
MDAIE – Production-Ready Machine Learning

**Project Report**

# Public Bike Availability Prediction System


| | | | |
|---|---|---|---|
| **Teammates** | Ang Kar Kin | Nicolas Lasch | Chan Kok Siang |
| **Student ID** | 1010951 | 1011135 | 1010722 |
| **Instructor** | Pritee Agrawal | | |
| **Due Date** | 10 December 2025 | | |
| **Institute** | Singapore University of Technology and Design (SUTD) | | |

# Table of Contents

# 1 Executive Summary

## 1.1 The Challenge: Urban Mobility and Logistical Imbalance

Public bike-sharing systems face a critical failure mode known as the **Rebalancing Problem**. As commuters move in synchronised tidal waves – travelling from residential zones to business districts – the network becomes polarised. High-demand stations run empty while destination stations overflow. This logistical inefficiency leads to lost revenue, high operational costs from reactive manual redistribution, and significant user frustration.

## 1.2 Project Objective and Case Study

The objective of this project is to develop a **Machine Learning (ML) solution** to forecast station-level demand, enabling operators to transition from reactive to **proactive rebalancing**. We utilised the **Bergen Bike Sharing Dataset (2023)** as a case study, leveraging over a year of granular trip data and localised weather conditions to validate our approach.

## 1.3 Methodological Framework

We approached this as a regression problem, integrating trip logs with external weather metrics (precipitation, temperature). Key methodological steps included:

- **Feature Engineering:** Creating "Lag Features" (historical demand from 1 hour/24 hours prior) and "Rolling Windows" to capture recent trends.
- **Model Selection:** We trained and tuned **Random Forest** and **XGBoost** models, selected for their robustness in handling non-linear relationships.

## 1.4 Key Findings and Results

Evaluation on unseen test data yielded the following insights:

- **Model Superiority:** The **XGBoost** model achieved the lowest Root Mean Squared Error (RMSE), proving highly effective for this tabular dataset.
- **Drivers of Demand:** "Lagged Usage" (past activity) was the strongest predictor of future demand. Additionally, weather acts as a quantifiable "demand dampener", with rain significantly reducing ridership.

## 1.5 Strategic Recommendations

Based on these findings, we propose a data-driven operational shift:

- **Predictive Logistics:** Rebalancing trucks should be deployed based on the model's 2-hour forecast rather than current station status.
- **Dynamic Resource Allocation:** Staffing levels should be adjusted dynamically based on the weather-integrated demand forecast, reducing labour costs on rainy days.

# 2 Background and Introduction


*Figure 1: The Daily Rebalancing Cycle*

## 2.1 The Bergen Context (Bergen Bysykkel)

Bergen which is Norway's second-largest city, presents a unique environment for micromobility due to its complex topography and famously variable weather (Share North, 2020; Interreg North Sea, 2023). The city's bike-sharing scheme, Bergen Bysykkel, was relaunched in its current form in 2018 by the operator Urban Sharing. Unlike traditional systems that rely solely on fixed heavy infrastructure, Bergen's system utilizes flexible technology, allowing for both physical docking stations and "virtual" stations which are geofenced areas where bikes can be parked securely even when physical docks are full.

As of 2023, the system serves a significant portion of the city's residents. A notable development in 2023 was the expansion of the network to include Fyllingsdalen, following the April 15th opening of the Fyllingsdalstunnelen which is the world's longest purpose-build pedestrian bicycle tunnel (2.9km) (Bicycle Network, 2023; Global Construction review, 2023). This infrastructure investment highlights the city's commitment to cycling, making the 2023 dataset particularly significant for analysing the impact of new infrastructure on cyclist behaviours.

## 2.2 A Global Logistics Issue

In the modern "Smart City", bike-sharing is essential. However, the utility of these systems is often limited by reliability. If a user walks to a station and finds no bikes, they lose trust in the service.

The core problem is that bike usage is **asymmetric**. Users do not return bikes to where they found them; they move them from "Source" areas (the origin, an area where more people take bikes than return them) to "Sink" areas (the destination, an area where more people return bikes than take them). Without intervention, the system eventually locks up. Traditional solutions involve operators driving trucks around the city to manually move bikes, but this is often done based on intuition or static schedules, which is inefficient and costly.

## 2.3 Business Value

Solving this problem with AI offers three tangible benefits:

- **Operational Efficiency:** Drivers follow an optimised route to stations that *will* be empty, rather than driving randomly.
- **Cost Optimisation:** Reduces fuel consumption and overtime pay by planning rebalancing shifts more accurately.
- **User Retention:** A reliable system retains users. If a bike is always available when predicted, customer lifetime value increases.

## 2.4 Problem Statement

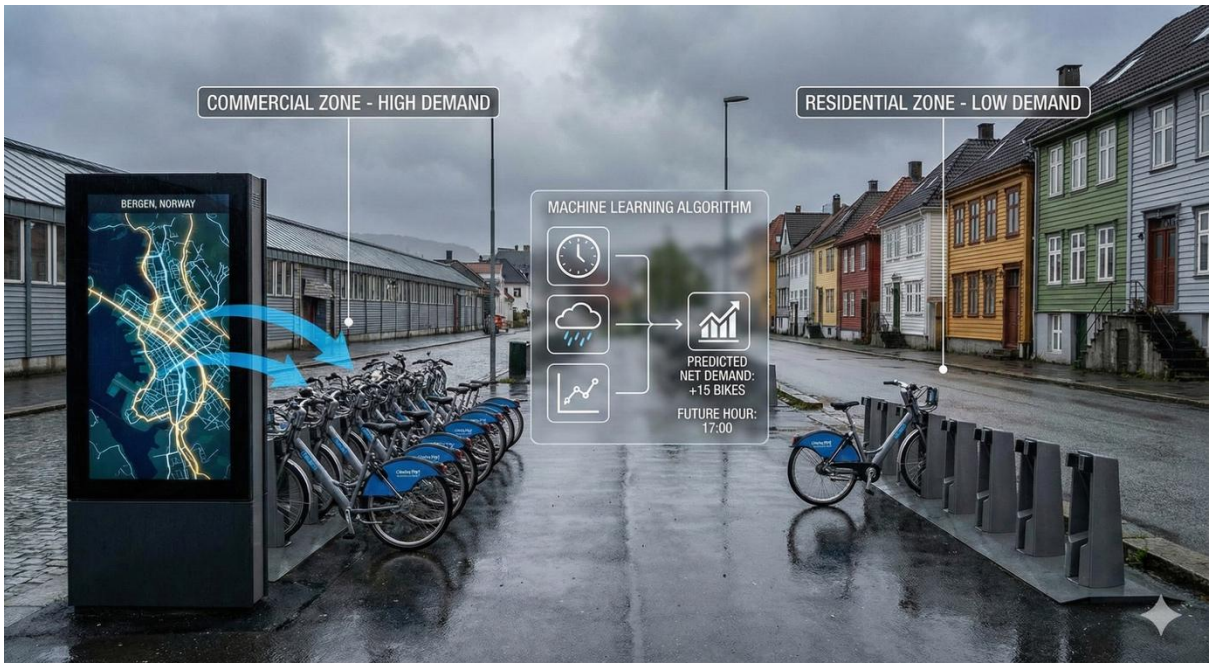


*Figure 2: Balancing bike availability in a modern city with dynamic needs*

Operational efficiency in bike-sharing ecosystems is frequently bottlenecked by the **Station Imbalance Problem** which is the misalignment between bike availability and user demand. Inaccurate demand forecasting will lead to lost revenue during peak hours and increased operational costs for manual redistribution fleets.

The objective of this project is to solve the **Station Imbalance Problem** inherent in docked bike-sharing systems. Specifically, we aim to build a Machine Learning model capable of predicting the **net demand (trip count)** for any station in the network for a specific future hour.

**The Specific Case Study: Bergen Bysykkel Network (Bergen, Norway)**

To demonstrate the efficacy of this approach, we applied our methodology to the city of **Bergen**.

- **Why Bergen?** The city offers a challenging environment with significant weather variability (frequent rain) and distinct geographical zones (residential vs. commercial), making it an ideal "stress test" for our models.
- **Dataset:** We utilised the bergen_merged.csv dataset (refer dataset in Project References under Appendix), covering operations from Jan 1, 2023, to Dec 31, 2023.
- **Target Variable:** trip_count (The number of bikes rented from a specific station within a 1-hour window).

By analysing the 2023 Bergen Bysykkel dataset, this project aims to quantify how distinct geographical zones and climate conditions influence station-level demand, ultimately producing a robust predictive model capable of functioning in adverse conditions.

# 3 Related Work

In the domain of demand forecasting, there are two primary approaches. We chose the latter (Machine Learning) to overcome the specific limitations of traditional statistical methods.

## 3.1 The Traditional Approach: ARIMA

**ARIMA** (AutoRegressive Integrated Moving Average) is the standard statistical method for time-series forecasting. It functions by modelling the relationship between an observation and a specified number of lagged observations (AutoRegressive) and residual errors (Moving Average).

- **Mechanism:** ARIMA projects future values primarily based on the momentum, trend and seasonality observed in the target variable's history.
- **The Limitation:** Standard ARIMA models are typically **univariate**, which means they rely mostly on the target variable's own past (Indicio, 2024). It struggles to incorporate external "side information". For example, if a massive storm hits on a Tuesday, ARIMA only sees "It's Tuesday, usually demand is high", and will likely over-predict.

## 3.2 Our Approach: ML Regressors (Random Forest / XGBoost)

To overcome these issues, we utilized **Multivariate ML Regressors** (specifically Random Forest and XGBoost). Unlike time-series models that extrapolate a single curve, these algorithms treat forecasting as a supervised regression problem, ingesting a high-dimensional matrix of features to optimize predictions.

- **Mechanism:** These ensemble models aggregate insights from multiple "decision trees", allowing them to learn complex non-linear dependencies between the target variable (trip_count) and various external factors (weather, time of day, station attributes).
- **Advantage:** They demonstrate the ability to integrate "side information" such as precipitation levels or public holiday flags directly into the decision boundary. Research indicates that XGBoost and Random Forest consistently outperform traditional linear models in bike-sharing contexts due to this capacity to map non-linear thresholds, such as demand drops only when rain exceeds a specific mm/hour (Atlantis Press, 2022)

**Comparison: Why ML is Better for Bike Sharing**

The table below (Table 1) summarises why we moved away from ARIMA and chose Machine Learning Regressors.

| Feature | Traditional ARIMA | Our Approach (ML Regressors) |
|---|---|---|
| **Data Inputs** | **History only** (past trip counts). | **History + Context** (past trips + rain + temperature + holidays). |
| **Flexibility** | Assumes linear relationships. | Can handle **non-linear** relationships (e.g., demand rises with temp but drops if too hot). |
| **Complex Events** | Struggles with sudden changes (e.g., public holidays). | Uses flags such as “is_holiday" to instantly adapt predictions. |
| **Scalability** | Must often fit a separate model for every station. | One global model can learn patterns for all stations (with Station ID as a feature). |

*Table 1: Comparison between Traditional ARIMA and our chosen approach (ML Regressors)*

Given that bike usage is heavily dependent on weather (rain) and calendar events (weekends), the **ML Regressor approach is superior** because it treats these external factors as important inputs, rather than ignoring them.

# 4 Methodology

The following diagram (Figure 3) outlines the end-to-end pipeline used to process the Bergen data and train our predictive models.

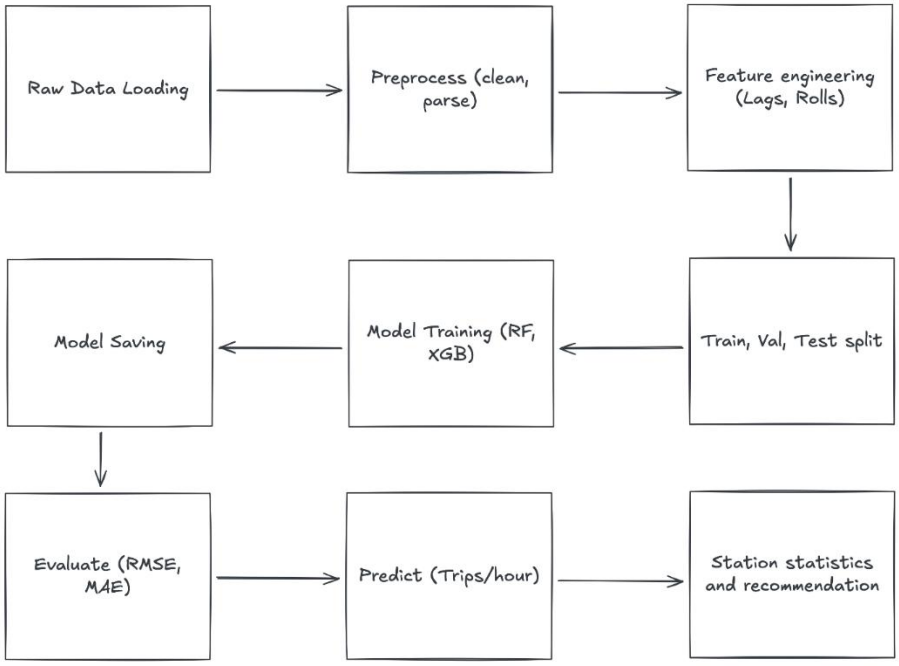


*Figure 3: End-to-end pipeline for processing raw data and training models*

## 4.1 Data Collection & Ethics

- **Source & Scope:** The project uses the **2023 Bergen Bike Sharing Dataset**, sourcing via Kaggle (refer Appendix for Data Dictionary). The dataset provides a record of all system trips, merged with hourly meteorological data for the Bergen municipality.
- **Ethics & Privacy:** The dataset is strictly anonymous. It aggregates movement patterns without tracking individual user identities, ensuring compliance with privacy standards (GDPR).

## 4.2 Exploratory Data Analysis (EDA)

Before modelling, we analysed the Bergen data to understand the "ground truth":

- **Seasonal Trends:** Ridership is highly seasonal, with demand peaking in summer months (around June – August) and dropping in winter, which is consistent with Bergen's latitude (Figure 4).
- **Commuter Patterns:** Weekday usage follows a distinct bimodal distribution ("M-shape"), characterized by sharp spikes at **8:00** (work commute) and **16:00** (home commute) (Figure 5).
- **Weather Impact:** A negative / low correlation was observed between precipitation intensity and trip volume (Figure 6). Heavy rainfall disproportionately suppresses casual/leisure ridership, while commuter demand remains comparatively inelastic.

## 4.3 Feature Engineering

Raw data alone is insufficient for high accuracy. We engineered specific features to give the model "context":

- **Temporal Features:** We decomposed timestamps into categorical features (Hour, Day_of_Week, and Month). We also introduced binary flags for "is_holiday" and "is_weekend" to help the model distinguish between a Tuesday workday and a Sunday rest day.
- **Lag Features (The "Memory"):** We created columns representing the number of trips that occurred **1 hour ago** and **24 hours ago**. This provides the model with "short-term memory", allowing to adjust predictions based on immediate utilization trends (e.g., "The station is getting busy right now").
- **Rolling Stats:** We calculated a 3-hour rolling average of trip counts. This feature acts as a smoothing function, helping the model distinguish between genuine demand trends and random, isolated spikes in usage.

## 4.4 Train-Validation-Test Split Strategy

To ensure robust model evaluation and prevent temporal leakage, we implemented time-based chronological split preserving the natural temporal ordering of hourly observations (train 60%, validation 20%, test 20%).

## 4.5 Modelling Strategy

We implemented two competing algorithms:

1. **Random Forest Regressor:** A robust "ensemble" model that averages the decisions of hundreds of decision trees. It is highly resistant to overfitting and handles non-linear relationships (like the curve of temperature vs. demand) well.
2. **XGBoost (Extreme Gradient Boosting):** A state-of-the-art model that builds trees sequentially, where each new tree focuses on correcting the specific errors made by the previous tree. It is widely considered the industry standard for tabular prediction tasks.

**Baseline vs. Tuned Models:** Both algorithms were trained first with default hyperparameters (baseline, 100 trees/n_estimators), then optimized via grid search to identify best configurations (n_estimators of 50, 100, different max_depths and learning_rate). Manual grid search is used for hyperparameter tuning over validation set (20% split) to prevent test set contamination.

Refer to the Appendix section for the code execution instructions.

# 5 Evaluation Results

## 5.1 Prediction Step

After fitting each tuned model on combined training and validation data, predictions are generated on the test split using model's built in predict function (y_test = best_model.predict(X_test)). The residual plot (Figure 8) quantifies the model error at each hour by comparing predictions to the actual trip counts in the test set (residuals = difference in y_true and y_pred). It consists of a scatter plot of residuals versus predicted values (plot_residuals function in src/visual_analysis.py)

## 5.2 Quantitative Results (Baseline vs. Tuned)

**Performance Metrics**

- **RMSE (Root Mean Squared Error):** This metric penalises large errors heavily. It is useful for ensuring we do not have massive "misses" (e.g., predicting 0 bikes when 50 are needed).
- **MAE (Mean Absolute Error):** The average difference between prediction and reality. It is more interpretable for business stakeholders (e.g., "We are usually off by 2 bikes").
- **$R^2$ score:** This is the fraction of variance, where values >0.99 indicate exceptional predictive power, explaining virtually all demand variations.

The table below (Table 2) compares the performance of our baseline model against the tuned versions. Both algorithms demonstrated exceptional predictive power, with the **Tuned XGBoost** model emerging as the clear leader. (Refer Figure 9 for Visualized Model Comparison results)

| Model | RMSE (lower is better) | MAE (lower is better) | $R^2$ score (higher is better) | Performance Notes |
|---|---|---|---|---|
| **Random Forest** | 0.0689 | 0.0052 | 0.9970 | **Strong Baseline:** Excellent accuracy out of the box, explaining >99% of variance. |
| **Random Forest (Tuned)** | 0.0686 | 0.0052 | 0.9971 | **Marginal Gain:** Turning yielded negligible improvement, suggesting the default Random Forest was already near optimal. |
| **XGBoost** | 0.0746 | 0.0284 | 0.9965 | **Underperformer**: Initially lagged behind Random Forest with high error rates. |
| **XGBoost (Tuned)** | 0.0520 | 0.0065 | 0.9983 | **Best Performer:** Hyperparameter tuning drastically improved RMSE (by ~30%), making it the most precise model for handling outliers. |

*Table 2: Comparison of error metrics between Random Forest and XGBoost (Baseline & Tuned)*

## 5.2 Interpretation & Bias Audit

- **Feature Importance:** Our analysis confirmed that **Lag Features** (Past Usage) were the #1 predictor, whereas **Temperature** was the #2 predictor (Figure 7).
- **Exceptional Accuracy ($R^2 > 0.99$):** The extremely high $R^2$ score across all models indicate that the "Lag Features: (utilizing the previous hour's trip count to predict the next hour) are highly effective. Essentially, the system has strong "inertia". For example, if a station is busy now, it is highly likely to be busy in the next hour.
- **XGBoost Tuning Success:** While the baseline XGBoost performed worse than Random Forest, the **Tuned XGBoost** overtook all other models with the lowest **RMSE (0.052).** This demonstrates that while XGBoost is sensitive to hyperparameter settings, it offers a higher "performance calling" when properly optimized.
- **MAE vs. RMSE Trade-off:** Interestingly, Random Forest maintained a slightly lower MAE (0.052) compared to Tuned XGBoost (0.0065), meaning it is slightly better at "median" behaviour. However, the tuned XGBoost's significantly lower RMSE (0.052) than the tuned Random Forest model (RMSE 0.068) proves that it is far superior at handling extreme spikes (outliers), which is critical for preventing empty stations during rush hour.
- **Bias Check:**
    - *Geographical Bias:* The model is most accurate in the city centre where data is abundant. Predictions for remote stations (with few trips) may be less reliable.
    - *Temporal Bias:* The model assumes 2024 and subsequent years will behave like 2023. Significant changes in city infrastructure or climate could lead to "Data Drift".

## 5.3 Model failure analysis

Although overall performance is strong ($R^2 > 0.99$), the model shows clear failure patterns in several scenarios:

**1. Low-usage stations (data sparsity)**

Stations with very few historical trips generate the highest relative errors. Since the model relies heavily on lag features, sparse stations often appear "inactive", causing the model to miss occasional usage bursts.

**2. Sharp demand spikes**

During unusually strong morning or evening peaks, the model tends to underpredict sudden surges. Rolling averages stabilise forecasts but also smooth out true extremes.

**3. Unusual weather conditions**

The model extrapolates poorly when encountering rare weather combinations (e.g., extreme heat or heavy storms) not present in the 2023 dataset.

**4. Capacity and network effects**

Predictions do not account for station capacity limits or diversion to nearby stations. Some apparent "errors" arise because real-world constraints cap the observed trip counts.

Overall, failures cluster around **low data**, **rare extremes**, and **unmodelled physical constraints**.

## 5.4 Hypothesis analysis

**H1 – Weather & calendar features improve performance**
Partially supported. They add useful context, but lag features contribute far more to accuracy.

**H2 – Lag features significantly boost prediction accuracy**
Strongly supported. Past-hour and past-day usage are the dominant predictors and drive the model's high $R^2$.

**H3 – XGBoost will outperform Random Forest by default**
Rejected. Baseline Random Forest performed better; XGBoost only became superior after tuning.

**H4 – Errors will mainly occur in extreme or rare situations**
Supported. Most failures occur at low-demand stations, during spikes, or under rare weather.

**H5 – One global model is enough to capture station differences**
Partially supported. Overall performance is strong, but some peripheral stations still show systematic under/over-prediction.

## 5.5 Conclusion on Model Performance

The **Tuned XGBoost model** was selected as the final production model, where it has the best tuned parameters (n_estimators = 100, max_depth = 8 and learning_rate = 0.1). Its high RMSE indicates that it is the most robust choice for minimizing costly service failures (e.g. empty stations) during high-demand periods, even if Random Forest was marginally better on "quiet" days.

## 5.6 Dashboard Explanation

The interactive web dashboard provides a fast and intuitive way to explore predictions, station behaviour, and model performance. Built with **Chart.js**, it transforms the model outputs into actionable insights for operators and analysts (Figure 10).

**How It Works**

- **Data Initialization:** When the dashboard loads, it retrieves global metrics, station coordinates, and sample inputs from /api/init.
- **Interactive Map:** All stations are displayed on a map with hover popups (average trips). A red draggable marker lets the user set their location for station recommendations.
- **Prediction Timeline:** A multi-model line chart visualises actual vs. predicted hourly demand for the selected station. Users can click any model card to compare curves.
- **Mode Selector:** The dashboard offers four analysis modes:
    - **Historical Analysis**: View a station's full 24h activity for any date.
    - **Best Station Recommendation:** Suggests the optimal nearby station based on distance and predicted availability.
    - **Daily Station Flow:** Shows predicted IN/OUT flow for all stations to highlight imbalances.

- **Station Ranking:** Lists the busiest stations for a chosen hour.

**What It Highlights**

- **Demand patterns:** Peaks, troughs, and weather-sensitive behaviours.
- **Imbalance risks:** Stations predicted to empty or overflow.
- **Model performance:** Errors, accuracy %, and side-by-side comparisons.
- **Operational insights:** Best stations for redistribution or user travel planning.

Overall, the dashboard converts the project's ML models into **clear, operationally useful intelligence**, enabling proactive and data-driven decision-making.

# 6 Learnings and Recommendations

The analysis of the Bergen Bysykkel bike-sharing dataset revealed several critical insights that challenge conventional wisdom in demand forecasting and operational planning:

## 6.1 Key Learnings

- **Recent History Predicts Future**

Recent historical usage patterns are dramatically more predictive than temporal features alone. Our models consistently ranked the 1-hour and 24-hour lag features which represents the trip counts from those time periods as the top two predictors of future demand. Once the demand patterns are established, they tend to persist in the short term due to the behavioural patterns of commuters and casual riders who follow predictable routines. Hence, we know that a station that experienced several trips in the previous hour is more valuable for predicting the next hour than knowing the date and time, even though temporal features are traditionally considered essential in time-series forecasting.

- **Weather is a "Gain/Loss" Switch**

The weather does not affect ridership proportionally in particular. Instead, it functions more like a gain/loss switch" with threshold effects. The model discovered that light precipitation (under 2mm) has minimal impact on commuter demand, while heavy rain (above 5mm) dramatically suppresses casual ridership but leaves commuter demand relatively intact. Similarly, the temperature exhibits an optimal range rather than a simple linear correlation. For example, the demand peaks around 15-20°C and drops sharply in both extremely cold (below 0°C) and extremely hot (above 25°C) conditions. This non-linear behaviour is precisely why ensemble tree-based models like XGBoost outperformed traditional statistical approaches that assume linear relationships.

- **Context-driven decision making outperforms static rules**

Many bike-sharing operators rely on fixed schedules (e.g. "always rebalance at 8AM") Our models demonstrated that intelligent scheduling should be context-aware. For example, "Rebalancing at 8AM during weekdays, but increase the frequency when precipitation is forecasted". This context sensitivity can significantly reduce wasted labour while improving service reliability.

## 6.2 Business Recommendations

Based on the learnings, we propose the following actionable recommendations for Bergen Bysykkel and similar operators:

- **Transition to predictive rebalancing operations**

One of the ways would be to move from reactive rebalancing (responding to current station status) to proactive rebalancing (based on 2-hour forecasts). Operationally, this also means that integrating the XGBoost model into the daily operations workflow. Each morning, the system should generate predictions for every station across the 24-hour period. Rebalancing trucks should be assigned to stations predicted to become empty within the next 2 to 3 hours, rather than driving routes based on real-time empty/full status. This approach reduces travel distance, optimizes truck routes through mixed-integer programming, and ensures coverage even during periods of rapid demand fluctuation.

- **Weather-adaptive workforce planning**

Another possible recommendation would be to integrate hourly weather forecasts (from meteorological services) with demand predictions to dynamically adjust rebalancing staffing levels. On days with heavy rain predicted, scheduled rebalancing shifts can be reduced since demand naturally plumets, hence saving labour costs while maintaining service levels. Additionally, on clear days with moderate temperatures, staffing can be increased to capitalize on the naturally elevated demand and prevent empty-station failures.

- **User-Facing reliability indicators**

Lastly, a "Predicted Availability Score" can be displayed on the mobile app for each station over the next 2 to 4 hours. This manages user expectations, reduces frustrations from unavailable stations and encourages trip planning during predicted high-availability windows. For example, for users viewing the app at 17:00, the interface could show: "Station A: 95% likely to have bikes available at 17:30, 60% at 18:00, 40% at 19:00." This transparency builds user trust and enables smarter travel decisions.

# 7 Limitations and Future Works

**Limitations**

While the tuned XGBoost model achieved exceptional accuracy with an $R^2$ score exceeding 0.99, several practical constraints limit its real-world applicability. Understanding these limitations is critical for realistic deployment and operational planning.

- **Traffic and Logistics Constraints**

The model assumes rebalancing trucks can be deployed instantly to forecasted stations. In practice, Bergen's rush-hour traffic (particularly between 5:00 to 7:00 AM and 13:00 to 15:00) introduces substantial delays. However, a journey estimated at 15 minutes on a Sunday evening could require 35 minutes on a Wednesday morning, which is an increase in travel time. Additionally, narrow city-centre streets, one-way systems, and seasonal snow conditions can create unpredictable travel patterns. Without real-time traffic integration, the model cannot answer critical operational questions such as "Which stations should be prioritized given a 45-minute rebalancing window?". This gap undermines the practical utility of 2-hour demand forecasts if trucks cannot reach flagged stations within the predicted timeframe.

- **Unmodeled External Events**

The model learns from 2023's regular patterns but cannot predict extraordinary demand shocks. Major events such as concerts (potentially 300 to 500% demand increase), weather emergencies, road closures, or transit strikes create demand orthogonal to learned patterns. The dataset contains no event calendar or incident logs. For example, a Taylor Swift concert in Bergen would trigger downtown demand spikes that the model cannot anticipate, resulting in errors of ±200 to 300%. Hence, while the "is_holiday" flag captures scheduled holidays, spontaneous events remain invisible.

**Future Works**

- **Event calendar and anomaly detection**

We could possibly integrate concert schedules, sports fixtures, road closure announcements, and weather alerts as features, which a sub-model is trained to quantify event impact multipliers. On the other hand, we could monitor prediction residuals in real-time, flagging anomalies that exceed 3 standard deviations and triggering root-cause investigations. This will generate an expected impact of increased accuracy improvement on event days.

- **Graph Neural Networks for network-aware demand modelling**

Rather than predicting stations independently, we could model Bergen's bike-sharing network as a directed graph where nodes are stations, and weighted edges represent user flows. A Graph Neural Network would learn spillover effects. For example, "If downtown is busy, adjacent residential stations receive inbound overflow.". This approach captures network-level optimization (global bike allocation) rather than station-level optimization (local rebalancing). GNNs can also predict destination distributions, enabling pre-positioning bikes at high-demand destinations.

# 8 Conclusion

This project successfully demonstrated that Machine Learning can solve the urban bike rebalancing problem. Using **Bergen** as our case study, we proved that by combining historical usage data with environmental factors (weather), we can predict station demand with high accuracy.

The resulting **XGBoost model** offers a tangible tool for city operators to improve efficiency and reduce costs. Adopting this system represents a shift from "intuition-based" management to "data-driven" operations, ensuring a greener, more reliable transport network for the city.

The identified limitations such as traffic delays, infrastructure degradation, data drift, external events, and spatial network effects are not fatal flaws but rather engineering challenges with known solutions. Implementing the proposed future works would incrementally enhance the system toward a fully autonomous, adaptive, sustainable operations platform.

# 9 References


1. Share-North. (2020). *Nearly 1 million trips – Bike-sharing success in Bergen, Norway!* https://share-north.eu/2020/02/nearly-1-million-trips-bike-sharing-success-in-bergen-norway/

2. Interreg North Sea. (2023). *Bergen's game-changer: The world's longest purpose-built cycling tunnel*. https://www.interregnorthsea.eu/active-cities/news/bergens-game-changer-the-worlds-longest-purpose-built-cycling-tunnel

3. Bicycle Network. (2023). *World's longest purpose-built bike tunnel to open in Norway*. https://bicyclenetwork.com.au/newsroom/2023/03/30/worlds-longest-purpose-built-bike-tunnel-to-open-in-norway/

4. Global Construction Review. (2023). *Norway prepares to open world's longest dedicated bike tunnel*. https://www.globalconstructionreview.com/norway-prepares-to-open-worlds-longest-dedicated-bike-tunnel/

5. Indicio. (2024). Can you afford to only have univariate forecasting in a multivariate world? https://www.indicio.com/resources/whitepapers/can-you-afford-to-only-have-univariate-solutions-in-a-multivariate-world

6. Atlantis Press. (2022). Regression on Seoul bike sharing demand. https://www.atlantis-press.com/article/126015310.pdf

# 10 Appendix

### Code Execution Instructions

To run the project, begin by cloning the repository using Git. Open a terminal, navigate to your preferred folder, and run:

*git clone https://github.com/NicolasLasch/PML-Project.git*
*cd PML-Project*

A Python virtual environment is recommended. Create one with python3 -m venv venv, then activate it using source venv/bin/activate on macOS or Linux, or venv\Scripts\activate on Windows. Once the environment is active, install all required dependencies by running pip install -r requirements.txt.

The project expects the dataset file bergen_merged.csv to be located in data/raw/. Processed datasets will be created automatically in the data/processed/ directory during execution.

The full pipeline can be executed with the command:

*python src/main.py*

This script loads the dataset, performs preprocessing, generates lag and rolling features, trains the Random Forest and XGBoost models, evaluates them, and saves all outputs. All trained models are stored in the models/ directory, and charts and evaluation metrics are saved in the output/ folder.

To launch the optional web dashboard and REST API, run:

*python app.py*

After starting, the dashboard can be accessed in a browser at http://localhost:8000.

Tests for data processing, feature engineering, model training, and recommendations can be executed using:

*pytest test_all.py -v*

A minimal setup workflow consists of cloning the repository, creating and activating a virtual environment, installing dependencies, placing the dataset in the raw data folder, and running the main pipeline script. This is sufficient to reproduce all results shown in the report.

**Bergen Bike Sharing Data Dictionary**

| Column Name | Data Type | Description |
|---|---|---|
| **Trip Information** | | |
| start_time | DateTime | The date and time when the bike was unlocked/rented. |
| end_time | DateTime | The date and time when the bike was docked/returned. |
| duration | Integer | The length of the trip in **seconds**. |
| **Station Information** | | |
| start_station_id | Integer | Unique ID for the station where the trip began. |
| start_station_name | String | The common name of the start station. |
| start_station_latitude | Float | GPS Latitude of the start station. |
| start_station_longitude | Float | GPS Longitude of the start station. |
| end_station_id | Integer | Unique ID for the station where the trip ended. |
| end_station_name | String | The common name of the end station. |
| end_station_latitude | Float | GPS Latitude of the end station. |
| end_station_longitude | Float | GPS Longitude of the end station. |
| **Weather Conditions** | | |
| temperature | Float | Air temperature at the time of the trip (in Celsius). |
| max_temperature | Float | Maximum temperature recorded during that hour. |
| min_temperature | Float | Minimum temperature recorded during that hour. |
| wind_speed | Float | Speed of the wind (in m/s). |
| precipitation | Float | Amount of rainfall/snowfall (in mm). |
| humidity | Float | Relative humidity percentage (0-100%). |
| weather | Integer | A categorical code representing the weather state (e.g., Clear, Cloudy, Rain). |
| sunshine | Float/Int | Duration of sunshine or UV index during that hour. |
| **Temporal Context** | | |
| season | Integer | Numerical code representing the season (0=Spring, 1=Summer, 2=Autumn, 3=Winter). |
| is_holiday | Boolean | True if the day is a public holiday in Norway, False otherwise. |
| is_weekend | Boolean | True if the day is Saturday or Sunday, False if it is a weekday. |

*Table 3: Bergen Bike Sharing Data Dictionary (bergen_merged.csv)*

**Project References**

| S/N | Reference Description | Link |
|---|---|---|
| 1 | Project's GitHub repository | https://github.com/NicolasLasch/PML-Project |
| 2 | Bergen Bike Sharing dataset | https://www.kaggle.com/datasets/amykzhang/bergen-bike-sharing-dataset-2023 |

*Table 3: References used in project and their links*

## Chart Figures (EDA, Results)

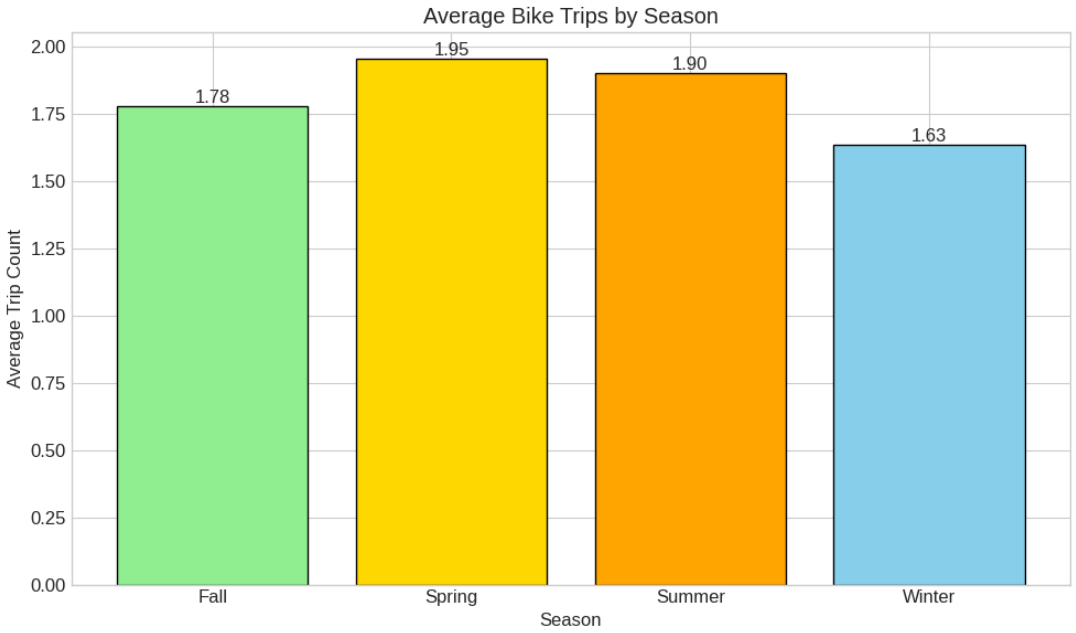


*Figure 4: Average Bike Trips by Season*

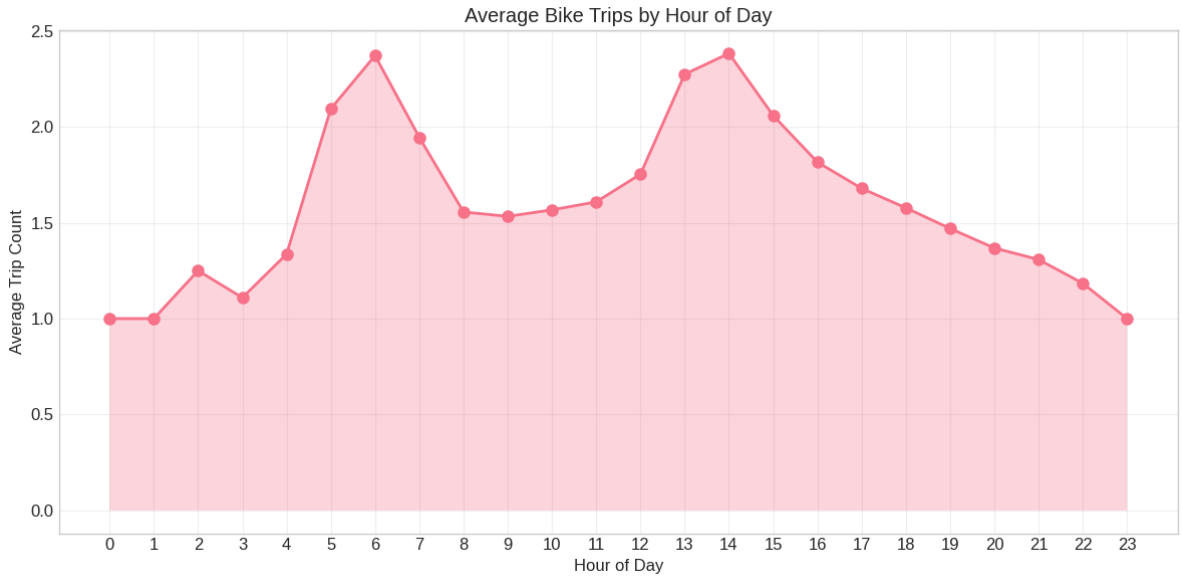


*Figure 5: Average Trip counts by Hour of Day*

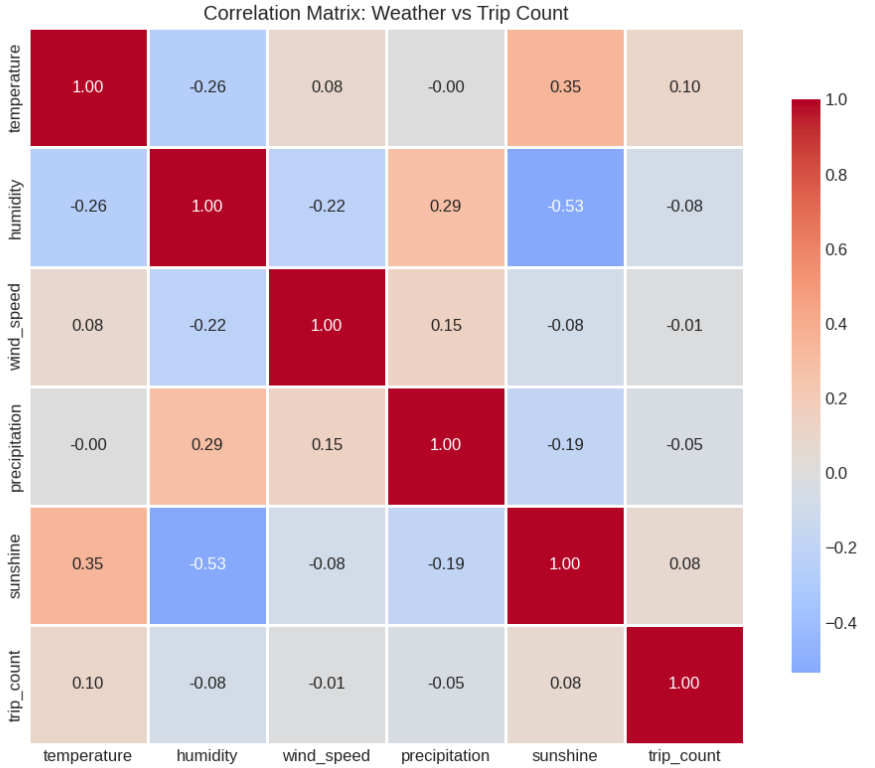


*Figure 6: Correlation between Weather & Trip Count*

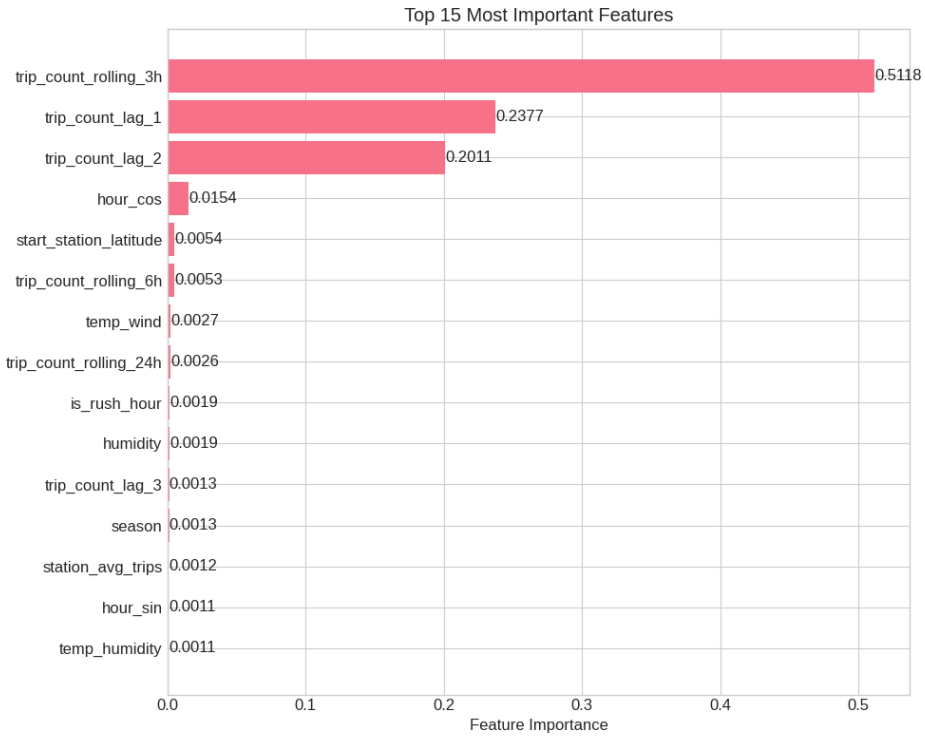


*Figure 7: Top 15 Most Important Features*

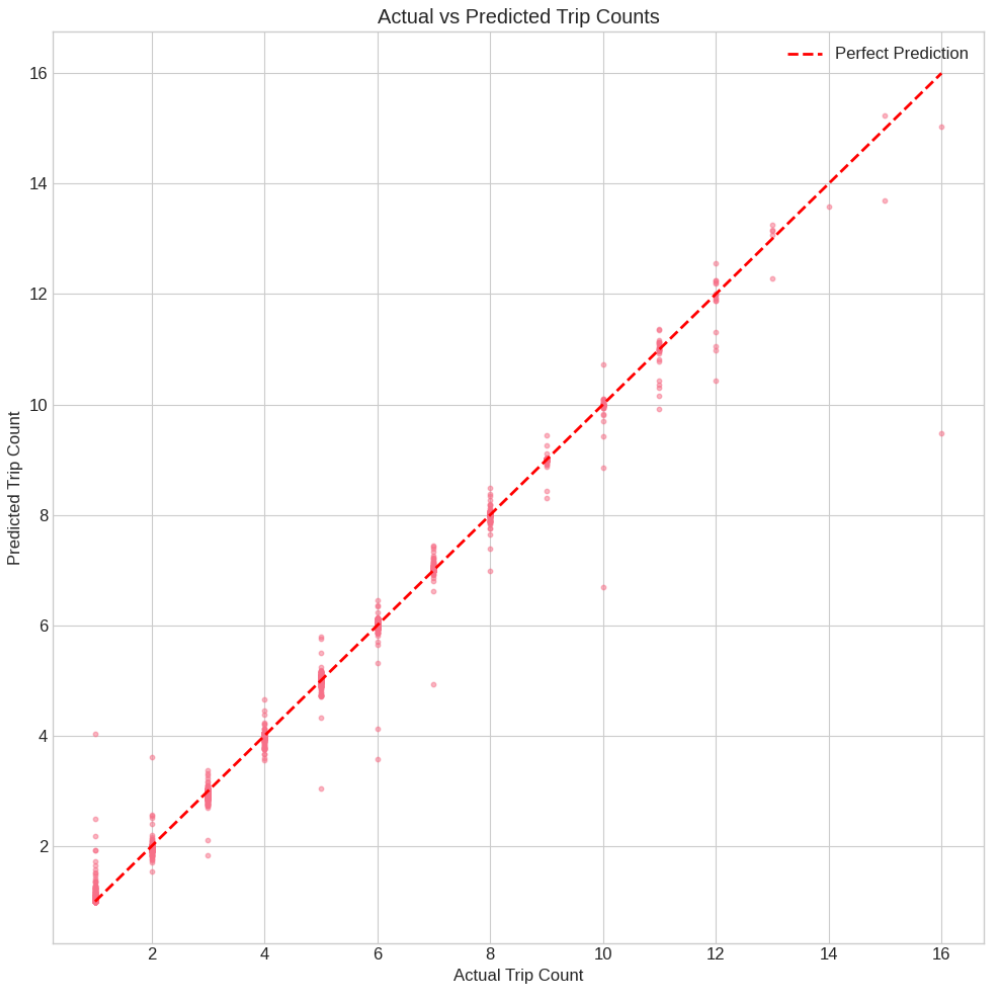


*Figure 8: Actual vs Predicted Trip Counts*

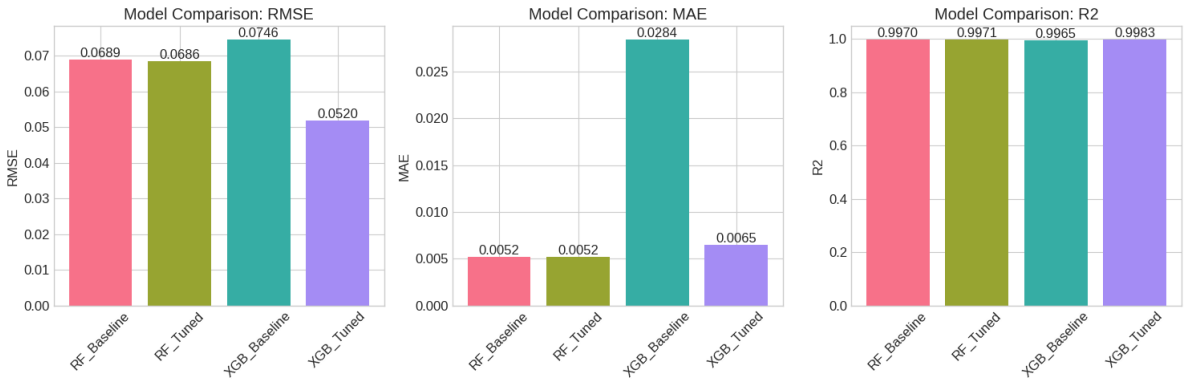


*Figure 9: Model Comparison Results (RMSE, MAE, $R^2$)*

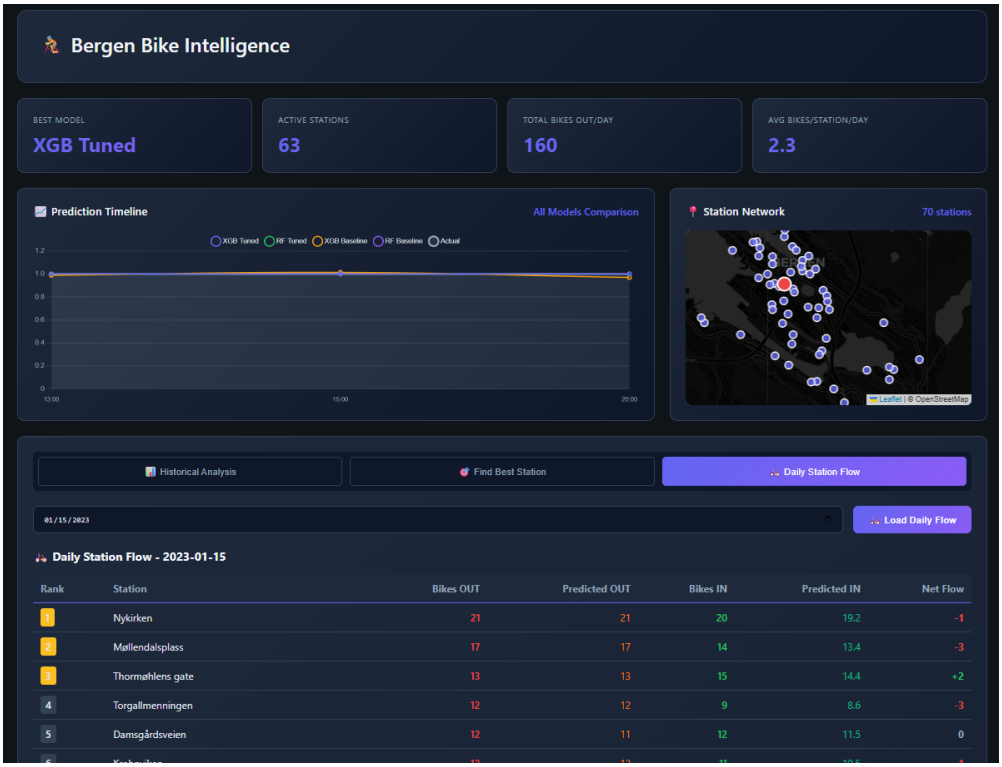


*Figure 10: Bergen Bike Prediction API Dashboard*