COMP SCI 7015

Software Engineering & Project

Sprint Retrospective 4

Rail Break Prediction AI

a1882259

Nilangi Maheesha Sithumini Edirisinghe

Group IF_PG1

List of Group Members:

a1873818 Zewei You, a1898921 Yong Kheng Beh, a1882259 Nilangi Edirisinghe, a1879038 Bhagya Indeewari Senanayake Senanayake Mudiyanselage, a1878048 Wenjing Shen, a1879321 Xinyue Ma, a1877644 Jiemin Zhanga, 1878387 Qiaoqiao Chen

1. Snapshots (Group)

1.1 Product Backlog and Task Board

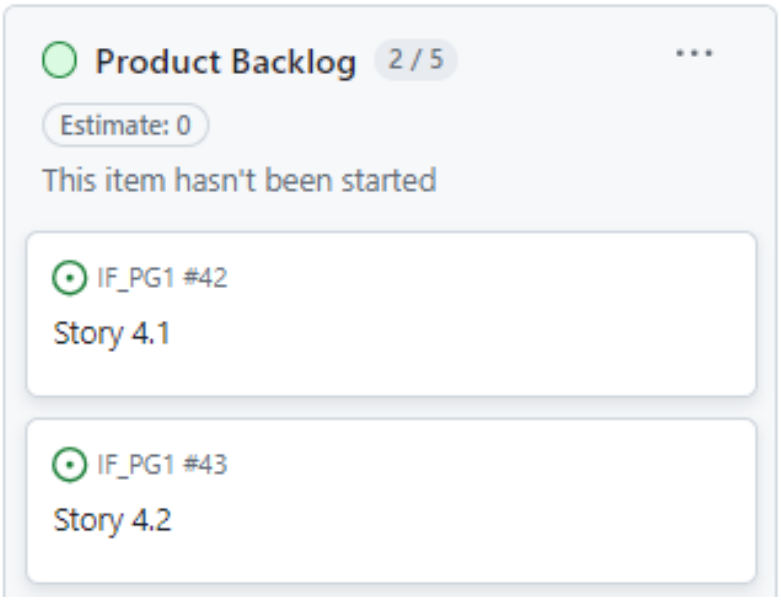

Figure 1: product backlog

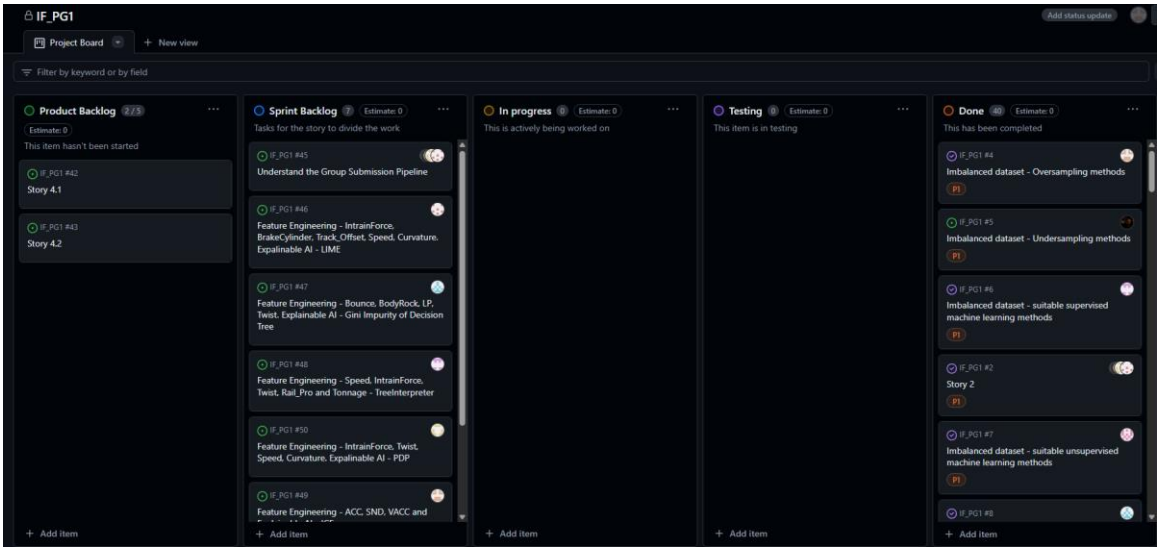

Figure 2: Task Board in First Week of Sprint 4 (From Snapshot 4.1)

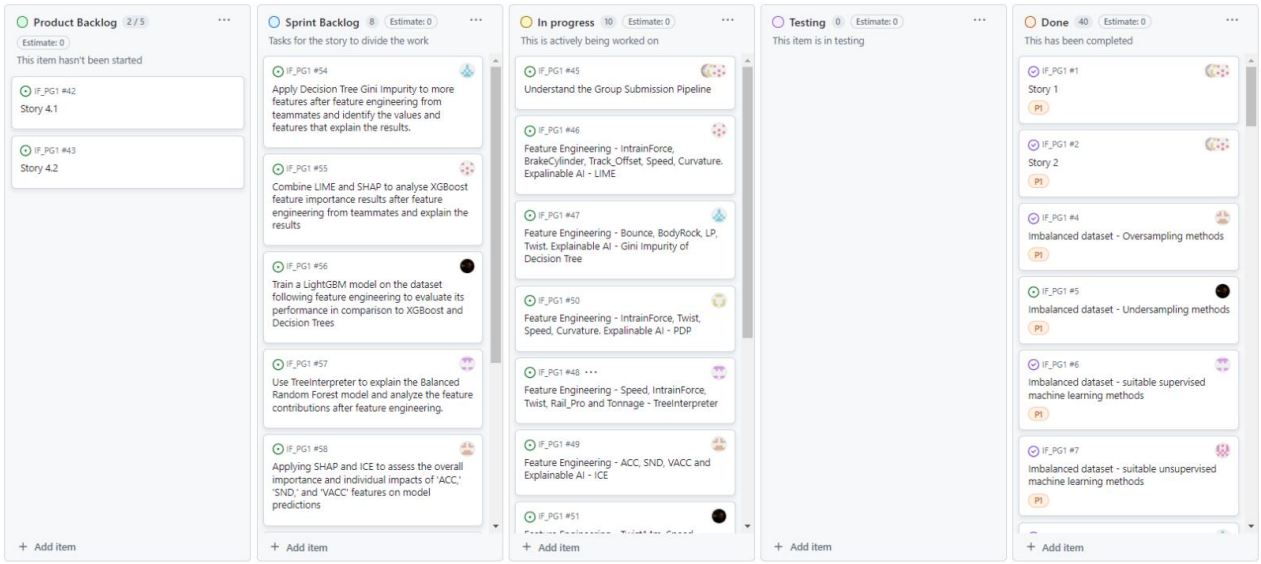

Figure 3: The Screenshot of the Task Board in First Week of Sprint 4 (From Snapshot 4.1)

1.2 Sprint Backlog and User Stories

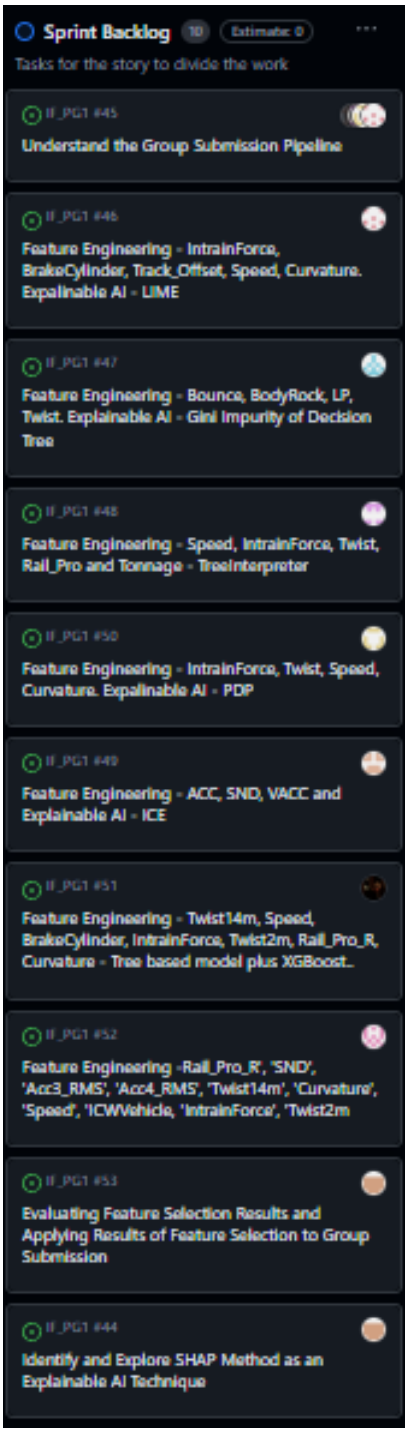

Figure 4: The Screenshot of the Sprint Backlog for First Week of Sprint 4 (From snapshot 4.1)

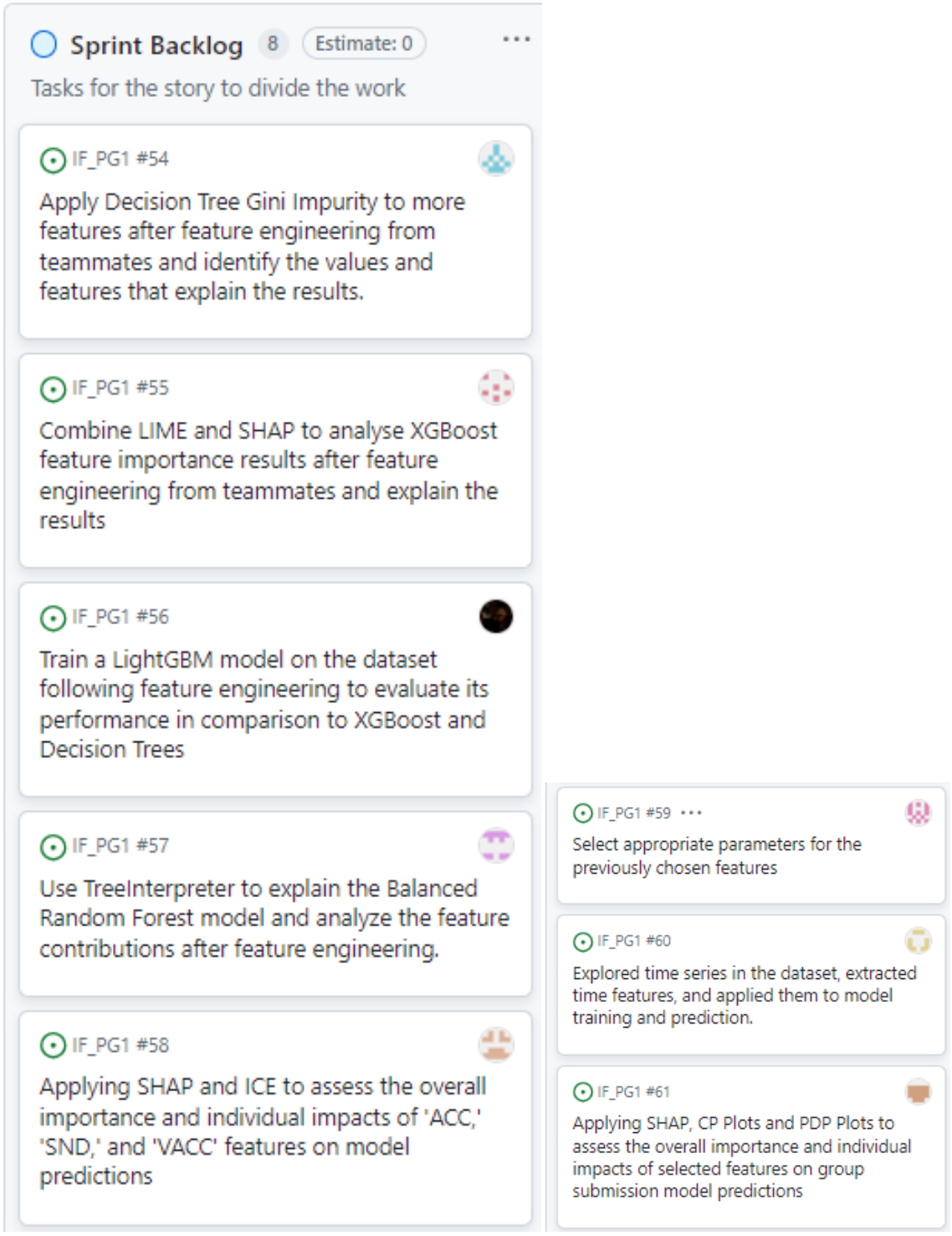

Figure 5: The Screenshot of the Sprint Backlog for Second Week of Sprint 4

1.3 User Story 4

As a software engineer, I want to try out:

1. **Different models while fine-tuning their hyperparameters**
   To improve the test set predictions and explainability, models were selected and optimized to exceed the target F1 score.

2. **Additional techniques for feature engineering, feature selection, and methods for addressing imbalanced datasets**
   These methods help to improve the model's performance and its interpretability. Feature importance and data distribution were assessed to ensure the model's predictions were reliable and not based on spurious correlations.

Acceptance Criteria:

1. **Fine-tune hyperparameters of at least 3 different models**
   Models:
   - Model 1: Random Forest
   - Model 2: Gradient Boosting
   - Model 3: XGBoost

Hyperparameter tuning:
Grid search and randomized search were employed to optimize parameters such as the number of estimators, learning rate, and maximum depth for these models. This improved the model's generalization ability while maintaining the interpretability.

2. **Feature Selection, Feature Engineering, and Imbalanced Dataset Handling Techniques**
   - **Feature Engineering**:
     Experimented with two additional techniques:
     1. Polynomial Features to explore higher-order interactions between variables.
     2. Normalization techniques (MinMaxScaler, StandardScaler) to address variability in feature ranges.
   - **Feature Selection**:
     Applied two additional techniques:
     1. Recursive Feature Elimination (RFE) to systematically remove less important features.
     2. Tree-based feature importance (from Random Forest and XGBoost) to retain only the most impactful features.
   - **Imbalanced Dataset Handling**:
     Implemented two more techniques to balance the dataset:

1. SMOTE (Synthetic Minority Over-sampling Technique) to oversample the minority class.
2. Class weight adjustment to penalize the majority class more heavily during training.

3. **Report on combinations of techniques and models that worked so far**
   The most effective combination was:
   - Model: **XGBoost**
   - Feature Engineering: Normalization combined with Polynomial Features
   - Imbalanced Data Handling: SMOTE worked best with Random Forest and XGBoost

This combination achieved a leaderboard score of **F1: 65%**, meeting the project's requirements.

Insights from Explainability (XAI):

1. **Training Explainability**:
   The SHAP (SHapley Additive exPlanations) technique was used to analyze the impact of each feature on the prediction of rail breaks. Features such as temperature and rail age showed the highest influence, especially within specific value ranges.

2. **Inference Explainability**:
   For inference, LIME (Local Interpretable Model-agnostic Explanations) was applied to explain individual predictions, highlighting which features (e.g., humidity and rail tension) drove the model's decision to predict a rail break in certain instances.

1.4 Definition of Done (DOD)

1. Code Standards

- **Readability and Maintainability**:
  - Code must be written with clarity to ensure it is understandable and easily modifiable.
  - All functions and classes must have clear, concise docstrings explaining their purpose, parameters, and return values to enhance maintainability and readability.
- **Testing**:
  - The entire pipeline, from data ingestion to model predictions, must run successfully without errors.
  - The prediction results must be successfully reflected on the leaderboard to signify the accuracy and performance of the submission.

2. Documentation

- **Reproducibility**:
    - All scripts and notebooks must be structured for easy re-runs by team members, ensuring the pipeline is easily understandable and executable.
- **Data Pipeline Documentation**:
    - Document the entire data pipeline, including data cleaning, feature engineering, feature selection, model decisions, and hyperparameter tuning.
    - Ensure that all assumptions, processes, and justifications for decisions are recorded for future reference, providing transparency and traceability.
- **Leaderboard Score Tracking**:
    - Record and track leaderboard scores during experimentation to guide improvements in the model.
    - Include dates and versions of models associated with specific scores to identify which iterations are the most effective.
- **Group Product Line Documentation**:
    - Every team member must contribute to the group product line, documenting any changes they make to the pipeline.
    - All changes should be backed by evidence from individual experiments, and documentation must explain how these changes improve model performance or stability.

3. Data Management

- **Data Cleaning**:
    - Define clear, reproducible steps for data cleaning (e.g., handling missing values, outliers), and log all changes for traceability.
    - Validate the dataset after cleaning to ensure no unexpected data loss or introduction of errors.
- **Feature Engineering and Selection**:
    - Document all feature engineering and selection processes, providing justification for decisions.
    - Regularly evaluate and log feature importance to track how features impact model performance.
- **Balanced Data Handling**:
    - Document the techniques used for rebalancing the dataset, such as SMOTE, undersampling, or ADASYN, and explain the rationale for selecting each method.

- Measure and log the performance of models before and after rebalancing to ensure the effectiveness of the balancing techniques.

4. Machine Learning Model

- **Model Selection and Validation**:
  - All models selected for experimentation must be validated using appropriate techniques, such as cross-validation or out-of-sample testing.
  - Performance metrics (e.g., F1 score, precision, recall) must be calculated for each model and compared against baseline models (e.g., logistic regression) to assess improvements.
- **Hyperparameter Tuning**:
  - The hyperparameter tuning process must be documented, including the parameters tuned, the range of values tested, and the final selected values that yielded the best results.
- **Model Interpretability**:
  - Each model must include interpretability steps (e.g., SHAP values, feature importance plots) to provide insights into how predictions are made.
  - Interpretability results must be documented and shared in a central location (e.g., Debug Zone) so that all team members can review and provide feedback.

5. Review and Sign-Off

- **Peer Review**:
  - All code, models, and documentation must undergo a peer review process before being marked as complete. Reviewers must verify that all DoD criteria have been met, ensuring code quality and correctness.
- **Sign-Off**:
  - The team must collectively sign off on each completed user story, confirming that all acceptance criteria have been satisfied and all necessary documentation is provided. This includes validation from both technical and business perspectives to ensure the solution is production-ready.

By adhering to the above standards, the team ensures that the project's development pipeline is robust, maintainable, and transparent, with clear documentation and validation of each process and model decision.

1.5 Summary of Changes

- **For Week 1 of Sprint 4 (Snapshot 4.1)**

During snapshot 4.1, we made two critical changes aimed at improving collaboration and enhancing model transparency. First, we shifted focus from individual product lines to a unified group product line. Previously, working on separate product lines led to fragmented progress, making it difficult to consolidate the team's efforts. By requiring all members to contribute to a single model with evidence-backed changes and proper documentation, we improved consistency and collective progress. Second, we made Explainable AI (XAI) mandatory, ensuring that each model includes interpretability techniques like SHAP values. This addition improved the transparency of our models, helping stakeholders understand the factors driving predictions. These changes resulted in better collaboration and more interpretable models.

- **For Week 2 of Sprint 4 (Snapshot 4.2)**

No major changes were made during Sprint 4.2. Our team continued to focus on Explainable AI (XAI) while exploring ways to enhance machine learning predictions with the goal of improving the F1 score. We leveraged the newfound knowledge from XAI techniques to refine our models further. Additionally, we introduced a shared Excel sheet where team members can document their model parameters and leaderboard scores. This shared resource fosters better collaboration and transparency, helping the team track progress and make data-driven decisions more effectively.

I attended the sprint planning meetings with my group on Monday, September 30th, to kick off the initial planning for Sprint 4. After completing the first week of the sprint, we had a follow-up discussion to assess progress and identify improvements for the next week. I attended our final sprint review meeting on Monday, October 7th where the group presented our current status to the product owner and got feedback.

2. What went well in the sprint?

During this sprint, several key improvements were successfully implemented, enhancing both collaboration and model transparency. A significant highlight was the shift from working on individual product lines to focusing on a unified group product line. This change streamlined our efforts, allowing the entire team to collaborate on a single model, with all modifications backed by evidence and thoroughly documented. This approach not only eliminated the fragmented progress experienced in previous sprints but also ensured that everyone contributed to the collective goal. This shift improved overall productivity and allowed us to make steady progress on a consolidated front.

Additionally, the introduction of Explainable AI (XAI) as a mandatory component for every model significantly enhanced transparency. By integrating interpretability techniques like SHAP values, we were able to understand and communicate the factors influencing the model's predictions. This provided crucial insights into model performance and helped identify areas for further improvement. Furthermore, the use of a shared Excel sheet for documenting model parameters and leaderboard scores improved collaboration by allowing everyone to track progress, compare results, and refine models collectively. The sprint's success was driven by stronger teamwork, clearer goals, and a more structured approach to model development and evaluation.

3. What Could be Improved?

During this sprint, one area that could be significantly improved is the team’s consideration and integration of individual analyses into the broader group strategy. For example, I shared my findings early on, showing that the feature selection analysis I conducted was correct, with **tonnage** identified as a key feature. Despite presenting this information, the group did not act on it until it was highlighted by our tutor during the sprint review meeting. This delay in acknowledging critical insights reduced our ability to optimize the model effectively from the start.

Throughout the sprint, the group tried various combinations of features, feature engineering techniques, and model explainability methods. However, achieving a leaderboard score of 65% was not possible. The team's lack of coordinated integration and failure to consider everyone's contributions fragmented progress and impacted on our ability to deliver the best possible solution. This approach hindered the development process by creating unnecessary bottlenecks, which ultimately delayed the implementation of the most effective methods. If the group had considered all insights and acted on them earlier, it would have led to a more efficient and successful software development process, improving both the model’s performance and overall collaboration.

4. What will the Group commit to Improve in the Next Sprint?

In the next sprint, the group will commit to improving the integration and collaboration of individual contributions. Specifically, we will establish a structured process for sharing findings early in the sprint, ensuring that all insights are discussed and evaluated during our regular meetings. This will involve setting up a weekly review of each member's analysis and results, where feature selection, engineering techniques, and model explainability approaches are critically assessed and combined into the group submission.

Additionally, we will implement a more organized approach to model experimentation. Rather than working in isolation, each member will contribute their results to a unified pipeline, ensuring that all tested techniques are incorporated in a way that optimizes the model's performance. Peer reviews of each other's work will be introduced to ensure that critical insights are not overlooked, and everyone's contributions are integrated before final model submissions.

This approach will streamline the software development process by preventing fragmented progress, improving collaboration, and ensuring that we make data-driven decisions based on collective findings, ultimately leading to more efficient and accurate model development.

5. Comment on My Progress This Sprint

During Sprint 4, my primary focus was to evaluate and verify the results of my feature selection from Sprint 3 and further refine our submission pipeline to improve the leaderboard score. Here's a summary of my contributions:

Evaluating Feature Selection Validity: I investigated whether the feature selection results from Sprint 3 were valid, as the group leader raised concerns about potential issues with the SQL query and data split. To address this, I modified the submission pipeline, applying different SQL queries to both the training and testing data and adjusting the data split according to the recording date. My analysis showed that neither the SQL queries nor the data split affected the leaderboard score, confirming that my feature selection results were correct. When I applied the selected feature set from Sprint 3, the leaderboard score improved, validating the accuracy of my analysis.

Dataset Imbalance and Resampling: While analyzing the F1 score variations, I detected that the resampling technique I had used was ineffective in addressing the imbalance in the dataset. This issue contributed to fluctuations in the overall F1 score and, more importantly, the positive class F1 score (rail break predictions), which is the metric used for the leaderboard score. Identifying this as a key area for improvement, I will focus on refining the resampling technique in the future.

Improving the Group Submission Pipeline: After reviewing the group's submission, I noticed certain important features, like tonnage, were missing. I created a new SQL query to include these features, then built a new feature engineering pipeline by aggregating the mean of the best feature set for the XGBoost model. However, this led to errors in the testing set due to mismatches with the previous pipeline. I resolved this by creating a new submission pipeline, leveraging the group's existing submission structure. This led to a leaderboard score of 0.35 for the newly built pipeline.

Explainable AI (XAI) Techniques: I applied several explainable AI methods, including SHAP values, CP (Conditional Probability) plots, and PDP (Partial Dependence Plots), to both my KNN model and the XGBoost model. I experimented with different feature combinations, which provided valuable insights into the models' behavior and the importance of the features. These insights will help further optimize the model's performance moving forward.

This sprint provided significant learnings regarding the impact of data imbalance, resampling techniques, and the integration of missing features, as well as valuable experience with explainable AI methods.

The following figures show some of the PDP, CP and SHAP plots that were obtained by applying the explainable AI techniques to the XG Boost model that I created using the group's submission as well as my KNN model.

5.1. Partial Dependence Plots (PDP) for Feature Analysis

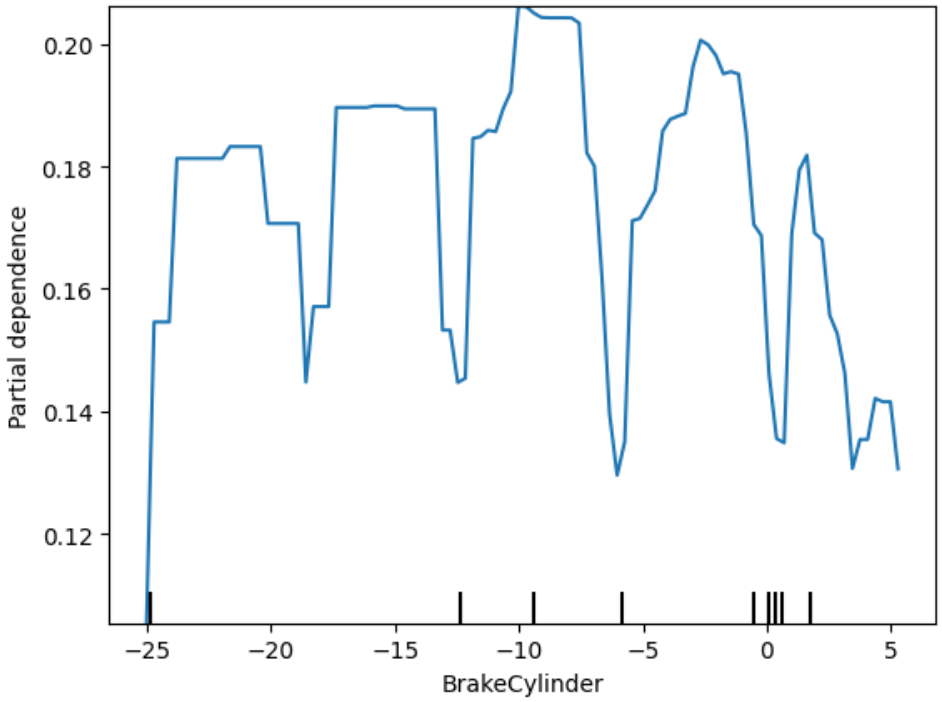

Figure 6: Partial Dependence Plot for BrakeCylinder

This plot shows the partial dependence of the feature 'BrakeCylinder' on the XG boost model's predicted probability. It indicates how variations in BrakeCylinder affect the model's output, with higher values generally leading to an increased prediction of rail breaks.

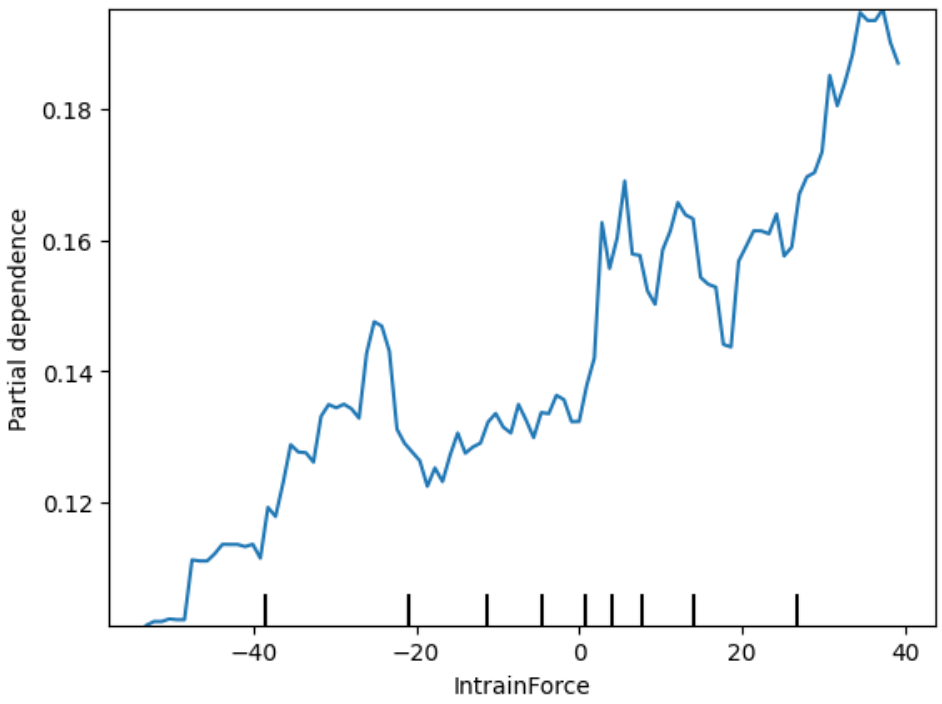

Figure 7: Partial Dependence Plot for IntrainForce

The partial dependence plot for 'IntrainForce' shows a positive trend, with higher values of this feature leading to higher predictions for rail breaks. This suggests that increased intrain force contributes significantly to the model's decision.

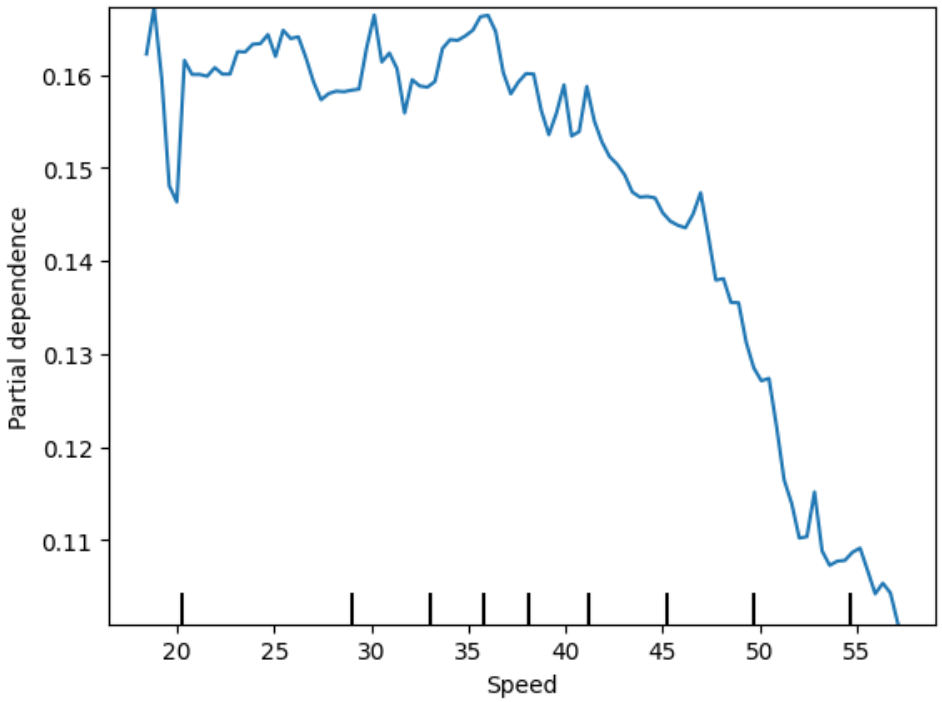

Figure 8: Partial Dependence Plot for Speed

This plot highlights how variations in 'Speed' impact the model's predictions. Initially, lower speed values have a high probability of rail breaks, but as speed increases beyond 45, the likelihood decreases, indicating that very high speeds are associated with a lower risk of rail breaks in the model.

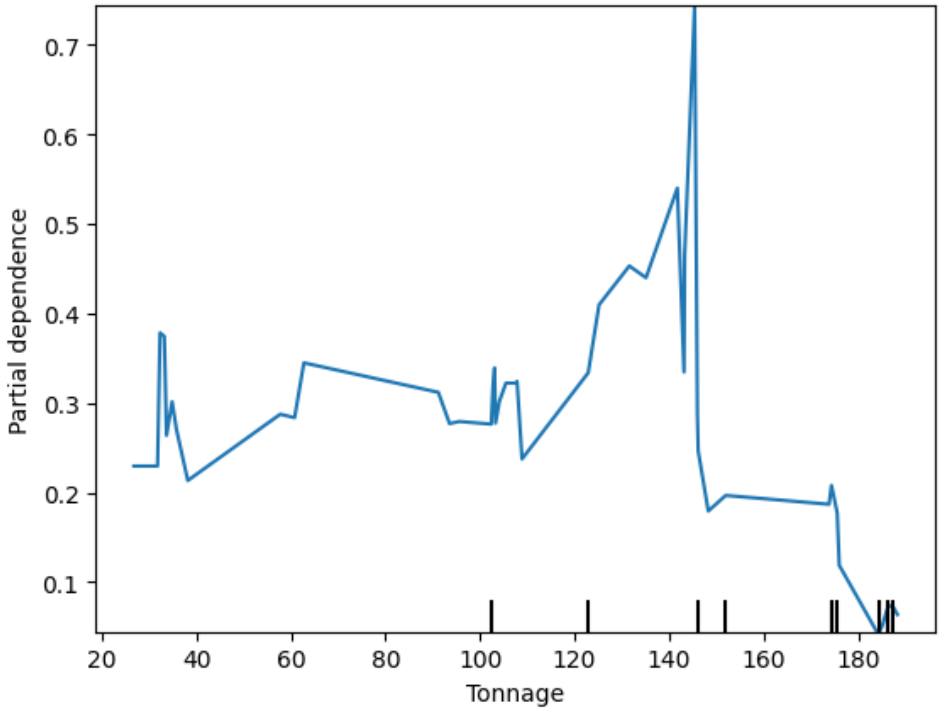

Figure 9: Partial Dependence Plot for Tonnage

The partial dependence plot for 'Tonnage' demonstrates that higher tonnage values tend to increase the model's predictions for rail breaks. The spikes in certain ranges suggest that this feature has non-linear interactions with the model's output.

The following code was used to generate the Partial Dependence Plots for various features in the dataset. It iterates over a list of selected features and plots the partial dependence of each one using the 'plot_partial_dependence' function from sklearn.

```
from sklearn.inspection import plot_partial_dependence

features = ['BrakeCylinder', 'Acc3', 'Tonnage', 'Twist14m', 'BodyRockFrt', 'Speed',
       'Rail_Pro_L', 'BounceFrt', 'Rail_Pro_R', 'Acc1', 'IntrainForce']

for feature in features:

    features = [feature]

    plot_partial_dependence(model, X_train, features)
    plt.show()
```

Figure 10: Code for Generating Partial Dependence Plots

5.2. SHAP Summary Plots for Model Explainability

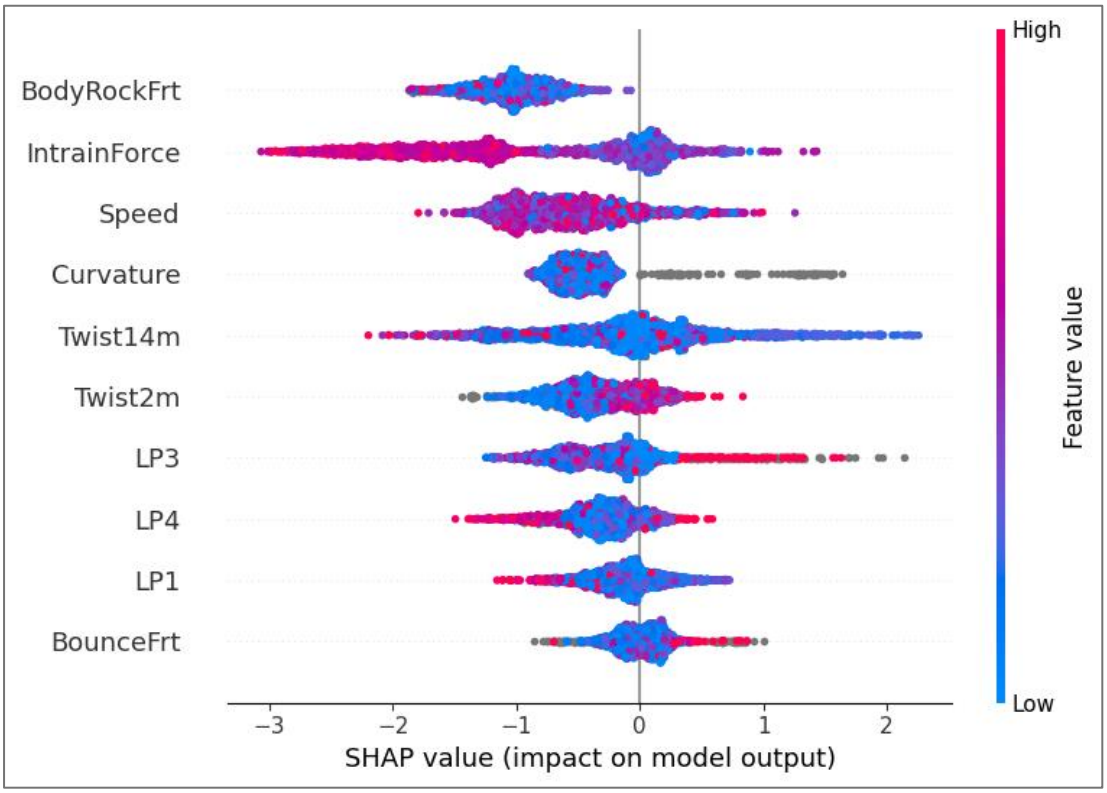

Figure 11: SHAP Summary Plot for Group Submission XGBoost Model

This plot shows the SHAP values for various features in the XGBoost model used in the group submission. The x-axis represents the SHAP value, which indicates the impact of the feature on the model's output. Positive SHAP values increase the likelihood of predicting a rail break. Key features like 'BodyRockFrt,' 'IntrainForce,' and 'Twist14m' are significant in determining the model's predictions.

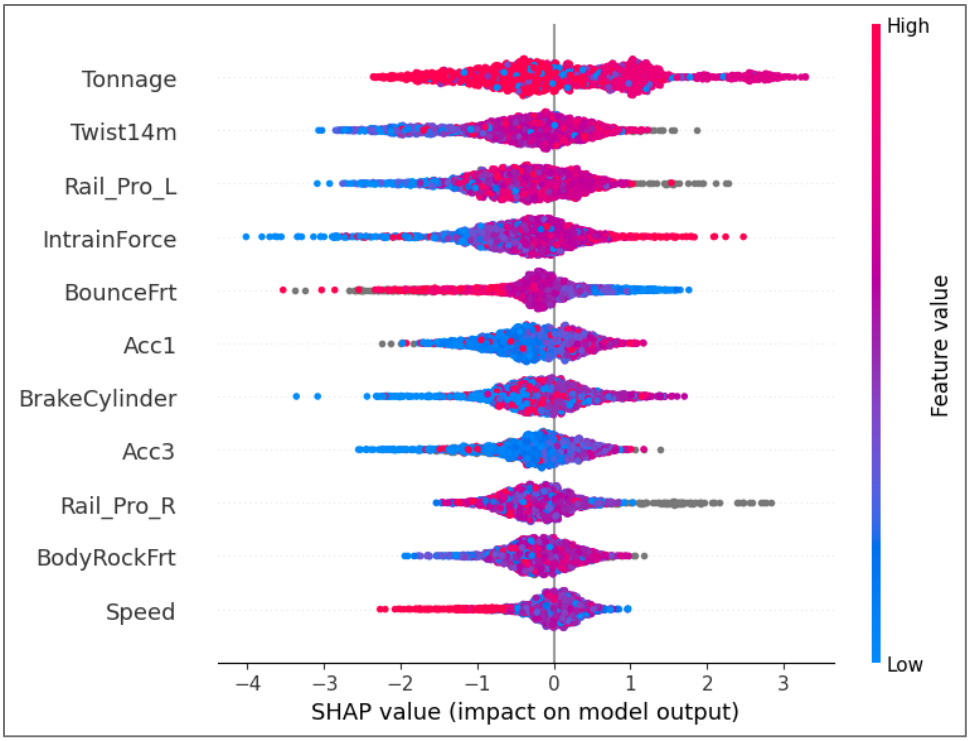

Figure 12: SHAP Summary Plot for XGBoost Model with Tonnage Included

This SHAP plot shows the impact of features, including 'Tonnage,' in an XGBoost model. 'Tonnage,' 'Twist14m,' and 'IntrainForce' emerge as critical features. The addition of 'Tonnage' seems to have a substantial effect, and high values of these features (shown in red) contribute significantly to higher SHAP values, indicating a higher likelihood of predicting a rail break.

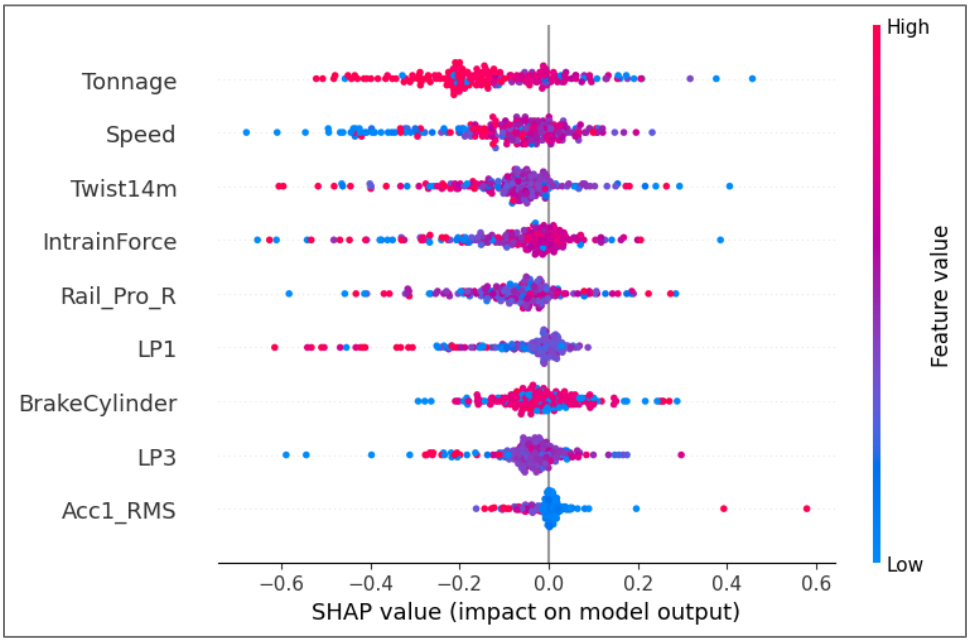

Figure 13: SHAP Summary Plot for KNN Model

The KNN model's SHAP plot shows the importance of features like 'Tonnage,' 'Speed,' and 'Twist14m.' These features have a smaller range of SHAP values compared to the XGBoost model, suggesting that the KNN model's predictions are less sensitive to these features. The distribution of feature values is more concentrated, indicating that the KNN model behaves differently from XGBoost in interpreting the data.

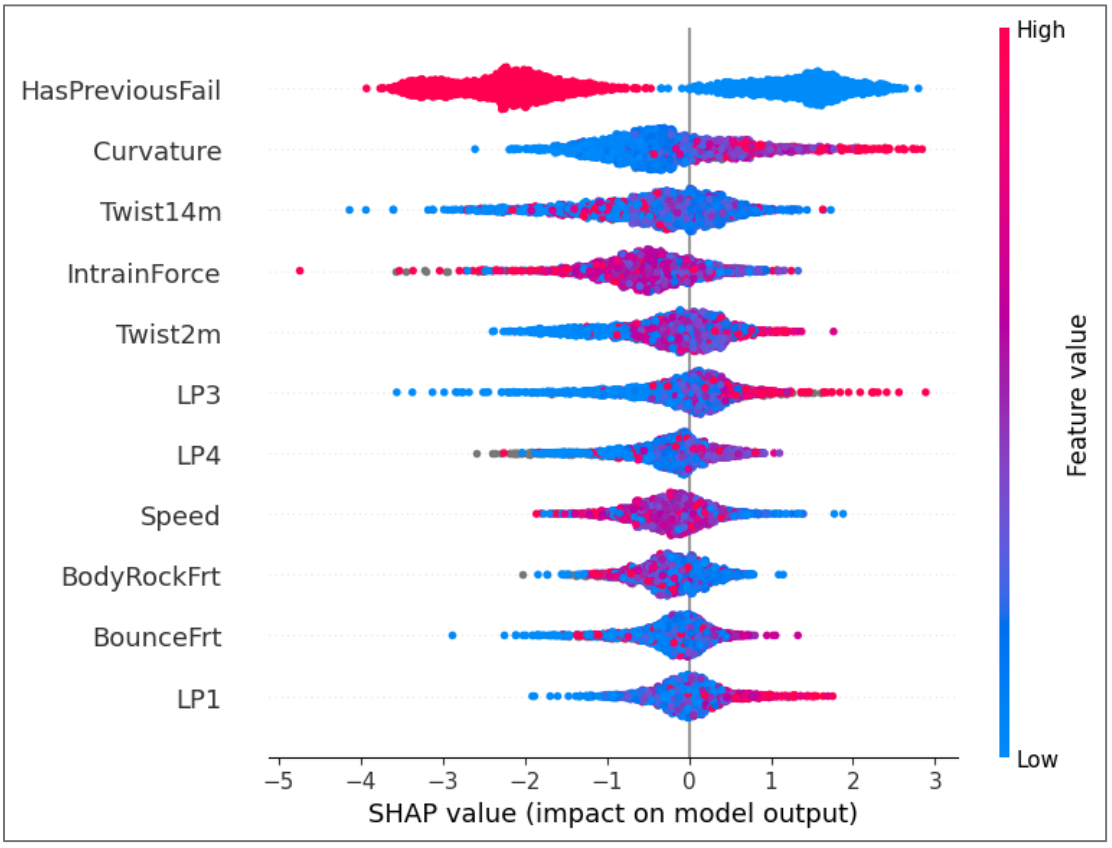

Figure 14: SHAP Summary Plot for Alternative XGBoost Model with Historical Features

This plot highlights the SHAP values for an XGBoost model that includes historical features such as 'HasPreviousFail,' 'Curvature,' and 'IntrainForce.' These features demonstrate a broader range of SHAP values, with the feature 'HasPreviousFail' having a significant positive impact on the prediction of rail breaks.

5.3. Ceteris Paribus Profiles (CP) for Feature Analysis

Figure 15: Ceteris Paribus Profile for Tonnage

This Ceteris Paribus plot shows how the model's prediction changes when 'Tonnage' is varied while keeping other features constant. The non-linear behavior in certain regions suggests that tonnage significantly affects the model's output in these ranges.

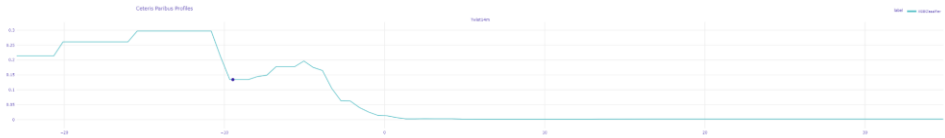

Figure 16: Ceteris Paribus Profile for Twist14m

This plot illustrates the impact of the 'Twist14m' feature on the model's predictions when it is varied. The model's predictions decline sharply after a certain threshold, suggesting that lower twist values lead to a higher probability of rail breaks.

Figure 17: Ceteris Paribus Profile for BodyRockFrt

This CP plot shows how 'BodyRockFrt' influences the model's predictions. As the value increases, so does the model's prediction for rail break likelihood, indicating that this feature has a strong positive correlation with the target.

Figure 18: Ceteris Paribus Profile for Speed

This plot highlights how the 'Speed' feature affects the model predictions. As speed increases, there is significant variation, suggesting that speed interacts non-linearly with other features to influence the predictions.

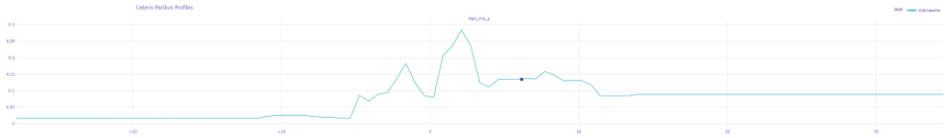

Figure 19: Ceteris Paribus Profile for Rail_Pro_L

This plot demonstrates how variations in the 'Rail_Pro_L' feature affect the model's prediction probability. It shows that certain ranges have a strong impact on the prediction, especially in the lower value regions.

Code for Generating Ceteris Paribus Plots:

The following code was used to generate the Ceteris Paribus (CP) Profiles for various features in the dataset. It iterates over a list of selected features and plots the CP profile using the 'Explainer' and 'predicted_profile' functions from dalex.

```
#CP Profile

from dalex import Explainer
import matplotlib.pyplot as plt

features = ['BrakeCylinder', 'Acc3', 'Tonnage', 'Twist14m', 'BodyRockFrt', 'Speed',
       'Rail_Pro_L', 'BounceFrt', 'Rail_Pro_R', 'Acc1', 'IntrainForce']


for feature in features:

    # Assuming `model` is your trained model, and X is your dataset, and y is your target
    explainer = Explainer(top_model, X_train, y_train)

    # Select an instance to explain (e.g., the first instance)
    instance = X_train.iloc[1]

    # Create the Ceteris Paribus plot for 'tonnage' feature
    cp_plot = explainer.predict_profile(instance, variables=[feature])

    # Plot the Ceteris Paribus plot
    cp_plot.plot()
    plt.show()
    print(instance)
```

Figure 20: Code snippet for Generating Ceteris Paribus Plots

5.4 GitHub Issues Covered:

I successfully completed the following issues assigned on GitHub during this sprint:

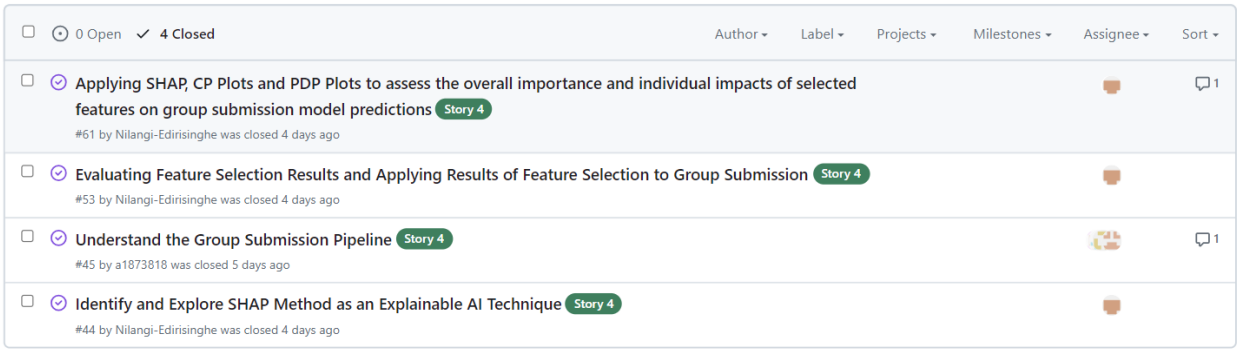

Figure 21: GitHub addressed successfully in sprint 4.

In Sprint 4, I successfully covered and completed the following GitHub issues, which directly contributed to our group's objectives:

1. Applying SHAP, CP Plots, and PDP Plots to assess the overall importance and individual impacts of selected features on group submission model predictions (Story 4) – This task involved implementing and analyzing these explainable AI methods to better understand the contribution of features to the model's performance.

2. Evaluating Feature Selection Results and Applying Results of Feature Selection to Group Submission (Story 4) – I critically assessed my feature selection analysis and confirmed its effectiveness by integrating it into the group submission, leading to improvements in our leaderboard score.

3. Understanding the Group Submission Pipeline (Story 4) – I took the initiative to study and refine the group submission pipeline, identifying key areas for enhancement and successfully integrating feature engineering.

4. Identify and Explore SHAP Method as an Explainable AI Technique (Story 4) – I applied the SHAP method to derive meaningful insights regarding the model and its behavior, contributing to a deeper understanding of the predictions.

These closed GitHub issues reflect my contributions to the project during Sprint 4.

6. Requirements Changes

In the upcoming sprint, we are planning to transition from shallow machine learning models to a simple deep neural network (DNN) due to time constraints. While shallow models like KNN and XGBoost have been useful, we believe a simple DNN may capture more complex patterns in the data, potentially improving our leaderboard score. Given the limited time available, we will focus on implementing a basic DNN architecture rather than an overly complex model. This approach will allow us to evaluate the feasibility and effectiveness of DNNs within our current project timeline.

Although deep neural networks typically require more time for training and tuning, we will aim to streamline the process by focusing on key parameters and using techniques like early stopping to prevent overfitting. This change in approach will affect our project timeline by requiring more focus on training, experimentation, and validation, but the simplified DNN architecture should help us manage these tasks efficiently within the sprint.