Velib' Project

G.O'Brien, A.Le Saux, S.Koskas, H.Al Muhdi

January 2025

Abstract

blah blah

1 Introduction

The rapid expansion of shared mobility services has profoundly reshaped urban transportation systems worldwide. In metropolitan areas like Paris, the Velib bike-sharing program has emerged as a critical component of sustainable urban mobility, serving a diverse user base that includes commuters, tourists, and local residents. Despite its widespread adoption, the Velib system faces significant operational challenges, particularly concerning the availability of bikes at docking stations. Users frequently encounter empty stations, leading to frustration and inefficiency as they are often forced to search multiple stations to find an available bike. This issue is especially problematic for daily commuters and students, who rely on Velib as a primary mode of transportation to adhere to strict schedules.

As students ourselves, we have personally experienced the frustration of relying on Velib for daily commutes, only to find no bikes available when needed. Recognizing that this inconvenience is not unique to us, but shared by many other students and commuters, we were motivated to tackle this issue. Recent data highlights that students constitute the largest proportion of Velib subscribers. Indeed, according to a 2019 report by Autolib' Velib' Métropole, individuals aged 14-24 years, which are for the most students, make up a significant 26% of Velib users (Statista, 2019). This demographic heavily depends on the service for daily commutes across Paris and its suburbs. The inability to reliably predict bike availability disproportionately affects this group, contributing to delays and inconvenience in their routines.

Currently, the Velib application provides real-time information on bike availability but lacks predictive capabilities and is not updated in real-time. This limitation prevents users from planning their trips effectively, particularly during peak hours or adverse weather conditions when bike demand fluctuates. Consequently, there is a pressing need for predictive models that can forecast the number of available bikes at specific stations and times, allowing users to plan their journeys in advance with greater accuracy. The primary objective of this project is to develop a robust machine learning model to predict the availability of Velib bikes across various stations in Paris. By leveraging historical usage data and external factors such as weather conditions, the proposed model aims to deliver accurate, station-specific forecasts. These predictions can empower users to make informed decisions about when and where to find available bikes, reducing the time and effort spent locating bikes. Additionally, these insights can support Velib's operational team in better understanding demand patterns, enabling more efficient bike redistribution and system management.

Addressing this problem not only improves user satisfaction but also contributes to more efficient and sustainable urban mobility. Accurate predictions of bike availability can mitigate user inconvenience, increase system efficiency, and support strategic decision-making for resource allocation. In the broader context, this research aligns with ongoing efforts to integrate intelligent systems into urban infrastructure, fostering smarter and more responsive cities.

2 Problem Definition

The primary objective of this project is to develop a predictive model capable of accurately forecasting the number of available Velib bikes at specific stations and times across Paris. Formally, this problem involves predicting the number of bikes available at a given station and future time by leveraging various influencing factors. Let $S = \{s_1, s_2, \ldots, s_n\}$ represent the set of all Velib stations, each characterized

by spatial, temporal, and environmental conditions, and let T denote the set of discrete future time intervals. The task is to predict the function f that maps station and temporal data to the expected bike availability:

$$\hat{Y}_{i,t} = f(S_i, t, X_{i,t})$$

where $\hat{Y}_{i,t}$ is the predicted number of available bikes at station S_i at time t, and $X_{i,t}$ represents the set of explanatory features influencing availability. The goal is to minimize the prediction error between the true number of available bikes $Y_{i,t}$ and the predicted value $\hat{Y}_{i,t}$.

To accurately model this problem, multiple data sources capturing the various factors affecting bike availability were considered. The Velib operational data provided detailed information on the real-time status of each station, including the total number of bikes available, the breakdown between mechanical and electric bikes, and the number of free docking points. Additionally, operational status indicators, such as whether a station was installed, renting, or accepting returns, were included to reflect the dynamic functionality of the bike-sharing system. Each station was geographically anchored by its latitude and longitude, enabling the model to account for spatial patterns in bike usage across different areas of Paris.

Complementing the operational data, weather data was incorporated to capture environmental conditions that could influence user behavior. Features such as temperature, feels-like temperature, precipitation levels, snow accumulation, wind gusts, and cloud cover were integrated into the model. Weather plays a critical role in the decision-making process of cyclists, with adverse conditions likely deterring usage. Incorporating these environmental variables aimed to enhance the model's ability to predict demand fluctuations linked to changing weather patterns.

The central objective of this predictive task is to minimize the discrepancy between the actual and predicted number of available bikes across stations and times. This is achieved by optimizing performance metrics that quantify model accuracy. Specifically:

- The **Mean Squared Error (MSE)** penalizes larger prediction errors more heavily, ensuring that significant mispredictions are minimized.
- The **Mean Absolute Error (MAE)** provides an interpretable measure of the average error magnitude.
- The **Root Mean Squared Error (RMSE)** translates prediction errors into the same units as the target variable, making it more tangible for interpretation.
- The R^2 **score** assesses how well the model explains the variance in bike availability, offering a holistic evaluation of model performance.
- **The Mean Absolute Percentage Error (MAPE)** measures the average magnitude of percentage errors between predicted and actual values, making it a useful metric for interoperability.

Several constraints and assumptions define the scope of this problem. One of the primary constraints was the limited timeframe of the project, which spanned less than two months. This restricted period significantly impacted the amount of data that could be collected and processed. Due to financial limitations, it was not feasible to scale up data collection beyond the existing infrastructure. Consequently, only three weeks of Velib data were gathered using the Velib API. This limited dataset does not fully capture long-term patterns and seasonal trends in bike usage that could emerge over several months or across different seasons. As a result, the model may not generalize well to conditions outside the sampled timeframe, such as extreme weather events, holidays, or city-wide disruptions. Additionally, predictions are generated on an hourly basis, aligning with the temporal granularity of the available data.

Finally, the model does not account for unforeseen disruptions such as strikes, major public events, or system-wide outages, which can cause sudden and significant shifts in bike demand. Predictive accuracy is inherently dependent on the quality and completeness of the input data, particularly the operational and weather datasets.

By integrating spatial, temporal, and environmental factors, this project aims to deliver accurate and actionable predictions of Velib bike availability.

3 Related Work

Predicting Bike Availability in Bike-Sharing Systems

The task of predicting bike availability in bike-sharing systems has been approached through various methodologies, broadly categorized into statistical models and machine learning/deep learning approaches. Each category offers distinct advantages and limitations in addressing the complexity of bike demand prediction.

Statistical Models

Traditional statistical models, particularly time series forecasting methods, have been widely used due to their simplicity and interpretability. These models typically assume linear relationships and focus on capturing temporal trends.

In the study *"Forecasting Citibike Ridership"*, Marvel applied a SARIMA model to predict bike usage in New York City, achieving 83% explained variance (EV). While this demonstrated strong performance in capturing seasonal and temporal patterns, the model's effectiveness was limited by the absence of external data, such as weather conditions or behavioral shifts triggered by events like the COVID-19 pandemic (Marvel, n.d.).

Similarly, an ARMA model was employed to predict bike availability in Barcelona's Bicing system. By incorporating spatial data from neighboring stations, the study improved short-term forecasts and captured station-specific demand fluctuations. However, it lacked integration with external influences, such as weather or maintenance operations, limiting its broader applicability (Reddy, 2010).

Another approach utilized the Piecewise Continuous-Time Markov Chain (PCTMC) in *"Moment-Based Availability Prediction for Santander Cycles"*. This model outperformed traditional Markov queueing models for short-term predictions by capturing stochastic variations in bike availability. Its computational efficiency made it suitable for real-time predictions, though it also lacked consideration of external variables like weather or city-wide events (ScienceDirect, 2017).

Strengths and Limitations of Statistical Models

Statistical models are computationally efficient and interpretable. However, their linear structure and lack of external data integration restrict their ability to capture the complex, nonlinear patterns often observed in bike-sharing systems.

Machine Learning and Deep Learning Approaches

To address the limitations of linear models, researchers have increasingly turned to machine learning and deep learning techniques, which excel at modeling nonlinear relationships and incorporating high-dimensional datasets.

In *"Predicting Vélib' Usage in Paris"*, Bergamasco utilized Random Forest and XGBoost regressors, integrating weather and temporal data to capture complex feature interactions. XGBoost outperformed Random Forest due to its superior handling of missing data and intricate patterns, highlighting the importance of combining operational and external data for improved accuracy (Bergamasco, n.d.).

A Graph Neural Network (GNN) was applied in *"Predicting Bike Availability at Bike Share Stations in Toronto"* to model both spatial and temporal dependencies. While the GNN achieved superior performance over baseline models by leveraging spatial relationships between stations, the study acknowledged the need for additional contextual data, such as weather or user demographics, to further enhance predictions (AI4SM, n.d.).

Deep learning methods have demonstrated exceptional predictive capabilities. In *"Bike Demand Forecasting—Time Series"*, a comparative analysis of LightGBM, SARIMAX, and a 1D Convolutional Neural Network (CNN) was conducted. The CNN achieved the highest accuracy, with a Mean Absolute Error (MAE) of 0.5 and a Mean Absolute Percentage Error (MAPE) of 0.06%. However, concerns about overfitting and limited interpretability persisted. LightGBM provided a balance between accuracy and explainability, while SARIMAX served as a computationally efficient baseline (Deladrière, n.d.).

Strengths and Limitations of Machine Learning and Deep Learning Models

Deep learning models offer superior predictive accuracy but often lack transparency, making it challenging to interpret the drivers behind their predictions. This trade-off has prompted interest in models that balance temporal dependency modeling with interpretability.

Positioning Our Work

Our project aligns with and builds upon existing research in bike availability forecasting by integrating

spatial, temporal, and environmental data. Like prior studies, we leveraged historical usage data to develop predictive models that capture time-based trends and station-specific usage patterns. However, our approach diverges in several key aspects:

Integration of External Variables

While studies like Bergamasco's incorporated external data sources, many relied solely on historical ridership or station data. Our project systematically integrates weather variables (e.g., temperature, precipitation, wind gusts) and temporal features (e.g., hour of the week) into statistical and machine learning models. This improves our ability to predict bike demand variability caused by environmental changes, an aspect often overlooked in traditional approaches.

Hybrid Modeling Approach

Unlike previous research that primarily focused on statistical or machine learning models in isolation, our project adopts a hybrid strategy. We implemented SARIMA, SARIMAX (with exogenous variables), Random Forest Regression, and advanced deep learning architectures, including LSTM and Bidirectional LSTM-GRU hybrids. This diverse approach enables us to compare performance across methodologies, balancing accuracy and interpretability.

Sequential Models for Temporal Dependencies

While other studies have applied CNNs, our use of sequential models like LSTM explicitly accounts for temporal dependencies and complex patterns in the data. This contrasts with CNNs, which, while effective, may over-rely on lag features without fully capturing time-series dynamics.

User-Centric Focus

Unlike studies aimed at long-term demand analysis or system optimization, our project specifically targets short-term bike availability predictions to assist users in real-time journey planning. This user-centric approach distinguishes our work from research focused primarily on operational efficiency.

4 Methodology

4.1 Data Collection

To retrieve real-time data on Velib stations and the number of bikes available (both mechanical and electric), we utilized the Velib API, as mentioned in the references. This API provides frequent updates on the current availability of Velib bikes at each station. Initially, since we were uncertain about the appropriate data collection frequency, we logged data for every station at 15-minute intervals. This approach allowed us to later adjust the intervals easily to hourly or 30-minute intervals, depending on our analysis requirements.

To gather this data, we set up a virtual machine (VM) on Google Cloud Platform (GCP) running the code provided in the *data_set_maker.py* script located in the zip file. Over the collection period, we retrieved more than 3 million data points from 1,488 stations. Data collection ran from November 14th to December 8th, which revealed a key limitation: the high cost of running a VM for such large-scale data gathering. Specifically, the process incurred a cost of €30 for three weeks, leading us to conclude that further data collection would not be feasible within our budget constraints.

To enhance our dataset, we mapped the precise location (latitude and longitude) of each station using another Velib API, provided in the references section. Additionally, to enrich the dataset further, we aimed to integrate weather data.

Weather data was sourced from Visual Crossing, a platform offering an API connected to a historical log of weather data for various regions. We were able to scrape hourly weather data for the three-week study period, as no finer time intervals were available. The initial weather data set consisted of 576 data points with 24 features.

Based on the results of our data collection, we decided to use hourly data for each station to perform regression predictions and time series analysis.

4.2 Data Preprocessing

4.2.1 Cleaning

In the Velib data set we initially had 11 features with 5 interests we identified to work on which were: the total number of bikes, the number of mechanical and electrical bikes, the *station_id* and finally the *api_call_time* which will be our timestamp.

The first issue to address was null values in the Velib data. Through analysis we realised that only features with null values were *stationCode*, *num_bikes_available_mechanical*, and *num_bikes_available_ebike*. A null value in the *stationCode* column corresponded directly to null values in the other two columns. Clearly, there was an issue with these stations' data collection or functioning so we decided to remove the data points corresponding to these stations completely. This corresponds to the removal of about 60,000 data points. Since the rest of the Velib data and the weather data were complete, we were left with no null values to impute.

Pulling more data from the Velib API we aligned the latitude and longitude with the *station_id*, since we believed that latitude and longitude values, set between -1 and 1, would be handled better by the model and required no scaling. *station_id* consisted of a huge range of numbers which could affect the functioning of linear models. To handle this, initially, the *station_id* was encoded to be the mean *num_bikes_available* for that specific station. This improved the results when ran through a Linear Regression. Scaling the *station_id* with the MinMaxScaler was also attempted for further improvements. The model performed best with latitude and longitude used as features instead of *station_id* so we decided to make this universal for all testing.

Certain Velib stations had continued to relay static data after the station stopped functioning properly. This was evident when analysing the difference between *last_reported and api_calltime*. It doesn't make sense to keep the bike data from these stations since it is representative of a previous time, and time is a key feature in this study. The weather data was collected in 1 hour intervals, so we decided to remove all data points with a difference between *last_reported* and *api_calltime* of more than an hour. This was achieved by creating a new feature, *time_differences*, and filtering based on this. Effectively, this removed the remaining faulty stations.

Finally, the two data sets had to be merged. Since the Velib data was collected every 15 minutes and the weather data was collected every hour, we decided to remove the Velib data which did not correspond to the weather data timestamp. Whilst this reduced our data set to 25% of its size, this was important for ensuring the data was properly aligned, removing any excess uncertainty. The final dataset contains about 754,000 points.

4.2.2 Feature Selection and Engineering

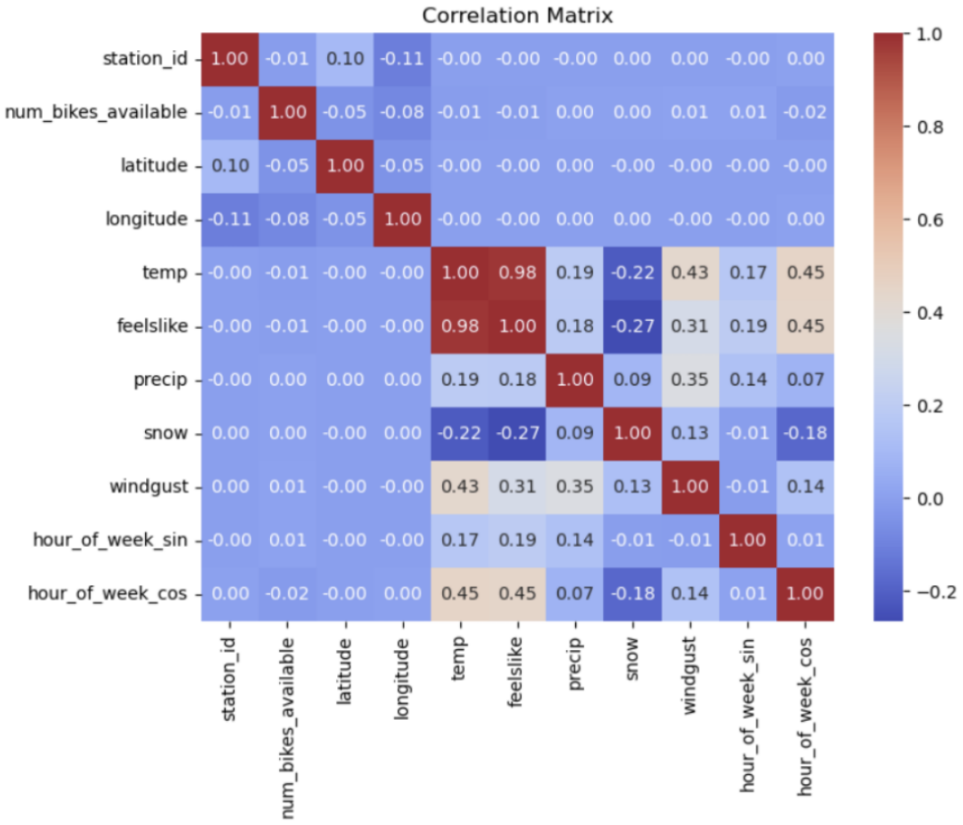


Figure 1: Correlation Matrix of Features

As previously mentioned, we decided to use latitude and longitude as features instead of *station_id* because they were more robust when used in linear models. We recognised that the time would be a key feature since people are more likely to use Velibs at certain times on certain days. For the models to process the time effectively, the *api_calltime* needed to be made cyclical and between -1 and 1. Given our data collection period was about three weeks the most granular solution was to convert it to the hour of the week, using Equation 1. This was then pushed through and Equations 2 and 3 to convert it to an hour of the week in sine and cosine.

$$\text{hour\_of\_week} = (\text{day\_of\_week} \times 24) + \text{hour} \tag{1}$$

$$\text{hour\_of\_week\_sin} = \sin\left(\frac{2\pi \cdot \text{hour\_of\_week}}{168}\right) \tag{2}$$

$$\text{hour\_of\_week\_cos} = \cos\left(\frac{2\pi \cdot \text{hour\_of\_week}}{168}\right) \tag{3}$$

This study is about predicting the number of bikes available at Velib stations around Paris based on time and weather. For this reason *num_docks_available*, *num_bikes_available_mechanical*, and *num_bikes_available_ebike* were not used as features. Furthermore, they are directly correlated to the total number of bikes available so training a model on them would improve our results without actually improving the model choice or tuning.

The weather data features had very little correlation to the number of bikes available, at an initial glance, so to select which features to use we had to use our intuition. As regular Velib users we were able to deduce which weather features have more of an impact on the decision to ride a bike at that time. For example, wind direction is unlikely to affect a user's decision whereas wind gust or wind speed might. We settled on temp, *feels_like*, *precip*, *snow*, *windgust*, and didn't use more because we thought they wouldn't improve results and it made any model training very computationally expensive. Reducing irrelevant features also reduces the chance of over fitting.

As seen in Figure 1 our features are very weakly correlated to each other. This implies that there is very little multicollinearity and, given good predictions can be made, they can be assumed to complement each other and the model can be assumed to be capturing non-linear patterns effectively. The redundancy is minimised but the risks associated with this are that features might have very low predictive value and no good model is easily found.

4.3 Models

4.3.1 Random Forest Regression

Random Forest Regression is an ensemble learning method that combines the predictions of multiple decision trees to produce a robust and accurate model. Each tree in the forest is trained independently on a bootstrap sample of the training data, ensuring diversity among the trees. To further enhance variability, only a random subset of features is considered at each split within a tree. The final prediction for regression tasks is obtained by averaging the outputs of all the trees, which reduces variance and makes the model resilient to noise and over fitting.

This algorithm excels at capturing complex, nonlinear relationships between features and the target variable without requiring explicit transformations. Random Forest is robust to outliers and missing data, making it suitable for a wide range of applications. Additionally, it provides interpretable feature importance measures, allowing practitioners to understand which features contribute most to the model's predictions. Despite its strengths, Random Forest can be computationally intensive due to the need to train multiple trees, and the overall model may lack the interpretability of simpler algorithms.

Throughout the whole evaluation process, Random Forest Regression remained the most accurate at each step achieving an R^2 Score of 90.8%. As seen in 5, location was the most important feature, followed by the hour of the week and then weather features.

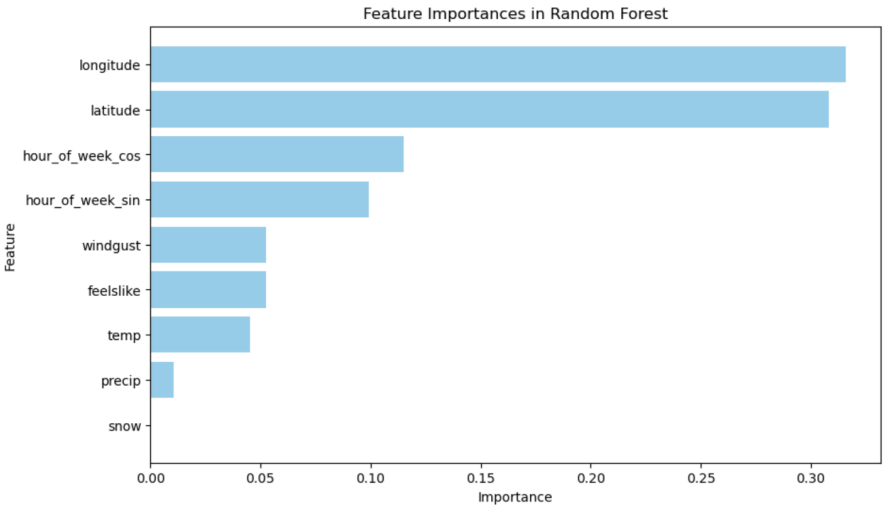


Figure 2: Random Forest Regression Feature Importance

4.4 Time-Series Forecasting Approach

For our time-series forecast, we used three different levels of data aggregation to make predictions:

1. A single station.
2. A group of stations around Falguière (7 stations).
3. All stations combined.

Additionally, we explored different prediction targets:

- The total number of bikes available.
- The total number of mechanical bikes.
- The total number of electric bikes (e-bikes).

We experimented with these combinations to identify the model and aggregation approach most likely to yield accurate results. For model evaluation, we predicted one week of data based on two weeks of the training dataset, equivalent to 70% training and 30% testing.

Our methodology was informed by the papers referenced in the report and external articles from Medium, which, while focused on slightly different datasets, addressed similar tasks (Deladrière, n.d.). And also from research papers for model intuitions. For clarity and consistency, we primarily used the Falguière dataset in this report due to its superior performance and ability to quantify errors clearly.

ARIMA

ARIMA, or Autoregressive Integrated Moving Average, is a statistical modeling approach commonly used for analyzing and forecasting time series data. Despite being developed in the 1970s, it remains one of the most widely used techniques for time series forecasting due to its simplicity and effectiveness. We chose to start with this model as it serves as the foundation for time series analysis.

To achieve reliable forecasts and apply statistical inferences, ARIMA requires the data to be stationary As shown in Figure 3, the different properties of stationary and non-stationary time series are illustrated.

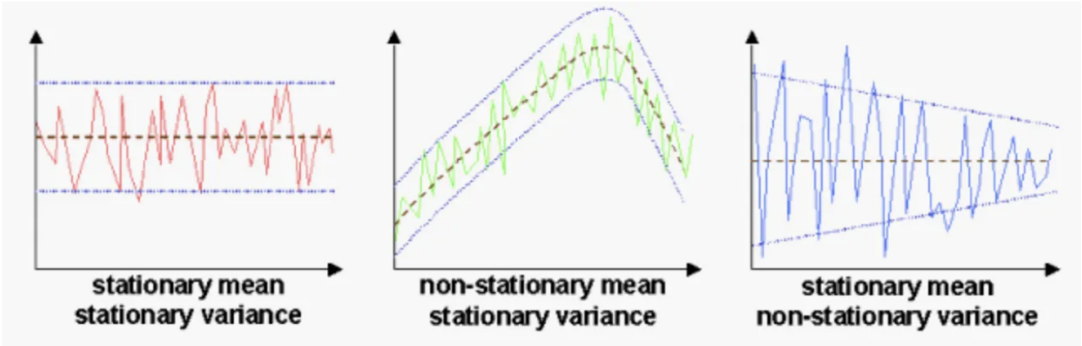


Figure 3: Illustration of stationary and non-stationary time series. The first shows stationary mean and variance, the second shows non-stationary mean with stationary variance, and the third shows stationary mean with non-stationary variance (Affeck, 2019).

Stationary data means that the mean and variance of the time series remain consistent over time. To determine if a time series is stationary, we performed an Augmented Dickey-Fuller (ADF) test. A result below the 5% threshold indicates that the data is statistically significant and stationary. Our time-series data showed a very low ADF test result, confirming it's stationary.

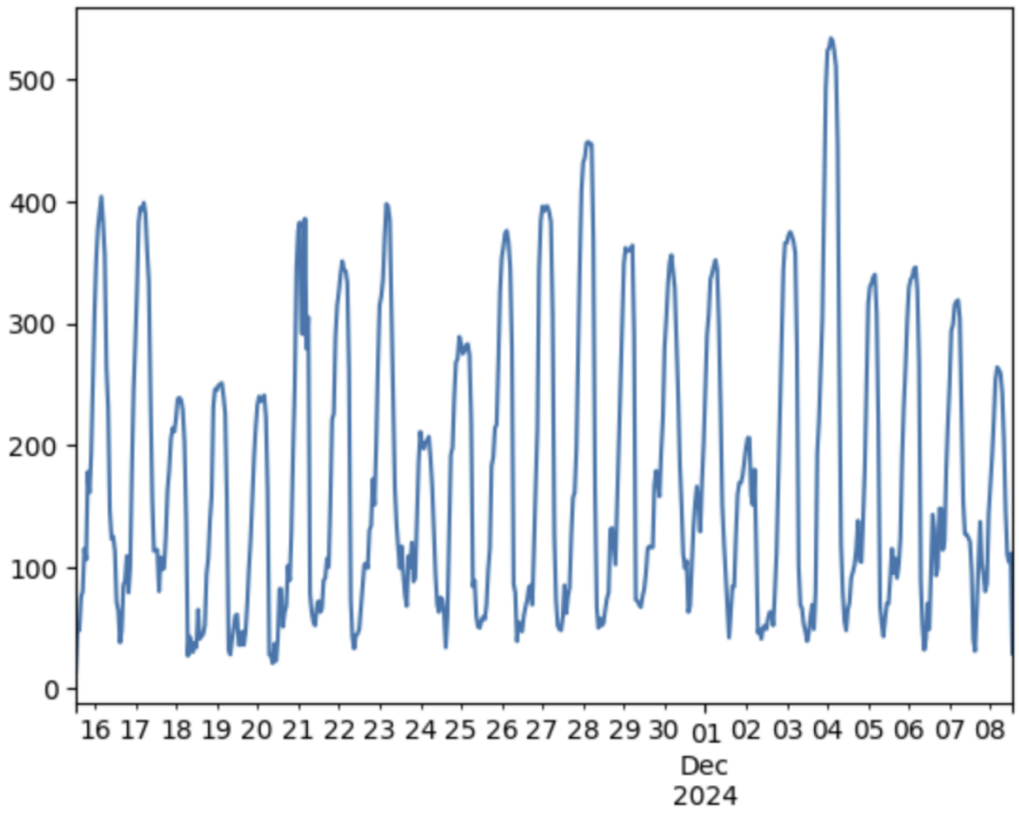


Figure 4: Time-series plot showing stationarity of our time serie.

If the time series is not stationary, it must be differentiated. Differentiation involves calculating the rate of change between consecutive data points, removing trends, and stabilizing the mean and variance.

$$\Delta x_t = x_t - x_{t-1} \tag{4}$$

Moreover, an ARIMA model requires two additional parameters:

- **Moving Average (MA)**: Represented as q, the MA component models the current value x_t as a linear combination of the past q error terms. In order, to optimize this parameters we need to look at the ACF graph and determine the number of lags corolated.

$$x_t = c + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q}$$

Where:

- ϵ_t: Current error term (white noise).
- θ_i: MA parameters (coefficients for lagged error terms).
- ϵ_{t-i}: Lagged error terms.

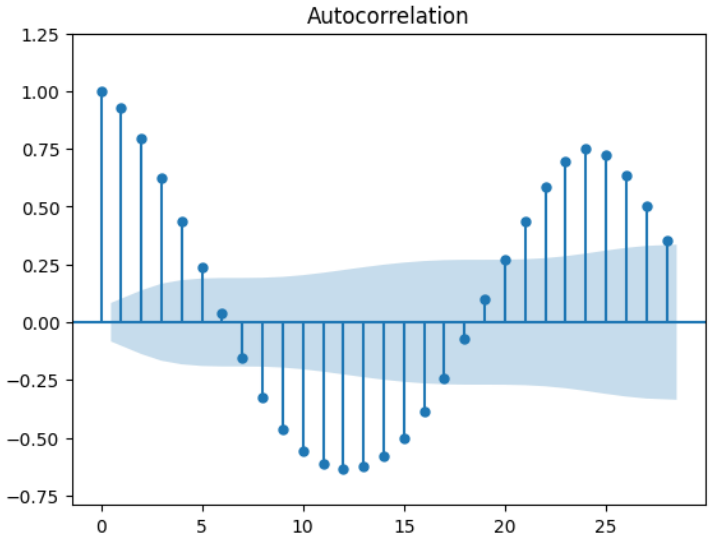


Figure 5: ACF graph for partial correaltion

- **Autoregressive (AR)**: Represented as p, the AR component expresses the current value x_t of the time series as a linear combination of its past p values. To optimize this parameter, we need to analyze the PACF graph and determine the number of significant lags.

$$x_t = c + \phi_1 x_{t-1} + \phi_2 x_{t-2} + \cdots + \phi_p x_{t-p} + \epsilon_t$$

Where:

- x_t: Current value of the time series.
- c: Constant term.
- ϕ_i: AR parameters (coefficients for lagged terms).
- x_{t-i}: Lagged values of the time series.
- ϵ_t: White noise (random error term).

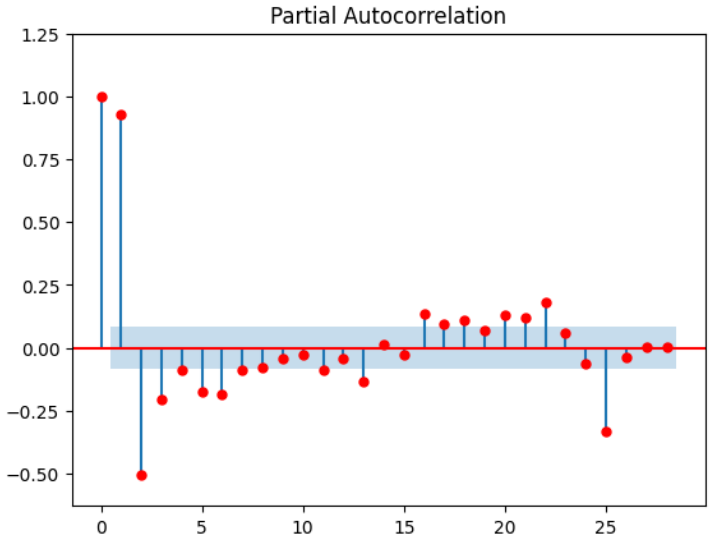


Figure 6: PACF graph for partial correlation

In our case, it appears that a high number of lags must be considered, indicating that our time series depends heavily on many past values.

A useful feature of the ARIMA model is the **Auto-ARIMA** function, which operates similarly to the Grid Search method discussed in class. By providing initial values for p, q, and d, Auto-ARIMA generates multiple time series models and estimates the best-fitting model. The optimal ARIMA model is determined by minimizing the Akaike Information Criterion (AIC). The AIC metric evaluates model fit and complexity, penalizing overfitting to enhance the statistical validity of the ARIMA model on the dataset.

$$\text{AIC} = -2\ln(L) + 2k$$

Where:

- L: The maximum likelihood of the model (how well the model fits the data).
- k: The number of parameters in the model (including both AR and MA terms, and the variance of the errors).
- ln: Natural logarithm.

While searching for the best model on our time series with found the best parameters of ARIMA(AIC = 3900, p = 3, d= 0, q = 1). The model provided us with a baseline for predictions to improve upon. However, as observed visually, the forecast fails to capture the seasonality and trends present in the data. Instead, it predicts a constant increase in the number of bikes over time and struggles to account for the trend in the test data after training.

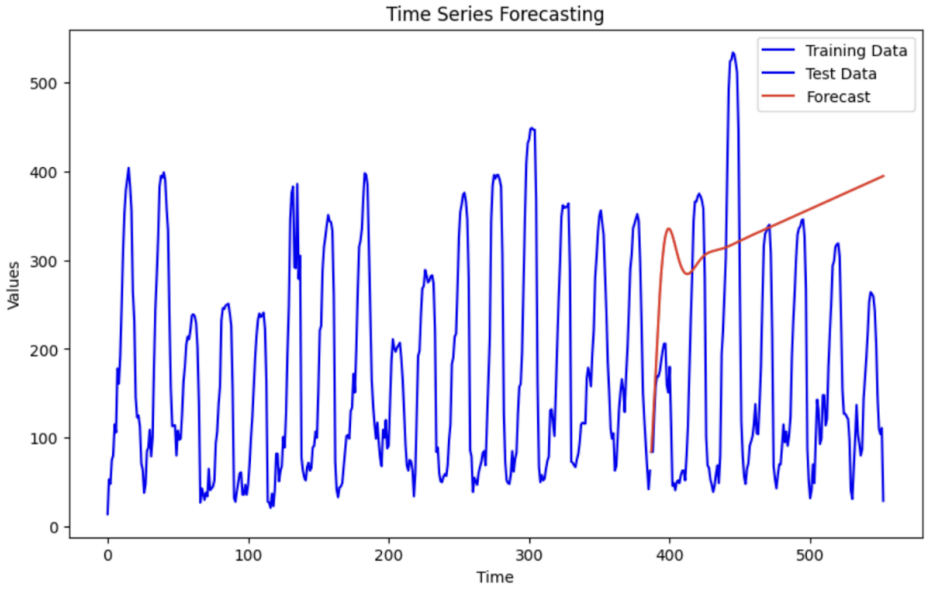


Figure 7: Time Series Forecasting: The graph shows the training data, test data, and forecast. The forecast fails to capture the seasonality and trends, resulting in a constant increase over time.

SARIMA

Knowing that our simple ARIMA model was unable to capture the seasonality in our data, we transitioned to a SARIMA model, which integrates seasonality.

To account for seasonality in our model, we subtracted the seasonal component of our time series using the `seasonal_decompose` method from the Statsmodels library. It was evident that our data exhibited strong seasonality, which needed to be removed for the statistical model to better capture the relationships between lags and improve future predictions.

After deseasonalizing the data, we revisited the PACF and ACF graphs. We observed that the variation in the number of bikes available was strongly influenced by its value 24 hours prior. Therefore,

we set the seasonal parameter $S = 24$, representing the seasonality period. This allowed the SARIMA model to account for seasonal parameters P, D, Q, S, where P, D, Q are seasonal counterparts of the ARIMA parameters and are determined by analyzing the deseasonalized PACF, ACF, and stationarity.

To optimize our seasonal model's performance, we conducted a grid search across all parameters. This significantly improved our results in terms of AIC, MAE,MSE and MAPE. As shown, after tuning the SARIMA parameters, our model began to predict trends in our data. While it still struggled with high-variation hours, it successfully identified peaks in the data's seasonality and incorporated them into the training dataset.

The next step will involve including exogenous data in our model to capture potential regression, correlations, and explained variance in our data points and records.

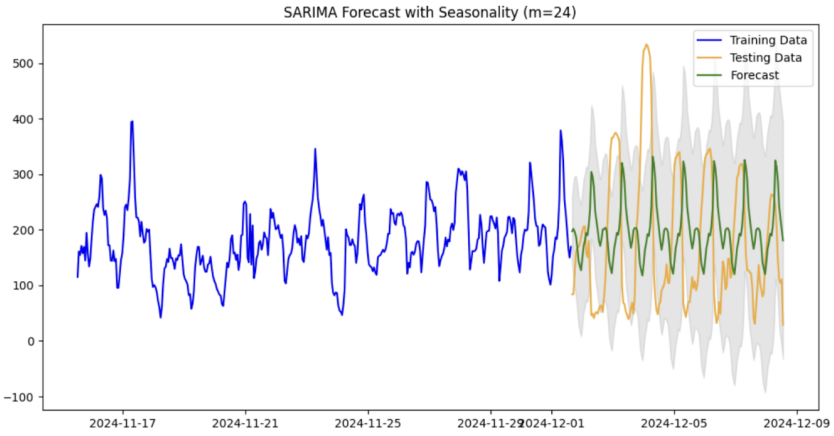


Figure 8: Sarima predictions

SARIMAX

To improve our time-series predictions over one week using two weeks of data, we incorporated seasonal features into the model. Specifically, we utilized SARIMAX, an extension of SARIMA that includes exogenous features (X) to better explain seasonality and patterns in the forecasts. This approach allows the model to include external factors that align with the same timestamps as the time series data, thereby improving prediction accuracy.

We decided to include the same features previously used in the Random Forest Regression model. After extensive parameter fine-tuning through grid searches and testing different types of time-series data (deseasonalized, raw, and differentiated), the SARIMAX model delivered the best results. It achieved the lowest MAE, MAPE, MSE, and AIC scores among all models tested.

Moreover, the forecast closely follows the observed patterns in the time series, representing a significant improvement over the initial ARIMA model and achieving greater accuracy than SARIMA.

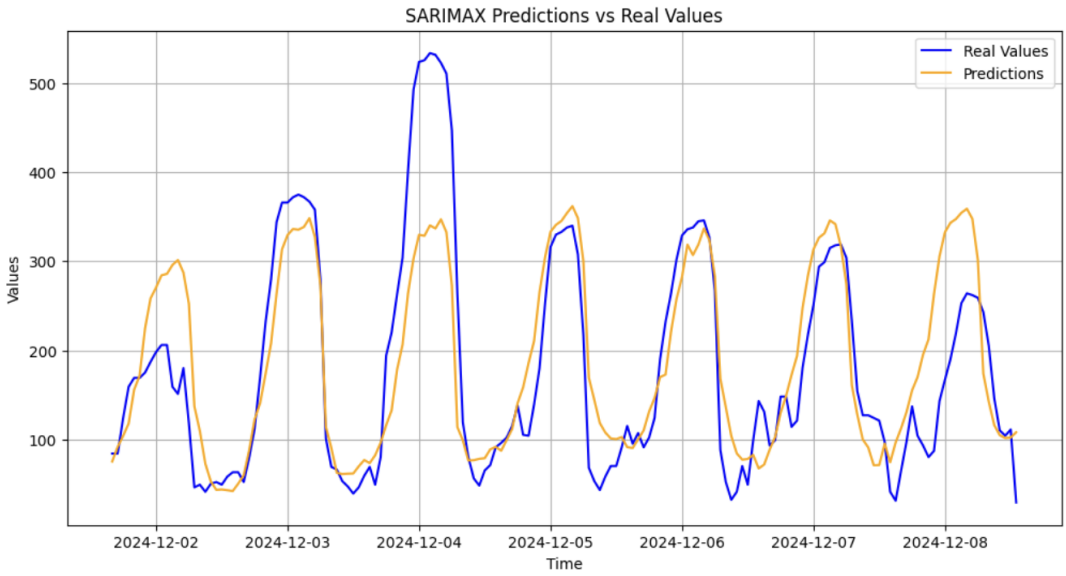


Figure 9: SARIMAX Predictions vs Real Values: The graph compares the predicted values (orange) with the real values (blue). The SARIMAX model effectively follows the overall trend and seasonality in the data.

Prophet

We also used pre-trained model to make our prediction. The first one we used is the Prophet model from Facebook shared open source. This pre-trained time-serie forecasting model requires the dataframes provided to be organized differently than the previous dataframes. It now requires the timestamp to not be as an index of the dataframe but as a column name ds and the time serie in column name y. We tried to run this model using only the time-serie at first and it provided us with the best results (lowest MAE, MAPE, MSE AIC not available). Also, the time for the fit was much faster than the time for fitting the hyperparameters tuning for the previous time series model. The behaviour of the prediction is close from the SARIMAX hypertuned but with slightly lower error levels

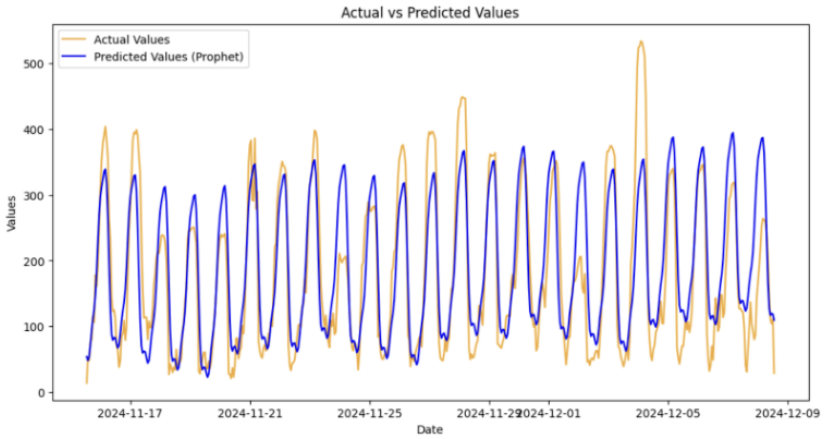


Figure 10: Actual vs Predicted Values: The graph compares actual values (orange) with predicted values from the Prophet model (blue). The model captures the overall trend and seasonality effectively.

Finally, we also tried the Prophet version using the exogenous features in it. Strangely, it did not provide better results, but still satisfying prediction.

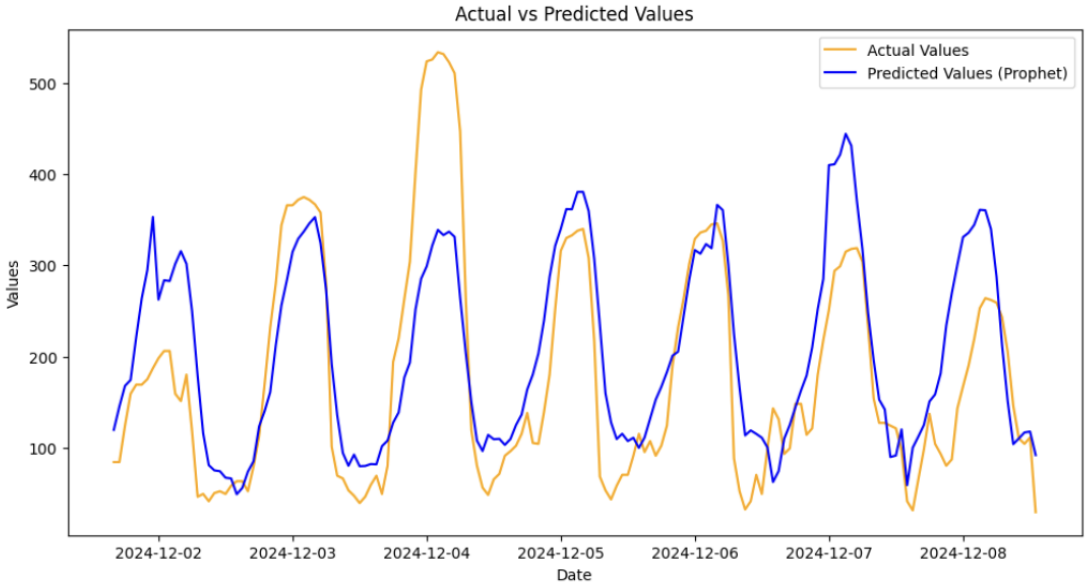


Figure 11: Actual vs Predicted Values: This graph compares the actual values (orange) with the predicted values from the Prophet model (blue). The model successfully identifies seasonal trends and general patterns in the dataset.

LSTM

Based on external studies we reviewed to improve our model performance [Reference], we ultimately opted to use a Deep Learning approach with an LSTM (Long Short-Term Memory) model. LSTM models are highly versatile and applicable in various domains, including image processing, video analysis, and natural language processing, due to their ability to retain "memory" over long-term data sequences. This feature allows them to maintain an adjustable focus on new batches of data while identifying patterns within the dataset.

Given that our dataset is relatively small, there is a risk of overfitting. However, even with a simple LSTM model, we achieved better results than those obtained from the previously used models.

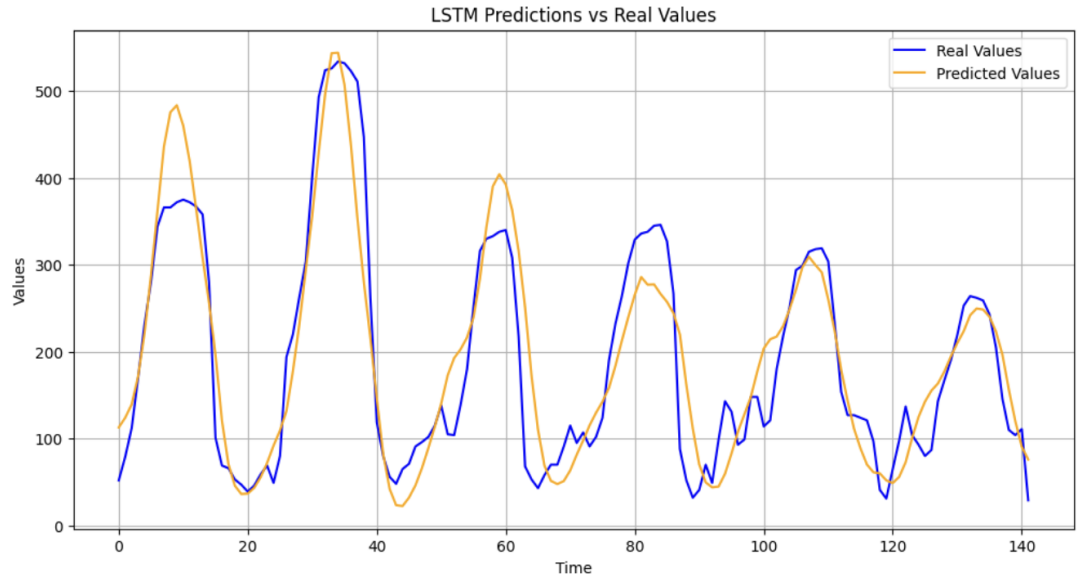


Figure 12: LSTM Predictions vs Real Values: The graph compares the predicted values (orange) with the actual values (blue) over time. The LSTM model effectively captures trends and general patterns in the dataset.

4.4.1 Other Models Attempted

XGBoost Regressor

The XGBoost Regressor demonstrated moderate performance in predicting the number of available bikes, with a root mean squared error (RMSE) of 5.09 and a coefficient of determination (R^2) score of 0.79. This indicates that the model explains 79% of the variance in the data, though its error rate is higher compared to the Random Forest model. XGBoost's ability to handle complex interactions between features makes it a robust choice for regression tasks, but in this case, it was slightly less effective than Random Forest, likely due to the specific characteristics of the dataset or the tuning of hyperparameters. Despite this, XGBoost remains a strong contender for modeling given its balance of performance and computational efficiency.

LightGBM Regressor

The LightGBM Regressor showed relatively poor performance in predicting the number of available bikes, with a root mean squared error (RMSE) of 7.02 and a coefficient of determination (R^2) score of 0.61. While the model explained 61% of the variance in the data, its high error rate suggests it struggled to capture the underlying relationships effectively. LightGBM is generally known for its computational efficiency and ability to handle large datasets, but in this case, its performance may have been hindered by the specific characteristics of the data or insufficient hyperparameter tuning. Despite its lower accuracy compared to other models, LightGBM remains a viable option for scenarios requiring speed and scalability.

Linear Regression

The Linear Regression model performed the poorest in predicting the number of available bikes, with a root mean squared error (RMSE) of 5.86 and a coefficient of determination (R^2) score of 0.20. This indicates that the model explained only 20% of the variance in the data, highlighting its inability to capture the complexity and non-linearity of the relationships between the features and the target variable. Linear Regression's simplicity makes it a useful baseline model, but its assumptions of linear relationships between variables limit its effectiveness in this scenario. The results suggest that more complex models, such as Random Forest or XGBoost, are better suited for this problem.

CNN

1D CNN should be accurate for analyzing time related series based on external sources and articles (Deladrière, n.d.), however, for us it performed badly due to lack of data and unfit task.

5 Evaluation

5.1 Regression Models

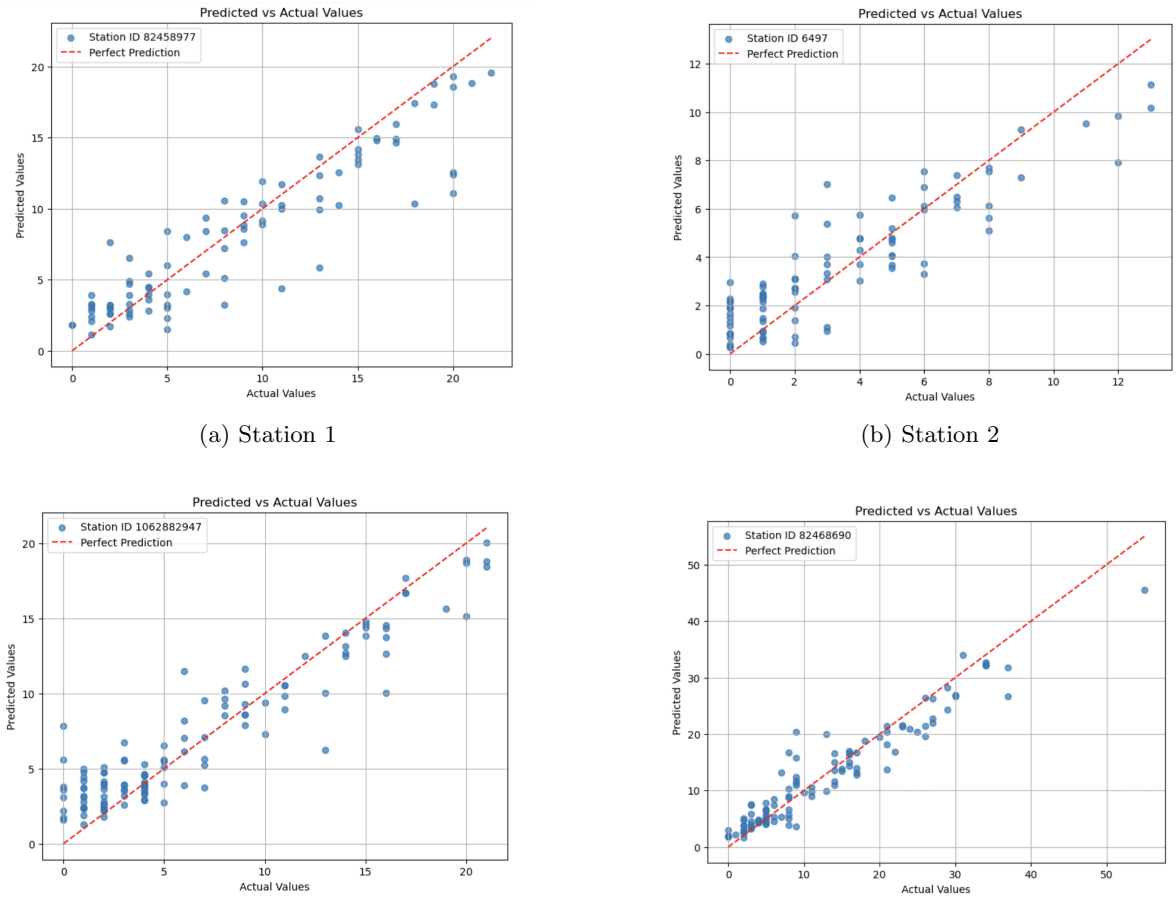
The results demonstrate that the Random Forest Regression model outperforms the other methods in terms of predictive accuracy, achieving the lowest RMSE (3.39) and the highest R^2 score (0.91). This indicates that it explains 91% of the variance in the data while maintaining a low error rate.

XGBoost Regression shows moderate performance, with an RMSE of 5.09 and an R^2 score of 0.79, reflecting decent but less robust predictions compared to Random Forest. LightGBM Regression and Linear Regression perform significantly worse, with LightGBM achieving an RMSE of 7.02 and an R^2 of 0.61, and Linear Regression showing the poorest performance with an RMSE of 5.86 and an R^2 of 0.20, suggesting it struggles to capture the variability in the data.

Overall, the results highlight the effectiveness of Random Forest in handling the given dataset and feature set, making it the most suitable model for this problem.

Model	RMSE	R2 Score
Random Forest Regression	3.39	0.91
XGBoost Regression	5.09	0.79
LightGBM Regression	7.02	0.61
Linear Regression	5.86	0.20

Figure 13: RMSE and R2 Score for the regression metrics tested.



(a) Station 1

(b) Station 2

(c) Station 3

(d) Station 4

Figure 14: Predicted vs actual values for the 4 stations closest to the Falguiere campus using the Random Forest Regression.

To evaluate the performance of the regression models used to predict the number of available bikes at a station, we employed two metrics: the coefficient of determination (R^2) and the root mean squared error (RMSE). These metrics provide complementary insights into model performance, ensuring a comprehensive evaluation.

The R^2 score measures the proportion of variance in the dependent variable that is explained by the independent variables. It is defined as:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

where y_i represents the actual values, $\hat{y}_i$ the predicted values, and $\bar{y}$ the mean of the actual values. An R^2 value close to 1 indicates that the model explains a large portion of the variance, which is useful for comparing the goodness-of-fit across different models.

RMSE, on the other hand, quantifies the average prediction error in the same units as the target variable. It is given by:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Unlike R^2, RMSE provides an absolute measure of the model's prediction error, making it easy to interpret in practical terms. While R^2 helps determine how well the model explains variability, RMSE evaluates the magnitude of errors, ensuring predictions are not only statistically robust but also operationally accurate. Combining these metrics enabled us to select a model that performed well both in terms of explanatory power and predictive accuracy.

5.2 Time-Series Analysis

To evaluate our time-series models, we initially used the AIC, as mentioned in the methodology section. However, to compare different models comprehensively,especially since AIC is not applicable to deep learning models. We quantified the prediction error using metrics such as MAE, MSE, RMSE, and MAPE. The table below presents the results of our predictions for one week of data for the station closest to Falguière.

Model / Metrics	MAE	MSE	RMSE	MAPE	AIC
Prophet	48	4398	66,31742	None	None
Prophet with exogenous data	64	6796	82,43786	None	None
ARIMA Basic	178	41322	203,2781	211	3888
SARIMA	127	22566	150,2198	126	3050
SARIMAX	52	5162	71,84706	39	2526
LSTM	39	2763	52,56425	33	None
GRU/Bidirectional	44	3233	56,85948	34	None

Figure 15: TABLE of results showing different metrics to compare Time-Series models

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100$$

6 Conclusion

Although several objectives outlined in our proposal paper could not be met, we are satisfied with the results of our research. Specifically, we were unable to achieve clustering or account for potential malfunctioning bikes due to missing or prohibitively expensive data. However, we made significant progress in predicting time series and monitoring correlations with exogenous variables through feature engineering in regression models.

We achieved improved results in iterative regression and time series analysis, all using only three weeks of data. Our models are now capable of providing decent predictions for one week of bike availability in specific areas or at individual stations, incorporating external features such as weather conditions.

Moreover, we learned that developing an intuition for the data—through visual representations—can greatly aid in designing models and setting research on the right track.

The next step would involve gathering more data over a longer period to enable predictions spanning months of bike demand. Additionally, exploring advanced techniques, such as implementing transformers or other time-series-dedicated libraries like DARTS, could further enhance our prediction models.

7 References

References

[1] Visual Crossing. (n.d.). *Visual Crossing weather data*. Available at: https://www.visualcrossing.com/weather-history/Paris%2CFrance

[2] Bergamasco, F. (n.d.). *The importance of data visualization for initiating a data science project*. Medium. Available at: https://medium.com/@bergamasco.florian/the-importance-of-data-visualization-for-initiating-a-data-science-project-662fd8ac7093

[3] Deladrière, H. (n.d.). *Bike demand forecasting — time series*. Medium. Available at: https://hugodeladriere.medium.com/bike-demand-forecasting-time-series-8d5f581dfde3

[4] Velib API. (n.d.). *Velib API to get real-time Velib data*. Available at: https://velib-metropole-opendata.smovengo.cloud/opendata/Velib_Metropole/station_status.json

[5] Velib API. (n.d.). *Velib API to get the latitude and longitude of stations*. Available at: https://velib-metropole-opendata.smovengo.cloud/opendata/Velib_Metropole/station_information.json

[6] Statista. (2019). *Distribution des personnes disposant d'un abonnement Vélib' en Île-de-France en 2019, selon l'âge*. Available at: https://fr-statista-com.essec.idm.oclc.org/statistiques/1219170/usagers-velib-ile-de-france-age/

[7] ScienceDirect. (2023). *Bike sharing paper and time series analysis*. Available at: https://pdf.sciencedirectassets.com/271506/1-s2.0-S0957417423X0030X/1-s2.0-S0957417423027665/main.pdf

[8] Marvel, J. (n.d.). *Forecasting Citibike Ridership*. Medium. Available at: https://medium.com/@jeffmarvel/forecasting-citibike-ridership-bdf72f97853a

[9] AI4SM. (n.d.). *Predicting bike availability at bike share stations in Toronto: Preliminary analysis*. Medium. Available at: https://medium.com/ai4sm/predicting-bike-availability-at-bike-share-stations-in-toronto-preliminary-analysis-cea5d849d63

[10] ScienceDirect. (2017). *Moment-based availability prediction for bike-sharing systems*. Available at: https://www.sciencedirect.com/science/article/abs/pii/S0166531617300822

[11] Bergamasco, F. (n.d.). *Velib' part 2 — Implementation of Data Science models to predict Vélib' utilization*. Medium. Available at: https://medium.com/@bergamasco.florian/velib-part-2-implementation-of-data-science-models-to-predict-v%C3%A9lib-utilization-64654065a986

[12] Reddy, S. (2010). *Urban cycles and mobility patterns: Exploring and predicting trends in a bicycle-based public transport system*. ScienceDirect. Available at: https://www-sciencedirect-com.essec.idm.oclc.org/science/article/pii/S1574119210000568