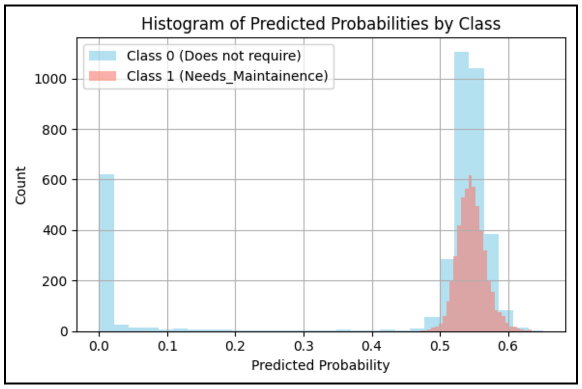
The TabNet model shows a relatively better performance in predicting maintenance needs compared to the Random Forest model. While the AUC-ROC score of 0.6212 still indicates only moderate discriminative power, it is an improvement over the Random Forest's 0.6091. The model exhibits very high recall for the "Needs Maintenance" class (Class 1) at 0.9904, meaning it successfully identifies nearly all actual positive cases. However, this comes at the cost of low recall for Class 0 (0.2193), indicating a high number of false positives.

The confusion matrix supports this—out of 3730 actual Class 0 instances, 2912 are misclassified as needing maintenance. On the other hand, only 51 out of 5328 Class 1 samples are misclassified. This behavior makes TabNet particularly suited for scenarios where missing a maintenance need is more critical than a false alarm.

Precision is also fairly strong for both classes, with Class 1 having 0.6444 and Class 0 reaching 0.9413. The overall accuracy stands at 67.29%, and the Matthews Correlation Coefficient (MCC) of 0.3505 indicates moderate predictive power with some class imbalance handling.

In summary, TabNet prioritizes capturing maintenance needs with very high recall at the expense of misclassifying many non-maintenance cases. It may be a good fit for preventive systems where it's more acceptable to over-predict than to miss actual failures.





TabNet shows a strong bias towards predicting that maintenance is needed, achieving a high recall of 0.99 for the "Needs Maintenance" class but performing poorly on the "Does Not

Require Maintenance" class, with a recall of just 0.22. This leads to a high number of false positives, as reflected in the confusion matrix and the predicted probability histogram, where many Class 0 instances are incorrectly assigned high probabilities. While the overall accuracy is 0.67 and the AUC-ROC score is 0.62, indicating moderate discrimination ability, the model sacrifices precision and balance in favor of minimizing false negatives for Class 1. This behavior could be beneficial in preventive maintenance scenarios where missing a needed repair is riskier than flagging a false one.

Decisions & Business Recommendations

The deployment of two specialized TabNet models—one for predicting failure events and the other for anticipating maintenance needs—offers a powerful dual-layered approach to asset management. The failure prediction model, chosen over Random Forest for its significantly higher recall and MCC, ensures critical failures are rarely missed, allowing for immediate corrective action. Meanwhile, the maintenance prediction model supports proactive scheduling, helping reduce downtime and optimize resource allocation. Together, these models enable a smart system: assets flagged by both models can be prioritized for urgent attention, while those needing only maintenance can be handled through routine servicing. This strategy not only enhances operational reliability but also minimizes unnecessary repairs and improves overall asset longevity. Regular retraining and incorporating model explainability can further strengthen model accuracy and adoption across technical teams.

The TabNet models can be effectively incorporated into:

- Failure Prediction Model: Prioritize urgent interventions by identifying assets at high risk of failure with high recall and precision, minimizing critical breakdowns.

- Maintenance Prediction Model: Schedule timely, cost-effective maintenance before failures occur, optimizing resource allocation and extending asset life.
- Combined Strategy: Assets flagged by both models should be escalated for immediate inspection, while others can be placed in routine servicing, creating a structured, tiered response system.
- Operational Efficiency: Use model outputs to guide technician deployment, spare part logistics, and preventive workflows, cutting costs from unnecessary checks and reducing downtime.