Final Report of Group IF_PG1

Rail Break Prediction AI

Nilangi Maheesha Sithumini Edirisinghe

A1882259

1. Project Vision

At the outset, our project aimed to develop a machine learning classification model to predict rail breaks across Australia's extensive railway network within a 30-day horizon. The vision centered around enhancing rail safety and operational efficiency, driven by predictive insights from historical and sensor data provided by ARTC. The team aspired not only to achieve high model accuracy but also to secure a leading position on the leaderboard, ultimately offering valuable decision-making support for rail network managers.

In alignment with this vision, our final model effectively met these objectives. Using an advanced XGBoost model, as shown in the final architecture, we achieved a positive class F1 score of 0.5224, which ranked us 6th on the leaderboard. This accomplishment validates the model's predictive power in identifying potential rail faults, thereby supporting the project's overarching safety and efficiency goals.

Additionally, by applying explainable AI techniques like SHAP, LIME, and PDP plots, we gained a deep understanding of feature importance and model behavior. This transparency reinforces the utility of our solution, offering interpretability that aligns well with the project's goal of actionable insights for railway management. Overall, our model closely embodies the vision set forth in the initial report, delivering a practical, safety-oriented tool for predictive rail maintenance.

2. Customer Q&A

Throughout our project, we held review meetings with our tutor, who acted as our client, providing essential guidance on critical technical questions, model-building strategies, and feedback on our progress.

- **Sprint 1 Review Meeting**: After presenting our initial research on machine learning methods, feature engineering, and feature selection, our tutor advised us to push a working model to the leaderboard by the sprint's end. This clear directive sets the groundwork for a results-driven approach.
- **Sprint 2 Review Meeting**: We questioned why high validation accuracy did not translate into a high leaderboard score. Our tutor clarified that certain features might not contribute significantly to the F1 score, suggesting alternative approaches to imputation and scaling. He prompted us to visualize feature distributions, explaining that misaligned scaling could distort the original data and reduce model accuracy. I presented my feature selection work, including heat maps for correlation coefficients, and raised concerns about the weak predictive power of individual correlations. The tutor advised that adjusting scales could reveal hidden predictive power, commending our progress and encouraging further scaling exploration.

- **Sprint 3 Review Meeting**: Presenting a feature selection analysis, I raised concerns about stagnation in our F1 score. The tutor highlighted potential data leakage in our validation split, suggesting careful data separation to improve accuracy. He also guided us on feature creation, suggesting we avoid binning continuous values and instead focus on aggregating or scaling to preserve essential patterns.
- **Sprint 4 Review Meeting**: We presented our best XGBoost model and sought advice on advancing our leaderboard score. The tutor recommended exploring a deep neural network (DNN) model, given our extensive experimentation with XGBoost, emphasizing the importance of shifting to a new approach for improvement. This pivot towards DNN introduced a new direction for optimizing performance.

Reflection on Client Meetings

The client meetings were instrumental in shaping our project's focus and approach. The tutor's guidance on feature scaling and handling data leakage directly strengthened our model's interpretability and reliability. One of our key strengths was the preparedness we demonstrated in each review, presenting both individual analyses and group progress, which facilitated structured and productive feedback. However, a notable shortcoming was our delayed adoption of deeper models like DNN, which may have limited our leaderboard performance.

In future projects, I would prioritize early experimentation with different model architectures and scaling techniques, ensuring alignment with project goals from the outset. The insights gained from these meetings have underscored the importance of iterative testing, stakeholder feedback, and the flexibility to pivot, which are critical for successfully meeting evolving project needs.

3. Users and User Stories

1. Railway Maintenance Team

Role Summary:

As members of the railway maintenance team, the primary responsibility lies in ensuring the railway infrastructure remains operational and safe. The rail break prediction model allows the team to anticipate potential rail breaks within a 30-day period, enabling proactive scheduling of inspections and prioritization of repairs. By acting on these predictions, the team can prevent disruptions and reduce the risk of accidents.

Key User Stories:

1. **Research and Identify Techniques** (User Story #1): This user story involved researching various feature engineering, feature selection, and machine learning techniques to approach a problem with an imbalanced dataset. This allowed the team to identify and explore relevant techniques for prediction accuracy, enabling a strong foundation for the maintenance team to trust the model's predictions and schedule preventive maintenance based on those predictions.
2. **Implement Techniques for Imbalanced Data** (User Story #2): This story focused on testing different techniques for feature engineering, feature selection, and handling imbalanced data to improve prediction reliability. By using SMOTE for balancing and XGBoost for handling skewed datasets, we were able to enhance the prediction model's F1 score, ensuring the maintenance team received more accurate risk assessments.
3. **Feature Identification for Key Insights** (User Story #5): In this user story, the team identified critical features driving rail breaks, such as tonnage and curvature. This insight enables the maintenance team to understand which areas or rail segments might need more immediate attention, allowing them to prioritize based on specific risk indicators and resource constraints.

2. Railway Operations Manager

Role Summary:

The railway operations manager relies on predictive data to optimize train schedules and manage track availability. Insights into potential rail breaks allow the manager to adjust operations preemptively, ensuring smooth train operations and minimizing delays due to unforeseen track issues.

Key User Stories:

1. **Submit Model to Leaderboard** (User Story #2): This user story involved submitting the initial model to the Insight Factory leaderboard, a valuable test of the model's robustness in a simulated environment. Achieving a good leaderboard score affirmed the model's reliability, helping operations managers confidently use predictions to adjust schedules as needed.
2. **Improve F1 Score Through Hyperparameter Tuning** (User Story #3): By experimenting with hyperparameter tuning and additional feature engineering methods, we aimed to enhance the model's F1 score. For the operations manager, a higher F1 score means fewer false positives and negatives, enabling better decision-making in operations planning.
3. **Use Explainability Techniques for Model Validation** (User Story #4.2): This story was essential for the operations manager, as it applied explainability techniques (like SHAP and LIME) to ensure model transparency. Understanding how the model arrived at predictions provides the manager with greater confidence in data-driven decisions, especially when rerouting or rescheduling is involved.

3. Railway Company Executive

Role Summary:

The executive uses the model's predictions to make strategic decisions around infrastructure investment, resource allocation, and risk management. The focus is on long-term sustainability and profitability, balanced with safety and regulatory compliance.

Key User Stories:

1. **Exploration of Model Explainability (XAI)** (User Story #4.1): In this story, we explored model explainability techniques, providing insights into the model's decision-making process. For executives, this clarity in model behavior supports strategic decision-making, allowing better justification for budget allocation towards risk-heavy areas identified by the model.
2. **Experiment with Different Model Techniques** (User Story #3): This story involved testing various machine learning models, which allowed us to find a robust model that balanced predictive accuracy with interpretability. The executive can leverage these results to guide strategic investments by understanding which models work best for different types of data and prediction requirements.
3. **Achieve F1 Score Exceeding 70%** (User Story #5): Although this ambitious F1 score goal was not fully reached, the attempts underscored the model's strengths

and limitations, giving the executive a realistic outlook on the prediction capabilities. This allows better planning for investments in infrastructure upgrades or additional monitoring technology.

4. Government Regulatory Body

Role Summary:

A representative from a regulatory body uses the model's outputs to enforce safety standards and compliance. Predictions help identify high-risk areas that may require closer inspection, ensuring public safety and adherence to legal standards.

Key User Stories:

1. **Implement XAI Techniques to Understand Feature Impact** (User Story #4.2): Implementing explainability techniques provided insights into feature importance, which is crucial for regulatory bodies. By identifying specific features that predict rail break risk, the regulatory body can establish more focused safety standards, ensuring compliance in areas most vulnerable to incidents.
2. **Track Key Risk Features for Compliance** (User Story #5): The identification of key features driving rail breaks was crucial for the regulatory body to track compliance in critical areas. With a clear understanding of high-risk factors, they can create policies targeting these elements, ensuring better safety standards across railway operations.
3. **Research ML Approaches to Enhance Accuracy** (User Story #1): This foundational story enabled us to experiment with techniques that improve the model's prediction accuracy, ultimately aiding the regulatory body in using data-backed insights to refine compliance checks and enforce standards effectively.

5. Insurance Company

Role Summary:

Insurance companies use predictive models to assess risk and determine premiums for railway operations. By quantifying the likelihood of rail breaks and their financial impact, the company can adjust premiums and offer risk mitigation recommendations to clients.

Key User Stories:

1. **Evaluate Prediction Model Using Leaderboard Metrics** (User Story #2): Submitting the model to the leaderboard helped evaluate its performance in an unbiased, standardized way. This validation allowed the insurance company to

base premium calculations on a well-tested model, ensuring financial assessments aligned with real-world risk.

2. **Enhance Model Accuracy for Risk Assessment** (User Story #3): By improving the model’s accuracy through tuning and balancing, we provided the insurance company with reliable predictions, enabling them to set premiums that reflect true operational risks and incentivize safer practices among clients.

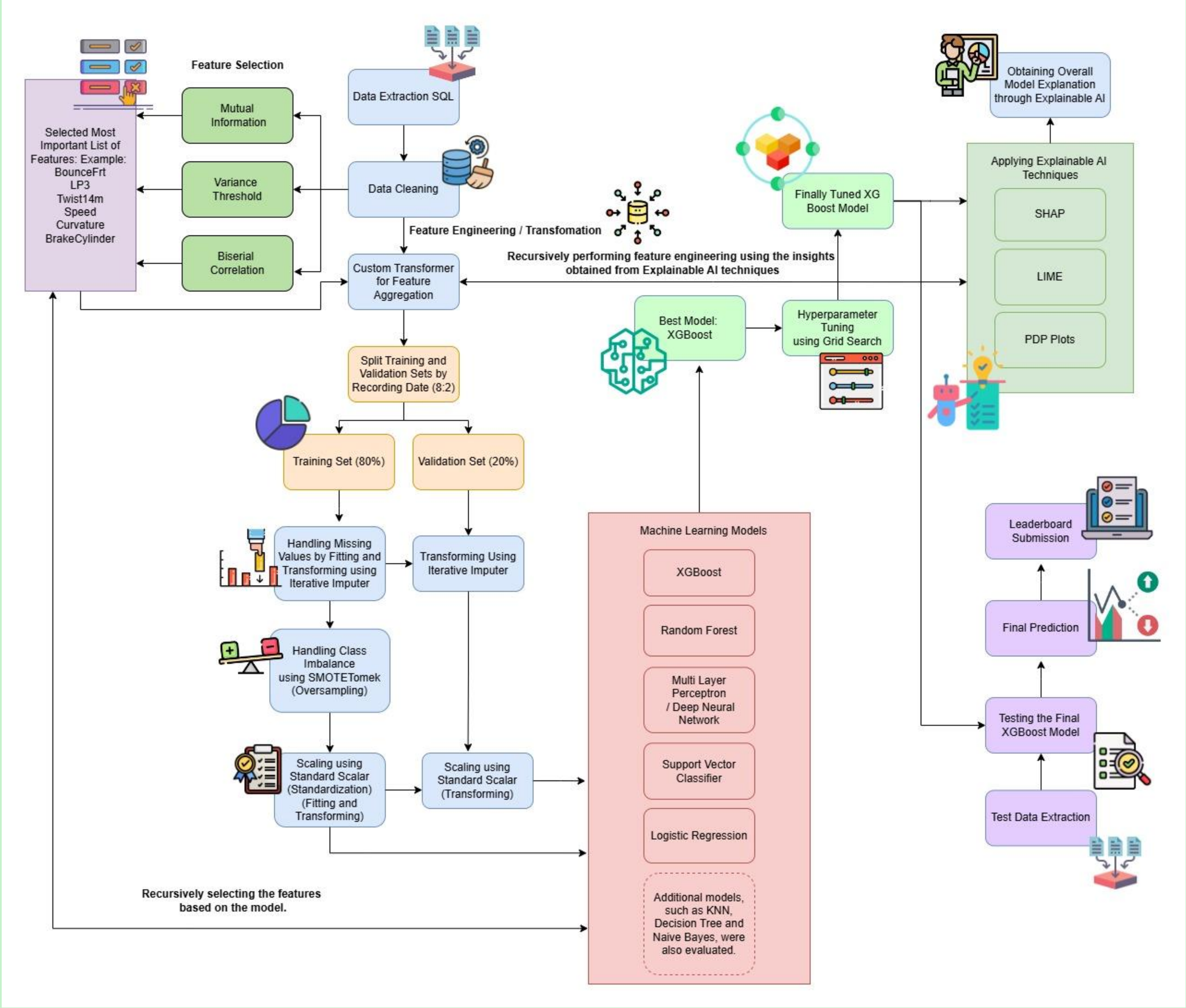
3. **Use Explainable AI for Transparent Premium Decisions** (User Story #4.2): Explainable AI techniques provided insights into which features impacted predictions most significantly. For the insurance company, this transparency supports more precise, justifiable premium adjustments based on the model’s assessment of operational risk factors.

Contrast with Initial Roles Identified

In our initial report, we anticipated that the primary users would include maintenance teams, operations managers, regulatory bodies, and insurance companies, with no significant changes in user identification throughout the project. However, as we progressed, we developed a deeper understanding of each user’s interaction with the system and specific needs. For example, we realized the maintenance team would benefit not only from predictive insights but also from detailed feature analysis to prioritize resources. The operations manager needed a high F1 score for reliable scheduling, while the insurance company required model explainability to make transparent premium decisions.

This expanded understanding of each role allowed us to tailor user stories more effectively, focusing on both prediction accuracy and explainability to meet diverse stakeholder needs. The roles remained largely as expected, but our enhanced focus on model interpretability and specific feature impacts aligned our final solution closely with the operational and strategic objectives of each user type.

4. Software Architecture



Feature Selection
Selected Most Important List of Features: Example: BounceFrt LP3 Twist14m Speed Curvature BrakeCylinder
Mutual Information
Variance Threshold
Biserial Correlation
Data Extraction SQL
Data Cleaning
Feature Engineering / Transfomation
Custom Transformer for Feature Aggregation
Recursively performing feature engineering using the insights obtained from Explainable AI techniques
Split Training and Validation Sets by Recording Date (8:2)
Training Set (80%)
Validation Set (20%)
Handling Missing Values by Fitting and Transforming using Iterative Imputer
Transforming Using Iterative Imputer
Handling Class Imbalance using SMOTETomek (Oversampling)
Scaling using Standard Scalar (Standardization) (Fitting and Transforming)
Scaling using Standard Scalar (Transforming)
Recursively selecting the features based on the model.
Machine Learning Models
XGBoost
Random Forest
Multi Layer Perceptron / Deep Neural Network
Support Vector Classifier
Logistic Regression
Additional models, such as KNN, Decision Tree and Naive Bayes, were also evaluated.
Best Model: XGBoost
Hyperparameter Tuning using Grid Search
Finally Tuned XG Boost Model
Applying Explainable AI Techniques
SHAP
LIME
PDP Plots
Obtaining Overall Model Explanation through Explainable AI
Leaderboard Submission
Final Prediction
Testing the Final XGBoost Model
Test Data Extraction

Figure 1: Software Architecture of Final Machine Learning Solution

Justification for Software Architecture

The chosen architecture is carefully designed to meet the predictive maintenance needs of railway infrastructure by not only forecasting potential rail breaks within the next 30 days but also providing a detailed, interpretable breakdown of feature importance. Starting with **feature selection** methods such as Mutual Information and Biserial Correlation, it focuses on filtering impactful features that enhance model interpretability and improve predictive accuracy. **Data cleaning** and **transformation stages** ensure high-quality, normalized data input, which is crucial for consistent model performance.

This architecture incorporates a **recursive approach to feature selection** and **feature engineering**. By iteratively refining the feature set based on model performance, we retain only the most predictive features while reducing noise. Additionally, **recursive feature engineering** utilizes insights from Explainable AI techniques (SHAP, LIME, PDP), allowing targeted transformations on high-impact features identified by model interpretability tools. This recursive process ensures the model reflects the most relevant factors influencing rail breaks.

Moreover, **Explainable AI integration** provides an overall model explanation, allowing users to receive reports on which features are crucial. This transparency is invaluable for stakeholders like the Railway Maintenance Team and Operations Managers, who rely on both predictions and detailed insights to guide maintenance and operational decisions. Finally, a **testing and submission process** enables ongoing model evaluation, aligning with the project's competitive goals and performance benchmarks, making this architecture adaptable and reliable for safety-critical, real-time predictions.

5. Tech Stack and Standards

Final Tech Stack

1. **Back-end**
    - **Programming Language**: Python – Chosen for its extensive machine learning libraries, ease of data manipulation, and broad support for statistical operations, Python was crucial for back-end processing.
    - **Libraries and Frameworks**:
        - **Pandas**: Used extensively for data manipulation and preprocessing, particularly for handling large, structured datasets.
        - **NumPy**: Assisted with efficient numerical computations, forming the basis for handling arrays and matrices required for machine learning tasks.
        - **Scikit-Learn**: Provided essential tools for preprocessing (e.g., StandardScaler, PowerTransformer), model evaluation (e.g., train_test_split, GridSearchCV), and initial model implementations (e.g., RandomForestClassifier).
        - **XGBoost**: Selected as the final model due to its performance with imbalanced datasets, which was critical for achieving a high F1 score in rail break prediction.

        - **Imbalanced-Learn**: SMOTETomek resampling techniques were applied to handle class imbalance, which was essential for improving model training on the minority class.
        - **LIME and SHAP**: These explainable AI tools allowed feature interpretation, providing essential transparency for users like the Railway Maintenance Team.
        - **Matplotlib**: Used for data visualization to analyze feature distributions and class imbalances, aiding in data preprocessing and model tuning.

2. **Front-end**
    - **InsightFactory.ai Platform**: This platform served as our main front-end interface, allowing structured workflows for data ingestion, model building, and testing. It eliminated the need for a separate IDE by providing a collaborative, cloud-based environment suitable for production pipelines.
3. **Development and Communication Tools**
    - **InsightFactory.ai**: The primary environment for managing data, building models, and testing predictions, offering robust integration with cloud storage and data visualization.
    - **Azure Databricks**: Enabled data extraction, initial exploration, and storage, essential for managing large datasets provided by the client.
    - **MS Teams and Zoom**: Used for communication with the product owner and team meetings, ensuring smooth coordination.
    - **WhatsApp and Google Sheets**: Used for daily scrum updates and informal communication, helping track individual progress and blockages.
    - **GitHub**: Maintained the repository, documentation, and version control for code and analysis, streamlining collaboration and progress tracking.

Coding Standards

- **PEP 8**: Followed PEP 8 guidelines for Python code to maintain consistency, readability, and structure across modules, particularly in terms of indentation, variable naming, and documentation.
- **Documentation**: Each function and class was documented to ensure transparency and ease of handover. Inline comments were also added for complex operations, especially in feature engineering.

Justification and Reflection

The final tech stack leveraged Python’s machine learning ecosystem, providing an efficient environment for model development and data processing. Using **Scikit-Learn** and **XGBoost** allowed flexible, high-performance model building, while **Imbalanced-Learn** helped overcome the challenges of class imbalance, a critical factor for accurate rail break prediction. Explainable AI tools like **LIME** and **SHAP** proved invaluable in aligning with user requirements for model transparency, providing actionable insights into critical features.

The **InsightFactory.ai** platform and **Azure Databricks** worked well as a collaborative environment, but its limitations in model customization occasionally hindered the full application of experimental features. Compared to the initial plan, which anticipated minimal platform limitations, these restrictions required us to explore workarounds to achieve optimal results. Communication tools like **MS Teams** and **WhatsApp** supported effective team coordination, while **GitHub** provided essential version control and centralized documentation.

In summary, the tech stack enabled robust model development despite some limitations, and the integration of explainable AI enhanced the software's transparency and user value. For future projects, exploring platform flexibility and early identification of potential limitations would improve project adaptability and efficiency.

6. Group Meetings and Team Member Roles

Meeting Frequency and Duration

Our team met virtually once a week through Zoom, adhering to the planned schedule of 8 PM every Monday. Each meeting was timeboxed to 1 hour, with a strict agenda managed by the Scrum Master to ensure discussions remained focused and productive. While timeboxing was generally effective, certain in-depth discussions, especially around model tuning and feature selection, required additional flexibility. Agile methodology encourages this adaptability, and balancing structured time with necessary flexibility was instrumental in addressing complex issues.

Sprint Retrospective Meetings

Sprint retrospectives took place at the end of each sprint on Mondays, from 6:40 PM to 7:05 PM. These sessions included structured presentations where each team member shared their completed tasks, key results, and blockers, followed by questions and feedback from the tutor. Although we often found the 25-minute timeframe tight, this setup encouraged efficient communication and concise reporting. A longer session would have provided room for more detailed feedback, but we adapted by focusing on critical points and writing on MS Teams for additional questions as required.

Additional Communication Channels with the Customer

To ensure we remained aligned with customer expectations, we established multiple communication channels with both the tutor and InsightFactory AI representatives. MS Teams served as our primary platform, allowing us to engage with both the tutor and the InsightFactory team, including Toby. InsightFactory also arranged a session during a lecture, which offered additional guidance on project specifics. Each sprint review meeting provided an opportunity to ask questions directly to the tutor, ensuring prompt feedback. These additional channels were invaluable, as they allowed us to refine our work continuously and address technical questions efficiently, which is crucial for Agile projects that rely on iterative customer feedback.

Scrum Masters for Each Sprint

Each team member took on the role of Scrum Master for a sprint, fostering leadership development and distributing responsibilities fairly:

- **Sprint 1**: Qiaoqiao Chen
- **Sprint 2**: Yong Kheng Beh
- **Sprint 3**: Bhagya Indeewari Senanayake Senanayake Mudiyanselage
- **Sprint 4**: Zewei You
- **Sprint 5**: Xinyue Ma

Personal Reflection on Group Meetings and Own Role

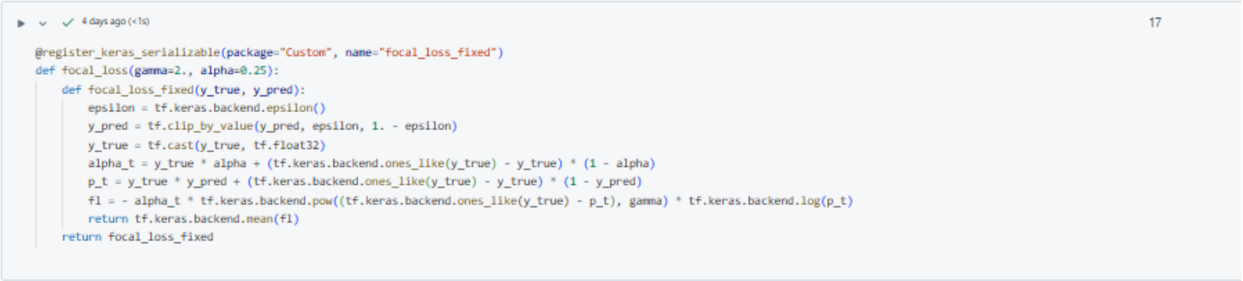
Throughout the project, I actively participated in group meetings and sprint reviews, presenting my individual work, sharing results, and seeking feedback. I consistently contributed by presenting my analysis findings, which allowed me to receive targeted feedback and ensured that the team was aligned to my progress. Beyond the meetings, I communicated effectively with teammates via WhatsApp and phone calls to resolve blockers quickly, fostering a collaborative atmosphere.

My role in the project was multifaceted, covering both technical and supportive responsibilities. Technically, I conducted an in-depth **feature selection analysis** using Mutual Information (MI), Variance Threshold (VT), and Biserial Correlation across five models—K-Nearest Neighbors (KNN), XGBoost, Random Forest, Logistic Regression, and Deep Neural Network (DNN). My analysis revealed which features influenced the F1 score positively or negatively for each model, providing critical insights for model optimization. This work supported the team's goal of achieving a balanced, unbiased experimentation process, helping us improve the robustness of our final model.

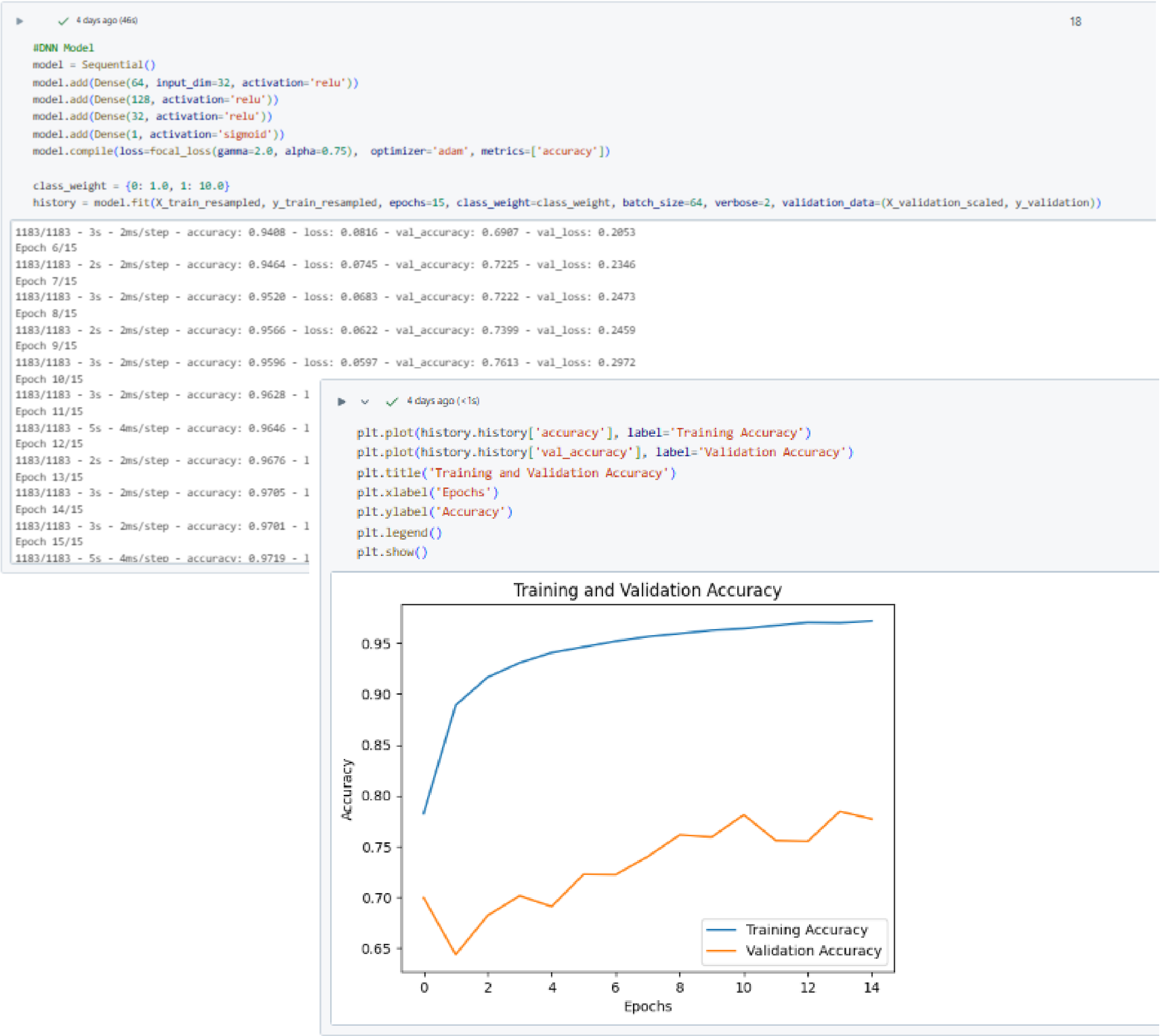
To further enhance the model's interpretability, I implemented **Explainable AI techniques** such as SHAP (Shapley Additive Explanations), PDP (Partial Dependence Plots), and CP (Conditional Plots). Through these methods, I derived valuable insights into how different models and specific features contributed to predictions. For instance, SHAP values highlighted the features with the highest impact on model decisions, while PDP plots provided a view of feature interactions, enhancing our understanding of the model's inner workings. This explainability was crucial for ensuring the model's reliability and for conveying key findings to stakeholders.

During the final phase, I focused on **developing a deep neural network** and **refining the group submission pipeline**. This involved studying the pipeline and structure in detail, which gave me a comprehensive understanding of the project's system architecture. Leveraging this knowledge, I **developed the software architecture** for the final report on behalf of the team. My work in feature selection, model tuning, and submission pipeline analysis contributed significantly to the final architecture, ensuring it aligned with our technical findings and supported the project's predictive objectives.

Reflecting on our Agile process, I recognize that our structured meetings, sprint retrospectives, and continuous communication channels enabled us to iterate and adapt quickly. While time constraints sometimes limited detailed discussions during retrospectives, our proactive communication outside formal meetings compensated for this, maintaining our project's momentum. This experience reinforced the importance of clear, continuous communication, iterative improvement, and a hands-on approach to problem-solving, all of which are invaluable in Agile software development.



```
@register_keras_serializable(package="Custom", name="focal_loss_fixed")
def focal_loss(gamma=2., alpha=0.25):
    def focal_loss_fixed(y_true, y_pred):
        epsilon = tf.keras.backend.epsilon()
        y_pred = tf.clip_by_value(y_pred, epsilon, 1. - epsilon)
        y_true = tf.cast(y_true, tf.float32)
        alpha_t = y_true * alpha + (tf.keras.backend.ones_like(y_true) - y_true) * (1 - alpha)
        p_t = y_true * y_pred + (tf.keras.backend.ones_like(y_true) - y_true) * (1 - y_pred)
        fl = - alpha_t * tf.keras.backend.pow((tf.keras.backend.ones_like(y_true) - p_t), gamma) * tf.keras.backend.log(p_t)
        return tf.keras.backend.mean(fl)
    return focal_loss_fixed
```

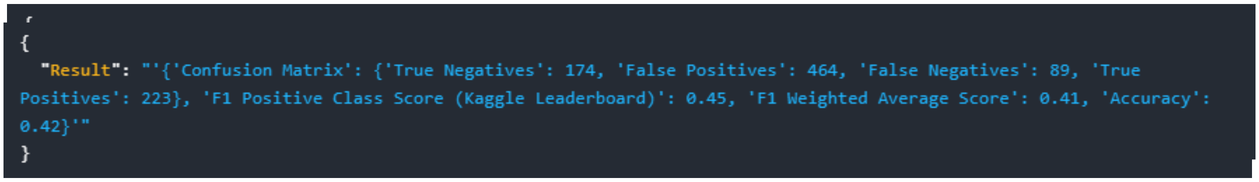


```
#DNN Model
model = Sequential()
model.add(Dense(64, input_dim=32, activation='relu'))
model.add(Dense(128, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss=focal_loss(gamma=2.0, alpha=0.75),  optimizer='adam', metrics=['accuracy'])

class_weight = {0: 1.0, 1: 10.0}
history = model.fit(X_train_resampled, y_train_resampled, epochs=15, class_weight=class_weight, batch_size=64, verbose=2, validation_data=(X_validation_scaled, y_validation))
```

```
1183/1183 - 3s - 2ms/step - accuracy: 0.9408 - loss: 0.0816 - val_accuracy: 0.6907 - val_loss: 0.2053
Epoch 6/15
1183/1183 - 2s - 2ms/step - accuracy: 0.9464 - loss: 0.0745 - val_accuracy: 0.7225 - val_loss: 0.2346
Epoch 7/15
1183/1183 - 3s - 2ms/step - accuracy: 0.9520 - loss: 0.0683 - val_accuracy: 0.7222 - val_loss: 0.2473
Epoch 8/15
1183/1183 - 2s - 2ms/step - accuracy: 0.9566 - loss: 0.0622 - val_accuracy: 0.7399 - val_loss: 0.2459
Epoch 9/15
1183/1183 - 3s - 2ms/step - accuracy: 0.9596 - loss: 0.0597 - val_accuracy: 0.7613 - val_loss: 0.2972
Epoch 10/15
1183/1183 - 3s - 2ms/step - accuracy: 0.9628 - l
Epoch 11/15
1183/1183 - 5s - 4ms/step - accuracy: 0.9646 - l
Epoch 12/15
1183/1183 - 2s - 2ms/step - accuracy: 0.9676 - l
Epoch 13/15
1183/1183 - 3s - 2ms/step - accuracy: 0.9705 - l
Epoch 14/15
1183/1183 - 3s - 2ms/step - accuracy: 0.9701 - l
Epoch 15/15
1183/1183 - 5s - 4ms/step - accuracy: 0.9719 - l
```

```
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```



```
{
  "Result": "'{'Confusion Matrix': {'True Negatives': 174, 'False Positives': 464, 'False Negatives': 89, 'True Positives': 223}, 'F1 Positive Class Score (Kaggle Leaderboard)': 0.45, 'F1 Weighted Average Score': 0.41, 'Accuracy': 0.42}'"
}
```

Figure 2: Some Screenshots that show my final individual contribution in sprint 5 (My role), where I developed a deep neural network for the project.

7. Personal Reflection on Software Process

Our team's implementation of Agile Scrum diverged from the traditional textbook approach in several ways, particularly in the flexibility we adopted for our sprint planning and review sessions. In a strict Scrum setup, sprints are typically timeboxed with pre-defined deliverables and objectives. While we adhered to the core Agile principle of iterative development, our sprints were more fluid to accommodate the experimental nature of machine learning and model tuning. This adaptability meant that our deliverables sometimes shifted within a sprint, especially as new insights emerged from tutor feedback or data analysis. This deviation from textbook Scrum was beneficial, as it allowed us to respond dynamically to challenges rather than adhering rigidly to pre-planned tasks. This flexibility ultimately contributed to the development of more effective software, as we could continuously refine our approach based on real-time learning.

One significant advantage of Agile Scrum in this project was the ability to break down the project into user stories and address them incrementally across sprints. This approach not only made the workload manageable but also provided a sense of accomplishment at the end of each sprint. The iterative feedback we received, particularly during sprint review meetings, was instrumental in guiding our development. Tutor input, combined with feedback from InsightFactory, allowed us to make informed decisions on model improvements and feature engineering, thereby enhancing our software incrementally. These Agile meetings, structured yet adaptable, were productive sessions that fostered an environment of open communication and collaborative problem-solving. This regular cadence of updates and feedback was key in helping us develop a robust and well-tested predictive model.

In contrast, a Waterfall approach would likely have been less effective for this project. Waterfall's linear nature would have required us to finalize each phase, such as feature selection, model tuning, and testing, before moving on to the next, without room for iterative feedback and adjustments. Given the experimental nature of machine learning, where refinement and experimentation are ongoing, a Waterfall approach would have limited our ability to adapt as we gained new insights from data and feedback. By the time we reached the testing phase, it would have been too late to make necessary adjustments without significantly impacting the project timeline. Waterfall's rigid structure could have hindered our progress by restricting flexibility, potentially resulting in a model that was less optimized or failed to meet evolving requirements.

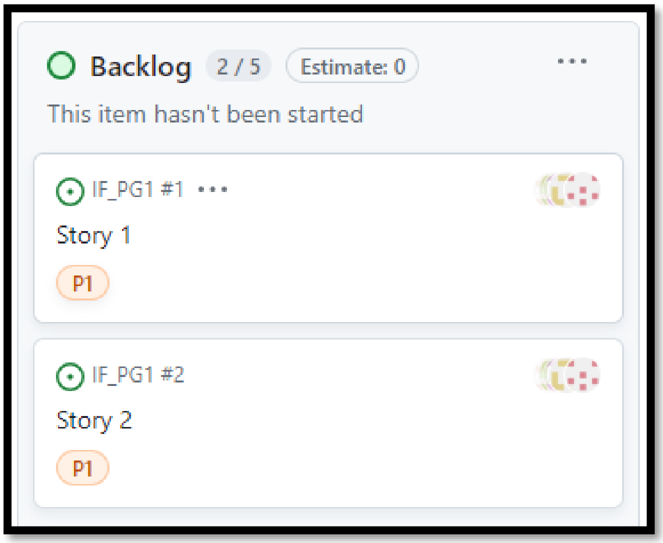
Reflecting on this experience, I have gained valuable insights into the effectiveness of Agile Scrum for projects that require iterative improvement and continuous adaptation. The successes we achieved, such as integrating tutor feedback into model refinement and achieving incremental milestones, underscore Agile's strength in facilitating adaptive development. However, one shortcoming was that our flexibility sometimes led to scope creep within sprints, which required careful management to avoid delays. In future projects, I will leverage Agile's iterative structure for work that demands regular adjustments and feedback but will also strive to maintain clearer sprint boundaries to avoid task overflow.

Overall, this project reinforced my appreciation for Agile's adaptability, especially in contexts that benefit from feedback and iterative progress. While Waterfall may be appropriate for projects with highly defined stages and limited need for adjustment, Agile's strengths lie in its responsiveness and ability to produce a continuously improving product. Moving forward, I am more equipped to assess which methodology best fits the project requirements, balancing flexibility with structure to achieve optimal outcomes.

8. Snapshots

a) Snapshots from Sprint 1

1.1 Product Backlog and Task Board



○ Backlog 2 / 5 Estimate: 0 ···

This item hasn't been started

IF_PG1 #1 ···

Story 1

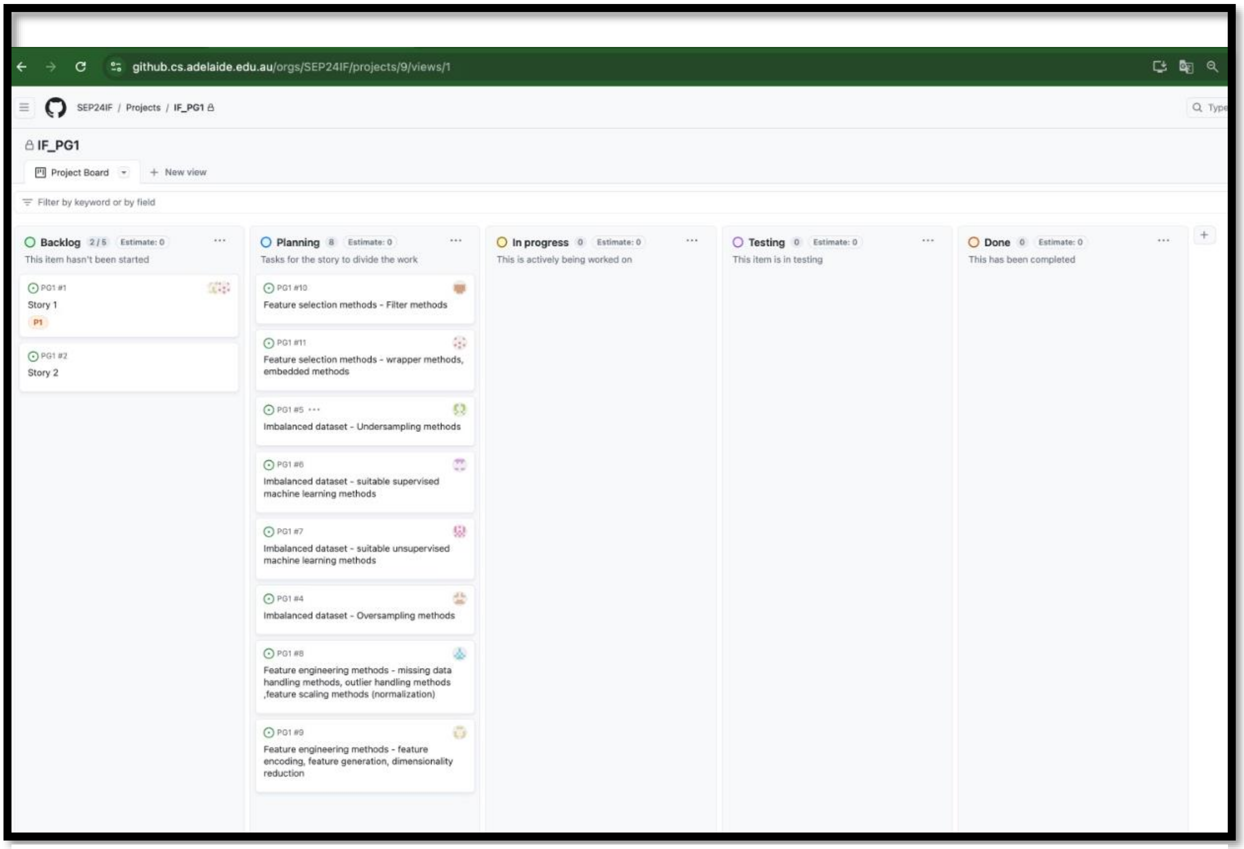
P1

IF_PG1 #2

Story 2

P1

Figure 3: The Screenshot of the Product Backlog



github.cs.adelaide.edu.au/orgs/SEP24IF/projects/9/views/1

SEP24IF / Projects / IF_PG1

IF_PG1

Project Board · + New view

Filter by keyword or by field

Backlog 2 / 5 Estimate: 0
This item hasn't been started

- PG1 #1 Story 1 P1
- PG1 #2 Story 2

Planning 8 Estimate: 0
Tasks for the story to divide the work

- PG1 #10 Feature selection methods - Filter methods
- PG1 #11 Feature selection methods - wrapper methods, embedded methods
- PG1 #5 Imbalanced dataset - Undersampling methods
- PG1 #6 Imbalanced dataset - suitable supervised machine learning methods
- PG1 #7 Imbalanced dataset - suitable unsupervised machine learning methods
- PG1 #4 Imbalanced dataset - Oversampling methods
- PG1 #8 Feature engineering methods - missing data handling methods, outlier handling methods ,feature scaling methods (normalization)
- PG1 #9 Feature engineering methods - feature encoding, feature generation, dimensionality reduction

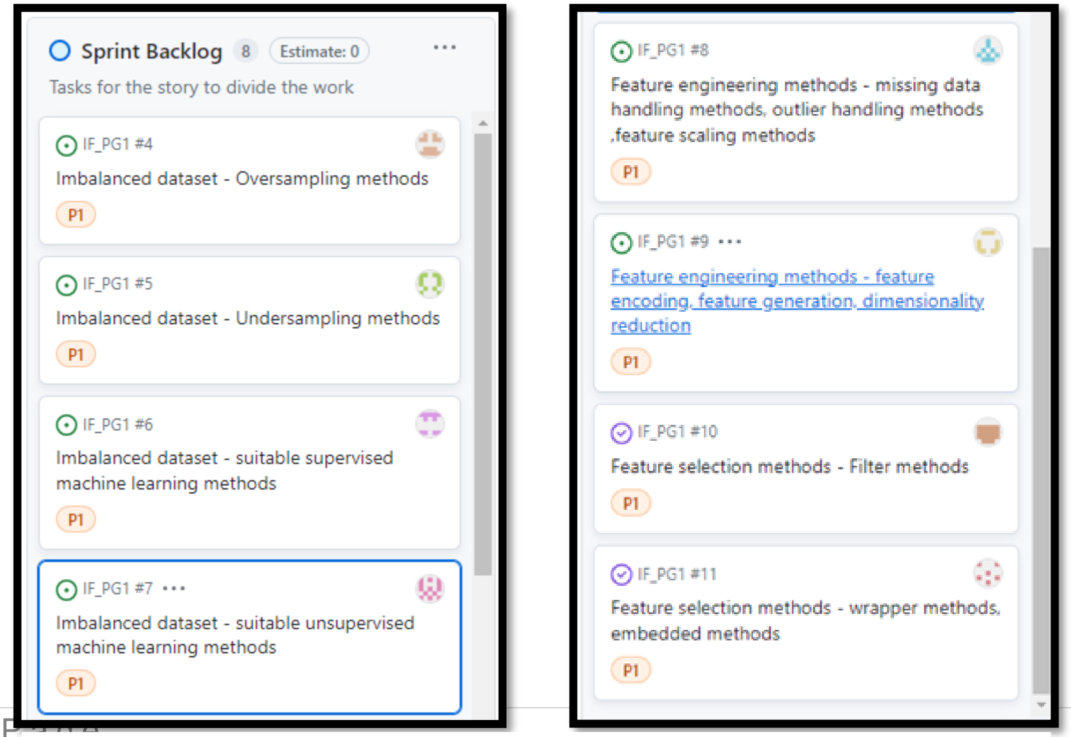
In progress 0 Estimate: 0
This is actively being worked on

Testing 0 Estimate: 0
This item is in testing

Done 0 Estimate: 0
This has been completed

Figure 4: The Screenshot of the Task Board

1.2 Sprint Backlog and User Stories



Sprint Backlog 8 Estimate: 0
Tasks for the story to divide the work

- IF_PG1 #4 Imbalanced dataset - Oversampling methods P1
- IF_PG1 #5 Imbalanced dataset - Undersampling methods P1
- IF_PG1 #6 Imbalanced dataset - suitable supervised machine learning methods P1
- IF_PG1 #7 Imbalanced dataset - suitable unsupervised machine learning methods P1
- IF_PG1 #8 Feature engineering methods - missing data handling methods, outlier handling methods ,feature scaling methods P1
- IF_PG1 #9 Feature engineering methods - feature encoding, feature generation, dimensionality reduction P1
- IF_PG1 #10 Feature selection methods - Filter methods P1
- IF_PG1 #11 Feature selection methods - wrapper methods, embedded methods P1

Figure 5: The Screenshot of the Sprint Backlog

User Story 1

As a software engineer, I want to research :

1) feature engineering methods

2) feature selection methods

3) Machine Learning (ML) techniques to approach a problem having an imbalanced dataset

Acceptance criteria :

1) Identify and explore at least 5 techniques each for:
   - Feature engineering methods
   - Feature selection methods
   - ML techniques to approach a problem with an imbalanced dataset.
2) Report your findings.

1.3 Definition of Done (DOD)

Due to the nature of the first sprint, which does not require committing code, the following the following DOD applies once we gain access to the platform. While conducting research and awaiting access to the InsightFactory platform, we have formulated our DOD to ensure we can work efficiently once we gain access and begin developing our machine-learning models. The DOD will be adjusted later if necessary.

Code Standards

1) Readability and Maintainability:
   - Code must be written with clarity and follow established style guides (e.g., PEP 8 for Python).
   - All functions and classes should have clear, concise docstrings explaining their purpose, parameters, and return values.
2) Testing:
   - Unit tests must cover critical functions, especially for data preprocessing, feature engineering, and model evaluation.
   - The code should pass all tests before being merged into the main branch.

Documentation

1) Reproducibility:
   - All scripts and notebooks must be structured to allow easy reruns by other team members.
   - Include a README file that details how to set up the environment, run the analysis, and where to find each component of the project.

2) Data Pipeline Documentation:

- The entire data pipeline from raw data ingestion to final model output should be documented.
- Include details on data cleaning steps, feature selection/engineering process, and the reasoning behind selecting specific machine learning models and hyperparameters.

3) Balanced Data Handling:

- Clearly, document the rebalancing techniques used (e.g., SMOTE, undersampling), along with the rationale for choosing them.
- Include performance metrics before and after rebalancing to illustrate the impact of the techniques.

Machine Learning Model

1) Model Selection and Validation:

- All selected models must be validated using appropriate techniques, including cross-validation and/or out-of-sample testing.
- Performance metrics must be calculated for each model and compared against baseline models (e.g., logistic regression).

2) Hyperparameter Tuning:

- Document the hyperparameter tuning process, including which parameters were tuned, the ranges tested, and the final selected values.

3) Model Interpretability:

- If possible, include model interpretability steps (e.g., feature importance, SHAP values) to provide insights into how the model makes predictions.

Review and Sign-Off

1) Peer Review:

- All code and documentation must undergo peer review before being marked as complete.
- Reviewers must verify that all DoD criteria have been met.

2) Sign-Off:

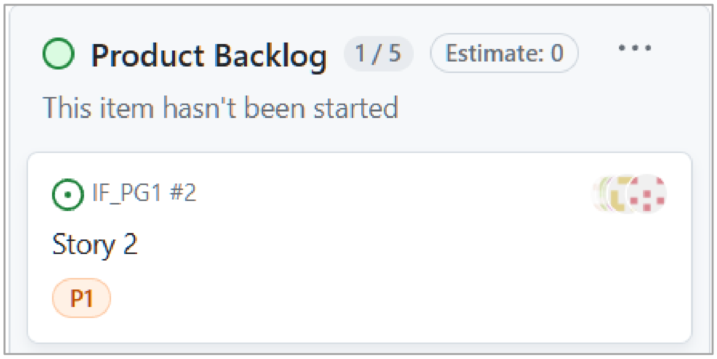
- The team must collectively sign off on each completed user story, ensuring all acceptance criteria are met and all necessary documentation is provided.

1.4 Summary of Changes

As this is the first snapshot of the sprint, this section is not applicable to this report.

b) Snapshots from Sprint 2

1.1 Product Backlog and Task Board



Product Backlog 1 / 5 Estimate: 0

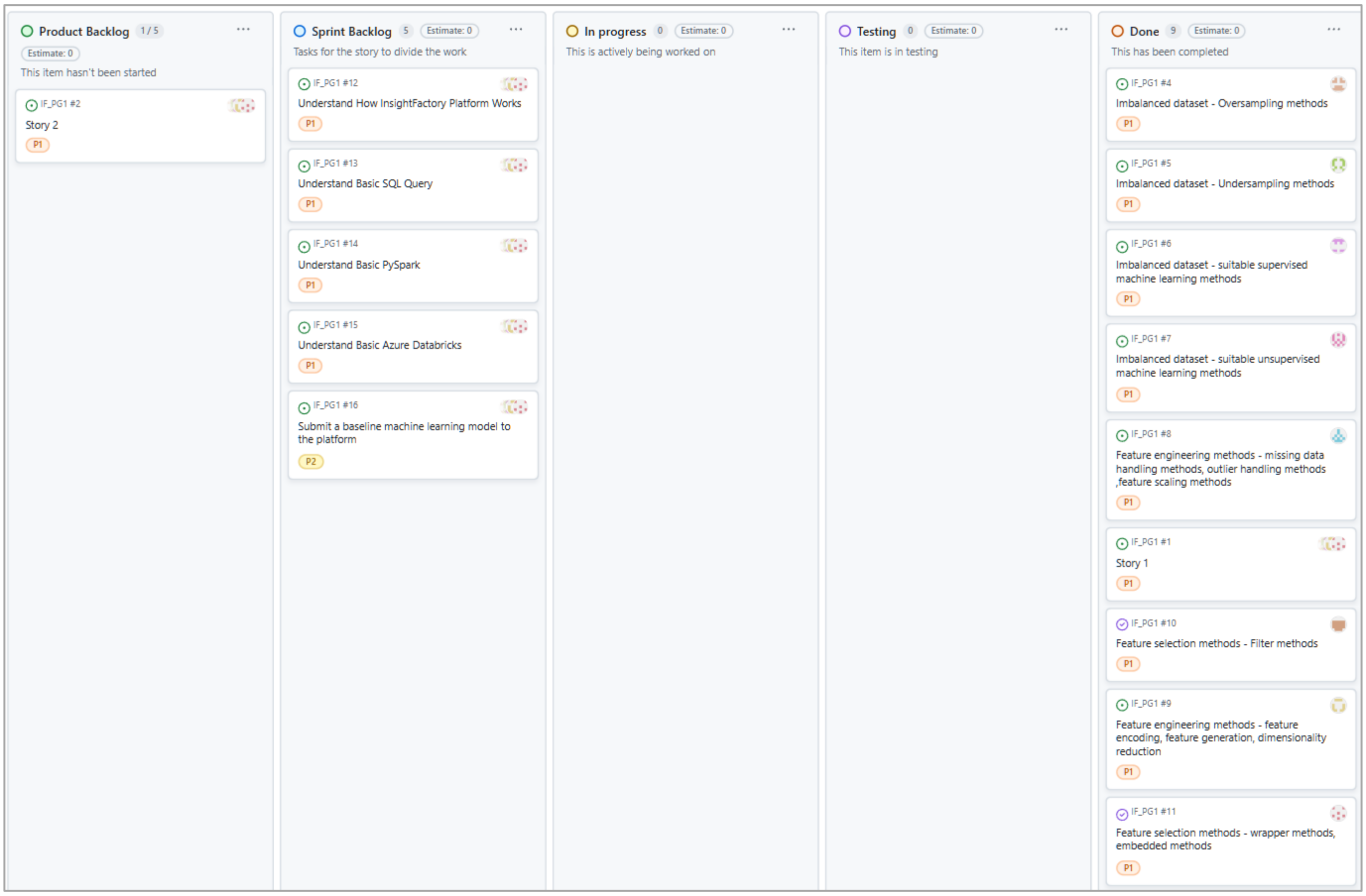
This item hasn't been started

IF_PG1 #2

Story 2

P1

Figure 6. The Screenshot of the Product Backlog



Product Backlog 1 / 5
Estimate: 0
This item hasn't been started

- IF_PG1 #2
  Story 2
  P1

Sprint Backlog 5 Estimate: 0
Tasks for the story to divide the work

- IF_PG1 #12
  Understand How InsightFactory Platform Works
  P1
- IF_PG1 #13
  Understand Basic SQL Query
  P1
- IF_PG1 #14
  Understand Basic PySpark
  P1
- IF_PG1 #15
  Understand Basic Azure Databricks
  P1
- IF_PG1 #16
  Submit a baseline machine learning model to the platform
  P2

In progress 0 Estimate: 0
This is actively being worked on

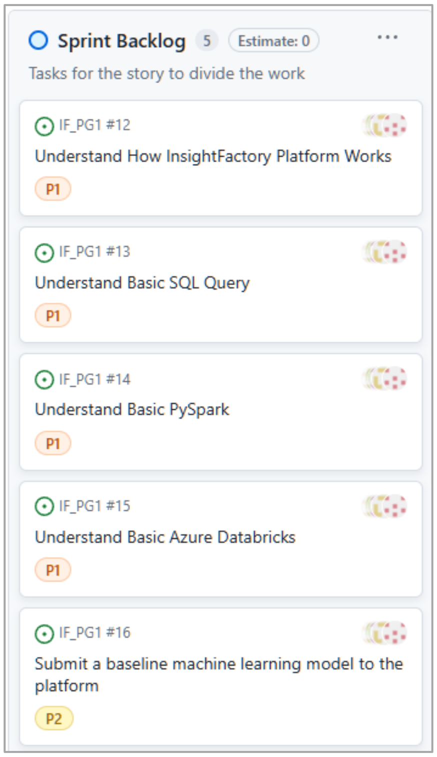
Testing 0 Estimate: 0
This item is in testing

Done 9 Estimate: 0
This has been completed

- IF_PG1 #4
  Imbalanced dataset - Oversampling methods
  P1
- IF_PG1 #5
  Imbalanced dataset - Undersampling methods
  P1
- IF_PG1 #6
  Imbalanced dataset - suitable supervised machine learning methods
  P1
- IF_PG1 #7
  Imbalanced dataset - suitable unsupervised machine learning methods
  P1
- IF_PG1 #8
  Feature engineering methods - missing data handling methods, outlier handling methods ,feature scaling methods
  P1
- IF_PG1 #1
  Story 1
  P1
- IF_PG1 #10
  Feature selection methods - Filter methods
  P1
- IF_PG1 #9
  Feature engineering methods - feature encoding, feature generation, dimensionality reduction
  P1
- IF_PG1 #11
  Feature selection methods - wrapper methods, embedded methods
  P1

Figure 7. The Screenshot of the Task Board

1.5 Sprint Backlog and User Stories



Sprint Backlog 5 Estimate: 0

Tasks for the story to divide the work

IF_PG1 #12
Understand How InsightFactory Platform Works
P1

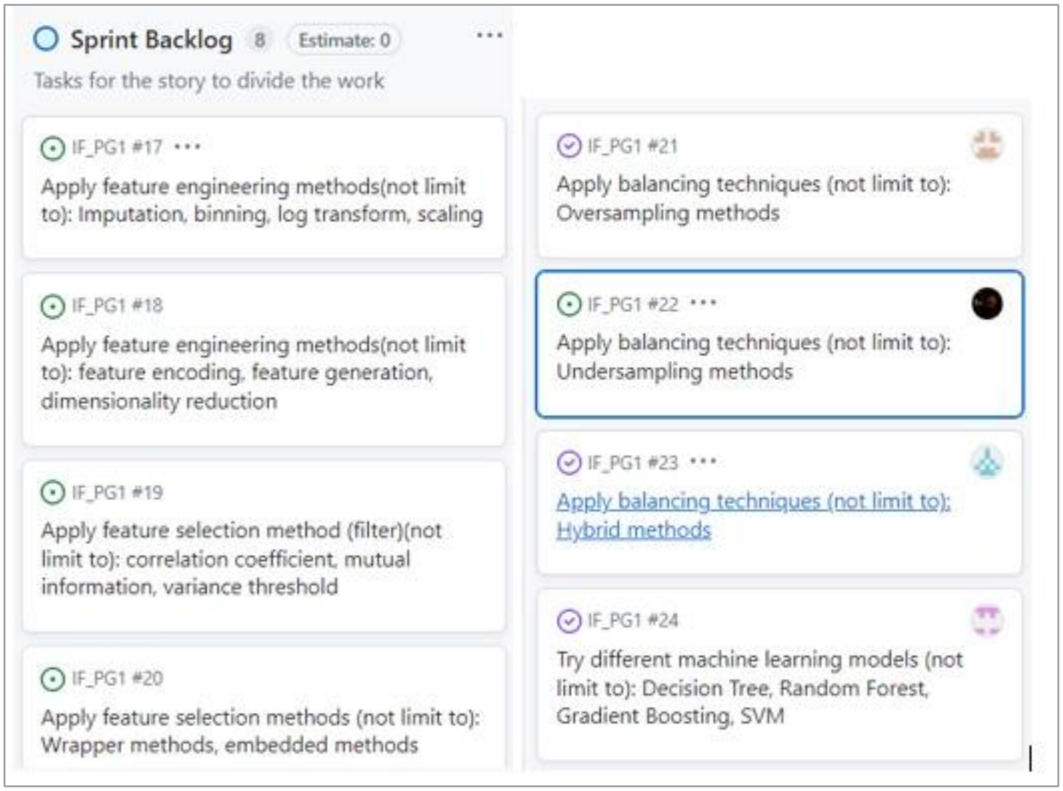
IF_PG1 #13
Understand Basic SQL Query
P1

IF_PG1 #14
Understand Basic PySpark
P1

IF_PG1 #15
Understand Basic Azure Databricks
P1

IF_PG1 #16
Submit a baseline machine learning model to the platform
P2

Figure 8. The Screenshot of the Sprint Backlog for First Week of Sprint 2



Sprint Backlog 8 Estimate: 0

Tasks for the story to divide the work

IF_PG1 #17
Apply feature engineering methods(not limit to): Imputation, binning, log transform, scaling

IF_PG1 #18
Apply feature engineering methods(not limit to): feature encoding, feature generation, dimensionality reduction

IF_PG1 #19
Apply feature selection method (filter)(not limit to): correlation coefficient, mutual information, variance threshold

IF_PG1 #20
Apply feature selection methods (not limit to): Wrapper methods, embedded methods

IF_PG1 #21
Apply balancing techniques (not limit to): Oversampling methods

IF_PG1 #22
Apply balancing techniques (not limit to): Undersampling methods

IF_PG1 #23
Apply balancing techniques (not limit to): Hybrid methods

IF_PG1 #24
Try different machine learning models (not limit to): Decision Tree, Random Forest, Gradient Boosting, SVM

Figure 9. The Screenshot of the Sprint Backlog for Second Week of Sprint 2

User Story 2

As a software engineer, I want to try out:

1) **Feature engineering methods**
2) **Feature selection methods and**
3) **Machine Learning (ML) techniques to approach a problem having an imbalanced dataset, so that I can produce my initial model to InsightFactory leaderboard**

Acceptance criteria :

1) **Implement and try out at least 3 techniques each for:**
   - **feature engineering methods**
   - **feature selection methods**
   - **ML techniques to approach a problem with an imbalanced dataset**
2) **Report what combination of techniques worked so far**
3) **Submit a model to InsightFactory leaderboard**

1. Definition of Done (DOD)

In the second sprint, as we gain access to the InsightFactory platform, the team will begin committing code while adhering to the following DoD to ensure that all defined standards are met and maintained. The DoD remains the same as in Sprint 1 but will be modified as needed when we begin actively committing code.

Code Standards

3) Readability and Maintainability:
   - Code must be written with clarity and follow established style guides (e.g., PEP 8 for Python).
   - All functions and classes should have clear, concise docstrings explaining their purpose, parameters, and return values.
4) Testing:
   - Unit tests must cover critical functions, especially for data preprocessing, feature engineering, and model evaluation.
   - Code should pass all tests before being merged into the main branch.

Documentation

4) Reproducibility:
   - All scripts and notebooks must be structured to allow easy reruns by other team members.
   - Include a README file that details how to set up the environment, run the analysis, and where to find each component of the project.
5) Data Pipeline Documentation:
   - The entire data pipeline from raw data ingestion to final model output should be documented.

- Include details on data cleaning steps, feature selection/engineering process, and the reasoning behind selecting specific machine learning models and hyperparameters.

6) Balanced Data Handling:
   - Clearly document the rebalancing techniques used (e.g., SMOTE, undersampling), along with the rationale for choosing them.
   - Include performance metrics before and after rebalancing to illustrate the impact of the techniques.

Machine Learning Model

3) Model Selection and Validation:
   - All selected models must be validated using appropriate techniques, including cross-validation and/or out-of-sample testing.
   - Performance metrics must be calculated for each model and compared against baseline models (e.g., logistic regression).
4) Hyperparameter Tuning:
   - Document the hyperparameter tuning process, including which parameters were tuned, the ranges tested, and the final selected values.
4) Model Interpretability:
   - If possible, include model interpretability steps (e.g., feature importance, SHAP values) to provide insights into how the model makes predictions.

Review and Sign-Off

3) Peer Review:
   - All code and documentation must undergo peer review before being marked as complete.
   - Reviewers must verify that all DoD criteria have been met.
4) Sign-Off:
   - The team must collectively sign off on each completed user story, ensuring all acceptance criteria are met and all necessary documentation is provided.

2. Summary of Changes:

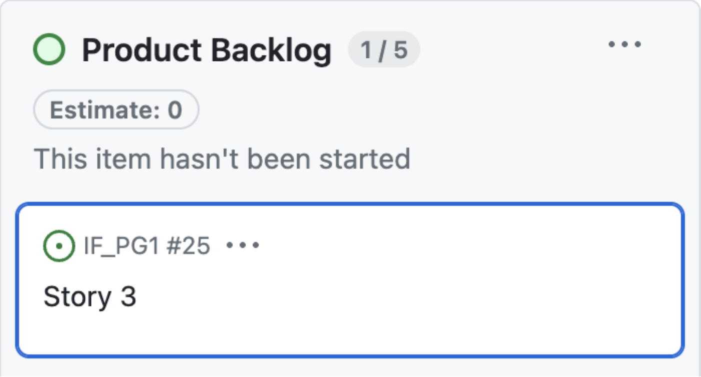
During the first sprint, the team concentrated on researching and identifying techniques for feature engineering, feature selection, and machine learning (ML) to manage imbalanced datasets.

In the first week of Sprint 2, the team gained access to the InsightFactory (IF) platform, which is essential for model implementation. Each member is individually exploring the platform to gain hands-on experience in creating a production line, including connecting databases, ingesting data into a data lake, performing data preprocessing, and training a machine learning model using Databricks notebooks. The team will discuss their findings and address any questions at the beginning of the second week of Sprint 2 while submitting initial models to the InsightFactory leaderboard.

I attended the sprint review/planning meetings on Monday, the 26th of August with my group and Monday, the 2nd of September 2024 with the tutor, and presented my results.

c) Snapshots from Sprint 3

1.1 Product Backlog and Task Board



Product Backlog 1 / 5

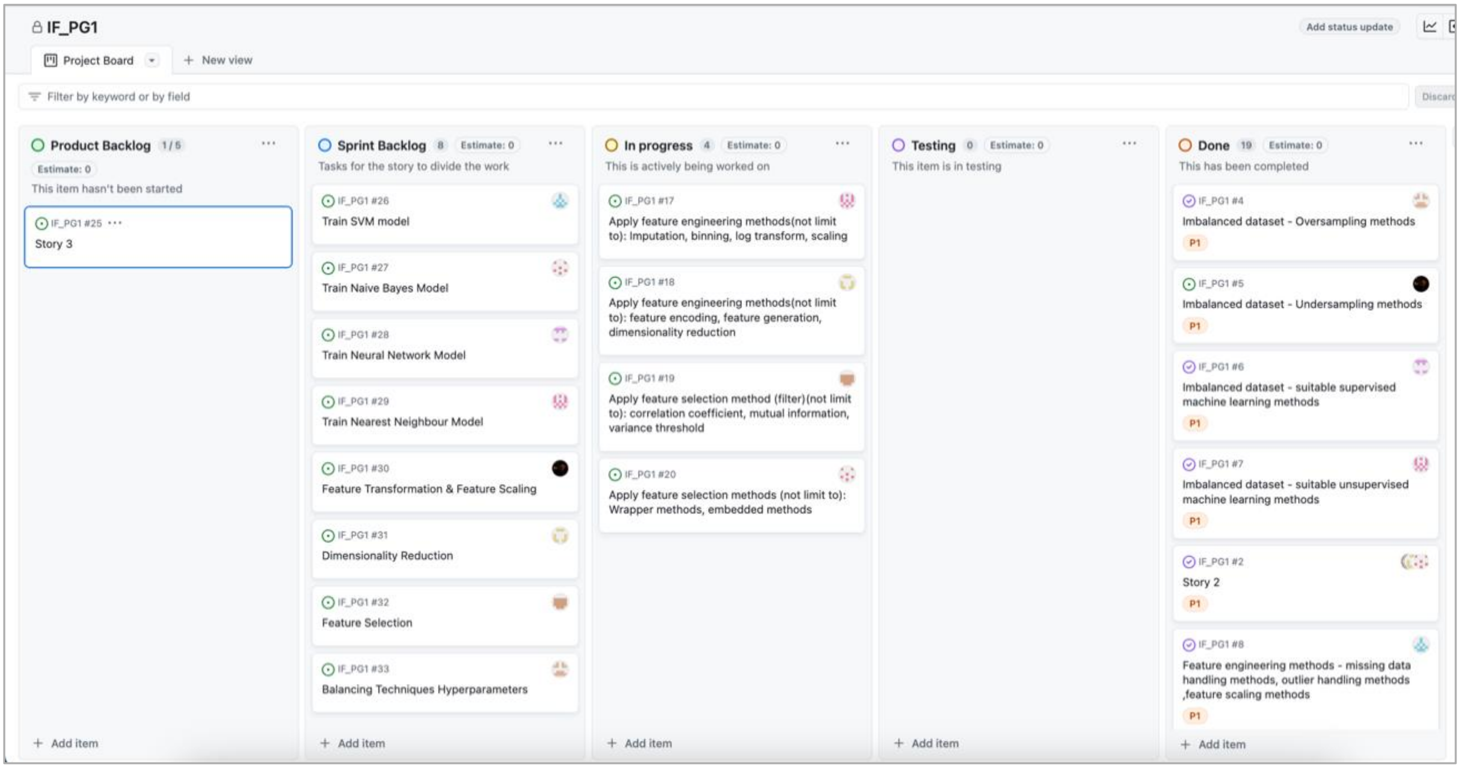
Estimate: 0

This item hasn't been started

IF_PG1 #25 •••

Story 3

Figure 10. The Screenshot of the Product Backlog



IF_PG1

Project Board | + New view | Add status update

Filter by keyword or by field | Discard

Product Backlog 1 / 5 · Estimate: 0

This item hasn't been started

- IF_PG1 #25 — Story 3

Sprint Backlog 8 · Estimate: 0

Tasks for the story to divide the work

- IF_PG1 #26 — Train SVM model
- IF_PG1 #27 — Train Naive Bayes Model
- IF_PG1 #28 — Train Neural Network Model
- IF_PG1 #29 — Train Nearest Neighbour Model
- IF_PG1 #30 — Feature Transformation & Feature Scaling
- IF_PG1 #31 — Dimensionality Reduction
- IF_PG1 #32 — Feature Selection
- IF_PG1 #33 — Balancing Techniques Hyperparameters

In progress 4 · Estimate: 0

This is actively being worked on

- IF_PG1 #17 — Apply feature engineering methods(not limit to): Imputation, binning, log transform, scaling
- IF_PG1 #18 — Apply feature engineering methods(not limit to): feature encoding, feature generation, dimensionality reduction
- IF_PG1 #19 — Apply feature selection method (filter)(not limit to): correlation coefficient, mutual information, variance threshold
- IF_PG1 #20 — Apply feature selection methods (not limit to): Wrapper methods, embedded methods

Testing 0 · Estimate: 0

This item is in testing

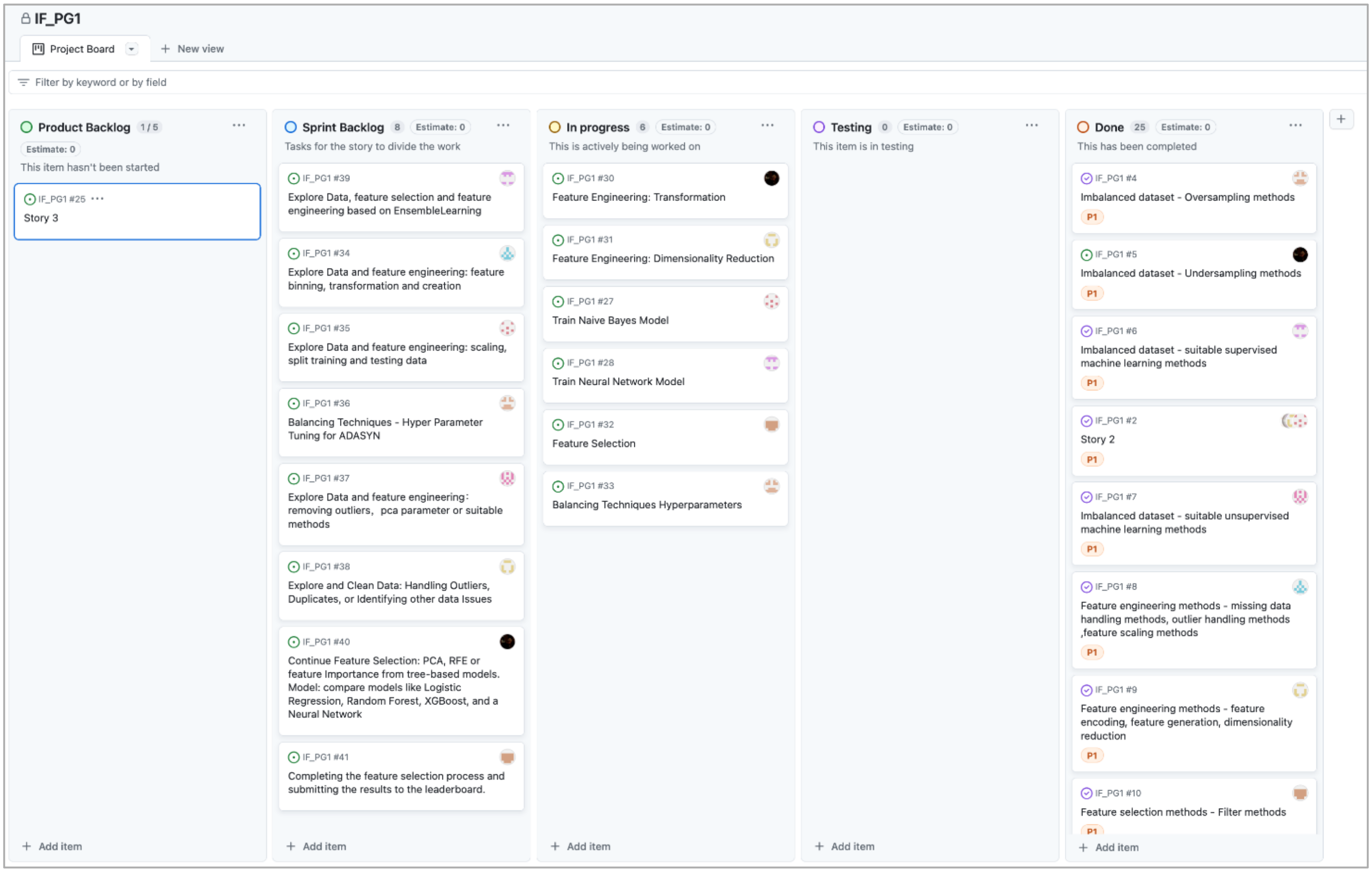
Done 19 · Estimate: 0

This has been completed

- IF_PG1 #4 — Imbalanced dataset - Oversampling methods — P1
- IF_PG1 #5 — Imbalanced dataset - Undersampling methods — P1
- IF_PG1 #6 — Imbalanced dataset - suitable supervised machine learning methods — P1
- IF_PG1 #7 — Imbalanced dataset - suitable unsupervised machine learning methods — P1
- IF_PG1 #2 — Story 2 — P1
- IF_PG1 #8 — Feature engineering methods - missing data handling methods, outlier handling methods ,feature scaling methods — P1

Figure 11. The Screenshot of the Task Board in First Week of Sprint 3

(From Snapshot 3.1)



IF_PG1

Project Board + New view

Filter by keyword or by field

Product Backlog 1 / 5 Estimate: 0

This item hasn't been started

- IF_PG1 #25 — Story 3

+ Add item

Sprint Backlog 8 Estimate: 0

Tasks for the story to divide the work

- IF_PG1 #39 — Explore Data, feature selection and feature engineering based on EnsembleLearning
- IF_PG1 #34 — Explore Data and feature engineering: feature binning, transformation and creation
- IF_PG1 #35 — Explore Data and feature engineering: scaling, split training and testing data
- IF_PG1 #36 — Balancing Techniques - Hyper Parameter Tuning for ADASYN
- IF_PG1 #37 — Explore Data and feature engineering: removing outliers, pca parameter or suitable methods
- IF_PG1 #38 — Explore and Clean Data: Handling Outliers, Duplicates, or Identifying other data Issues
- IF_PG1 #40 — Continue Feature Selection: PCA, RFE or feature Importance from tree-based models. Model: compare models like Logistic Regression, Random Forest, XGBoost, and a Neural Network
- IF_PG1 #41 — Completing the feature selection process and submitting the results to the leaderboard.

+ Add item

In progress 6 Estimate: 0

This is actively being worked on

- IF_PG1 #30 — Feature Engineering: Transformation
- IF_PG1 #31 — Feature Engineering: Dimensionality Reduction
- IF_PG1 #27 — Train Naive Bayes Model
- IF_PG1 #28 — Train Neural Network Model
- IF_PG1 #32 — Feature Selection
- IF_PG1 #33 — Balancing Techniques Hyperparameters

+ Add item

Testing 0 Estimate: 0

This item is in testing

+ Add item

Done 25 Estimate: 0

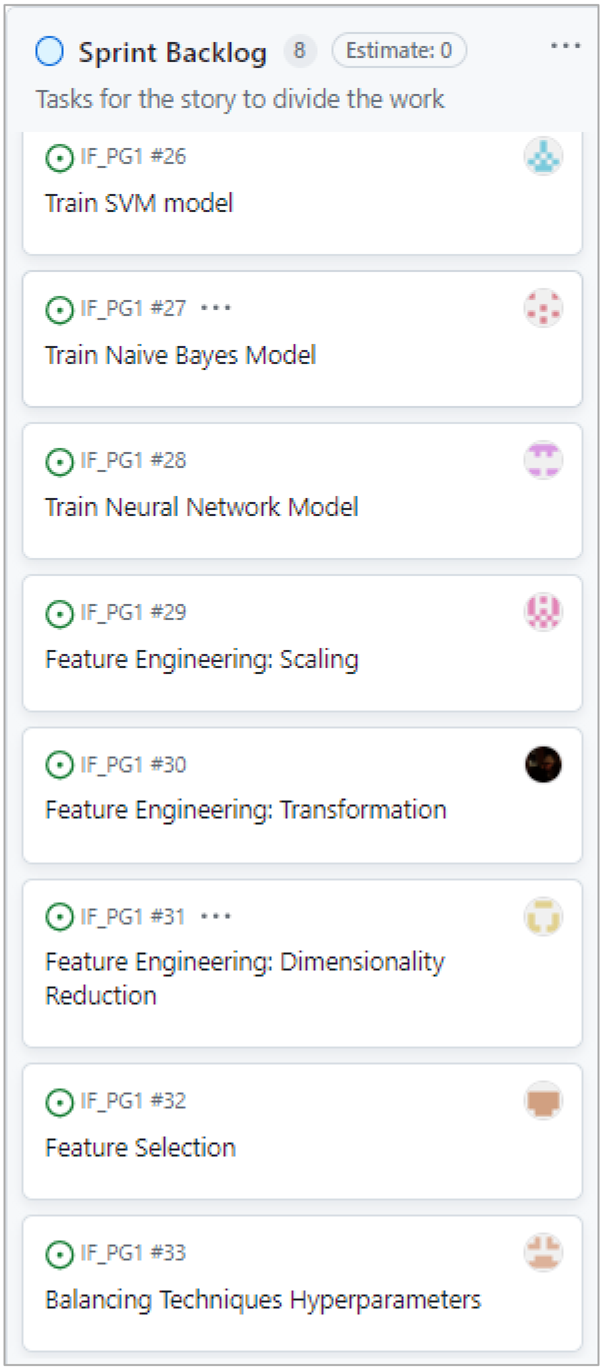
This has been completed

- IF_PG1 #4 — Imbalanced dataset - Oversampling methods — P1
- IF_PG1 #5 — Imbalanced dataset - Undersampling methods — P1
- IF_PG1 #6 — Imbalanced dataset - suitable supervised machine learning methods — P1
- IF_PG1 #2 — Story 2 — P1
- IF_PG1 #7 — Imbalanced dataset - suitable unsupervised machine learning methods — P1
- IF_PG1 #8 — Feature engineering methods - missing data handling methods, outlier handling methods ,feature scaling methods — P1
- IF_PG1 #9 — Feature engineering methods - feature encoding, feature generation, dimensionality reduction — P1
- IF_PG1 #10 — Feature selection methods - Filter methods — P1

+ Add item

Figure 12. The Screenshot of the Task Board in Second Week of Sprint 3 (From Snapshot 3.2)

1.2 Sprint Backlog and User Stories



○ Sprint Backlog 8 Estimate: 0 ···

Tasks for the story to divide the work

IF_PG1 #26
Train SVM model

IF_PG1 #27 ···
Train Naive Bayes Model

IF_PG1 #28
Train Neural Network Model

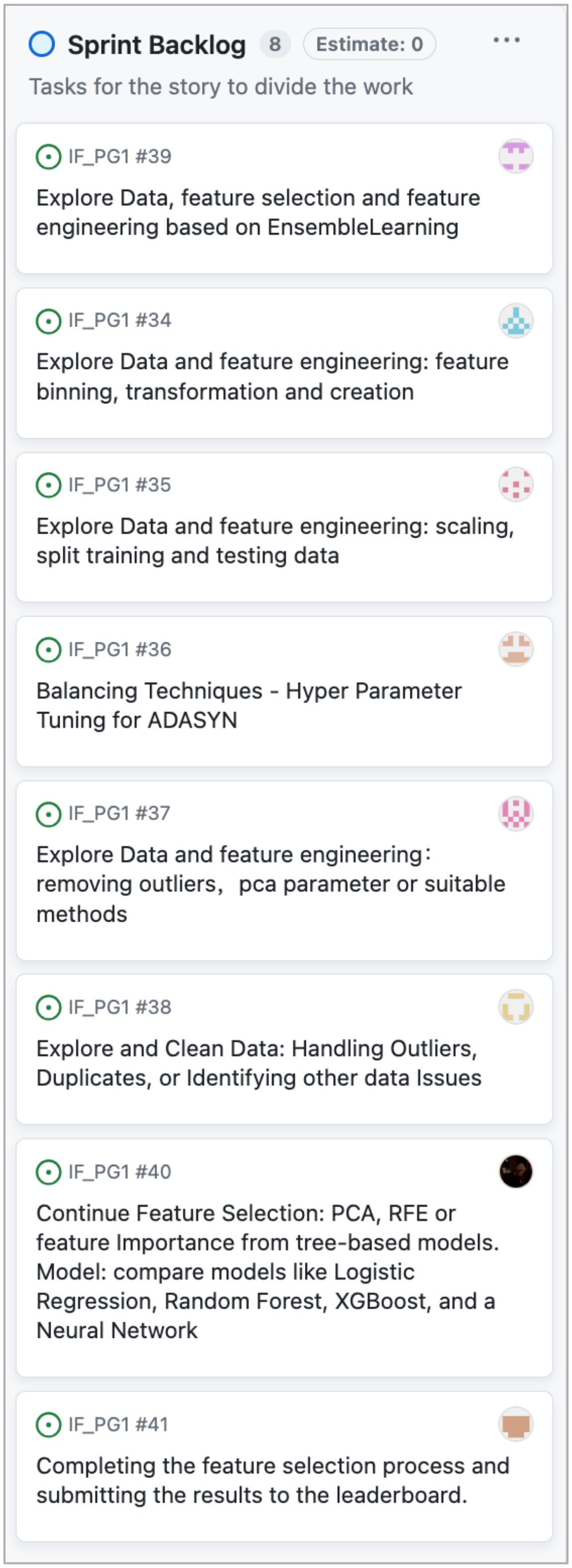
IF_PG1 #29
Feature Engineering: Scaling

IF_PG1 #30
Feature Engineering: Transformation

IF_PG1 #31 ···
Feature Engineering: Dimensionality Reduction

IF_PG1 #32
Feature Selection

IF_PG1 #33
Balancing Techniques Hyperparameters



○ Sprint Backlog 8 Estimate: 0 ···

Tasks for the story to divide the work

IF_PG1 #39
Explore Data, feature selection and feature engineering based on EnsembleLearning

IF_PG1 #34
Explore Data and feature engineering: feature binning, transformation and creation

IF_PG1 #35
Explore Data and feature engineering: scaling, split training and testing data

IF_PG1 #36
Balancing Techniques - Hyper Parameter Tuning for ADASYN

IF_PG1 #37
Explore Data and feature engineering: removing outliers, pca parameter or suitable methods

IF_PG1 #38
Explore and Clean Data: Handling Outliers, Duplicates, or Identifying other data Issues

IF_PG1 #40
Continue Feature Selection: PCA, RFE or feature Importance from tree-based models. Model: compare models like Logistic Regression, Random Forest, XGBoost, and a Neural Network

IF_PG1 #41
Completing the feature selection process and submitting the results to the leaderboard.

Figure 13. The Screenshot of the Sprint Backlog for First Week of Sprint 3.(From snapshot 3.1)

Figure 14. The Screenshot of the Sprint Backlog for Second Week of Sprint 3. (From snapshot 3.2)

1.3 User Story 3

As a software engineer, I want to tryout:

4) **different models while fine-tuning their hyperparameters.**
5) **additional techniques for feature engineering, feature selection, and methods for addressing imbalanced datasets, so that I can improve the test set predictions to achieve an F1 score exceeding 55%.**

Acceptance criteria :

1) **Fine-tune hyperparameters of:**
    - **at least 3 different models**
    - **feature selection, feature engineering, and imbalanced dataset handling techniques, if they have parameters to tune**
2) **Implement and try out at least 2 more techniques each for:**
    - **feature engineering methods**
    - **feature selection methods**
    - **ML techniques to approach a problem with an imbalanced dataset**
3) **Report what combination of techniques and what model worked so far**
4) **Achieve a minimum InsightFactory leaderboard score of 55%**

1.4 Definition of Done (DOD)

Code Standards

7) Readability and Maintainability:
    - Code must be written with clarity.
    - All functions and classes should have clear, concise docstrings explaining their purpose, parameters, and return values.
8) Testing:
    - The entire pipeline must be able to run successfully.
    - The result of the prediction must be reflected on the leaderboard to signify successful submission.

Documentation

1) Reproducibility:
    - Scripts and notebooks must be structured for easy reruns by team members.
2) Data Pipeline Documentation:
    - Document the entire pipeline, including data cleaning, feature engineering/selection, and model decisions.
    - Ensure that all assumptions, processes, and justifications are recorded for future reference.

3) Leaderboard Score Tracking:

- Record leaderboard scores during experimentation to guide model improvements.
- Include dates and versions of models associated with specific scores.

Data Management

5) Data Cleaning:

- Define clear steps for data cleaning (e.g., handling missing values, outliers) and log changes for traceability.
- Validate data after cleaning to ensure no unexpected data loss or errors.

6) Feature Engineering and Selection:

- Document feature engineering and selection processes, with justification for decisions. Regularly evaluate feature importance.

7) Balanced Data Handling:

- Clearly document the rebalancing techniques used, such as SMOTE, under sampling, or ADASYN, along with the rationale for choosing each method.
- Measure and log the performance of models before and after rebalancing to ensure the effectiveness of the technique.

Machine Learning Model

1) Model Selection and Validation:

- All selected models must be validated using appropriate techniques, including cross-validation and/or out-of-sample testing.
- Performance metrics must be calculated for each model and compared against baseline models (e.g., logistic regression).

2) Hyperparameter Tuning:

- Document the hyperparameter tuning process, including which parameters were tuned, the ranges tested, and the final selected values.

3) Model Interpretability:

- If possible, include model interpretability steps (e.g., feature importance, SHAP values) to provide insights into how the model makes predictions.

Review and Sign-Off

5) Peer Review:

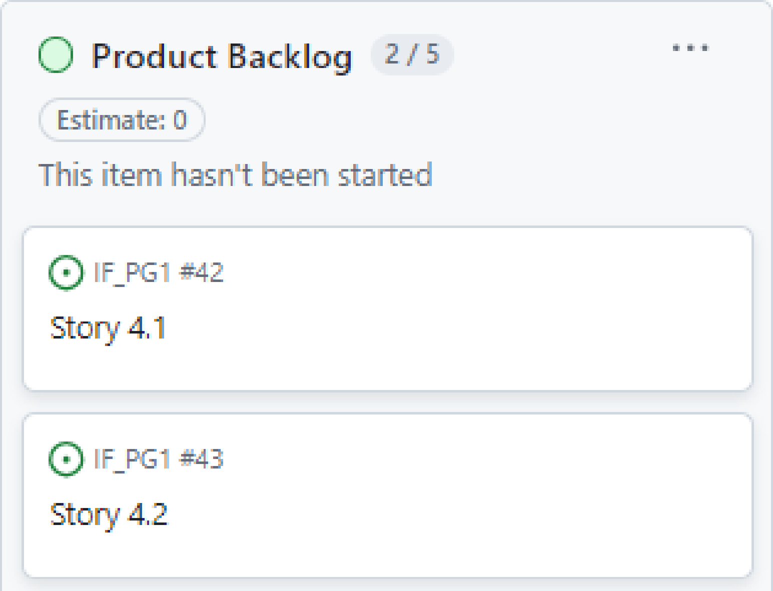
- All code and documentation must undergo peer review before being marked as complete.
- Reviewers must verify that all DoD criteria have been met.

6) Sign-Off:

- The team must collectively sign off on each completed user story, ensuring all acceptance criteria are met and all necessary documentation is provided.

d) Snapshots from Sprint 4

1.1 Product Backlog and Task Board



Product Backlog 2 / 5

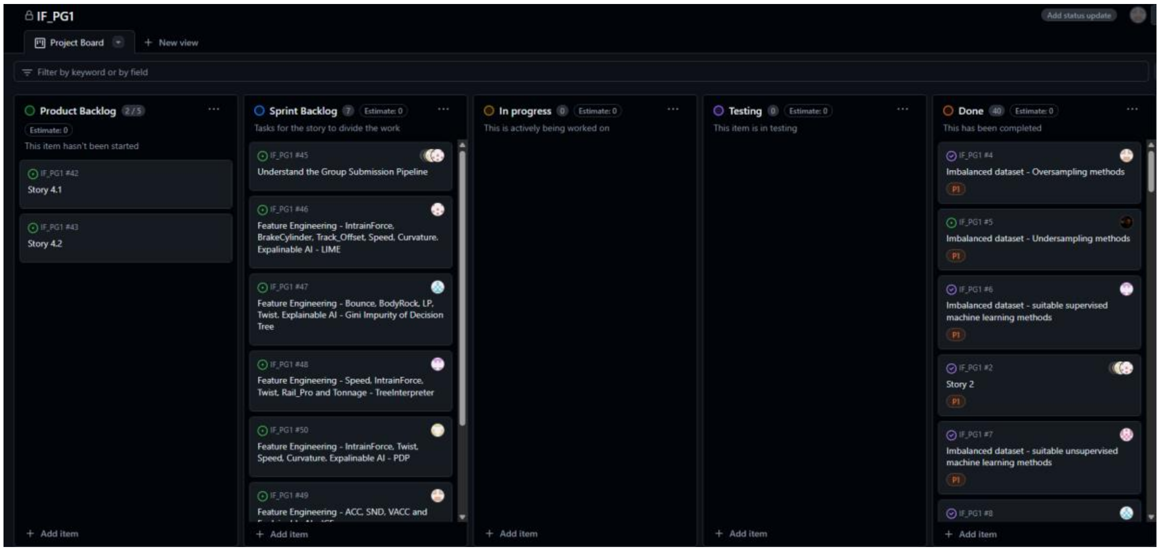
Estimate: 0

This item hasn't been started

IF_PG1 #42
Story 4.1

IF_PG1 #43
Story 4.2

Figure 15: Product Backlog



Product Backlog 2 / 5
Estimate: 0
This item hasn't been started

IF_PG1 #42
Story 4.1

IF_PG1 #43
Story 4.2

Sprint Backlog 7 Estimate: 0
Tasks for the story to divide the work

IF_PG1 #45
Understand the Group Submission Pipeline

IF_PG1 #46
Feature Engineering - IntrainForce, BrakeCylinder, Track_Offset, Speed, Curvature, Expalinable AI - LIME

IF_PG1 #47
Feature Engineering - Bounce, BodyRock, LP, Twist, Explainable AI - Gini Impurity of Decision Tree

IF_PG1 #48
Feature Engineering - Speed, IntrainForce, Twist, Rail_Pro and Tonnage - TreeInterpreter

IF_PG1 #50
Feature Engineering - IntrainForce, Twist, Speed, Curvature, Expalinable AI - PDP

IF_PG1 #49
Feature Engineering - ACC, SND, VACC and

In progress 0 Estimate: 0
This is actively being worked on

Testing 0 Estimate: 0
This item is in testing

Done 40 Estimate: 0
This has been completed

IF_PG1 #4
Imbalanced dataset - Oversampling methods
P1

IF_PG1 #5
Imbalanced dataset - Undersampling methods
P1

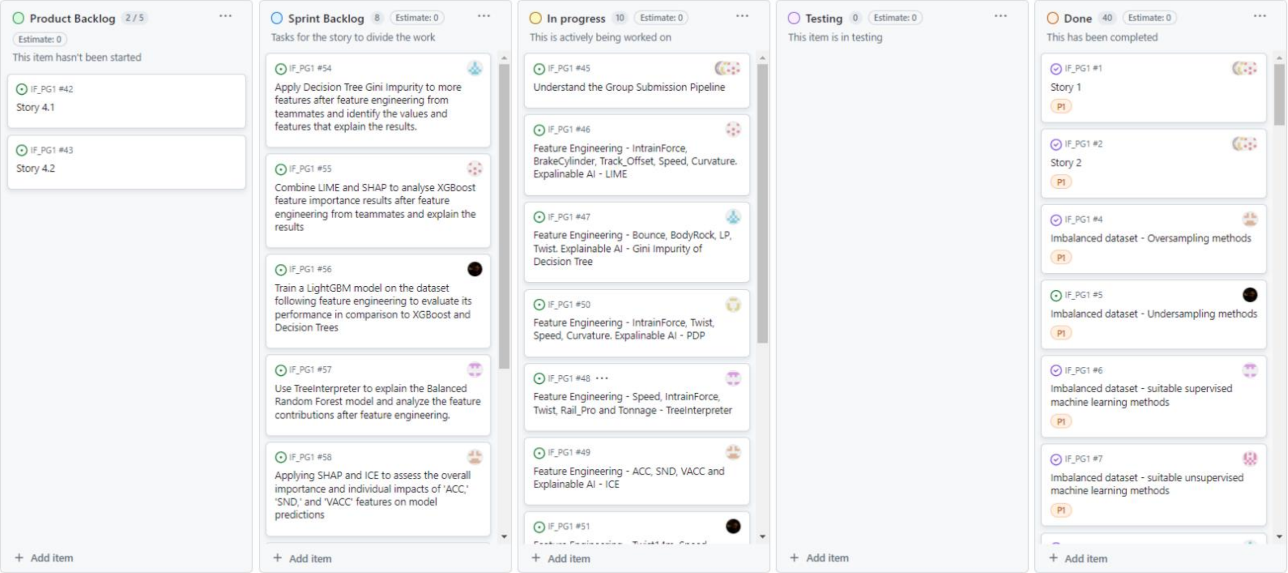
IF_PG1 #6
Imbalanced dataset - suitable supervised machine learning methods
P1

IF_PG1 #2
Story 2
P1

IF_PG1 #7
Imbalanced dataset - suitable unsupervised machine learning methods
P1

IF_PG1 #8

Figure 16: Task Board in First Week of Sprint 4 (From Snapshot 4.1)



Product Backlog 2 / 5 Estimate: 0

This item hasn't been started

IF_PG1 #42
Story 4.1

IF_PG1 #43
Story 4.2

Sprint Backlog 8 Estimate: 0

Tasks for the story to divide the work

IF_PG1 #54
Apply Decision Tree Gini Impurity to more features after feature engineering from teammates and identify the values and features that explain the results.

IF_PG1 #55
Combine LIME and SHAP to analyse XGBoost feature importance results after feature engineering from teammates and explain the results

IF_PG1 #56
Train a LightGBM model on the dataset following feature engineering to evaluate its performance in comparison to XGBoost and Decision Trees

IF_PG1 #57
Use TreeInterpreter to explain the Balanced Random Forest model and analyze the feature contributions after feature engineering.

IF_PG1 #58
Applying SHAP and ICE to assess the overall importance and individual impacts of 'ACC,' 'SND,' and 'VACC' features on model predictions

In progress 10 Estimate: 0

This is actively being worked on

IF_PG1 #45
Understand the Group Submission Pipeline

IF_PG1 #46
Feature Engineering - IntrainForce, BrakeCylinder, Track_Offset, Speed, Curvature. Expalinable AI - LIME

IF_PG1 #47
Feature Engineering - Bounce, BodyRock, LP, Twist. Explainable AI - Gini Impurity of Decision Tree

IF_PG1 #50
Feature Engineering - IntrainForce, Twist, Speed, Curvature. Expalinable AI - PDP

IF_PG1 #48
Feature Engineering - Speed, IntrainForce, Twist, Rail_Pro and Tonnage - TreeInterpreter

IF_PG1 #49
Feature Engineering - ACC, SND, VACC and Explainable AI - ICE

IF_PG1 #51

Testing 0 Estimate: 0

This item is in testing

Done 40 Estimate: 0

This has been completed

IF_PG1 #1
Story 1
P1

IF_PG1 #2
Story 2
P1

IF_PG1 #4
Imbalanced dataset - Oversampling methods
P1

IF_PG1 #5
Imbalanced dataset - Undersampling methods
P1

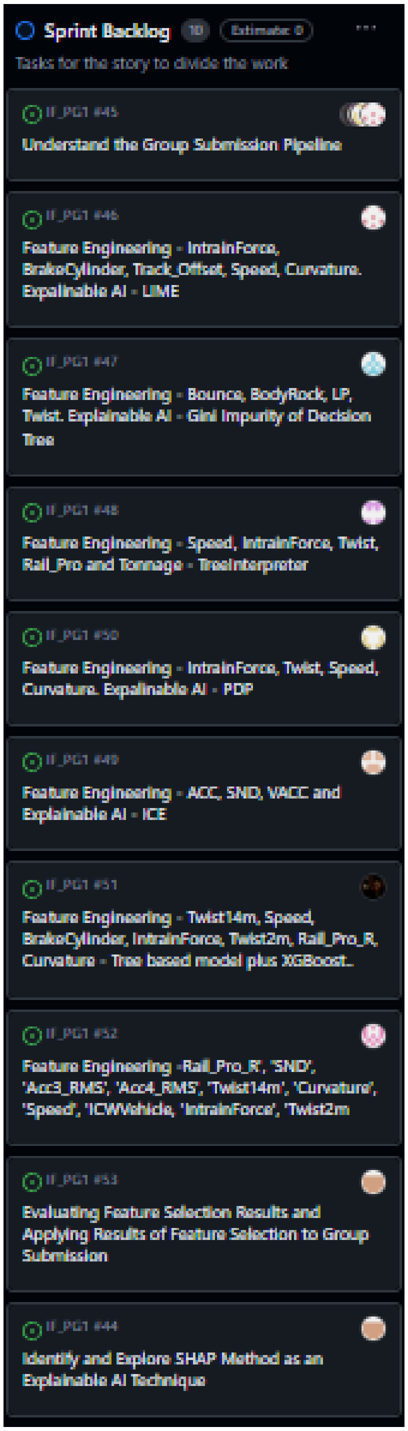
IF_PG1 #6
Imbalanced dataset - suitable supervised machine learning methods
P1

IF_PG1 #7
Imbalanced dataset - suitable unsupervised machine learning methods
P1

Figure 17: The Screenshot of the Task Board in First Week of Sprint 4

(From Snapshot 4.1)

1.4 Sprint Backlog and User Stories



Sprint Backlog 10 Estimate: 0

Tasks for the story to divide the work

IF_PG1 #45

Understand the Group Submission Pipeline

IF_PG1 #46

Feature Engineering - IntrainForce, BrakeCylinder, Track_Offset, Speed, Curvature. Expalinable AI - LIME

IF_PG1 #47

Feature Engineering - Bounce, BodyRock, LP, Twist. Explainable AI - Gini Impurity of Decision Tree

IF_PG1 #48

Feature Engineering - Speed, IntrainForce, Twist, Rail_Pro and Tonnage - TreeInterpreter

IF_PG1 #50

Feature Engineering - IntrainForce, Twist, Speed, Curvature. Expalinable AI - PDP

IF_PG1 #49

Feature Engineering - ACC, SND, VACC and Explainable AI - ICE

IF_PG1 #51

Feature Engineering - Twist14m, Speed, BrakeCylinder, IntrainForce, Twist2m, Rail_Pro_R, Curvature - Tree based model plus XGBoost..

IF_PG1 #52

Feature Engineering -Rail_Pro_R', 'SND', 'Acc3_RMS', 'Acc4_RMS', 'Twist14m', 'Curvature', 'Speed', 'ICWVehicle, 'IntrainForce', 'Twist2m

IF_PG1 #53

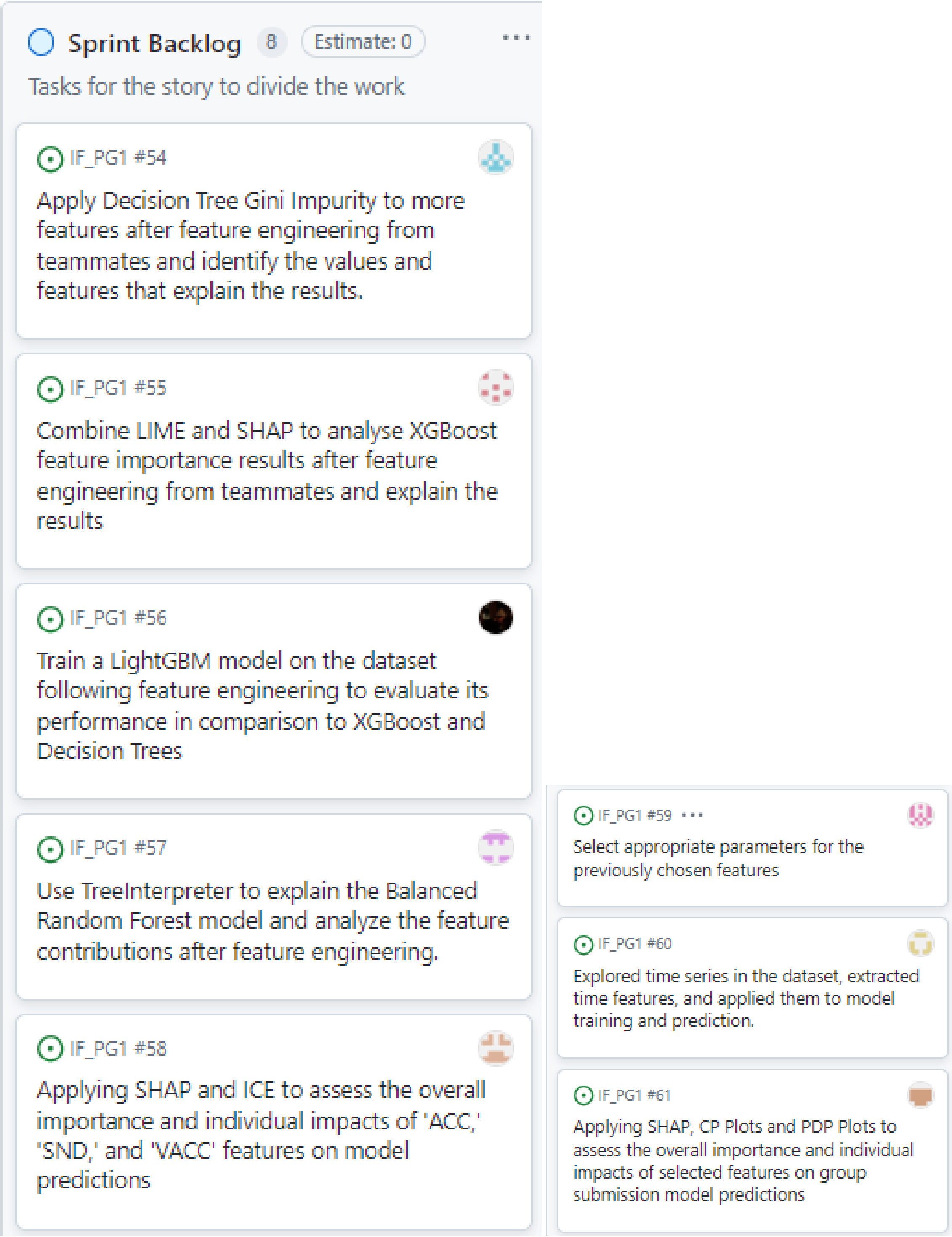
Evaluating Feature Selection Results and Applying Results of Feature Selection to Group Submission

IF_PG1 #44

Identify and Explore SHAP Method as an Explainable AI Technique

Figure 18: The Screenshot of the Sprint Backlog for First Week of Sprint 4

(From snapshot 4.1)



Sprint Backlog 8 Estimate: 0

Tasks for the story to divide the work

IF_PG1 #54

Apply Decision Tree Gini Impurity to more features after feature engineering from teammates and identify the values and features that explain the results.

IF_PG1 #55

Combine LIME and SHAP to analyse XGBoost feature importance results after feature engineering from teammates and explain the results

IF_PG1 #56

Train a LightGBM model on the dataset following feature engineering to evaluate its performance in comparison to XGBoost and Decision Trees

IF_PG1 #57

Use TreeInterpreter to explain the Balanced Random Forest model and analyze the feature contributions after feature engineering.

IF_PG1 #58

Applying SHAP and ICE to assess the overall importance and individual impacts of 'ACC,' 'SND,' and 'VACC' features on model predictions

IF_PG1 #59

Select appropriate parameters for the previously chosen features

IF_PG1 #60

Explored time series in the dataset, extracted time features, and applied them to model training and prediction.

IF_PG1 #61

Applying SHAP, CP Plots and PDP Plots to assess the overall importance and individual impacts of selected features on group submission model predictions

Figure 19: The Screenshot of the Sprint Backlog for Second Week of Sprint 4

1.5 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 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.

1.5 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.

1.6 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.

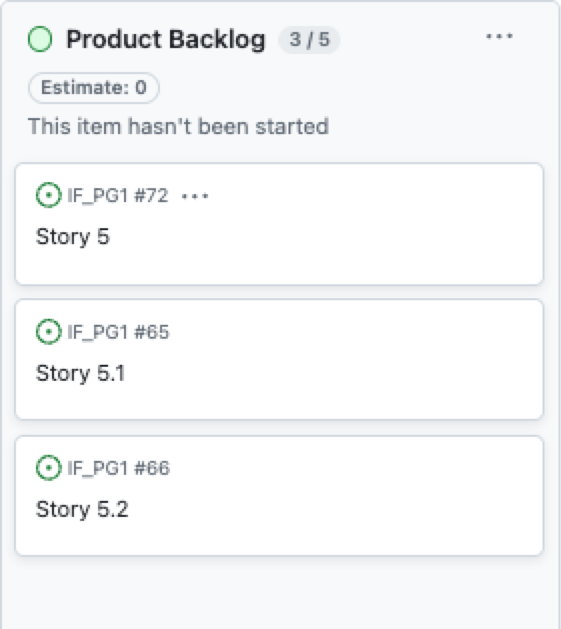
- ### For Week 2 of Sprint 4 (Snapshot 4.2)

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.

e) Snapshots from Sprint 5.1

Product Backlog and Task Board



Product Backlog 3 / 5

Estimate: 0

This item hasn't been started

IF_PG1 #72

Story 5

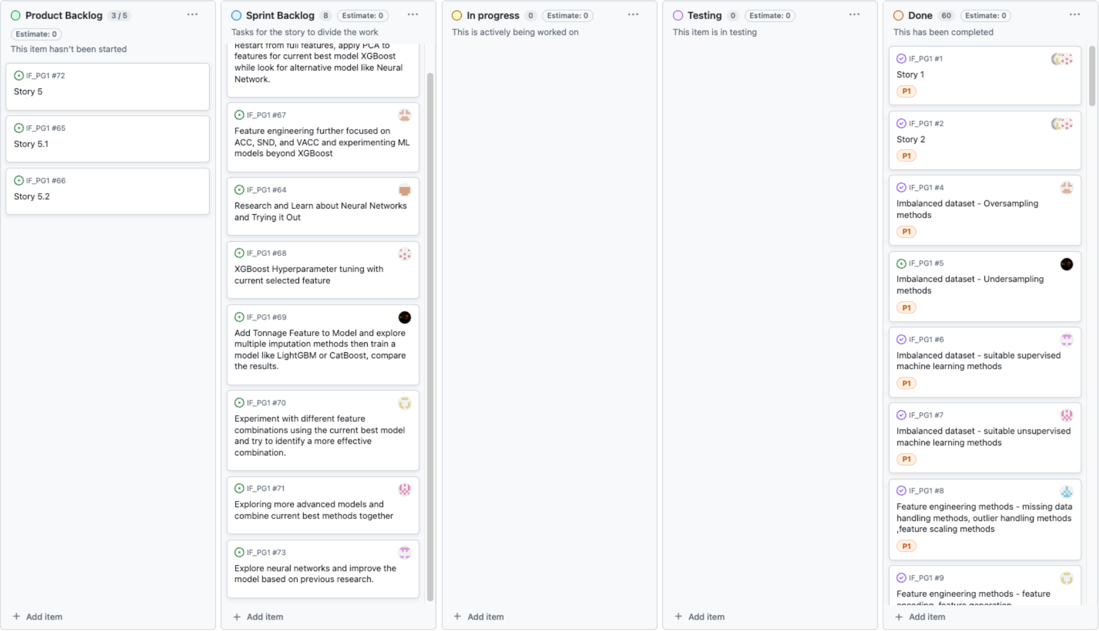
IF_PG1 #65

Story 5.1

IF_PG1 #66

Story 5.2

Figure 20. The Screenshot of the Product Backlog



Product Backlog 3 / 5

Estimate: 0

This item hasn't been started

IF_PG1 #72
Story 5

IF_PG1 #65
Story 5.1

IF_PG1 #66
Story 5.2

Sprint Backlog 8 Estimate: 0

Tasks for the story to divide the work

Restart from full features, apply PCA to features for current best model XGBoost while look for alternative model like Neural Network.

IF_PG1 #67
Feature engineering further focused on ACC, SND, and VACC and experimenting ML models beyond XGBoost

IF_PG1 #64
Research and Learn about Neural Networks and Trying it Out

IF_PG1 #68
XGBoost Hyperparameter tuning with current selected feature

IF_PG1 #69
Add Tonnage Feature to Model and explore multiple imputation methods then train a model like LightGBM or CatBoost, compare the results.

IF_PG1 #70
Experiment with different feature combinations using the current best model and try to identify a more effective combination.

IF_PG1 #71
Exploring more advanced models and combine current best methods together

IF_PG1 #73
Explore neural networks and improve the model based on previous research.

In progress 0 Estimate: 0

This is actively being worked on

Testing 0 Estimate: 0

This item is in testing

Done 60 Estimate: 0

This has been completed

IF_PG1 #1
Story 1
P1

IF_PG1 #2
Story 2
P1

IF_PG1 #4
Imbalanced dataset - Oversampling methods
P1

IF_PG1 #5
Imbalanced dataset - Undersampling methods
P1

IF_PG1 #6
Imbalanced dataset - suitable supervised machine learning methods
P1

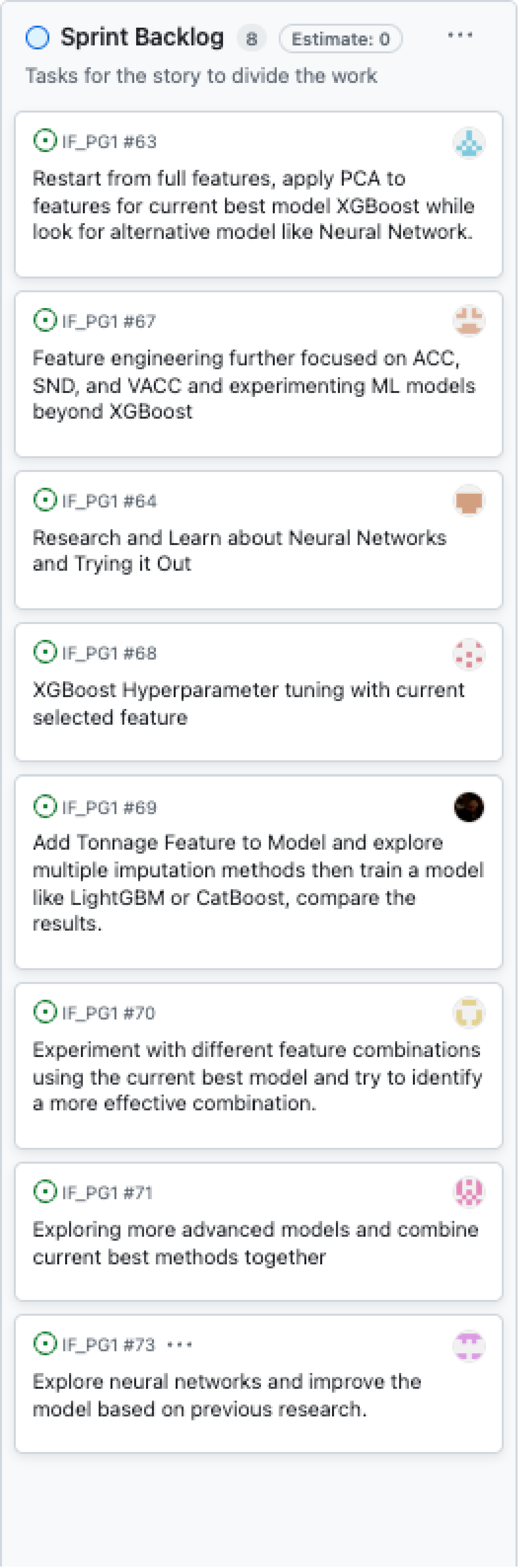
IF_PG1 #7
Imbalanced dataset - suitable unsupervised machine learning methods
P1

IF_PG1 #8
Feature engineering methods - missing data handling methods, outlier handling methods ,feature scaling methods
P1

IF_PG1 #9
Feature engineering methods - feature encoding, feature generation

Figure 21. The Screenshot of the Task Board

3. *Sprint Backlog and User Stories*



○ Sprint Backlog 8 Estimate: 0 ···

Tasks for the story to divide the work

IF_PG1 #63

Restart from full features, apply PCA to features for current best model XGBoost while look for alternative model like Neural Network.

IF_PG1 #67

Feature engineering further focused on ACC, SND, and VACC and experimenting ML models beyond XGBoost

IF_PG1 #64

Research and Learn about Neural Networks and Trying it Out

IF_PG1 #68

XGBoost Hyperparameter tuning with current selected feature

IF_PG1 #69

Add Tonnage Feature to Model and explore multiple imputation methods then train a model like LightGBM or CatBoost, compare the results.

IF_PG1 #70

Experiment with different feature combinations using the current best model and try to identify a more effective combination.

IF_PG1 #71

Exploring more advanced models and combine current best methods together

IF_PG1 #73 ···

Explore neural networks and improve the model based on previous research.

Figure 22. The Screenshot of the Sprint Backlog

User Story 5.1

Description

As a software engineer, I want to identify the key features driving rail breaks within the training data as well as to improve the F1 score on the test set to exceed 70%.

Acceptance criteria

1. **Using the training set,**
    a. **Identify the critical features that increase the risk of rail breaks.**
    b. **Determine the ranges or thresholds of these features that correspond to a higher likelihood of rail failure.**
2. **Continue fine-tuning hyperparameters of the model (and any other technique has), that worked best so far.**
3. **Report what combination of techniques, model and explainability method worked so far.**
4. **Achieve a minimum InsightFactory leaderboard score of 70%**

Notes

The goal of this sprint is twofold: achieve model explainability to understand the risk factors for rail breaks, and also optimize the model for performance on the test set.

You are allowed to make a separate models, one for model exaplainability and one for achiving best F1 score. Because, sometimes the best performing model may not be the best explainable model.

4. Definition of Done (DOD)

Code Standards

9) Readability and Maintainability:
    - Code must be written with clarity.
    - All functions and classes should have clear, concise docstrings explaining their purpose, parameters, and return values.
10) Testing:
    - The entire pipeline must be able to run successfully.
    - The result of the prediction must be reflected on the leaderboard to signify successful submission.

Documentation

4) Reproducibility:
   - Scripts and notebooks must be structured for easy reruns by team members.

5) Data Pipeline Documentation:
   - Document the entire pipeline, including data cleaning, feature engineering/selection, and model decisions.
   - Ensure that all assumptions, processes, and justifications are recorded for future reference.

6) Leaderboard Score Tracking:
   - Record leaderboard scores during experimentation to guide model improvements.
   - Include dates and versions of models associated with specific scores.

7) Group Product Line Documentation:
   - Every team member must contribute to the group product line and document any changes they make.
   - Changes to the group line should be backed by evidence from individual experiments and should include reasons and proof of how the changes improve model performance or stability.

Data Management

8) Data Cleaning:
   - Define clear steps for data cleaning (e.g., handling missing values, outliers) and log changes for traceability.
   - Validate data after cleaning to ensure no unexpected data loss or errors.

9) Feature Engineering and Selection:
   - Document feature engineering and selection processes, with justification for decisions. Regularly evaluate feature importance.

10)Balanced Data Handling:
   - Clearly document the rebalancing techniques used, such as SMOTE, undersampling, or ADASYN, along with the rationale for choosing each method.
   - Measure and log the performance of models before and after rebalancing to ensure the effectiveness of the technique.

Machine Learning Model

4) Model Selection and Validation:
   - All selected models must be validated using appropriate techniques, including cross-validation and/or out-of-sample testing.
   - Performance metrics must be calculated for each model and compared against baseline models (e.g., logistic regression).

5) Hyperparameter Tuning:
   - Document the hyperparameter tuning process, including which parameters

were tuned, the ranges tested, and the final selected values.

6) Model Interpretability:
   - Each model must include interpretability steps (e.g., SHAP values, feature importance plots) to provide insights into how the model makes predictions.
   - Interpretability results should be documented and shared in a central location (e.g., Debug Zone in the group submission line) for all team members to review.

Review and Sign-Off

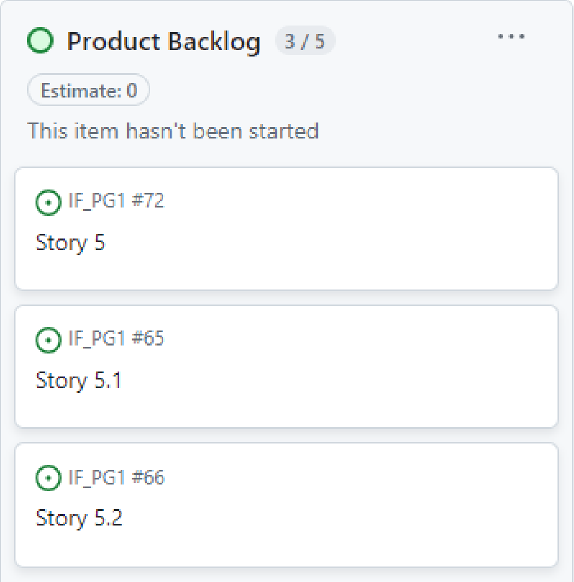
7) Peer Review:
   - All code and documentation must undergo peer review before being marked as complete.
   - Reviewers must verify that all DoD criteria have been met.
8) Sign-Off:
   - The team must collectively sign off on each completed user story, ensuring all acceptance criteria are met and all necessary documentation is provided.

5. Summary of Changes:

No major changes were made for Sprint 5.1. Our team is still experimenting with advanced models, including Neural Networks and alternatives like LightGBM and CatBoost, alongside XGBoost tuning. We are building on research from previous sprints to identify the best combination of approaches, with the ultimate goal of achieving model explainability to understand the risk factors for rail breaks and optimizing the model's performance on the test set.

f) Snapshots from Sprint 5.2

Product Backlog and Task Board



Product Backlog 3 / 5

Estimate: 0

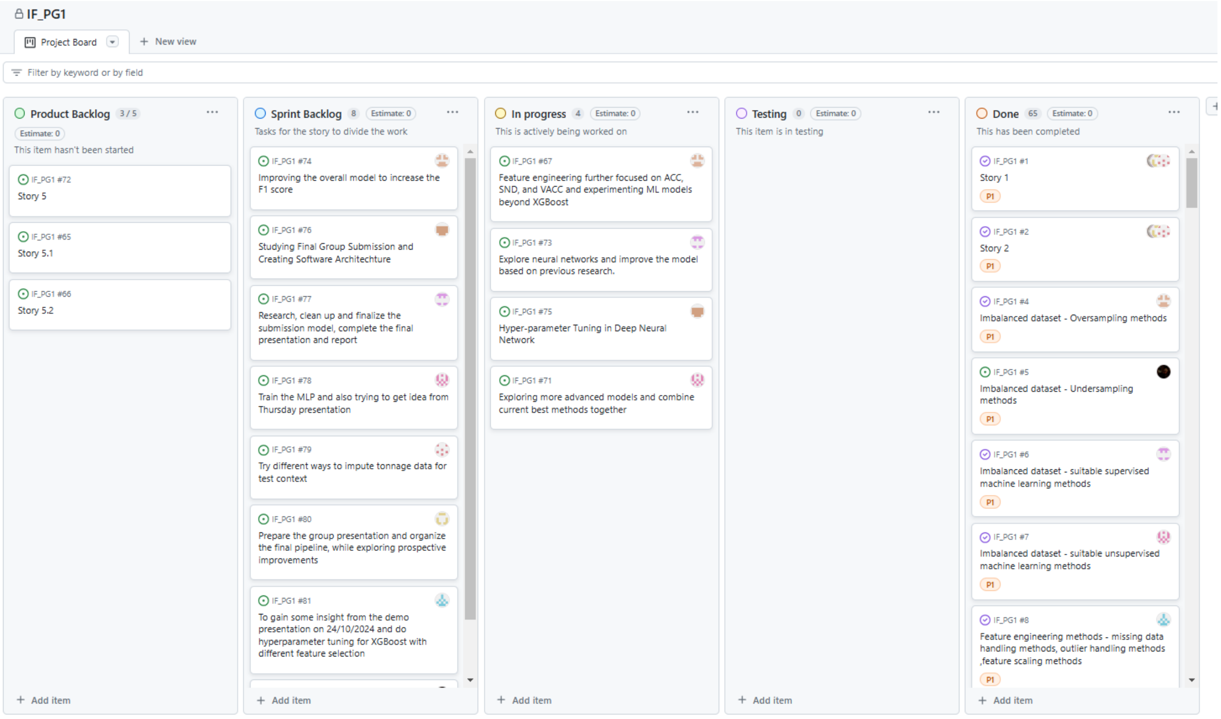
This item hasn't been started

IF_PG1 #72
Story 5

IF_PG1 #65
Story 5.1

IF_PG1 #66
Story 5.2

Figure 23. The Screenshot of the Product Backlog



Product Backlog 3 / 5 — Estimate: 0

This item hasn't been started

- IF_PG1 #72 — Story 5
- IF_PG1 #65 — Story 5.1
- IF_PG1 #66 — Story 5.2

Sprint Backlog 8 — Estimate: 0

Tasks for the story to divide the work

- IF_PG1 #74 — Improving the overall model to increase the F1 score
- IF_PG1 #76 — Studying Final Group Submission and Creating Software Architechture
- IF_PG1 #77 — Research, clean up and finalize the submission model, complete the final presentation and report
- IF_PG1 #78 — Train the MLP and also trying to get idea from Thursday presentation
- IF_PG1 #79 — Try different ways to impute tonnage data for test context
- IF_PG1 #80 — Prepare the group presentation and organize the final pipeline, while exploring prospective improvements
- IF_PG1 #81 — To gain some insight from the demo presentation on 24/10/2024 and do hyperparameter tuning for XGBoost with different feature selection

In progress 4 — Estimate: 0

This is actively being worked on

- IF_PG1 #67 — Feature engineering further focused on ACC, SND, and VACC and experimenting ML models beyond XGBoost
- IF_PG1 #73 — Explore neural networks and improve the model based on previous research.
- IF_PG1 #75 — Hyper-parameter Tuning in Deep Neural Network
- IF_PG1 #71 — Exploring more advanced models and combine current best methods together

Testing 0 — Estimate: 0

This item is in testing

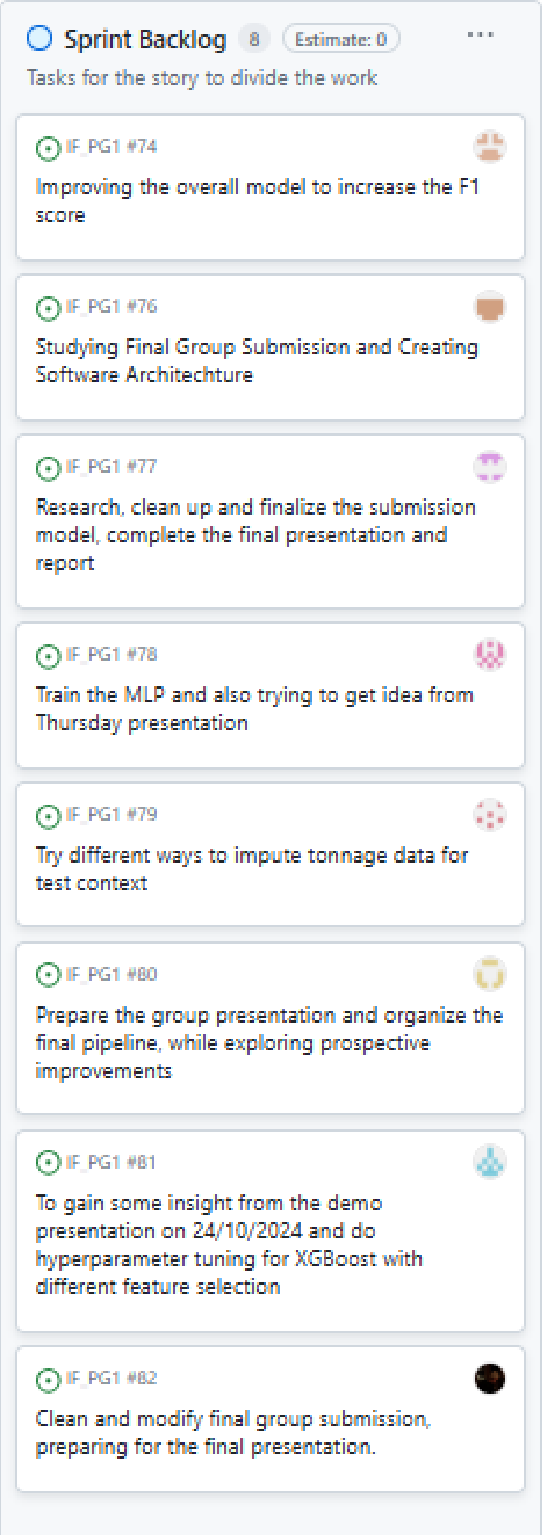
Done 65 — Estimate: 0

This has been completed

- IF_PG1 #1 — Story 1 — P1
- IF_PG1 #2 — Story 2 — P1
- IF_PG1 #4 — Imbalanced dataset - Oversampling methods — P1
- IF_PG1 #5 — Imbalanced dataset - Undersampling methods — P1
- IF_PG1 #6 — Imbalanced dataset - suitable supervised machine learning methods — P1
- IF_PG1 #7 — Imbalanced dataset - suitable unsupervised machine learning methods — P1
- IF_PG1 #8 — Feature engineering methods - missing data handling methods, outlier handling methods ,feature scaling methods — P1

Figure 24. The Screenshot of the Task Board

6. *Sprint Backlog and User Stories*



Sprint Backlog 8 Estimate: 0

Tasks for the story to divide the work

IF_PG1 #74
Improving the overall model to increase the F1 score

IF_PG1 #76
Studying Final Group Submission and Creating Software Architechture

IF_PG1 #77
Research, clean up and finalize the submission model, complete the final presentation and report

IF_PG1 #78
Train the MLP and also trying to get idea from Thursday presentation

IF_PG1 #79
Try different ways to impute tonnage data for test context

IF_PG1 #80
Prepare the group presentation and organize the final pipeline, while exploring prospective improvements

IF_PG1 #81
To gain some insight from the demo presentation on 24/10/2024 and do hyperparameter tuning for XGBoost with different feature selection

IF_PG1 #82
Clean and modify final group submission, preparing for the final presentation.

Figure 25. The Screenshot of the Sprint Backlog

User Story 5

Description

As a software engineer, I want to identify the key features driving rail breaks within the training data as well as to improve the F1 score on the test set to exceed 70%.

Acceptance criteria

5. **Using the training set,**
    - c. **Identify the critical features that increase the risk of rail breaks.**
    - d. **Determine the ranges or thresholds of these features that correspond to a higher likelihood of rail failure.**
6. **Continue fine-tuning hyperparameters of the model (and any other technique has), that worked best so far.**
7. **Report what combination of techniques, model and explainability method worked so far.**
8. **Achieve a minimum InsightFactory leaderboard score of 70%**

Notes

The goal of this sprint is twofold: achieve model explainability to understand the risk factors for rail breaks, and also optimize the model for performance on the test set.

You are allowed to make a separate models, one for model exaplainability and one for achiving best F1 score. Because, sometimes the best performing model may not be the best explainable model.

7. Definition of Done (DOD)

Code Standards

11) Readability and Maintainability:
    - Code must be written with clarity.
    - All functions and classes should have clear, concise docstrings explaining their purpose, parameters, and return values.
12) Testing:
    - The entire pipeline must be able to run successfully.
    - The result of the prediction must be reflected on the leaderboard to signify successful submission.

Documentation

8) Reproducibility:
   - Scripts and notebooks must be structured for easy reruns by team members.

9) Data Pipeline Documentation:
   - Document the entire pipeline, including data cleaning, feature engineering/selection, and model decisions.
   - Ensure that all assumptions, processes, and justifications are recorded for future reference.

10) Leaderboard Score Tracking:
   - Record leaderboard scores during experimentation to guide model improvements.
   - Include dates and versions of models associated with specific scores.

11) Group Product Line Documentation:
   - As tasks are assigned, the specific team members responsible for the group submission can contribute to the group product line and document any changes they make.
   - Changes to the group line should be backed by evidence from individual experiments and should include reasons for and proof of how the changes improve model performance or stability.

Data Management

11)Data Cleaning:
   - Define clear steps for data cleaning (e.g., handling missing values, outliers) and log changes for traceability.
   - Validate data after cleaning to ensure no unexpected data loss or errors.

12)Feature Engineering and Selection:
   - Document feature engineering and selection processes, with justification for decisions. Regularly evaluate feature importance.

13)Balanced Data Handling:
   - Clearly document the rebalancing techniques used, such as SMOTE, undersampling, or ADASYN, along with the rationale for choosing each method.
   - Measure and log the performance of models before and after rebalancing to ensure the effectiveness of the technique.

Machine Learning Model

7) Model Selection and Validation:
   - All selected models must be validated using appropriate techniques, including cross-validation and/or out-of-sample testing.
   - Performance metrics must be calculated for each model and compared against baseline models (e.g., logistic regression).

8) Hyperparameter Tuning:

- Document the hyperparameter tuning process, including which parameters were tuned, the ranges tested, and the final selected values.

9) Model Interpretability:
   - Each model must include interpretability steps (e.g., SHAP values, feature importance plots) to provide insights into how the model makes predictions.
   - Interpretability results should be documented and shared in a central location (e.g., Debug Zone in the group submission line) for all team members to review.

Review and Sign-Off

9) Peer Review:
   - All code and documentation must undergo peer review before being marked as complete.
   - Reviewers must verify that all DoD criteria have been met.

10) Sign-Off:
   - The team must collectively sign off on each completed user story, ensuring all acceptance criteria are met and all necessary documentation is provided.

8. Summary of Changes:

For Sprint 5.2, we will be focusing on finalizing and improving the model, while cleaning and optimizing the group submission pipeline. In addition to improving the current model, we will be conducting experiments with deep neural networks (DNNs) to evaluate their potential for further enhancing the F1 score. This sprint marks the conclusion of our project, with the final submission prepared, ensuring both performance and explainability are balanced.