KTH Library Seating Predictor

Contents

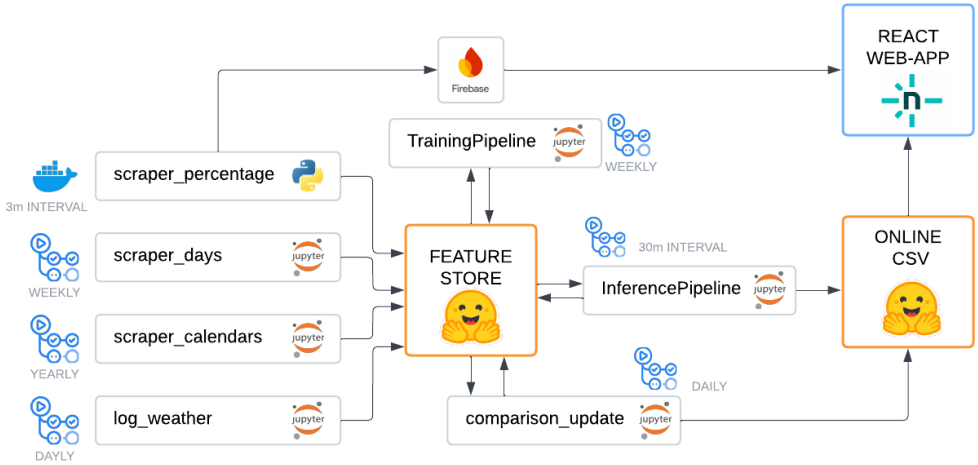
Introduction

This project focuses on a KTH seating predictor designed for the library. It allows students to check the current occupancy status at any time and view predictions for future hours and the following day. The website is always online, and we plan to maintain and support it in the future. The notebook runs periodically using GitHub Actions.
It can be accessed at the following URL:

`https://kthseating.netlify.app/`

For more details and the complete code, please refer to the GitHub repository:

`https://github.com/davidenascivera/KTH_seat_predictor/`

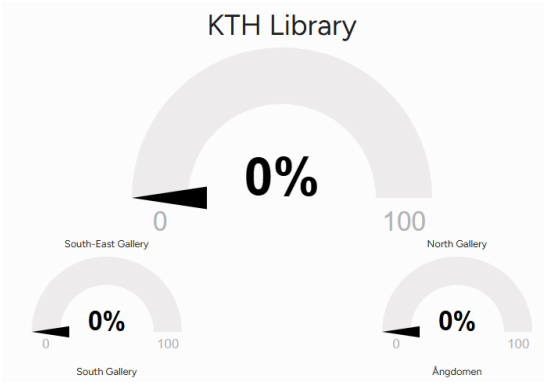


1 Dynamic Data Sources

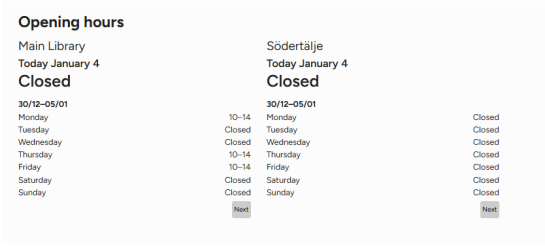
For the creation of this project multiple dynamic data sources have been used. For the exception of the weather, neither of those data had an API so we had to create some scraper to get the information from the official KTH website.

1.1 Scraper for the Library Occupancy

To obtain real-time occupancy data for the various zones of the KTH library, we utilized the official library website: KTH Library Visitors in Real Time. The site includes a widget displaying the percentage occupancy for each zone.



(a) Real-time Occupancy Widget on the KTH Library Website



(b) Dynamic Display of Library Opening Hours on the KTH Website

Using Selenium, we implemented a web scraper to access the site, extract its HTML content, and parse the occupancy data into a structured row format. This process is executed every three minutes, with each new row appended sequentially to the dataset. Every ten entries (approximately every 30 minutes), the dataset is pushed to both Hugging Face and Hopsworks for storage and analysis. The Hugging Face dataset is accessible at: `https://huggingface.co/datasets/davnas/occupancy_perc`.

Timestamp	South-EastGallery	NorthGallery	SouthGallery	Ångdomen	Newton
2024-12-09 18:16:16	35	66	60	17	21
2024-12-09 18:19:24	34	66	59	14	19

Table 1: Occupancy data from different library zones with timestamps.

To ensure reliability, the scraper is containerized using Docker and runs on both a Raspberry Pi and an additional server. It is configured to restart daily at 5 a.m., ensuring continuous operation and data collection.

1.2 Scraper for the Opening Hours

During the initial phases of exploratory data analysis (EDA), we identified that the opening hours of the library were a crucial feature for our predictions. The opening hours are dynamically published on the KTH website: `https://www.kth.se/en/biblioteket/anvanda-biblioteket/oppettider-kontakt/oppettider-och-kontakt`. The relevant information for parsing is shown in Figure 1b.

One challenge with this system is that the website only displays the opening hours for the current week. This limitation is particularly problematic on Sundays, as the opening hours for the following Monday are not available on the main page. To address this, the scraper also interacts with the "Next" button to retrieve the subsequent week's data, ensuring continuity in our dataset.

The collected dataset is regularly updated and published on Hugging Face: `https://huggingface.co/datasets/davnas/date_kth` Figure 1b. Updates are automated and occur three times per week using GitHub Actions, ensuring the dataset remains up-to-date and reliable.

Timestamp	Day	OpeningHour	ClosingHour	IsOpen
2024-12-09T00:00:00	Monday	18	21	1
2024-12-10T00:00:00	Tuesday	10	21	1
2024-12-11T00:00:00	Wednesday	8	21	1

Table 2: Library opening hours schedule

1.3 Scraper for KTH Calendar

To ensure our model accounts for academic schedules, we scrape the KTH academic calendar annually from the following link: `https://intra.kth.se/en/utbildning/schema-och-lokalbokning/lasarsindelning/lasaret-2024-2025`.

The URL is dynamically updated each year to reflect the new academic schedule. The scraper's implementation is similar to the previous ones and is designed to handle updates seamlessly. The extracted calendar data is published on Hugging Face for easy access and integration into our pipeline: `https://huggingface.co/datasets/andreitut/kth-academic-scraper`.

Date	Year	Day ofYear	Day ofWeek	Days UntilExam	Event
2024-08-26	2,024	239	1	52	Normal
2024-08-27	2,024	240	2	51	Normal
2024-08-28	2,024	241	3	50	Normal

Table 3: Academic calendar and event schedule

1.4 Weather API

Recognizing that library occupancy is influenced by weather conditions, we integrated a weather API into our data pipeline. The same API used in an earlier lab project was adopted for this purpose.

Every three hours, historical weather data is collected and uploaded to the following Hugging Face repository: `https://huggingface.co/datasets/andreitut/weatherDatasetProject`.

This dataset enables our model to correlate weather patterns, such as temperature, precipitation, and other variables, with library usage. By incorporating weather data, we improve the accuracy of our predictions and account for external factors that influence occupancy trends.

Time	Temp (°C)	Precip (mm)	Wind (m/s)	Dir (°)
2024-01-02 00:00	-4.0	0.2	16.6	85
2024-01-02 01:00	-4.0	0.2	17.2	79
2024-01-02 02:00	-4.4	0.2	17.0	84

Table 4: Hourly weather measurements

2 Training and Inference of the System

The Training pipeline che be found at the following link: `https://github.com/davidenascivera/KTH_seat_predictor/blob/main/training_inference/Training_4_original.ipynb`

Its a notebook composed of multiple sections.

2.1 Downloading All the Data

In this step, we retrieve all the previously uploaded dataframes and merge them into a single dataset using the common key, the timestamp. The merging process is performed as a backward-asof join, which ensures that for each timestamp in the main dataset, the nearest earlier timestamp from the secondary dataset is used. At the end of this process we get the full data with merged with the index being the date.

2.2 Model Independent Transformation (MIT)

In this phase, we perform the Model Independent Transformation (MIT), a crucial step to ensure consistency between the training and inference pipelines. Maintaining identical preprocessing steps in both phases is essential to avoid model skew and ensure the model operates as intended during real-world inference.

To achieve this, all transformation functions are centralized in a utility file named `utils`, which is imported as needed in the respective pipelines. This modular approach enhances maintainability and reduces redundancy.

The key transformations performed during the MIT phase are as follows:

1. Remap the `event` column from strings to numerical values using a predefined, hard-coded mapping.

2. Calculate `time_until_opening` and `time_until_closure` to provide temporal context for each data point.

3. Recompute the `is_Open` column to ensure it accurately reflects whether the library is open (`is_Open = 1`) or closed (`is_Open = 0`) based on the library's operating hours.

2.3 Model Selection

At the beginning of our project, we experimented with time-series models, specifically Facebook Prophet and XGBoost with rolling features. However, due to the limited amount of historical data, as shown in Figure 2, we realized that training models requiring extensive data, such as Prophet or rolling-feature-based XGBoost, was not feasible. Consequently, we opted to proceed with a simplified approach using XGBoost without rolling features.

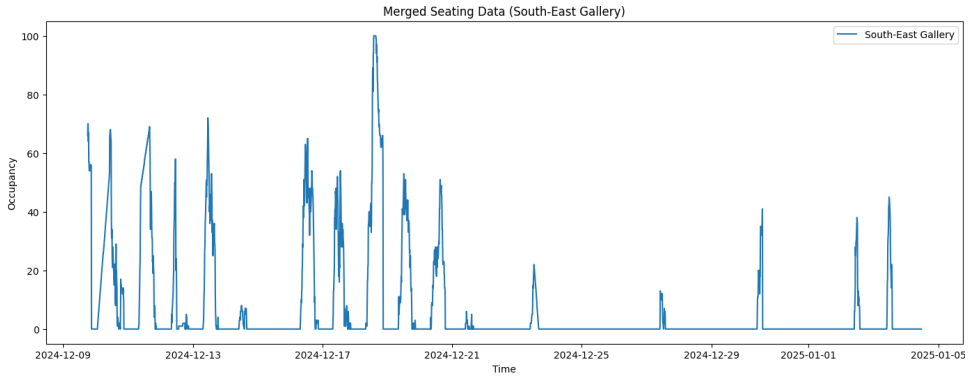


Figure 2: Historical Data Overview

During the exploratory data analysis (EDA) phase, we also tested Prophet. However, as illustrated in Figure 3, we found that Prophet is not well-suited for predicting percentages. The model struggled to capture the significance of critical variables, such as `is_Open`, which plays a pivotal role in library occupancy predictions.

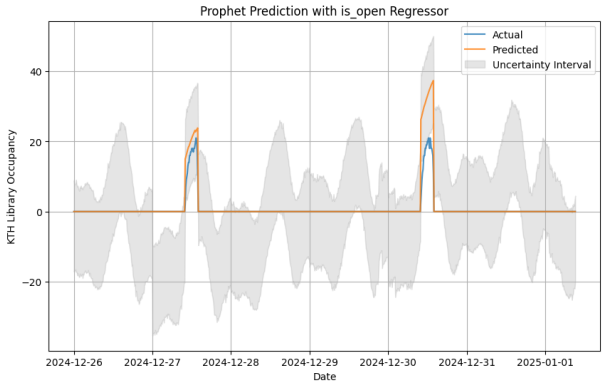


Figure 3: Prophet Model Performance. The model's prediction corresponds to the mean of the shaded gray region. The prediction is flat and equal to zero when the `is_open` variable is set to 0, indicating closed hours.

Based on these findings, we decided to rely solely on XGBoost without the rolling windows, which provided greater flexibility and adaptability in handling the data we had available.

2.4 Prediction and Upload of the Predictions

After completing the training phase, individual models are trained for each location and uploaded to a private Hugging Face repository named `davnas/library_model`. The repository is organized such that each model is named according to its corresponding room, ensuring seamless identification and retrieval during the inference phase.

During inference, the same preprocessing steps used in the training phase—downloading, merging, and Model Independent Transformation (MIT)—are performed to ensure consistency. From the resulting dataset, only data for the current day is retained. Predictions are then generated for both the current day and the following day.

Once the predictions are generated, the results are visualized and stored. The final datasets are uploaded to the following Hugging Face repository links:

- `https://huggingface.co/datasets/davnas/library-occupancy/blob/main/forecast_tomorrow.csv` – Contains predictions for the following day.
- `https://huggingface.co/datasets/davnas/library-occupancy/blob/main/data_2.csv` – Stores the current day measurement and the forecast for the current day.

The inference notebook is executed daily at 1-hour intervals between 7:00 AM and 10:00 PM, automated through GitHub Actions.

2.5 Logging the Predictions

To maintain a record of daily predictions, the forecast for the following day is logged every evening at 11 PM. This process ensures a comprehensive historical log of predicted occupancy data for tracking and analysis purposes.

The predictions are appended to an existing dataset and then reuploaded to the following Hugging Face repository: `https://huggingface.co/datasets/davnas/library-occupancy`.

3 UI and Deployment

The system is designed to be serverless, with automatic updates occurring hourly. The working website can be accessed at: `https://kthseating.netlify.app/`

The website was built using React and is hosted on Netlify. Netlify provides a seamless service for hosting GitHub repositories, enabling effortless updates and deployment. The website is static and fetches data from the previously created Hugging Face repository in CSV format.

The website is structured into two main sections:

1. **Current Occupancy and Predictions**: This section displays the daily predictions alongside real-time occupancy data. The real-time data is managed using a Firebase database, which is updated each time the scraper collects occupancy information from the KTH website.

2. **Prediction vs. Reality**: This section provides a comparison of the predictions against the actual occupancy data from the previous day, allowing for evaluation of the model's performance.

By leveraging React and Netlify, the website remains lightweight, easily maintainable, and capable of efficiently displaying dynamic data for users.

4 Future Development

The project has been deployed using Hugginface for database management and is nearly fully integrated with the Hopsworks API. The corresponding code can be accessed on GitHub through the new branch named `hopsworks`:

`https://github.com/davidenascivera/KTH_seat_predictor/tree/Hopsworks`

The only remaining task for full integration is porting the training script to upload the model to the model registry. For display purposes, the current predictions and one-day-ahead predictions are still being stored on Hugging Face.