



Demand Forecasting Short-Term Ecommerce Sales Using Machine Learning Models

Author: Yehia Heiba

First supervisor: Dr. Sieuwert van Otterloo

Second reader: Dr. Jacco van Ossenburggen

Bachelor Thesis

Department of Computer Science

Vrije Universiteit Amsterdam

In Fulfillment of the Requirements

August 2024

Abstract

The book publishing industry faces significant challenges in logistics and planning, particularly with the rise of Print-on-Demand (POD) technology. POD offers numerous benefits, such as reduced stock risk and faster delivery times, but it also requires precise demand forecasting to align resources effectively. This thesis explores the application of various machine learning models to predict the ecommerce order volumes for CB, a Dutch logistics company serving the book industry (www.cb.nl/media). The study evaluates the performance of five models: ARIMA, Prophet, BILSTM, XGBoost, and Vertex AI AutoML. The XGBoost outperformed the other models with a reduction of 24% on the RMSE and 36% on the MAE when compared to the ARIMA (baseline model).

The XGBoost is used to predict the sales for each book title. The models were trained and validated using historical sales data and Google search data. Integrating Google search trends is especially useful for new books and sales spikes. This caused a 50% reduction in MAE compared to the XGBoost model, which doesn't use Google Searches this month. The findings indicate that while traditional time series models like ARIMA have limitations in capturing sudden fluctuations, advanced machine learning models, particularly tree-based models like XGBoost, provide more accurate and stable forecasts. This research highlights the potential of integrating machine learning solutions and Google Searches into logistics planning to enhance e-commerce businesses' operational efficiency.

Table of Contents

1. Introduction to Demand Forecasting.....	5
1.1 Background and Context of CB.....	5
1.2 Central Issue.....	6
1.3 Data Provided.....	6
2. Related Work.....	10
2.1 Models.....	10
2.1.1 Base Model - ARIMA.....	10
2.1.2 BILSTM.....	13
2.1.3 Transformer.....	15
2.1.4 XGBoost.....	16
2.1.5 Prophet.....	16
2.1.6 Vertex AI AutoML.....	18
2.2 Predicting on a Unit Basis.....	19
2.3 Validation and Evaluation.....	21
3. Research Methods.....	25
3.1 Proposed Models and Process.....	25
3.2 Model Parameters and Hyperparameter Tuning.....	28
3.2.1 ARIMA.....	28
3.2.2 Prophet.....	28
3.2.3 BILSTM.....	29
3.2.4 XGBoost.....	30
3.2.5 AutoML.....	30
3.3 Unit Basis Prediction.....	30
3.3.1 Google Keywords.....	31
3.4 Validation and Evaluation.....	33
4. Results.....	34
4.1.1 Results for the Models.....	34
4.1.2 Results for Unit Predict Model.....	38
4.2 Model Parameters and Hyperparameter Tuning.....	44
4.2.1 ARIMA.....	44
4.2.2 Prophet.....	44
4.2.3 BILSTM.....	45
4.2.4 XGBoost.....	45
4.3 XGBoost for Unit Prediction.....	46
5. Discussion.....	47
5.1 Models.....	47
5.1.1 ARIMA - Baseline.....	47
5.1.2 Prophet.....	48
5.1.3 BILSTM.....	48

5.1.4 XGBoost.....	49
5.1.5 AutoML.....	49
5.2 Unit Prediction XGBoost.....	50
5.3 Limitations.....	51
5.4 Further Improvements.....	52
6. Conclusion.....	53
References.....	54
Appendix.....	57
Appendix A.....	57
Google Keyword Planner Walkthrough.....	57
Appendix B.....	60
Uploading a file to Google Vertex AI.....	60
Training the model.....	65
Appendix C.....	69
Code:.....	69

1. Introduction to Demand Forecasting

1.1 Background and Context of CB

Effective logistics and planning are critical for the book publishing industry, where timely delivery, reduced inventory risk, and responsiveness to customer demand are key factors for success (McIlroy, 2015). Print-on-demand (POD) technology has emerged as a promising solution to address these challenges, offering benefits such as reduced stock risk, sustainable use, and fast deliveries. However, implementing POD requires careful alignment of resources and accurate prediction of order volumes - tasks that can be difficult to manage manually.

The Dutch logistics company CB, a centralized storage, processing, and delivery center for the book industry, has heavily invested in POD technology to address these challenges. As a leading provider of logistics services to booksellers in the Netherlands and Belgium, CB's experiences and approaches to leveraging POD and other innovations offer valuable insights into the practical realities of implementing advanced logistics solutions in the book publishing sector.

POD provides less capital risk as no book is printed without a demand for it. This eliminates the case of leftover unsold books, which account for 30% of the books and contribute to losses and storage costs (Birkenshaw, 2003). It also offers the booksellers the opportunity to always have their books in stock as they can always order them with next-day delivery. However, POD has cheaper production costs per unit at lower quantities but becomes more expensive when it comes to scaling >1000 books per order (Keif, 2007).

POD could be quite convenient for the end customer but quite a hassle for CB as they have a limited time span to satisfy their end consumer. It requires CB to correctly align the number of employees and materials required according to the expected number of orders. This task is done manually; hence, the company fails to correctly predict the range of expected orders. With the ongoing revolution regarding AI, CB believes that such a technology can be the solution to their continuous pain.

CB offers bookstores a dashboard for analytics to measure the turnover of the books and guide the bookstores' ordering process. Their data storage contains information about the orders since 2006. They also offer (B2C) solutions, where online shoppers can order a book through bol.com, and they would fulfill the order on demand. According to our interviews with CB, the staff believes it would be more difficult to predict consumer behavior as they believe there is no distinct pattern. However, their core business is (B2B), and selling to bookstores is deemed to be more predictable as a business is less likely to order a book on holidays, for example.

1.2 Central Issue

CB doesn't have a program that forecasts the possibility of incoming orders and faces severe consequences from irregularities in orders in terms of scheduling employees. With the sudden growth of AI, machine learning models could provide a possible solution to CB. Hence, as a team, our aim is to help CB achieve its goal by possibly forecasting orders. For the B2C sector, CB would like to examine consumer behavior and how it affects the sales of books. The model will only provide insights about incoming orders and will not examine the operational side of CB in terms of setting the schedule or managing the operation. In addition, the issue of integrating the model into their current system lies outside of the scope as this study solely focuses on the feasibility of an ML solution for forecasting orders.

1.3 Data Provided

The current data collected is split into 3 categories. The first is `sales_daily`, which contains the overall number of orders per day from January 1, 2017.

Figure 1.

sales_daily.csv

ONTVANGST_DATUM (Date of orders)	DISTRIBUTION_CHANNE L (B2C - E-Commerce) (B2B - Bookstores)	EXEMPLAREN_AANT (number of copies)
16-06-2020 00:00	B2C	44427
5/8/2019 12:00:00 AM	B2B	201430
8/11/2017 12:00:00 AM	B2B	193427

Note. The total number of rows is 5204, which is equivalent to 365 days across about seven years, with two values for each day: B2C and B2B.

The second CSV is *articles.csv* which contains information about the book such as the book ID, title, publisher, and publication date. The following CSV file has 52 columns however, only the columns marked as important will be displayed. The publication date wasn't marked as an important column but was added as the release date could affect the book sale patterns.

Figure 2.

articles.csv

ARTIKEL_NR (Book ID)	BOEKSOORT (Book Genre)	NUGI (Item category) https://www.boek.nl/nur	TITEL (Book Name)	VERSCHIJNINGSDAT (Publication Date)
12665604	A	12	Monsters	17-11-2023 00:00
521993	W	730	Vertoog over de ongelijkheid	14-05-2003 00:00

521993	S	163	Bedrijfsinformat iesystemen	7/11/2017 12:00:00
--------	---	-----	--------------------------------	-----------------------

Note. The file contains 368,956 rows, which represent the total number of unique books.

The third is `Orders_2024`, which contains each order on a daily basis, and the distribution channel B2C or B2B is split among 7 CSVs, one for each year. There are ten columns however, the four relevant columns are the book ID, date, amount ordered, and sales channel. The following example is from `Orders_2024`

Figure 3.

Orders_2024.csv

ONTVANGST_DTD (Datetime of Order)	EXEMPLAREN_AANT (Amount Ordered)	ARTIKEL_KD (Book ID)	DISTRIBUTION_CC CHANNEL (B2C - E-commerce) (B2B - Bookstores)
31-JAN-24 21:27:53	1	9789403205786	B2C
29-JAN-24 17:37:34	1	9789053527979	B2B
08-FEB-24 09:11:09	1	9789043032018	B2C

Note. `Orders_2024` is one of 7 other files from 2017 to 2024.

2017 has 15,345,568 rows.

2018 has 20,258,558 rows.

2019 has 20,357,570 rows.

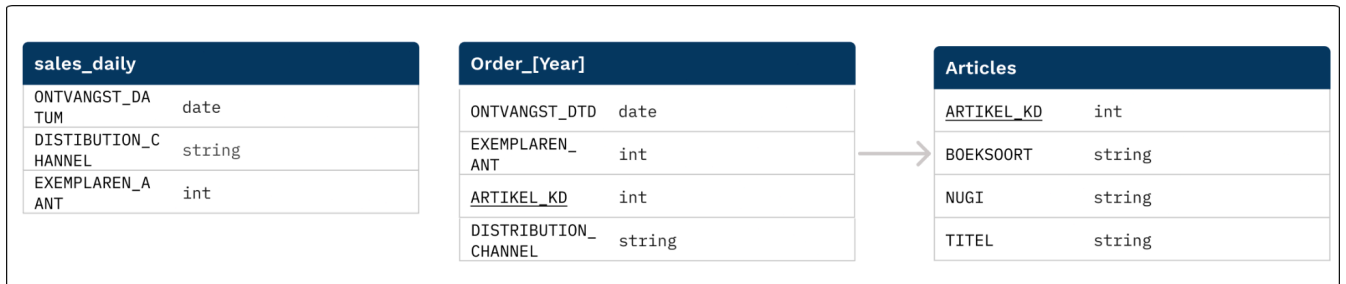
2020 has 22,445,843 rows.

2021 has 23,697,567 rows.

2022 has 22,313,284 rows.

2023 has 21,539,044 rows.

Figure 4.

UML diagram for data structure

2. Related Work

2.1 Models

2.1.1 Base Model - ARIMA

Currently, the emphasis in forecasting and machine learning has predominantly been on time series analysis ARIMA, which tends to involve relatively straightforward models (Kumar, 2021). Time series forecasting is a widely used approach for sequential data (Kumar et al., 2021). The Auto-regressive integrated moving average (ARIMA) is a statistical analysis model that uses time series data to predict future trends. ARIMA is composed of an Autoregressive (AR), which processes the previous information to the model, and a moving average (MA), which limits the noisy data of previous instances. ARIMA only works on stationary data points where there is a constant mean, variance and its autocorrelation structure does not change over time. There are two unit root tests that check the stationary process of the data, augmented Dickey-Fuller (ADF) and Philippe-Perron (PP) (Fattah et al., 2018). The ARIMA works best with linear data and is not suitable for complex non-linear data.

The following equation is the general form of the AR if the autoregressive coefficient is \varnothing and p is the order of the autoregressive part (the number of previous instances):

$$X_t = c + \sum_{i=1}^p \phi_i X_{t-i} + \epsilon_t$$

eq1. Autoregressive formula

Where:

- X_t Is the value of the series at the time t .
- c is a constant term.
- ϕ_i are the parameters of the model that weigh the influence of the lagged values.
- X_{t-i} are the past values of the series.
- ϵ_t is the white noise error term.

The following equation represents the MA part of the ARIMA, with a variable q input which determines how many past residual errors are used to model the current observation:

$$X_t = \mu + \epsilon_t + \sum_{j=1}^q \phi_j + \epsilon_{t-j}$$

eq2. Moving Average formula

Where:

- X_t is the value of the series at the time t .
- μ is the mean of the series.
- ϵ_t is the error term at time t .
- ϕ_j are the MA coefficients for the past errors.
- ϵ_{t-j} are the lagged error terms.

The integration part of ARIMA, also denoted as d , refers to the number of times the current series is different from the previous series to make it stationary. This helps in removing the trend and seasonality from the series.

$$\Delta X_t = X_t - X_{t-1}$$

eq3. ARIMA first difference (d)

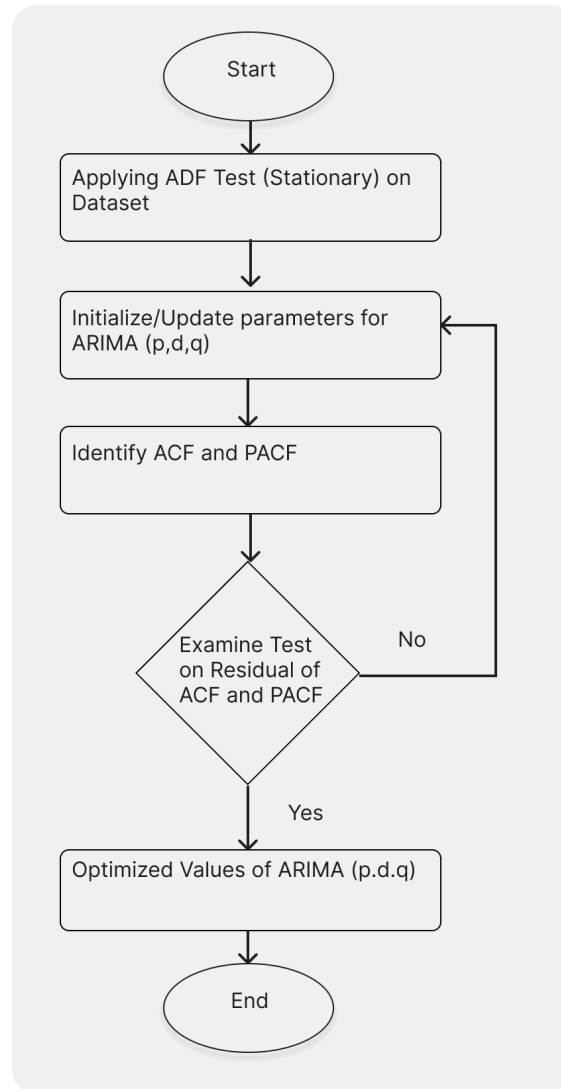
Where:

- X_t is the value of the series at the time t .

By integrating AR and MA components, the ARIMA model can adaptly handle complex and non-stationary time series data. The AR component leverages historical data points to forecast future values, while the MA component corrects future forecasts based on the error of past ones.

Figure 5.

Process of training an ARIMA model from start to end after preprocessing



The ACF measures the correlation between observations in a series and their respective lags, encapsulating both direct and indirect correlations across various time intervals. This function is instrumental in detecting the presence of autocorrelation and in determining the Moving Average (MA) component of an ARIMA model, thereby guiding the selection of MA terms for modeling.

Conversely, the PACF isolates the direct correlation between an observation and its lag, controlling for the influences of intervening observations. This nuanced measure is crucial for identifying the AutoRegressive (AR) component of an ARIMA model, specifying the number of AR terms required by filtering out the indirect effects captured by the ACF.

There are important considerations to make when using an ARIMA model. The model assumes that the future will resemble the past hence, it performs poorly with crises and periods of rapid technological change. It is not suitable for long-term predictions and it is also poor in identifying turning points due to the usage of a moving average component.

2.1.2 BILSTM

Recently AANs have been used such as LSTM, RNN, and CNN. In several instances, the LSTM has performed best on time series due to its context-aware structure (Shiri et al., 2023). The CNN performs better on image classification but not time series analysis. Unlike CNNs, LSTMs, an advanced variation of RNNs, are designed to mitigate the problem of diminishing weights for older data points.

Empirical studies show that deep learning algorithms such as LSTM outperform the ARIMA model with an 84 expected reduction in error (Siami-Namini et al., 2019). That is primarily due to preserving and training the features of given data for a longer period of time. Their study also suggests that BILSTM has a 37% increase in accuracy on average when compared to the LSTM. The exact model structure of the BILSTM, whether the model is a (right-to-left) or (left-to-right), doesn't affect the performance.

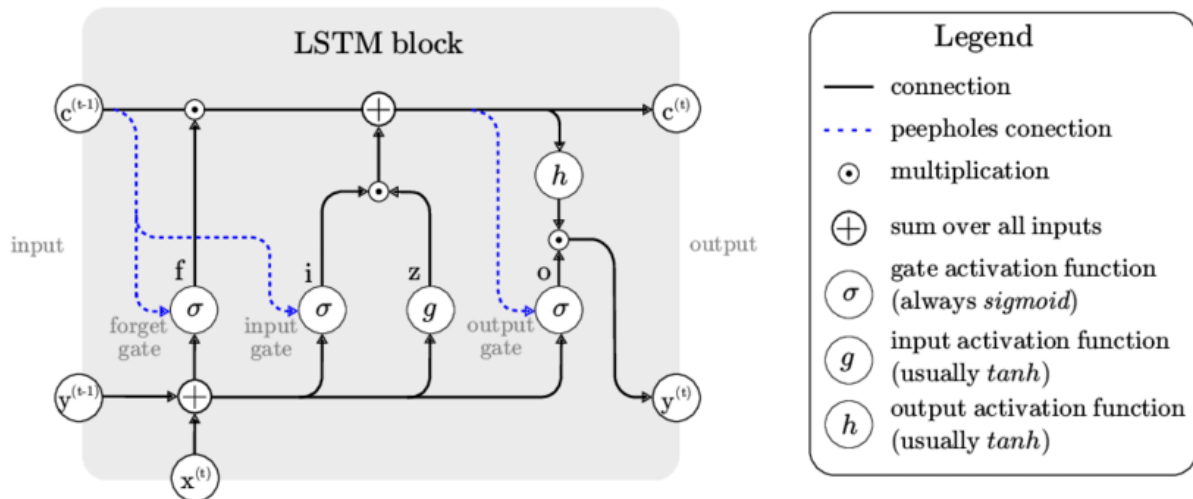
Krauss et al. (2016) reported that deep learning-based modeling under-performed gradient-boosted trees and random forests for forecasting tasks. Additionally, Krauss et al. (2016) report that training neural networks and, consequently deep learning-based algorithms is very difficult.

LSTM is a type of Recurrent Neural Network with the capacity to remember values from earlier stages (Van Houdt et al., 2020). A basic neural network is composed of three layers; an

input layer, a hidden layer, and an output layer. The number of nodes in the input layer is equivalent to the dimensions of the data and nodes of each layer are connected using weights. The hidden layer applies an activation function (e.g. sigmoid or tangent hyperbolic). A neural network basically learns by adjusting the weight for each iteration through a process called backpropagation. The errors are “backpropagated” into the network from the output layer towards the hidden layers and as a result, the weights are adjusted (Van Houdt et al., 2020). The recurrent neural network (RNN) is a special type of neural network that aims to predict the next step based on the previous steps. This is possible due to an internal storage allocated on the hidden layers. They have coined the term “recurrent” because they perform the same task for every element of the sequence while making use of the internal storage it has. However, RNNs are not suitable for longer sequences as they lack a memory line, which is solved by the LSTM. The memory line alongside some gates offers an LSTM that can memorize the sequence of data.

Figure 6.

LSTM block (Van Houdt et al., 2020)



Within the architecture of Long Short-Term Memory (LSTM) networks, sigmoid layers play a pivotal role by outputting values confined to the $[0, 1]$ interval. This operational characteristic signifies the extent to which data segments are permitted passage through each cell, with a score of 0 effectively indicating the complete blockade of data transmission, and a score of 1 facilitating unhindered passage. LSTM networks incorporate a triad of distinct

gating mechanisms, each tailored to modulate the cell state dynamically. The forget gate in a BiLSTM decides which information to discard from the cell state, while the memory gate selects new information to update the cell state. Determining the final output from each cell, the output gate integrates the current cell state with filtered and newly admitted data to produce the output value.

A new extension built on top of the LSTM is the Bi LSTM. It is a bidirectional LSTM hence, the input sequence is fed in a forward form and then in a reverse form. As a result, it improves long-term dependency and reduces the error on average by 37% (Siami-Niamini et al., 2019).

To implement the algorithm, the keras library Theano should be installed, and the parameters and the hyperparameters will be discovered through training. The “mean squared error” and “ADAM” are used as the loss function and the optimization algorithm, respectively (Siami-Namini et al., 2019). If the data provided is too big, it is better to use batch training and split the data into batches of about 250 instances each. As demonstrated in the study executed by Siami-Namini et al. (2019), there is no evidence of the number of epochs and training on the same dataset affecting the accuracy of the prediction.

2.1.3 Transformer

Transformers are the model chatGPT uses, hence the name Generative Pretrained Transformer. The main strength of transformers is the multi-head self-attention mechanism (Zeng et al., 2023). It is usually used for extracting semantic correlations among large sequences, either texts or images. The only downfall is that self-attention is permutation invariant; while it can preserve some ordering in the information, it still has temporal information loss. This is not a concern for NLP tasks where ordering is less important than semantic meaning. However, in time series tasks, the order of the data significantly affects pattern recognition. Based on Zeng et al.'s (2023) findings, a transformer cannot perform better than a linear model; therefore, it will not be further examined.

2.1.4 XGBoost

Bohdan Pavlyshenko (2019) mentions that regression models can often exceed time series models in forecasting. Regression models are trained to predict future patterns based on historical data. Hence, the future patterns are demand forecasts in this scenario.

As stated by Chen et al. (2016), “XGBoost scales beyond billions of examples using far fewer resources than existing systems.” The concept of extreme boosting is built on top of gradient boosting, where a powerful ensemble technique corrects errors that have been made by previous models sequentially. Hence, it optimizes for the loss while maintaining flexibility and performs well on diverse datasets. The added benefit of the XGBoost is the use of L1 and L2 regularizers that penalize model complexity and overfitting (Chen et al., 2016). Subsequently, it has a sparsity-aware algorithm for split finding to work with the sparse data caused by regularizers. Lastly, efficient tree pruning allows XGBoost algorithms to stop growing using the gain on the training loss, even if the maximum depth hasn't been reached yet. Just like any other tree-based ML model, the tree can handle both numerical and categorical features. In this case, the features of the XGBoost are an aggregation of different factors, such as book genre, with the lagged target variable similar to the one used by Pavlyshenko (2019).

2.1.5 Prophet

In 2017, Meta, which was Facebook at that time, launched an open-source forecasting tool, the FB Prophet. Similar to an ARIMA, the model takes into account the trend, seasonality, and noise. It is specially designed to meet the requirements of businesses as it also takes into account holidays, break intervals, and outlier detection (Singh et al., 2020). The model uses modular regression for the growing trend, a Fourier series for the seasonal component, a weekly cyclical component, and user-recommended holidays. The FB prophet uses the Bayesian model to fit the curve for the trend and doesn't require much time-series data. In the study done by Chan (2020), the prophet model was used to predict stock prices. The research findings suggested that ARIMA outperformed the prophet model in daily to weekly transactions. At the same time, the prophet model outperforms on longer periods, from monthly to yearly.

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t$$

eq4. Prophet Model Equation (Kumar Jha, 2021)

Where:

- $y(t)$ is the value of the time series at time t .
- $g(t)$ is the trend component, the general change over time.
- $s(t)$ is the seasonal component that captures the periodic changes.
- $h(t)$ accounts for the effects of holidays and special events.
- ϵ_t is the error term.

$$g(t) = (k + a(t)^\top \delta)t + (m + a(t)^\top \gamma)$$

Eq5. Trend Component $g(t)$

Where:

- k is the initial growth rate.
- m is the initial offset.
- δ is the rate of change.
- γ is the offset change.
- $a(t)$ is an indicator function that handles changes in the trend.

$$s(t) = \sum_{n=1}^N (a_n \cos(\frac{2\pi nt}{P}) + b_n \sin(\frac{2\pi nt}{P}))$$

Eq6. Seasonality Component $s(t)$

Where:

- P is the period window (7 for a week, 31 for a month, etc.)
- N is the number of terms in the Fourier series.
- a_n, b_n are the coefficients.

$$h(t) = \sum_{i=1}^K \alpha_i D_i(t)$$

Eq7. Holiday Component $h(t)$

Where:

- K is the number of holidays/events.
- α_i is the effect of the i -th holiday.
- $D_i(t)$ is an indicator function for the i -th holiday.

$$\epsilon_t$$

Eq8. Error Component ϵ_t

Where:

- The error term ϵ_t captures the residuals or noise in the data. Prophet models assume that errors are normally distributed.

2.1.6 Vertex AI AutoML

A currently existing product for forecasting is the Vertex AI AutoML solution (Google Cloud, 2024). The user simply uploads the historical data and assigns each column a meaning. The user does not need any technical background or knowledge about ML models to conduct the experiment. Vertex AI trains, validates, and deploys the model. Like other AutoML, it doesn't simply use a single model; however, it combines different models with varied parameters and chooses one that performs best on the validation set. The model can then be used for predictions using the console or API integrations. Back in 2021, AutoML solutions didn't significantly outperform naive models such as the Holt-Winters (Paldino, 2021). There was a slight improvement when tested on short-term forecasting. However, the AutoML solutions have evolved significantly since.

2.2 Predicting on a Unit Basis

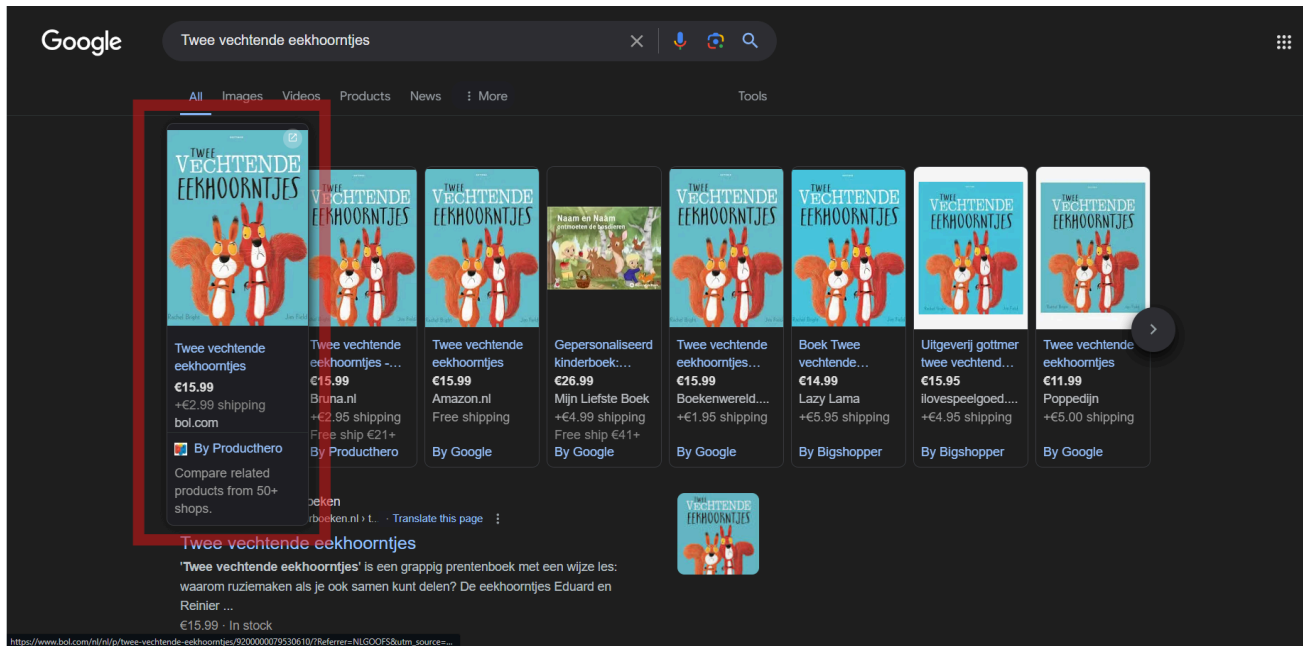
The use of external data can explain the behaviors of consumers online (e-commerce shoppers) and try to build an ML model to predict order patterns. Metrics such as the number of searches or the virality of a certain book may provide essential data regarding short-term trends. The key stakeholders in this problem are CB, Bol.com, Google.com, and the end consumer. CB sales through bol.com and Google Ads were responsible for a 50% increase in bol.com sales

(bol.com, 2020). In addition, there is a lag period of about a week between the user's first search and the purchase event, and it can sometimes reach up to three weeks (Lehmann, 2016). Google and Bol.com serve as intermediaries that link the end consumer looking to purchase a book with CB, the supplier. The proposed solution should not offer an analysis on the same day, as it could lead to biased or irrelevant results, which wouldn't benefit CB. Rather, it should predict future trends for the days ahead.

When examining the task, one could assume that there is a strong correlation between the online searches for a specific book and the books sold on bol.com (CB's online e-commerce partner). As seen in *Figure 7* below, when choosing a random book from the list of books, Twee Vechtende Eekhoortjes, in this case, Bol.com usually appears the first.

Figure 7.

Consumer View of Bol.com's Search Ads

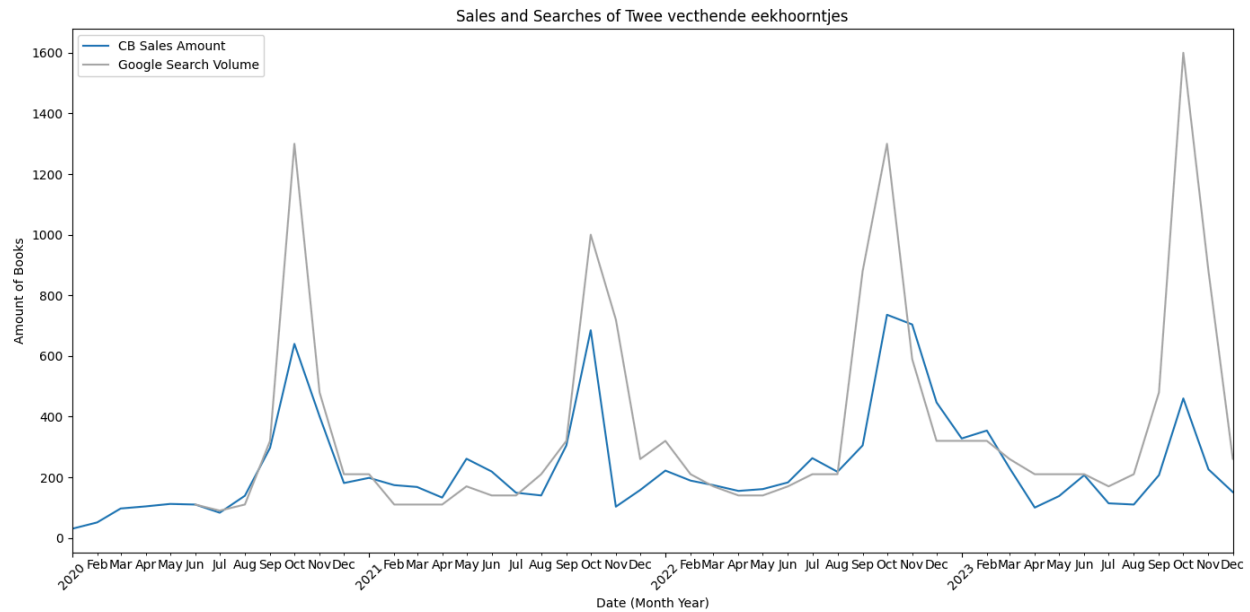


The number of searches for each book can be extracted from the open-source Google platform called Keyword Search Planner (<https://ads.google.com/aw/keywordplanner/>). This would align with the number of orders as the dataset lies between 2017-2024. There is always a lag period between the first search a user makes and the day they buy a product, as depicted in

the following sources, where the correlation between Google Trends and TV sales was examined (Lehmann, 2016). The following Figure 8 represents the search graph of Tweekvechtende oekhoortjes and the corresponding sales for CB. The gap between the number of searches and the number of sales is known as the conversion rate or dropout rate. Factors like pricing, shipping days, shipping costs, and payment methods significantly affect the dropout rate.

Figure 8.

Google Search Volume and Sales of Tweekvechtende oekhoortjes for CB



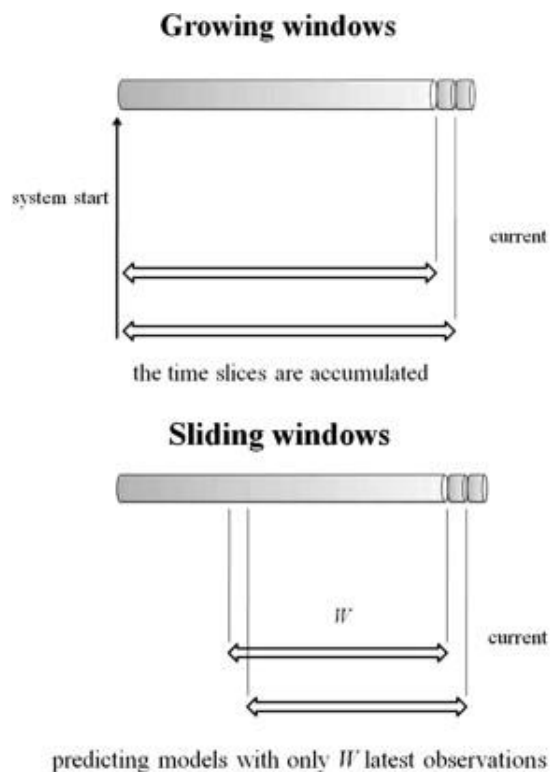
2.3 Validation and Evaluation

The evaluation metric for the forecasting task is the Root Mean Square Error (RMSE) due to its robustness and adaptability (González-Sopeña et al., 2021). RMSE is the standard evaluation metric as it is the most representative because it penalizes large errors more significantly. This would be used to compare each model to the base time series model when tested on the validation set. Other well-renowned metrics are Mean Absolute Error (MAE) and Mean Absolute Percent Error (MAPE). The following metrics penalize all errors equally, large or small. However, the MAE preserves the same scale of unit and, hence, can be more easily interpreted.

Before training the model, the data has to be split into training and testing data, 80% for training and 20% for testing (Medar et al., 2017). This split prevents the model from overfitting while ensuring there is enough data to train the model. In the case of time-based forecasts, the split has to be in place, and there are generally two strategies: the growing window and the sliding window. The growing window, as seen in *Figure 9* below, starts with the first data point for training and keeps expanding the training data iteratively while maintaining the sequential structure. On the other hand, the sliding window maintains the same window size and also moves sequentially. Herrera's (2010) predictive model for water demand using Monte Carlo simulation explains the use cases between the growing window and sliding window. The growing window performs better for tasks that have a long period of dependency or tasks affected by seasonality. In the meantime, the sliding window performs better for tasks where the predictions rely only on a few sequential data points before it.

Figure 9.

Growing window vs sliding window diagram (Herrera et al., 2010)



The search space defines the possible parameters and architecture of a model (Liashchynskyi et al., 2019). Each model has a different architecture hence, a different variety of hyperparameters. For the hyperparameter tuning, there are two search strategies: random search and grid search. Random search takes less time as it simply chooses random values within a boundary and tests them. On the other hand, the grid search strategy iterates through all the possible values within a boundary (Liashchynskyi et al., 2019). High dimensional data is much more difficult. However, it can be parallelized, saving time but not resources. A benefit of the grid search is that it is reproducible. The results it will yield will always be the same, unlike the random search.

Figure 10.

Grid Search Illustration (Liashchynskyi et al., 2019)

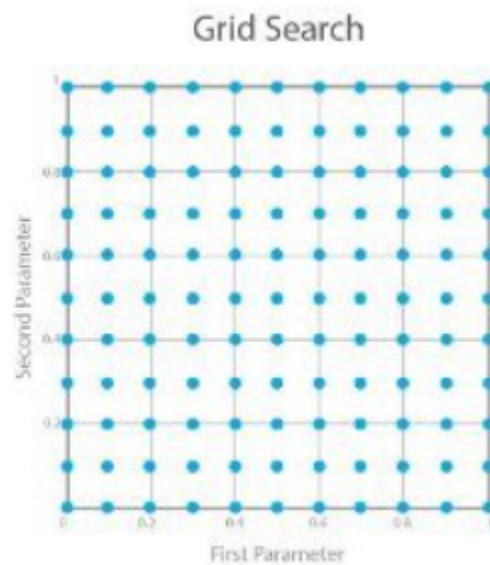
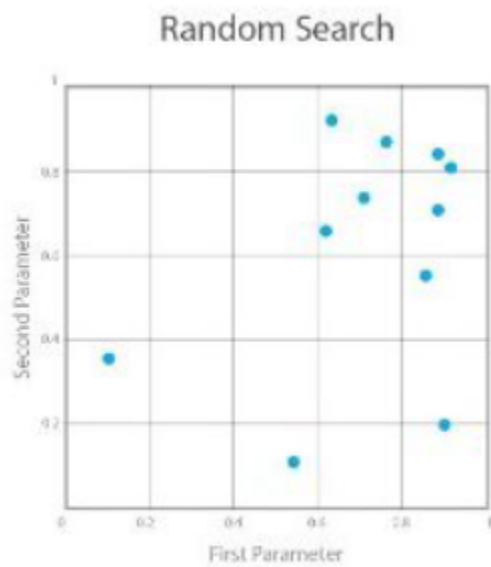


Figure 11.

Random Search Illustration (Liashchynskyi et al., 2019)



3. Research Methods

Research Question:

How can machine learning models and techniques reduce RMSE in forecasting demand for e-commerce?

Subquestion:

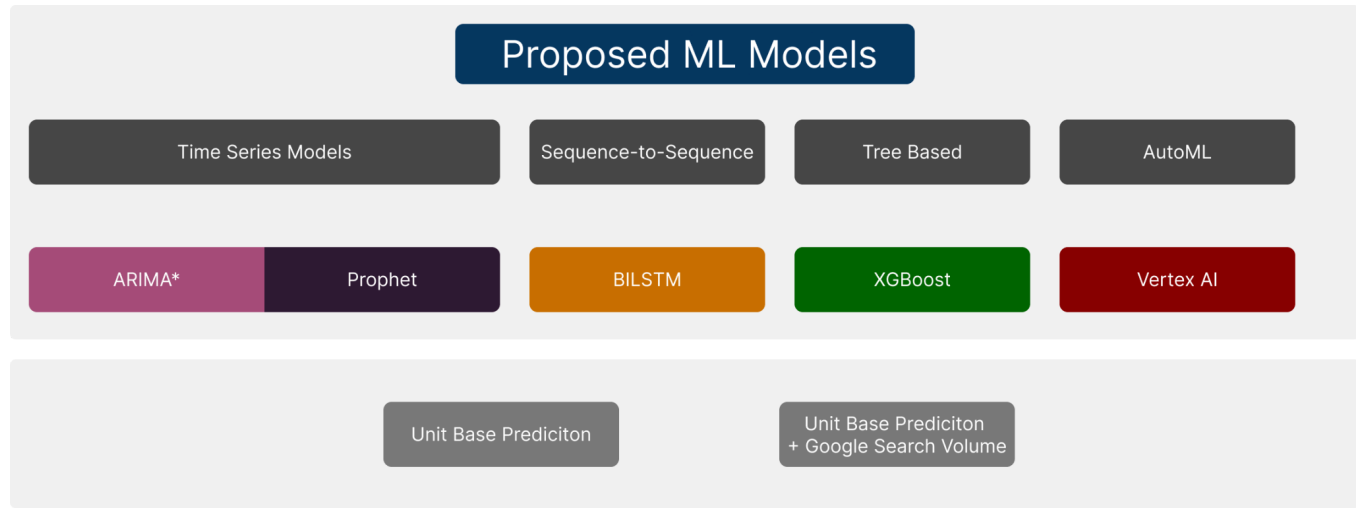
- 1. Which current state-of-the-art ML models are best for demand forecasting: time-series models, sequence-to-sequence models, or tree-based models?
- 2. Can external data from Google Keywords help reduce RMSE for demand forecasting on a unit basis?

3.1 Proposed Models and Process

The figure below shows the categories and each corresponding model. The four proposed models are ARIMA (Base model), Prophet, BILSTM, XGBoost, and AutoML.

Figure 12.

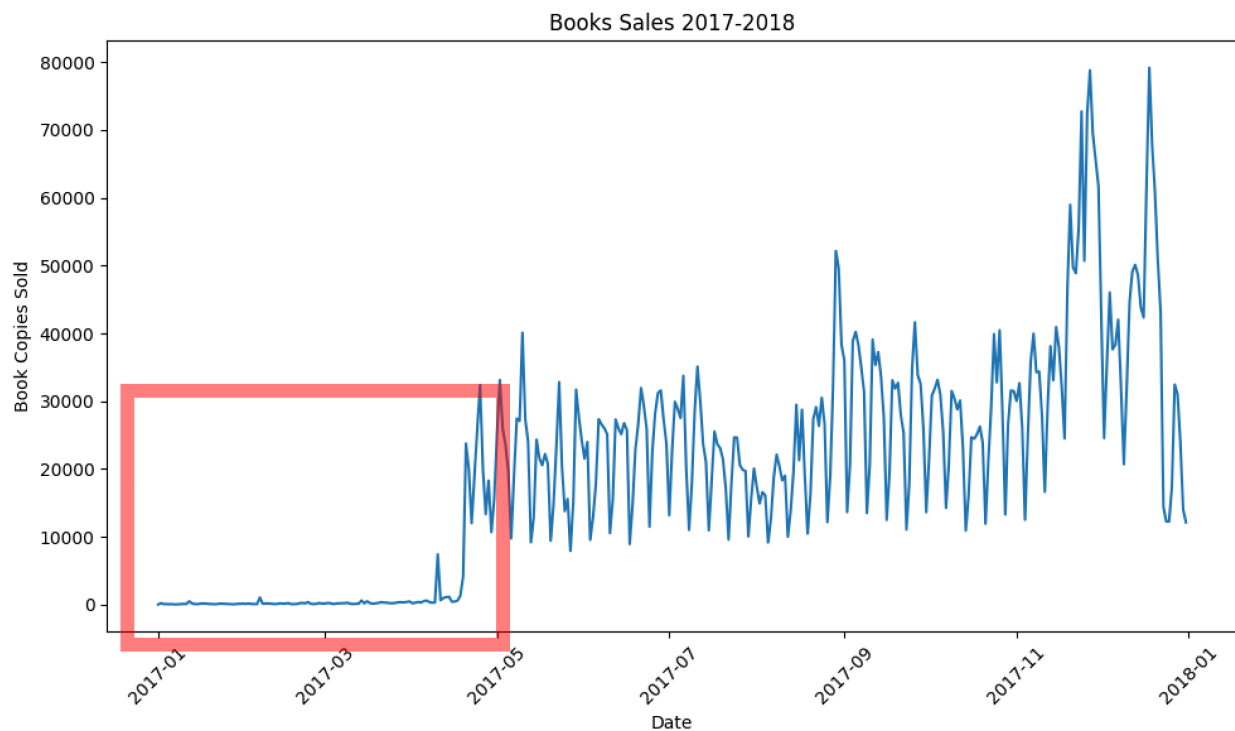
Proposed Machine Learning Models



The best-performing model will yield the lowest RMSE on a predicted horizon of 7 days in the future. All four models will incur the same training and validation process to ensure fair trial among the different models. The models will be trained and tested on data from 2018-01-01 to 2023-12-31. The dataset includes data from 2017; however, when further examined, there is misleading data from the first quarter of the year. Hence, it would work best to remove the year to avoid training on misleading data. The reason behind the removal of the whole year is that the data has an annual seasonal pattern; therefore training the data from the start to end will ensure that it is balanced across seasons.

Figure 13.

Missing Data from Books Sales 2017-2018



After the preprocessing process (the code is available in Appendix C), the expected input for the model will have the following format:

Figure 13.

Sample Input for the models

ONTVANGST_DATUM (Date of orders)	EXEMPLAREN_AANT (number of copies)
4/8/2019	44427
5/8/2019	65215
6/8/2019	45020

3.2 Model Parameters and Hyperparameter Tuning

3.2.1 ARIMA

The ARIMA model has only 3 model parameters (p , d , q). The necessity of the parameter d is determined by the Augmented Dickey-Fuller (ADF) test, which measures whether the data is stationary or not. The Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) determine the values for p and q respectively. The value for q is the point where there are significant spikes then drops off. In contrast, the value of p is the point where the PACF plot spikes and then drops off. For the ARIMA model, the PACF and ACF graphs are used to determine these values; unlike the other models, there is no need for extra hyperparameter tuning.

3.2.2 Prophet

The following parameters correspond to the Prophet model:

```
Prophet(
    growth='linear',
    changepoints=changepoints_dataframe,
```

```

n_changepoints=10,
changeoint_prior_scale=0.1,
seasonality_mode='additive',
yearly_seasonality=True,
weekly_seasonality=True,
daily_seasonality=False,
holidays=holidays_dataframe,
interval_width=0.95,
uncertainty_samples=1000
)

```

These are the default parameters for the model. A grid search will be applied to achieve the most relevant parameters for forecasting sales. A few parameters that won't be changed are growth, changeoints, seasonality_mode, and holidays. Growth and seasonality_mode are already known from the EDA as linear and additive due to the trend of the graph. In addition, holidays in the Netherlands are already known, and the changeoints are irrelevant to the current study as it measures dates with significant changes.

Holidays List (Government of the Netherlands, 2024):

- Nieuwjaarsdag (New Year's Day): Monday 1 January 2024
- Goede vrijdag (Good Friday): Friday 29 March 2024
- Eerste en tweede paasdag (Easter Sunday and Easter Monday): Sunday 31 March and Monday 1 April 2024
- Koningsdag (King's Day): Saturday 27 April 2024
- Bevrijdingsdag (Liberation Day): Sunday 5 May 2024
- Hemelvaartsdag (Ascension Day): Thursday 9 May 2024
- Eerste en tweede pinksterdag (Whit Sunday and Whit Monday): Sunday 19 and Monday 20 May 2024
- Eerste en tweede kerstdag (Christmas Day and Boxing Day): Wednesday 25 and Thursday 26 December 2024

Book Related Holidays:

- World Book Day: April 23, 2025

3.2.3 BILSTM

For the BILSTM, two components require tuning, the inputs and model parameters. For the inputs, the lookback window size will be tested to achieve the best size. The variety of choices lies between 1-5 times larger than the horizon size (7) hence, the possibilities are between 7-35.

After that, hyperparameter tuning will be applied to the number of neurons in each layer, number of layers, dropout rate, batch size, and learning rate. A grid search will be applied due to its reproducibility and understandable structure. All the inputs to the BILSTM have to be normalized and scaled as BILSTM is a NN; therefore, it requires a unified scale.

3.2.4 XGBoost

The XGBoost will require tuning similar to the BILSTM, inputs, and model parameters. The lookback window size will be tuned to the values that result in the least RMSE. The model parameters here that will be discovered using grid search are learning rate, number of estimators(trees), max depth of a tree, regularization parameters (L1 and L2), and Colsample_bytree (fraction of features used by each tree).

3.2.5 AutoML

The steps to achieve the forecasting model are documented below in Appendix B. The input for the model will be the input found in Figure 13 above. It will simply contain the date and the number of sales in addition to a column called reference ID, which will store the same value, 1 for all rows. Details about the validation are not apparent on Vertex AI, and the user cannot access the generated code. However, the platform claims that it does an 80/10/10 train-validate-test split.

3.3 Unit Basis Prediction

The best-performing model from the above four models will be used to predict the sales of books on a unit basis. The top 1,000 book IDs sold each year in terms of units will be extracted from the Orders_[year].csv files. This will be done by grouping the sales on book ID. The reason behind choosing only the high-moving books is that some books are sold only a few times a year. Therefore, building a forecasting model for these books would yield low, irrelevant results.

Google keyword planner provides only five years of historical searches on a term for a monthly basis. To fairly experiment with the results against the historical searches, the base model will have the same structure. The data input will be monthly, starting from 2020 instead of 2018. The inputs to the following model will be the sales of the book across the previous three months, and the output is predicted with a month's horizon. The book category, book genre, and publication date will be incorporated as input and will be further used if they yield positive results.

3.3.1 Google Keywords

Google Keyword Planner accepts a CSV file with up to 10,000 keywords at a time. The input must include only letters and numbers. Special characters are not acceptable inputs and must be removed before processing. The output of the program is a variety of metrics regarding competition, pricing, and the search volume per month. The search volume for the last three months will be inputted as a feature of the model besides the previously mentioned features. The search volume expands from 2020-06 to 2024-06. Figure 14 below is a sample of the output when merged with the ARTIKEL_KD. A walkthrough instruction of the keyword planner will be provided in Appendix A.

Figure 14.

Sample Output of Keyword Planner

ARIKEL_KD (Book Id)	Keyword (Title)	Searches: Jun 2020	Searches: Jul 2020	Searches: Aug 2020	Searches: Sep 2020	Searches: Oct 2020	...
978902576 7341	twee vechtende eekhoorn tjes	110	90	110	320	1300	...

The input for the Google keyword planner will be the titles of the books. Another possible input considered was the book author however, it shows that about 27.7% of the book authors have a value of null.

Figure 15 below shows the top-selling books with no author or publisher. When any of the following books are searched for, the author and publisher appear. For “Meer Koolhydraatarme Recepten Oanh’s Kitchen,” the author is Oanh Ha Thi Ngoc, and the Publisher is Bjornbooks. However, in the metadata provided by CB, the following was not found. Possible reasons could be the insignificance of the following in terms of logistics. As a consequence, it would not be possible to incorporate the author's name into Google keyword searches unless the entries for >30,000 were edited.

Figure 15.

Top selling books with no authors or publisher metadata

Book Title	Number of Books Sold Per Year (B2C)
Meer Koolhydraatarme Recepten Oanh's Kitchen	2982
Koolhydraatarme baksels uit Oanh's kitchen	1262

Anesthesie en de normale zwangerschap	1047
Samen in therapie	957
De slimste scheurkalender ter wereld 2019	691
Tuin smakelijk	617
Wat kun je doen als je te veel piekert	609
Permacultuur in je moestuin	520
Blij suikervrij zoet!	505
Onderwijs voor de 21ste eeuw	455

3.4 Validation and Evaluation

The dataset from 2018-01-01 to 2022-12-31 will be used to train the models, and the dataset from 2023-01-01 to 2023-12-31 will be used to validate the results achieved in the training phase. All of the data will be in place, meaning there will be no shuffling of the data to maintain its temporal significance.

As the horizon is only 7 days, a strategy of a sliding window must be implemented across all models to cross-validate, ensuring that the model generalizes well on unseen data. A sliding window is preferred over an expanding window as the dependency window size in this case is not that large therefore, it would allow for more memory-efficient calculations.

The main criteria for measuring performance will be RMSE however, the following metrics will be calculated; MAE, MAPE, and percentage of days where calculations are within 5%.

4. Results

4.1.1 Results for the Models

Figure 16.

Evaluation metrics for each model

Model	RMSE	MAE	MAPE	Percentage of days (predictions within 5% error)
ARIMA (baseline)	7324	6108	18%	28%
Prophet	6041	5009	16%	0%
BILSTM	7261	4986	40%	31%
XGBoost	5555	3860	11%	35%
AutoML	5681	4155	12.8%	N.A.

Figure 17.

Graph of ARIMA Predictions vs Actual Book Sales 2023 using a sliding window

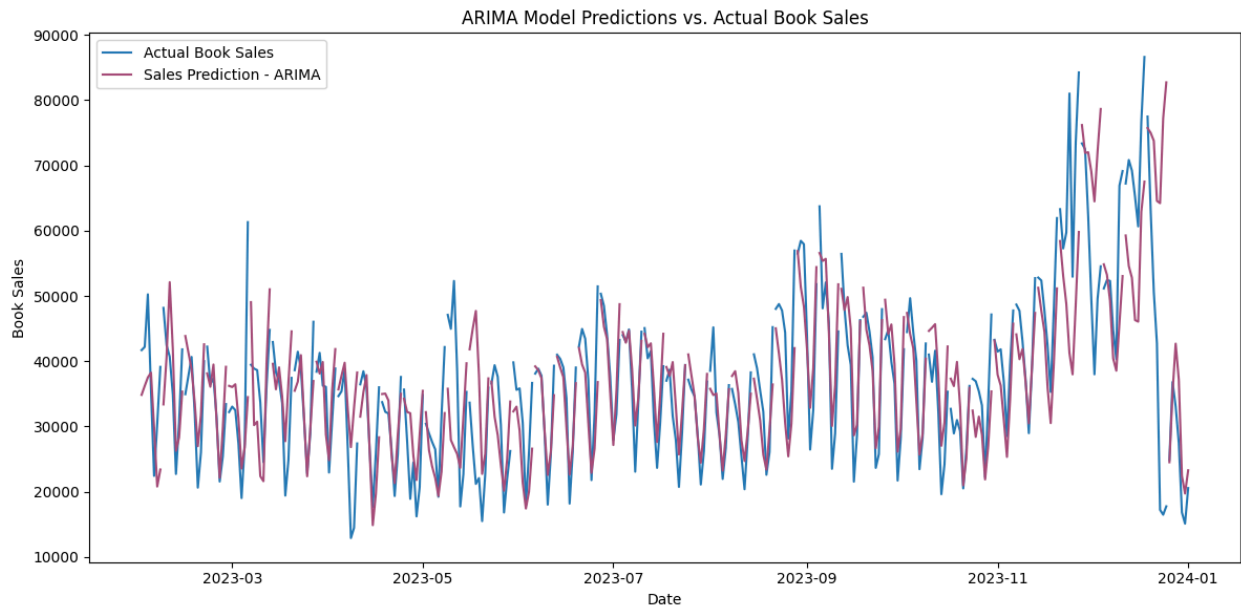


Figure 18.

Prophet Predictions vs. Actuals for total book sales in 2023 using a sliding window

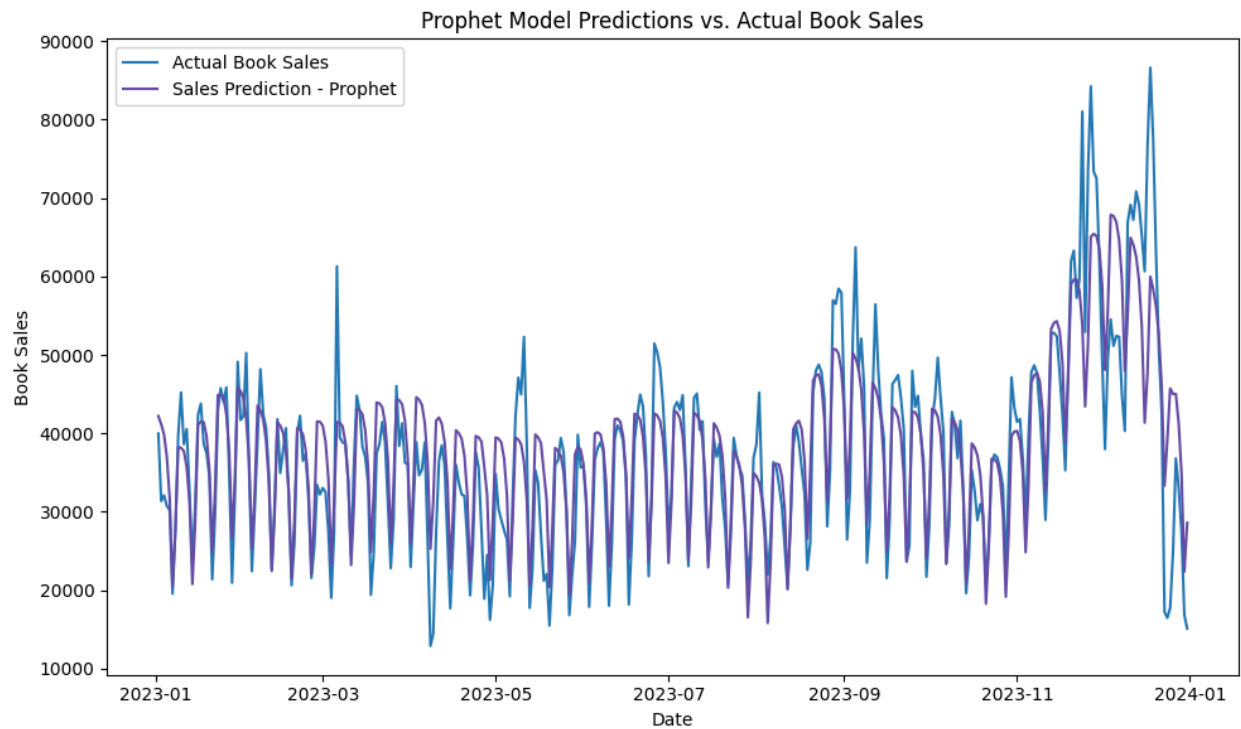


Figure 19.

BILSTM Predictions vs. Actuals for total book sales 2023 using a sliding window

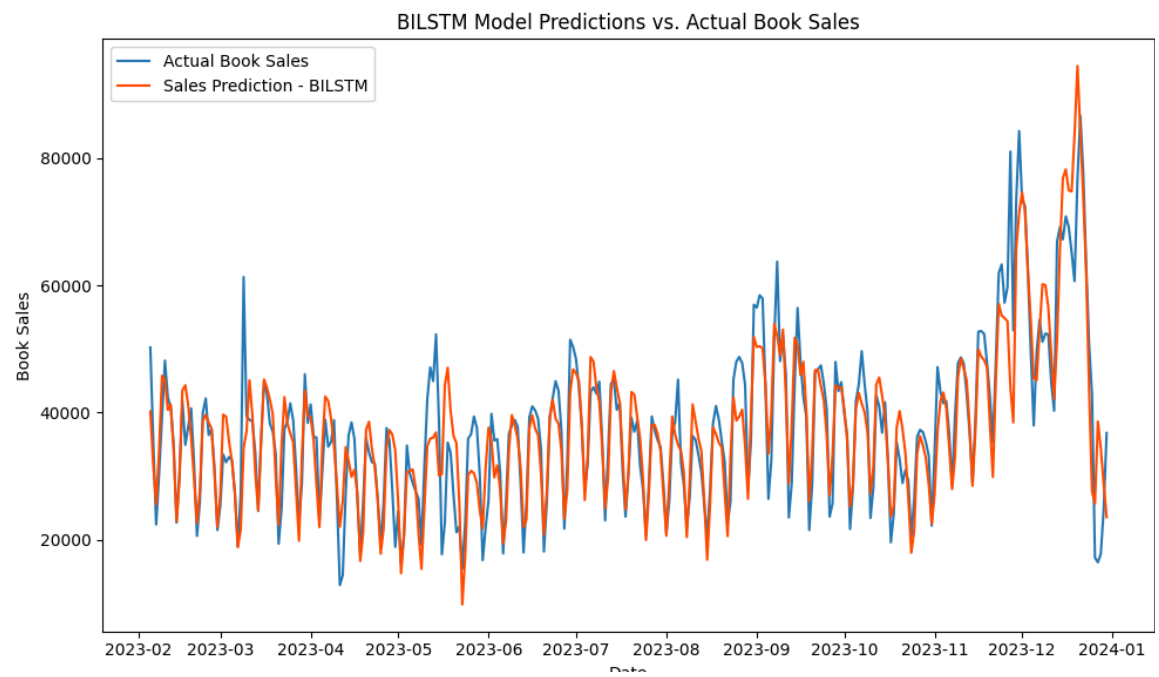
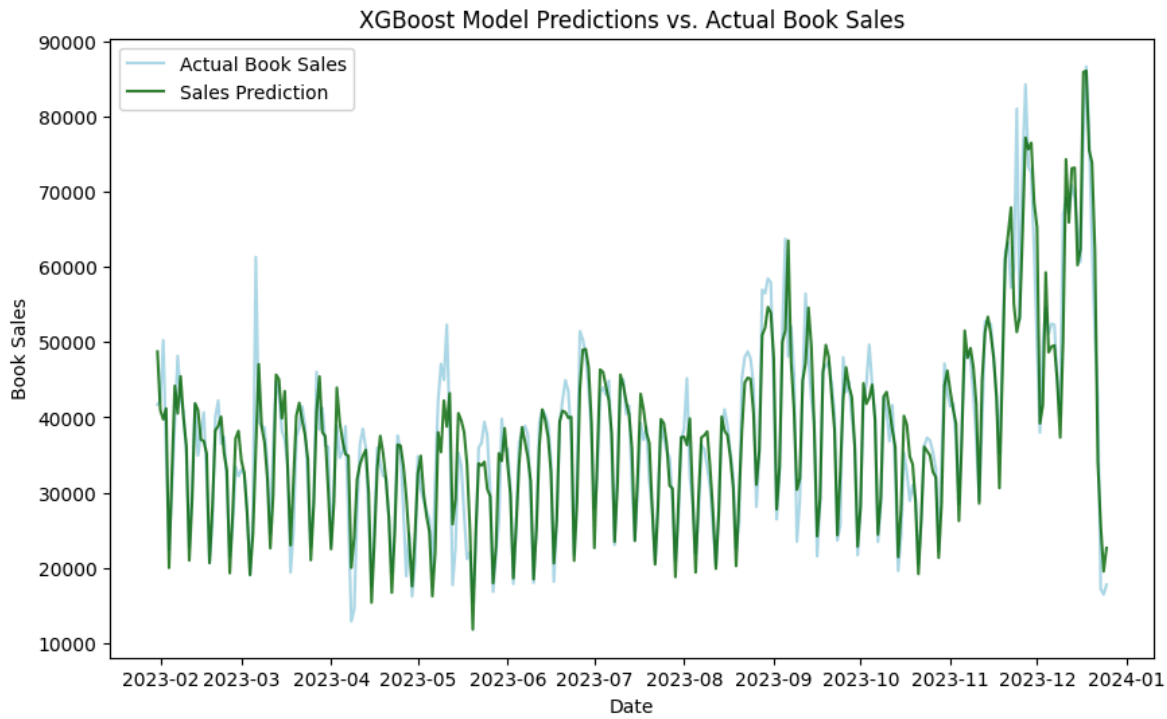


Figure 20.

XGBoost Predictions vs. Actuals for total book sales 2023 using a sliding window



4.1.2 Results for Unit Prediction Model

Figure 21.

Unit Basis Prediction for Book Sales 2023

Model Inputs	RMSE	MAE	MAPE	Percentage of days (predictions within 5% error)
Sales Per Book (last three months Sales + Book Category + Book Genre + Days from first Publication)	417	98	483%*	2%
Sales Per Book + Google Searches (last three months + Book Category + Book Genre + Days from first Publication + Google Searches last three months)	459	95	443%*	5%
Sales Per Book + Current Month	409	84	244%*	7%

Google Searches (Last three months sales + Book Category + Book Genre + Days from first Publication + Google Searches last three months + Google Searches Current Month)				
--	--	--	--	--

Note. When calculating the MAPE here, values of 0 were omitted as they would skew the metric due to a lot of books having sales of 0 in some months. Otherwise, the metric would give 0.

Top 3 selling books in 2023: Predictions vs Actual Book Sales:

Figure 22.

Book Sales vs Predictions for Atlas in 2023

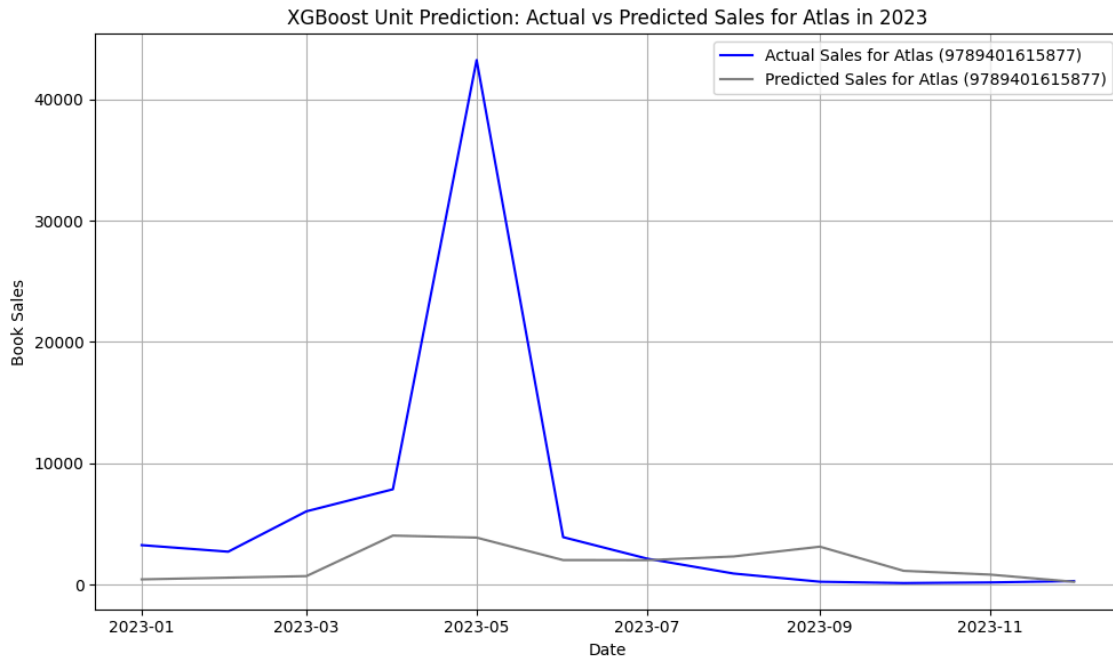


Figure 23.

Book Sales vs Predictions for Master Your Mindset in 2023

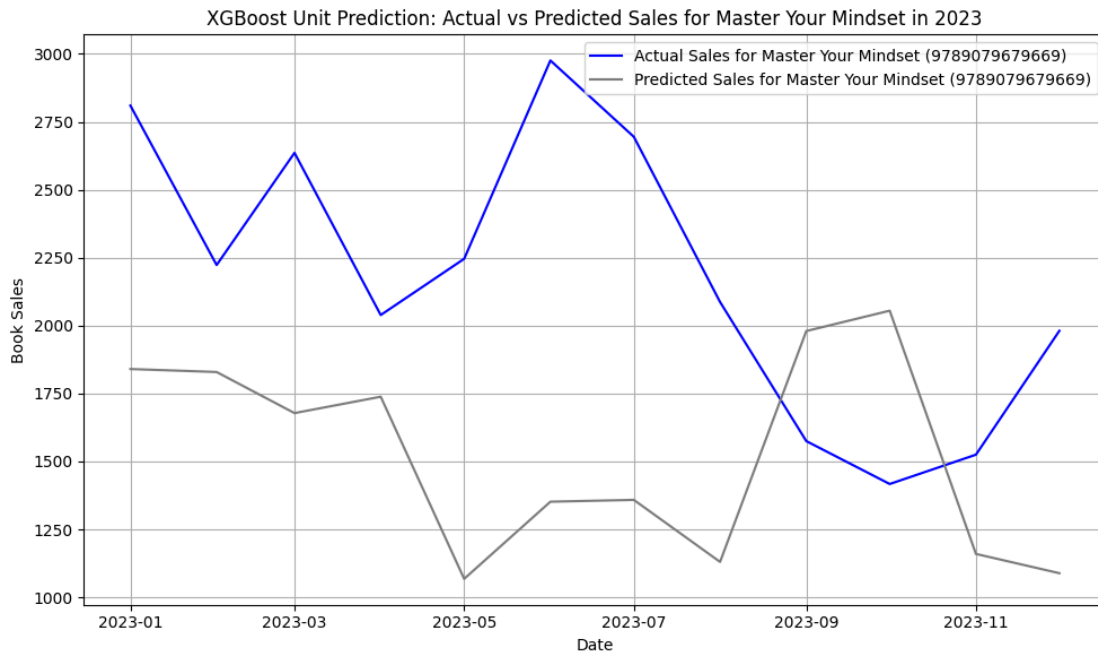
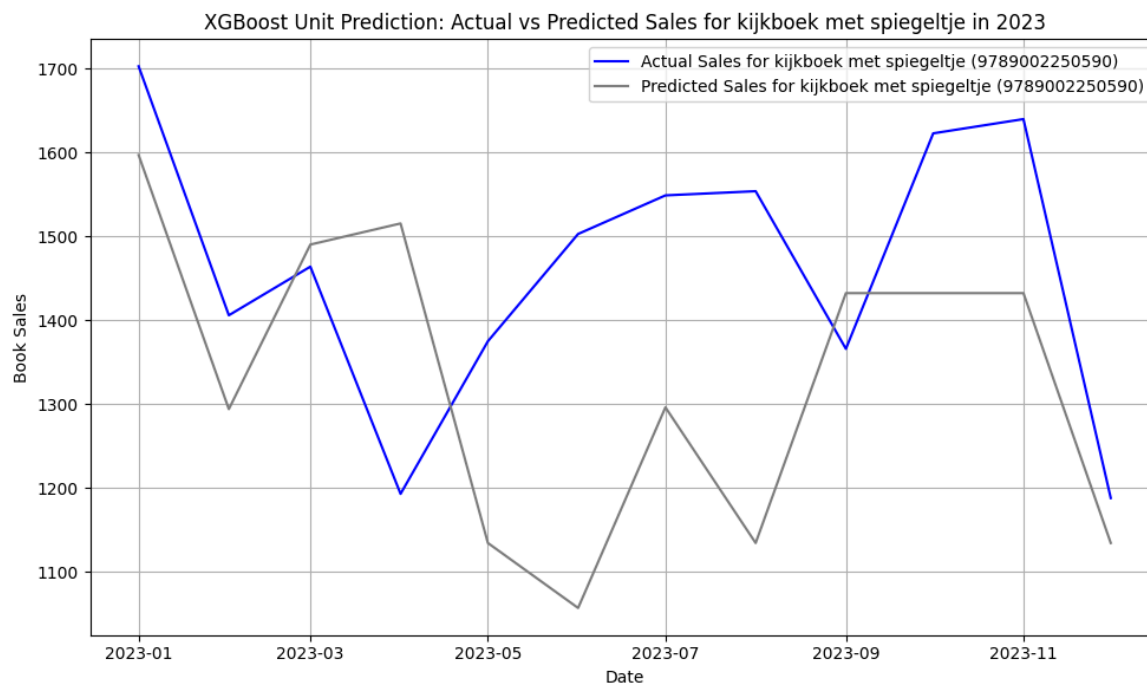


Figure 24.

Book Sales vs. Predictions for Kijkboek Met Spiegeltje in 2023



Top 3 selling books in 2023: Predictions vs Actual Book Sales + Google Searches
in 2023:

Figure 25.

Book Sales vs. Predictions for Atlas using Google Searches in 2023

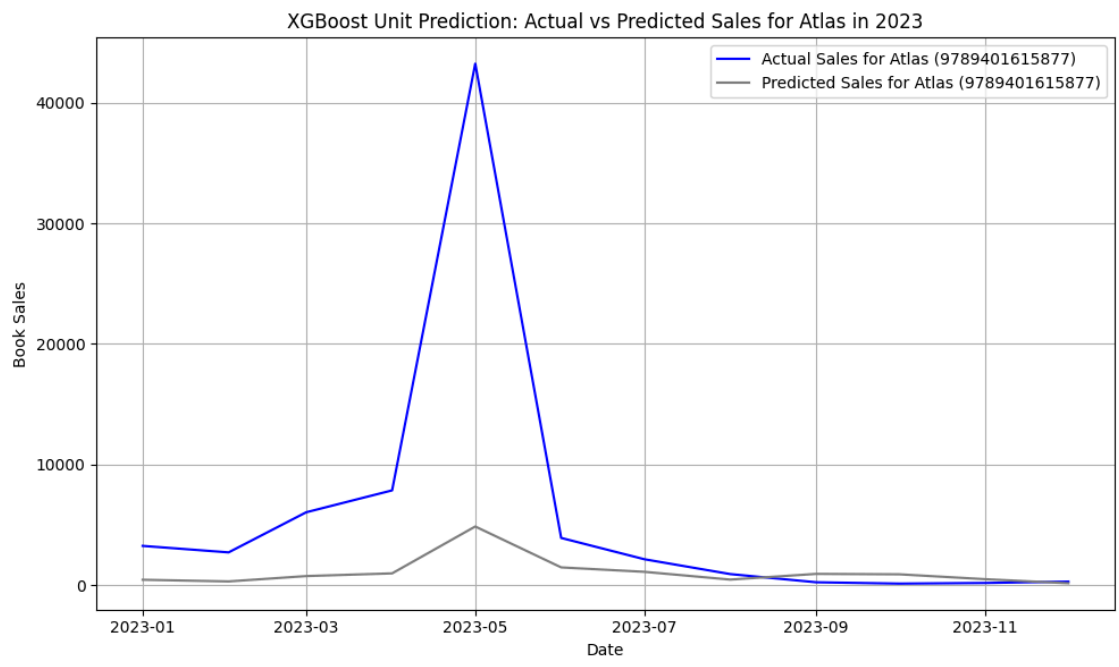


Figure 26.

Book Sales vs. Predictions for Master Your Mindset Using Google Searches in 2023

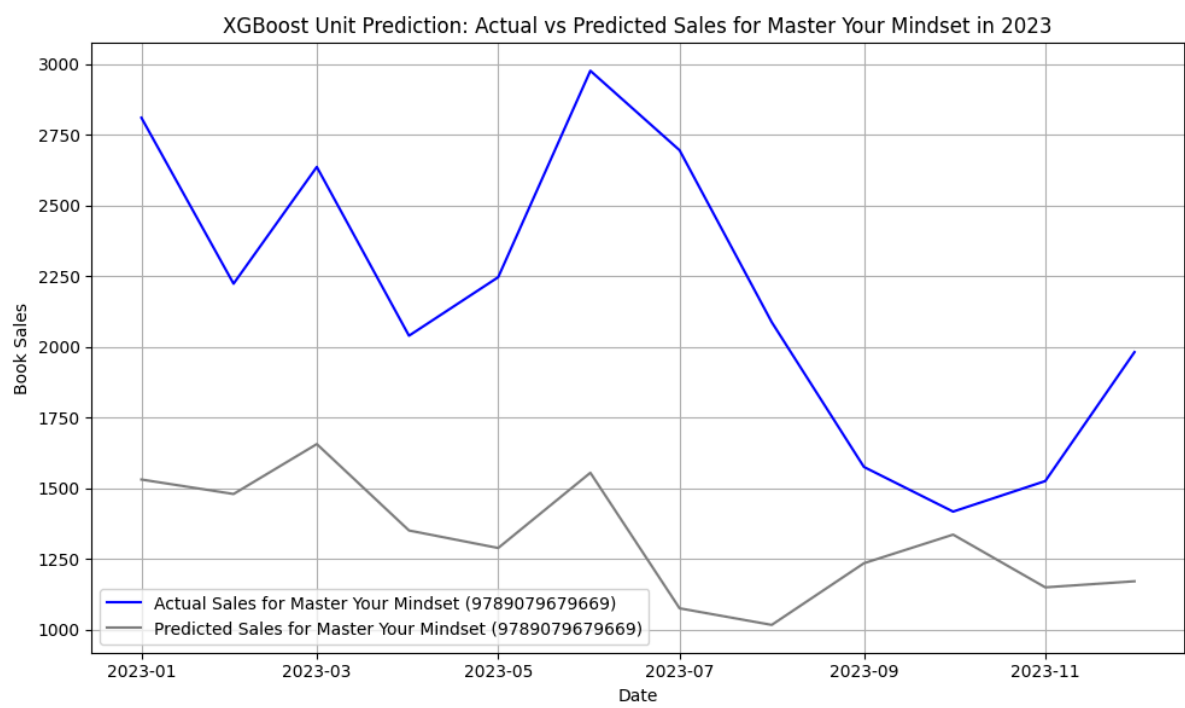


Figure 27.

Book Sales vs. Predictions for Kijkboek Met Spiegeltje using Google Searches in 2023

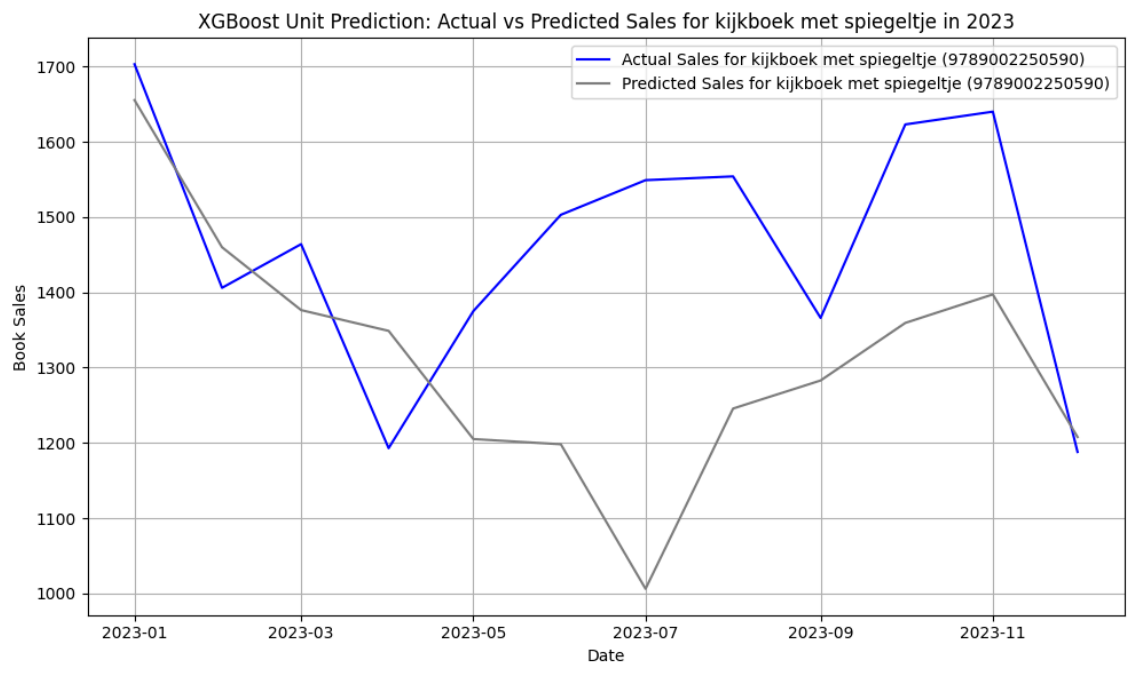
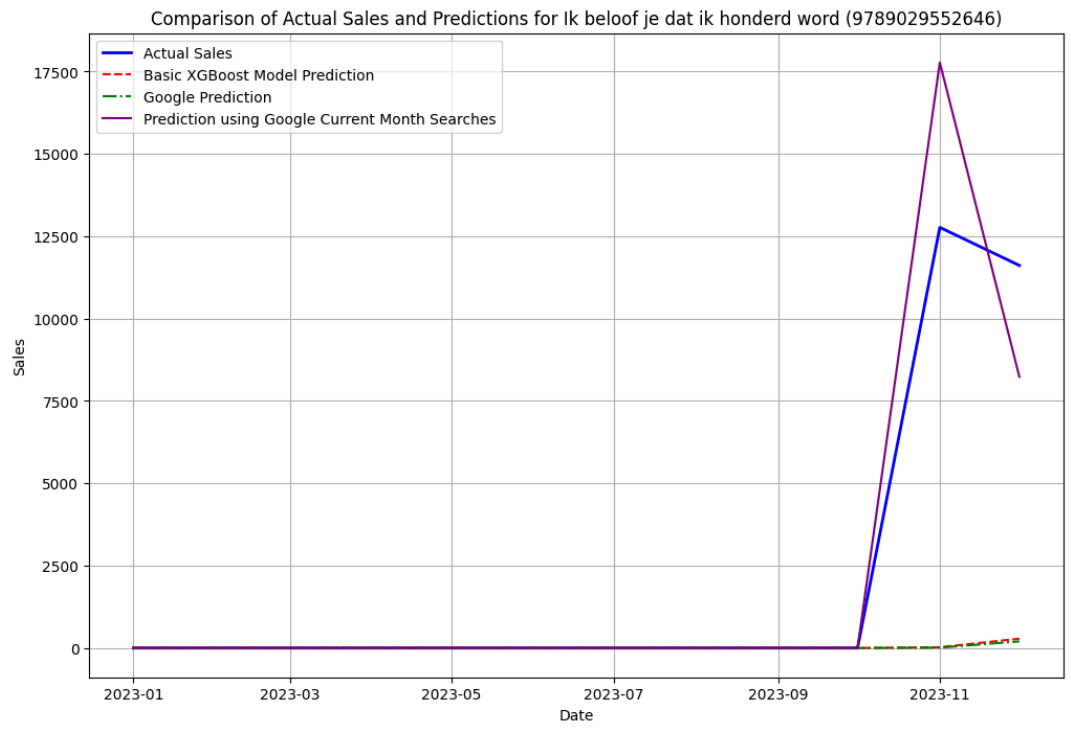


Figure 28.

Book with first publication during 2023



Note. The following diagram shows all the unit basis prediction models for comparison on newly published books.

4.2 Model Parameters and Hyperparameter Tuning

4.2.1 ARIMA

Parameter d is set to 1 as the data is non-stationary (p-value=0.072). Parameter p is set at six as the PACF cuts off at 6. Parameter q is set to 4 as the ACF cuts off at 4. The corresponding graphs can be found in the code attached, which is called ARIMA.ipynb.

4.2.2 Prophet

Figure 29.

Top 3 Model Parameters for Prophet Model from Grid Search

Changepoint_prior_scale	seasonality_prior_scale	holidays_prior_scale	seasonality_mode	RMSE
0.5	1.0	0.01	additive	4611
0.5	1.0	0.1	additive	4611
0.5	1.0	1.0	additive	4611

4.2.3 BILSTM

Figure 30.

Top Model Parameter for BILSTM Model from Grid Search

epochs	batch_size	validation_split	RMSE
50	16	0.1	7182

50	32	0.2	7303
50	16	0.2	7151

4.2.4 XGBoost

Figure 31.

Model parameters comparison for XGBoost

n_estimators	max_depth	learning_rate	Holidays	Day of the Week and Month	RMSE
50	16	0.2	No	No	6532
50	16	0.2	Yes	No	5555
50	16	0.2	Yes	Yes	5619

Figure 32.

Top 3 hyperparameter values for XGBoost

n_estimators	max_depth	learning_rate	subsample	colsample_by tree	RMSE
150	3	0.1	0.8	1.0	2804
100	3	0.1	0.8	1.0	2805
100	3	0.1	1.0	0.8	2805

4.3 XGBoost for Unit Prediction

Figure 33.

Model parameters comparison for XGBoost

n_estimators	learning_rate	lag_period	BOEKS OORT (Book Genre)	NUGI (Item Category)	OPGEN OMEN_ DAT (Publicati on Date)	Google Search	RMSE
100	0.1	3	No	No	No	No	440
100	0.1	6	No	No	No	No	395
100	0.1	3	No	Yes	No	No	431
100	0.1	3	Yes	No	No	No	432
100	0.1	3	Yes	Yes	No	No	430
100	0.1	3	Yes	Yes	Yes	Yes	459
100	0.1	3	Yes	Yes	Yes	Yes (+current month)	409

Note. The hyperparameter value of 6 months for context windows was discontinued as the testing data is only one year, and this would affect the model's generalizability.

5. Discussion

5.1 Models

5.1.1 ARIMA - Baseline

Out of the five chosen models, the XGBoost outperformed all other models on all metrics. As seen in Figure 17, the ARIMA model couldn't accurately detect sudden changes in peaks in the graph. This is primarily due to the model's dependency on the general trend of the data to predict future points, the linear regression component.

However, it did not perform the worst across all metrics. This is due to the moving average component, which primarily focuses on staying within a certain range and avoiding any fluctuations. As imagined, this would perform best in weeks with little to no fluctuations. The parameters chosen were based on the tests applied. The results from the ADF test suggested that the data is non-stationary and it would require differencing, hence, the value of d being 1. The Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) suggested values of $p = 6$ and $q = 4$, respectively.

5.1.2 Prophet

The prophet model performs significantly better on the RMSE and MAE. The prophet model takes holidays as a model parameter, which explains the reduction in the RMSE, as holidays usually face fluctuations in book sales. In addition, the prophet model has a seasonal component that assists in handling the monthly changes. Across all years, CB encounters peaks towards the last quarter of the year with a 30-50% increase in units sold. The unexpected part is that the model predicted 0 days within 5% of the actual amount of books sold. Excluding holidays, the prophet model is not robust with outliers.

The prophet model is additive; in other words, it decomposes the trends, seasonality, and holidays to generate predictions. This can be useful for Meta to forecast activity on their applications, as social media activity is heavily reliant on trends and seasonality. In this case, the

model manages to pick up the trend. However, the noise, possibly caused by the surge of sales during the COVID-19 pandemic, makes it difficult for the model to predict within the 5% range.

5.1.3 BILSTM

Unlike the findings of Siami-Namini et al. (2018), the BILSTM didn't see any reduction in the RMSE when compared to the ARIMA model. A possible reason could be the small context window. In this study, the context window was the previous 30 days. RNNs and LSTM, in particular, require larger amounts of data to accurately detect the pattern. A longer context window can enhance the performance of the BILSTM. The BILSTM, however, performed better on the percentage of days with 5% and the MAE. This further suggests that LSTM predicts within a closer range on a large proportion of the days, but on the days it predicts incorrectly, the model is far off from the actual results.

BILSTMs, by default, don't accept holidays and measure seasonality, which could be the reason behind the misalignment of outlier sales. Another concern was the degradation in the prediction from day 1 to day 7. Most models can face this concern. However, the BILSTM encountered it heavily. For horizon day 1, the RMSE was 5418. For horizon day 7, it reached 8084, a 50% increase in errors. This might suggest that BILSTM performs better for shorter horizons than longer as its performance decreases significantly.

5.1.4 XGBoost

The XGBoost was the best-performing model across all metrics. The model achieved a reduction of about 40% on the MAPE when compared to the baseline ARIMA. This further proves Pavlyshenko's (2019) theory that XGBoost and random forest perform well for not only classification and regression tasks but also forecasting tasks. The model was put under a train/test split just like the other models, which ensures that the model did not overfit, contrary to beliefs towards trees. When further examining the results, XGBoost faces little to no degradation across the weak horizon, from 5065 to 5619. The L1 and L2 regularizers built into the XGBoost prevent the model from overfitting and generalizing well on new points. Unlike the previous

models, the XGBoost is a tree, non-sequential model. Each prediction is separate from the previous prediction, reducing the risk of an error propagating forward.

The initial model parameters were chosen through trial and error; apparently adding the day of the week and the month as separate parameters didn't help the model. When the sales across the week were plotted, there was no distinct difference between the days, which explains the inefficiency of it as a metric. The metric that helped improve the prediction accuracy was the use of holidays as a parameter of the model. In addition, the hyperparameter tuning overfitted the model as it performed worse on the validation set than on the training set after adding the tuned parameters.

5.1.5 AutoML

The AutoML performed quite similarly to the XGBoost which suggests that the model could be using a tree model in the backend. The exact architecture cannot be extracted as it is a closed-source program. Additionally, the calculation for the percentage of days with 5% couldn't be calculated as it is not a default metric for Vertex AI AutoML and the validation process is also a closed source.

A common theme across the models was the tradeoff between stability and anomaly detection. Models that detect sudden changes perform slightly worse on overall day-to-day sales (General Trend). An ensemble or blending between the models would help average out these differences.

5.2 Unit Prediction XGBoost

Both predicting sales per unit with and without Google searches faced the same problem of unexpected changes. As seen in the above figures 22-28, each book doesn't have a predictable pattern or even a unified magnitude of units sold. Hence, both have high RMSE values. When integrating the Google Searches of the past three months, only the percentage of book sales within 5% improved. However, when integrating the search volume of the past three months and the current month, the MAPE value was reduced by 51%, and the percentage of book sales within 5% improved from 2% to 7%. The improvement was primarily due to books that had not

been sold before, such as the example in Figure 28. It is difficult to predict book sales solely based on category and genre. The model would always output the same value for any combination of genre and book. Therefore, the use of Google searches assists the model in such a scenario.

The model parameters for the unit prediction started as an XGBoost with a lag period of 3 months. Then, six months was also tested and performed better. However, it was not further selected as the test data is only one year, and having a context window of 6 months would affect the model's generalizability. Consequently, the book category, book genre, and publication date were added, all of which improved the metrics' values. Lastly, the Google searches for the last three months were added, and then the Google searches for this month were added, resulting in a 51% improvement. No further hyperparameter tuning was applied as the models are already complex enough.

5.3 Limitations

Using the current month to forecast sales is clearly limited due to leaky data. This would assume that the consumer searches first and then decides to buy after a few days. This assumption would fail if consumers buy the book and then search for it. Hence, further testing regarding the correlation between searches and sales must be done to ensure there is no leaking data.

Another limitation across all models is the COVID-19 irregular surge in sales period. The period lies in the middle of data, making it difficult to omit and significantly affecting the data trend. Using the COVID data makes the model more robust against possible irregularities but affects the model's overall performance. Adjusting or synthesizing sales data for this period is a possible solution.

When using Google search volume, a strict difference between user intent and user keywords must be drawn. Not all consumers search for a book by writing its title and definitely not the full title. At the same time, a book title can also be a movie name or TV series. Hence, the choice of keywords should be further examined to ensure the quality of results. In this study,

the word ‘boek’ was added behind any single-worded title to ensure the clarification that it is a book. However, in a lot of cases, the consumer doesn't necessarily write a book after the word. Additional data from bol.com to explore the full customer journey can help reveal new keywords.

Besides the irregularity in sales for some books, the magnitude differed significantly for ‘Atlas Boek’, which can be seen in Figure 25. The sales went from 2-3 thousand sales per month to almost 45,000 books sold in May, and then back to 2-3 thousand sales per month. Meanwhile, there are only a few other books sold per month. This makes it difficult for the ML model to forecast unless there is some distinction or classification between them. In addition, each book can have a different conversion rate, such as the number of people who purchase the book. Attributes like pricing and delivery speed affect the consumer’s dropout rate.

5.4 Further Improvements

A clear improvement is to assess the units sold per day or per week. Google keyword planner offers only the last 30 days daily. Hence, continuous data collection must be done several months before training the model. The use of a daily search will help eliminate the risk of leaky data. In addition, customers usually take around seven days between their first search intent and purchase event, according to Lehmann (2016). This ensures that the search corresponds to the sale event rather than being an indicator of the expected amount of orders.

A classification of books in categories based on how frequently they are sold would assist the XGBoost model; for instance, it can be classified into new book, dead mover, slow mover, medium mover, and fast-moving. The model would then be a classification model which detects any changes in a book’s sales. Additionally, the conversion rate can be useful for each book to measure the percentage of the people who searched for this book and how many actually bought it in the previous months. These metrics can assist the model when dealing with fluctuations.

Since each model specializes in a certain aspect, trying a blend or ensemble between the models can yield better results by reducing variance and increasing robustness. A stacking model like the one used by Pavlyshenko improves the model's generalizability.

6. Conclusion

This study aimed to investigate the best performing ML model for ecommerce demand forecasting. Through the evaluation of five machine learning models—ARIMA, Prophet, BILSTM, XGBoost, and Vertex AI AutoM, XGBoost was identified as the most effective model for predicting total order volume each day. The model demonstrated superior performance in terms of accuracy and stability, reducing errors by 25% on the RMSE and 36% on the MAE when compared to the baseline ARIMA model.

The research underscores the importance of incorporating advanced machine-learning techniques and aggregating new features. The use of external data, such as Google search trends, further enhanced the model's predictive capabilities, particularly for new items or items with irregular sales patterns. The use of external data sources, such as Google searches, reduced the MAE by an incredible 50%. However, the study also identified limitations related to data irregularities, such as the impact of the COVID-19 pandemic on sales trends. Despite the reduction in the error, the study highlighted the potential for leaky data when using current month search data for forecasting.

Future improvements could involve the continuous collection of daily search data to reproduce the experiment on a daily or weekly basis, mitigating leaky data risks. The classification of books based on their sales patterns would refine the model and possibly reduce the error. Additionally, ensemble methods, combining different models, could offer a balanced approach to handling both stability and anomaly detection in sales forecasting.

In conclusion, this thesis provides a comprehensive analysis of machine learning models for demand forecasting in the e-commerce industry, demonstrating that advanced models like XGBoost can significantly enhance the accuracy of order predictions, thereby improving logistical efficiency and reducing operational risks for companies like CB.

References

- Birkenshaw, J. (2003). *Life Cycle Costing of Print on Demand Digital Printing of Books and Packaging Materials*.
<https://www.imaging.org/common/uploaded%20files/pdfs/Papers/2003/DPP-0-289/8931.pdf>
- Bol.com. (2020). Whitepaper bol.com & Google: A strategic partnership. Retrieved June 6, 2024, from
https://partnerplatform.bol.com/content/uploads/2020/11/Whitepaper_bol_Google_DEF.pdf
- CB. (2024). Print on Demand. CB. <https://www.cb.nl/en/media/expertises/print-on-demand>
- ecommerceDB. (n.d.). bol.com revenue. Retrieved June 6, 2024, from
<https://ecommercedb.com/store/bol.com>
- Chan, W. N. (2020). Time series data mining: Comparative study of ARIMA and Prophet methods for forecasting closing prices of Myanmar Stock Exchange. *Journal of Computer Applications and Research*, 1(1), 75-80. Faculty of Information Science, University of Computer Studies (Hpa-an). Retrieved from
https://www.ucstgi.edu.mm/storage/2020/10/JCAR2020_75_80.pdf
- Chen, T., & Guestrin, C. (2016). XGBoost: a Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, 785–794. <https://doi.org/10.1145/2939672.2939785>
- Fattah, J., Ezzine, L., Aman, Z., El Moussami, H., & Lachhab, A. (2018). Forecasting of demand using ARIMA model. *International Journal of Engineering Business Management*, 10(1), 184797901880867. <https://doi.org/10.1177/1847979018808673>
- González-Sopeña, J. M., Pakrashi, V., & Ghosh, B. (2021). An overview of performance evaluation metrics for short-term statistical wind power forecasting. *Renewable and Sustainable Energy Reviews*, 138, 110515. <https://doi.org/10.1016/j.rser.2020.110515>
- Google Keyword Planner. (2024). Keyword Planner. Retrieved June 6, 2024, from
<https://ads.google.com/aw/keywordplanner/home>
- Google Cloud. (2024.). Vertex AI. Retrieved June 6, 2024, from
<https://cloud.google.com/vertex-ai>

- Government of the Netherlands. (2024). Which days are official public holidays in the Netherlands?. Retrieved June 11, 2024, from <https://www.government.nl/topics/working-hours/question-and-answer/public-holidays-in-the-netherlands>
- Herrera, M., Torgo, L., Izquierdo, J., & Perez-Garcia, R. (2010). Predictive models for forecasting hourly urban water demand. ScienceDirect. <https://www.sciencedirect.com/science/article/abs/pii/S0022169410001861>
- Keif, M. G. (2007). Cost-estimating for commercial digital printing. Document Recognition and Retrieval XIV. https://www.academia.edu/33368162/Cost_Estimating_for_Commercial_Digital_Printing
- Krauss, C., Do, X., Anh, & Huck, N. (2016). *Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500*. <https://www.econstor.eu/bitstream/10419/130166/1/856307327.pdf>
- Kumar Jha, B., & Pande, S. (2021). Time Series Forecasting Model for Supermarket Sales using FB-Prophet. *2021 5th International Conference on Computing Methodologies and Communication (ICCMC)*. <https://doi.org/10.1109/iccmc51019.2021.9418033>
- Kumar, R., Kumar, P., & Kumar, Y. (2021). Multi-step time series analysis and forecasting strategy using ARIMA and evolutionary algorithms. *International Journal of Information Technology, 14(1)*, 359–373. <https://doi.org/10.1007/s41870-021-00741-8>
- Lehmann, L. P. (2016). Predicting the sales figures of TVs using data from Google Trends (Master's thesis, University of Twente, School of Management and Governance). https://essay.utwente.nl/70721/1/Lehmann_BA_BMS.pdf
- Liashchynskyi, P., & Liashchynskyi, P. (2019). Grid Search, Random Search, Genetic Algorithm: A Big Comparison for NAS. <https://arxiv.org/pdf/1912.06059>
- Medar, R., Rajpurohit, V., & Bachulkar, R. (2017). *Impact of Training and Testing Data Splits on Accuracy of Time Series Forecasting in Machine Learning*. <https://ieeexplore.ieee.org/abstract/document/8463779>
- McIlroy, T. (2015, March 5). *Reengineering the Book Publishing Supply Chain*. Thad McIlroy - Future of Publishing.

- <https://thefutureofpublishing.com/2015/03/reengineering-the-book-publishing-supply-chain/>
- Paldino, G. M., De Stefani, J., De Caro, F., & Bontempi, G. (2021). Does AutoML Outperform Naive Forecasting? *Engineering Proceedings*, 5(1), 36.
<https://doi.org/10.3390/engproc2021005036>
- Pavlyshenko, B. M. (2019). Machine-Learning Models for Sales Time Series Forecasting. *Data*, 4(1), 15. <https://doi.org/10.3390/data4010015>
- Shiri, F. M., Perumal, T., Mustapha, N., & Mohamed, R. (2023). *A Comprehensive Overview and Comparative Analysis on Deep Learning Models: CNN, RNN, LSTM, GRU*.
<https://arxiv.org/pdf/2305.17473>
- Siami-Namini, S., Tavakoli, N., & Siami Namin, A. (2018). A comparison of ARIMA and LSTM in forecasting time series. In 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA) (pp. 1394-1401). IEEE.
<https://doi.org/10.1109/ICMLA.2018.00227>
- Singh, K., Booma, P. M., & Eaganathan, U. (2020). E-Commerce System for Sale Prediction Using Machine Learning Technique. *Journal of Physics: Conference Series*, 1712, 012042. <https://doi.org/10.1088/1742-6596/1712/1/012042>
- Swami, D., Shah, A. D., & Ray, S. K. B. (2020, August 18). Predicting Future Sales of Retail Products using Machine Learning. University of Southern California.
<https://arxiv.org/pdf/2008.07779>
- Van Houdt, G., Mosquera, C., & Nápoles, G. (2020). A review on the long short-term memory model. *Artificial Intelligence Review*, 53(8). <https://doi.org/10.1007/s10462-020-09838-1>
- Zeng, A., Chen, M., Zhang, L., & Xu, Q. (2023). Are Transformers Effective for Time Series Forecasting? *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(9), 11121–11128. <https://doi.org/10.1609/aaai.v37i9.26317>

Appendix

Appendix A

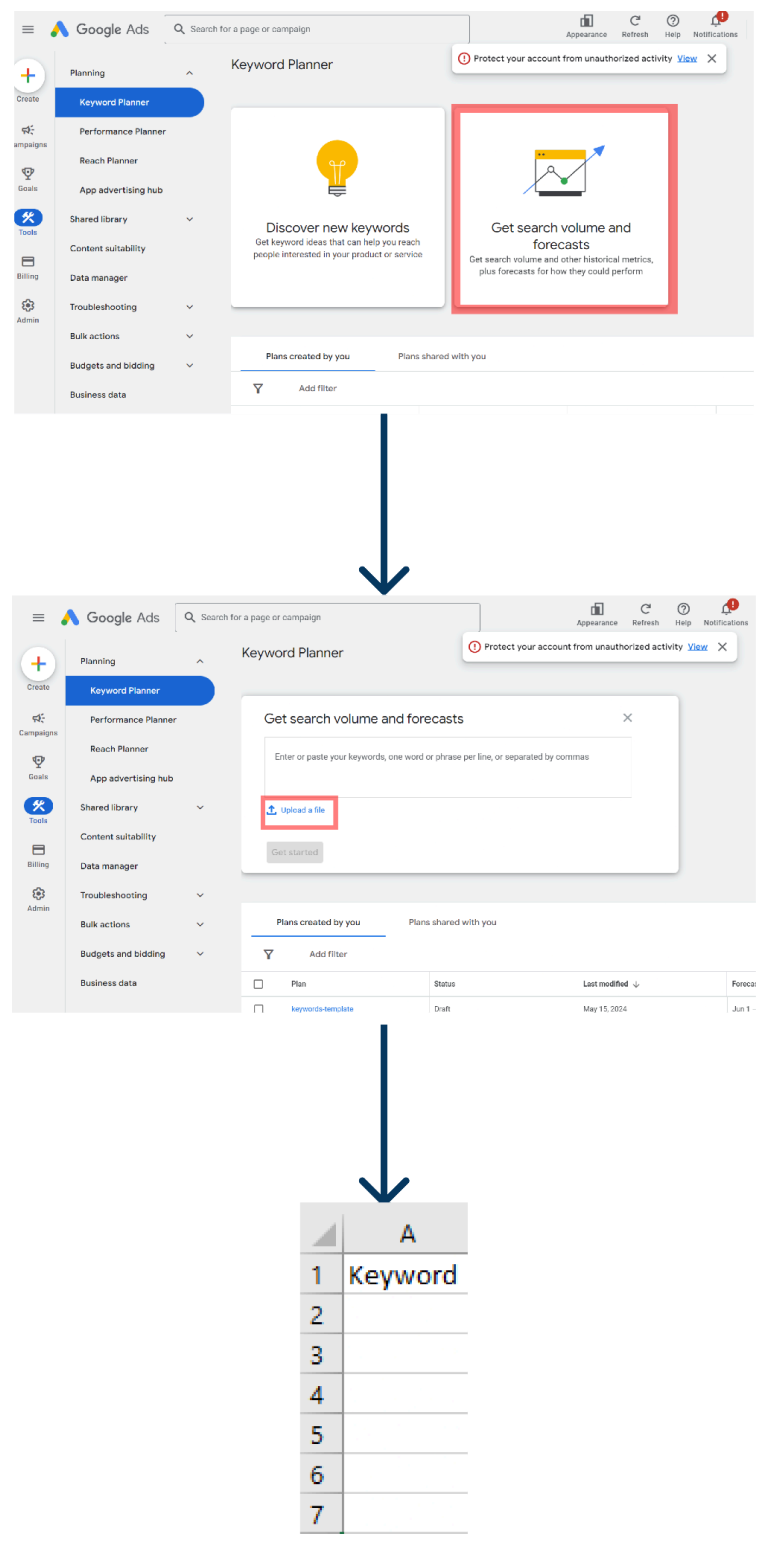
Google Keyword Planner Walkthrough

1. Click on the following link and sign in with Google account:

<https://ads.google.com/aw/keywordplanner/home>

2. After clicking on the “Get Search volume”, you reach this page which contains an input of a CSV file with a template as below.

Figure A1.
Process of Google Keyword Planner



Appendix B

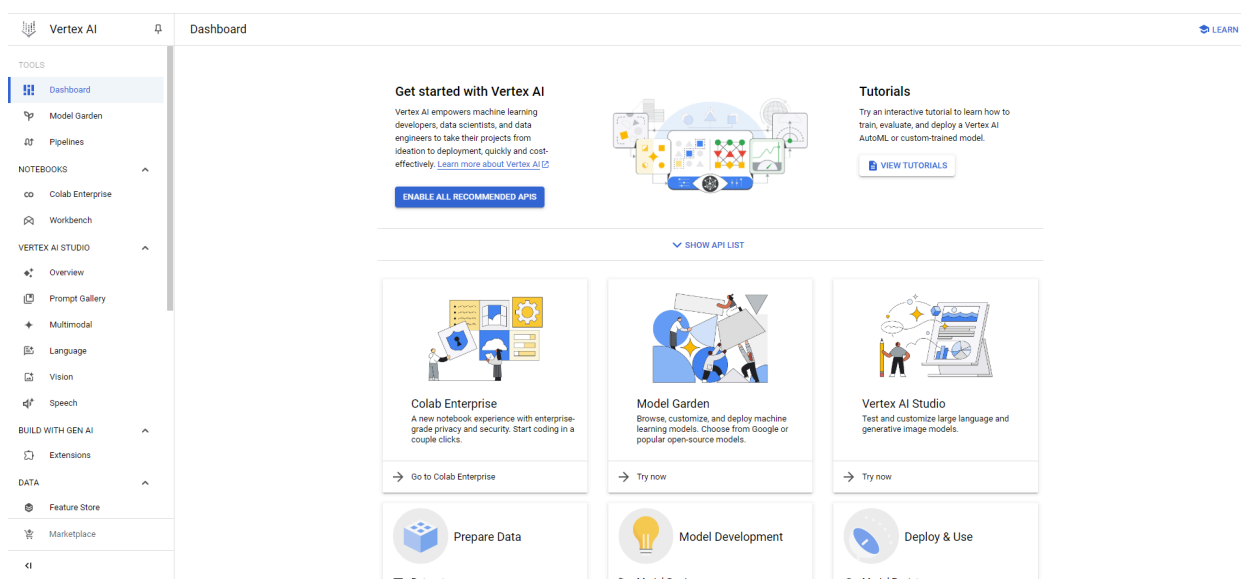
Uploading a file to Google Vertex AI

1. Sign in to the Gcloud console and click on Vertex AI or access through the following link:

<https://console.cloud.google.com/vertex-ai>

Figure B1.

Vertex AI Home Page



2. Scrolling through the sidebar on the left until Datasets appears.
3. Click on Datasets. (Currently the Vertex AutoML doesn't offer data cleaning, hence, the data has to be already cleaned)

Figure B2.

Datasets on Vertex AI

The screenshot shows the Vertex AI Dashboard interface. On the left, a navigation sidebar lists various categories: Vertex AI, Language, Vision, Speech, BUILD WITH GEN AI, Extensions, DATA, Feature Store, **Datasets** (highlighted with a red box), Labeling tasks, MODEL DEVELOPMENT, Training, Experiments, Metadata, DEPLOY AND USE, Model Registry, Online prediction, Batch predictions, Monitoring, Vertex Connect, and Marketplace. The main dashboard area features a 'Get started with Vertex AI' section with a 'VIEW TUTORIALS' button and an 'ENABLE ALL RECOMMENDED APIS' button. Below this is a 'SHOW API LIST' link. The dashboard also displays three main cards: 'Colab Enterprise' (with a 'Go to Colab Enterprise' button), 'Model Garden' (with a 'Try now' button), and 'Vertex AI Studio' (with a 'Try now' button). At the bottom, there is a navigation bar with three icons: 'Prepare Data', 'Model Development', and 'Deploy & Use'.

4. Click on “Create” button on the top.

Figure B4.

Choose the format and name of the dataset

← Create dataset

Dataset name *
untitled_1718262078233
Can use up to 124 characters.

Select a data type and objective

First select the type of data your dataset will contain. Then select an objective, which is the outcome that you want to achieve with the trained model. [Learn more](#)

IMAGE **TABULAR** TEXT VIDEO

Regression/classification
Predict a target column's value. Supports tables with hundreds of columns and millions of rows.

Forecasting
Predict the likelihood of certain events or demand.

Region
us-central1 (Iowa) ▼ ?

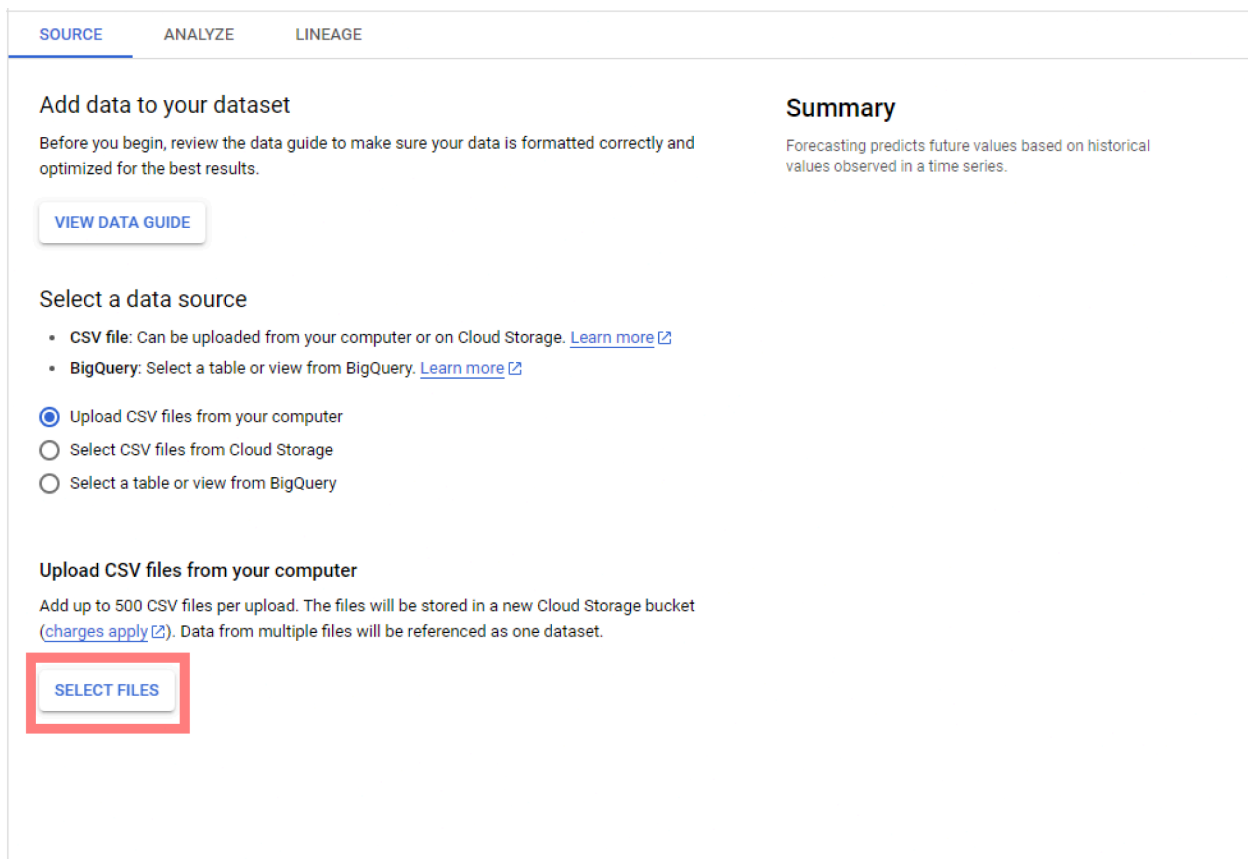
▼ ADVANCED OPTIONS

CREATE CANCEL

6. Select the csv file form local computer storage

Figure B5.

Select the files and upload them from the local computer storage



The screenshot shows the 'Add data to your dataset' interface. At the top, there are three tabs: 'SOURCE', 'ANALYZE', and 'LINEAGE'. The 'SOURCE' tab is selected. Below the tabs, there is a section titled 'Add data to your dataset' with a sub-header 'Before you begin, review the data guide to make sure your data is formatted correctly and optimized for the best results.' and a 'VIEW DATA GUIDE' button. Below this is a section titled 'Select a data source' with three radio button options: 'Upload CSV files from your computer' (selected), 'Select CSV files from Cloud Storage', and 'Select a table or view from BigQuery'. Below the radio buttons is a section titled 'Upload CSV files from your computer' with a sub-header 'Add up to 500 CSV files per upload. The files will be stored in a new Cloud Storage bucket (charges apply)'. Data from multiple files will be referenced as one dataset.' and a 'SELECT FILES' button, which is highlighted with a red box.

7. Files have been uploaded successfully. Next, training the model.

Training the model

1. Open Vertex AI console, click on training under model development in the left sidebar. Then, click on ‘Train a new model’.

Figure B6.

Select the files and upload them from the local computer storage

The screenshot shows the Google Cloud Vertex AI console interface. At the top, there's a search bar with 'forecast' and a dropdown menu showing 'airbnb-clone'. The left sidebar is categorized into 'BUILD WITH GEN AI', 'DATA', 'MODEL DEVELOPMENT', and 'DEPLOY AND USE'. Under 'MODEL DEVELOPMENT', the 'Training' option is highlighted with a red box. The main content area is titled 'Training' and features a '+ TRAIN NEW MODEL' button, also highlighted with a red box, and a 'REFRESH' button. Below this, there are tabs for 'TRAINING PIPELINES', 'CUSTOM JOBS', 'HYPERPARAMETER TUNING JOBS', 'NAS JOBS', and 'PERSISTENT'. A text block explains that training pipelines are the primary model training workflow. Below this is a 'Region' dropdown menu set to 'us-central1 (Iowa)'. A 'Filter' input field is present above a table of training pipelines.

Name	ID	Status	Job type	Model type
CBDataset	7495803183856353280	Finished	Training pipeline	Tabular forecasting

2. Choose the correct dataset. Use AutoML as a training method and continue.

Figure B7.

selecting dataset and training method

The screenshot shows the 'Train new model' interface. On the left, there is a sidebar with four steps: 1. Training method (highlighted), 2. Model details, 3. Training options, and 4. Compute and pricing. Below the sidebar are 'START TRAINING' and 'CANCEL' buttons. The main content area shows a 'Dataset +' dropdown menu with 'CBDataset' selected. Below this is a message: 'Please refer to the pricing guide for more details (and available deployment options) for each method.' A warning message states: 'Custom training with a managed dataset is not currently available. Learn more about custom model training'. Under 'Model training method', there are four radio button options: 'Time series Dense Encoder (TIDE)', 'Temporal Fusion Transformer (TFT)', 'AutoML (Default)' (which is selected and highlighted with a red box), and 'Seq2seq+'. Below these is a 'CONTINUE' button, also highlighted with a red box.

3. Add a Model Name: CBModel

Add a reference to the 'Target column': EXEMPLAREN_AANT (amount of books sold)

Add a reference to the 'series identifier column': SeriesIdentifier (used for multivariate, in this case, it is a column with value 1 in all rows)

Add a timestamp column: ONTVANGST_DATUM (Date)

Add data granularity: Daily

Holiday regions: Netherlands

Forecast Horizon: 7

Context window: 30

Click Continue

Figure B8.

Model Parameters and Feature References

Train new model

- Training method
- 2** Model details
- 3** Training options
- 4** Compute and pricing

START TRAINING CANCEL

Train new model
Creates a new model group and assigns the trained model as version 1

Train new version
Trains model as a version of an existing model

Name *
CBModel

Description

Target column *
EXEMPLAREN_AAANT

Series identifier column *
Seriesidentifier

Timestamp column *
ONTVANGST_DATUM

Forecasting configuration

Data granularity *
Daily

The granularity level of the timestamp column. Granularity must be the same for all rows. For example, if "days" is selected, timestamps must be within one day of each other. Data granularity also sets the time period granularity for the forecast horizon and context window.

Holiday regions
Netherlands

Adds a holiday column to your dataset for each selected region. A holiday column adds holiday name values to rows with holiday timestamps. [Learn more about holiday regions](#)

Forecast horizon *
7

The number of time periods into the future for which forecasts will be created. Future periods start from the most recent timestamp in the dataset.

Context window *
30

Defines the input lags to the model for each time series. For most use cases, the context window is between 0-5 times the forecast horizon value. For a starting point, try setting the context window equal to the forecast horizon value. [Learn more](#)

Export test dataset to BigQuery

4. *Preview of the selected features show and use can click continue to precede.*
5. *A text input shows with the budget for the training. It is measured in hours and the minimum is 1. In addition, it is recommended as follows:*

The data contains it will require 1-3	<100,000 rows:	1 - 3 hours	less than 100,000 rows hence, hours.
	100,000 - 1,000,000 rows:	1 - 6 hours	
	1,000,000 - 10,000,000 rows:	1 - 12 hours	
	>10,000,000 rows:	3 - 24 hours	

Click 'Start Training

Figure B9.

Budget Allocation

Train new model

- Training method
- Model details
- Training options
- 4 Compute and pricing**

START TRAINING
CANCEL

Enter the maximum number of node hours you want to spend training your model.

You can train for as little as 1 node hour. You may also be eligible to train with free node hours. [Pricing guide](#)

Budget * Maximum node hours ?

Estimated completion: 1 hour

Factors like dataset size and evaluation metrics generation can make training take longer than estimated

Figure B10.

AutoML Metric Results

Target column EXEMPLAREN_AA NT numeric	MAE ? 4,155.69	MAPE ? 12.814	RMSE ? 5,681.69
RMSLE ? 0.171	r^2 ? 0.878		

Appendix C

The full code for the study can be found using the following link:

https://drive.google.com/drive/folders/1TdAXJisS0Bx3PJjKqj9G2zqHm_2Zy-Ln?usp=drive_link

Note: the README file in the directory can help guide the user.