Machine Learning Approaches for Aftermarket Demand Forecasting: Tackling Intermittent Time Series Challenges

Sarvesh Kumar Rajavelloo

A Thesis

in

The Department

of

Supply Chain & Business Technology Management

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Supply Chain Management at

Concordia University

Montréal, Quebec, Canada

December 2023

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared,

By: Sarvesh Kumar Rajavelloo

Entitled: Machine Learning Approaches for Aftermarket Demand Forecasting: Tackling Intermittent Time Series Challenges

and submitted in partial fulfillment of the requirements for the degree of

Master of Supply Chain Management

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final Examining Committee:

_____ Chair

Dr. Navneet Vidyarthi

_____ Examiner

Dr. Navneet Vidyarthi

_____ Examiner

Dr. Xiaodan Pan

_____ Supervisor

Dr. Chaher Alzaman

Approved by:

_____

Dr. Satyaveer S. Chauhan, Graduate Program Director

_____

Dr. Anne-Marie Croteau, Dean of Faculty

Date: _____

December 6th 2023

ABSTRACT

Machine Learning Approaches for Aftermarket Demand Forecasting: Tackling Intermittent Time Series Challenges

Sarvesh Kumar Rajavelloo

This thesis addresses the significant challenge of achieving precise demand prediction within the aviation aftermarket maintenance and spare parts management sector, particularly concerning intermittent parts. These components, characterized by irregular demand occurrences, present a formidable challenge due to the difficulty in accurately estimating their demand and setting appropriate stock levels. Historical approaches, relying on conventional demand forecasting techniques, often yielded inaccurate forecasts, resulting in slow inventory turnover and increased warehousing costs. To address this challenge, a broad spectrum of techniques was examined, ranging from traditional statistical models to modern machine learning and deep learning methods falling under the broader domain of artificial intelligence. Deep learning has garnered substantial attention in time series analysis for its exceptional forecasting performance. Real-world data from an aviation company was used to implement various forecasting models, including traditional methods like the exponential smoothing, and Croston, as well as machine learning models like SVR, Random Forest, and K-nearest neighbour. Deep learning techniques, including LSTM, GRU, and CNN, were prominently featured, with customized error metrics tailored to intermittent demand forecasting. The findings highlight that, on average, deep learning models, especially Gated CNN and LSTM, outperform other models and offer highly accurate forecasts for intermittent demand. This study serves as a reference point for choosing the most effective forecasting method to support inventory planning in the aviation aftermarket, reducing costs, and enhancing service reliability. Moreover, its relevance extends to various industries dealing with intermittent demand, offering valuable insights for improved demand forecasting.

Keywords: Demand forecasting, Aviation aftermarket parts management, Intermittent time series, Deep learning

ACKNOWLEDGEMENT

I would like to sincerely thank everyone who contributed to the realization of this thesis. My sincere gratitude goes out to Dr. Chaher Alzaman, who served as my thesis supervisor and provided invaluable guidance, support, and encouragement throughout the research process. His counselling and expertise have been crucial in forming this thesis. This academic and personal journey has been profoundly transformative, and it owes its success to the support, direction, and motivation I received from numerous individuals and organizations.

I would also like to extend my gratitude to my mother and father for their unwavering patience, understanding, and inspiration throughout the thesis process. My inspiration has always come from their confidence in my ability. Finally, I would like to thank my friends and coworkers for their insightful comments, and I sincerely appreciate what they have contributed to this academic project.

TABLE OF CONTENTS

List of figures

List of tables

List of Abbreviations

ADAM	Adaptive Moment estimation
ADI	Average demand interval
ADIDA	Aggregate-disaggregate intermittent demand approach
AMAPE	Asymmetric mean absolute scaled error
ANN	Artificial neural network
ARIMA	Autoregressive integrated moving average
AUC	Area under the curve
BO	Bayesian Optimisation
CFE	Cumulative forecast error
CNN	Convolutional Neural Network
CV^2	Squared coefficient of variance.
DIVIDE	Diversity-based intermittent demand forecasting
ELM	Extreme learning machine
ES	Exponential smoothing
ETS	Error Trend and Seasonality
FFNN	Feed forward neural network
FIDE	Feature-based Intermittent demand forecasting
GMAE	Geometric mean absolute error
GRU	Gated Recurrent Unit
HHO	Harris Hawks optimisation
IMAPA	Intermittent multiple aggregation prediction algorithm
KNN	K-nearest neighbours
LSTM	Long Short-Term Memory
MA	Moving average
MAD	Mean absolute deviation
MAE	Mean absolute error
MAPE	Mean absolute percentage error
MASE	Mean absolute scaled error
ME	Mean error
MFV	Most frequent value
MLR	Multiple linear regression
MPE	Mean percentage error
MSBA	Modified SBA
NN	Neural Network
OWA	Overall weighted average
PB	Percentage better
RAE	Relative absolute error
RAND	Random sampling from past values with jittering
RBF	Radial basis function
RCNN	Recurrent convolutional Neural Network
RelMAE	Relative mean absolute error
ReLU	Rectified linear unit
RGRMSE	Relative geometric root mean square error

RMSE	Root mean squared error.
RNN	Recurrent neural network
ROC	Receiver operating characteristic
SARIMAX	Seasonal Autoregressive Integrated Moving Average Exogenous model
SES	Simple exponential smoothing
SKUs	Stock keeping units.
SBA	Syntetos-Boylan approximation
SMAE	Scaled mean absolute error
SMAPIS	Scaled mean absolute periods in stock
SME	Scaled mean error
SMPIS	Scaled mean periods in stock
SMSE	Scaled mean squared error
SPEC	Stock-keeping-oriented Prediction Error Costs
STLM-AR	Seasonal and trend decomposition using loss with AR modeling of the seasonally adjusted series
SVM	Support vector machine
SVR	Support vector regression
SWB	Sliding window bootstrapping
TBATA	Trigonometric seasonality
TSB	Teunter-Syntetos-Babia
WMA	Weighted moving average
WRMSSE	Weighted root mean squared scaled error
WSS	Willemain–Smart–Schwarz model

CHAPTER 1: Introduction

Demand forecasting, at its heart, entails anticipating future client requirements and is a critical area of work for most businesses. For instance, a firm's net revenue would be significantly impacted by even a small decrease in predicting accuracy. Demand forecasting is widely relevant in a variety of sectors where it is used as a key technique for predicting future demand for goods or services. The spare parts and service business aftermarket is a vital and inseparable aspect of the commercial aviation industry. The aviation aftermarket spare component market is worth approximately USD 22.44 billion in 2020 and is predicted to grow to USD 47.33 billion by 2028 (Fortune Business Insights, 2021). Intermittent demand patterns emerge periodically for various parts, with certain periods displaying no demand at all and when the demand exists, the size of the demand may be stable or vary significantly in size (Syntetos & Boylan, 2010). A significant difficulty arises in balancing the cost of supply chain operations against customer's demands for high availability, managing the extremely large number of stock-keeping units (SKUs), the rising cost of downtime for end users, and the high cost of managing backorders (Bacchetti & Saccani, 2012). A corporation must maintain an effective management system for its aftermarket spare parts business that covers tasks like demand forecasting, storage, part distribution, and service coordination with other aftermarket supply chain partners. For businesses that deal with aftermarket replacement parts, demand forecasting accuracy becomes critical, particularly when creating supply chain strategies that depend on accurate demand projections for individual SKUs. (Fildes et al., 2009) To enhance customer satisfaction and mitigate the potential additional expenses associated with extended lead times, companies aim to minimize delivery times to customers while maintaining a high level of customer retention. Hence, efforts must be dedicated to tackling the challenges of this complex demand pattern by enhancing the accuracy of part predictions and maintaining a high level of customer satisfaction while managing an extensive range of stock-keeping units in the aftermarket spare parts business.

1.1 Research problem

Aircraft maintenance and repair are critical for guaranteeing aircraft safety and dependability, and the timely availability of replacement parts is a critical factor in this process. Aircraft aftermarket firms play a critical role in delivering these parts and ensuring that aircraft ground time is maintained to a minimum. When an AOG situation happens, these instances are aggregated, and the components must be ready for shipment. The prevalence of irregular and inconsistent demand patterns in the aviation aftermarket part sales presents a particular challenge for these organizations. In contrast to normal items, which have more predictable and consistent demand trends, demand for aviation aftermarket components is erratic, with periodic spikes and declines. Traditional forecasting methods fail to consider the distinct attributes of intermittent demand, often resulting in inaccuracies, leading to either underestimation or overestimation running up inventory costs. This volatility makes managing inventory and overall organizational efficiency in the aviation repair and maintenance industry difficult. This shortcoming is especially problematic in a sector where the timely supply of replacement parts is critical for airline maintenance service providers and the overall smooth operation of all channels.

This study aims to enhance forecasting of intermittent part sales in the aviation aftermarket business by employing modern machine learning and deep learning models that are widely employed in other industries thereby presenting the industry leaders with practical solutions. This master's thesis aims to enhance the company's existing forecasting model, offering a more resilient approach that leads to improved punctuality in part deliveries and reduced inventory expenditures.

We aim to address the following research questions. To begin, in the context of our company's requirement to predict intermittent and irregular demand for airplane replacement parts, our first inquiry centers on identifying the most effective methods for achieving precise demand forecasts. Our second objective is to assess how optimizing hyperparameters influences the accuracy of our forecasting outcomes. Furthermore, we aim to establish whether machine learning models surpass traditional statistical techniques, such as smoothing-based methods, in forecasting intermittent data. As part of our investigation, we also strive to comprehend how various error metrics play a role in evaluating the effectiveness of these forecasting models.

1.2 Thesis structure

There will be five primary chapters in the thesis. The second chapter, "Literature Review," will provide a thorough analysis of the present concepts and research in the field of intermittent demand forecasting. It will offer the academic groundwork for the investigation, incorporating both conventional and advanced forecasting techniques. The third chapter, "Data," which follows, will include information on data sources, data cleansing, and data categorization. The models employed, model structure, and methodology will be described in the "Methodology" chapter, which will be the fourth in the series. The findings and insights from the model assessments and optimizations will be presented in Chapter 5, "Results," with an emphasis on performance measures. The "Conclusion" chapter will conclude by summarising the most important findings and their implications as well as proposing potential research directions.

CHAPTER 2: Literature review

The objective of this research is to investigate various advanced machine learning models and basic statistical models to assess their efficiency in predicting intermittent demand patterns. In addition, within the domain of spare parts inventory management, demand classification, and forecasting based on the demand pattern are essential elements. The present chapter delves into the fundamental elements of demand categorization, underscoring the crucial importance of possessing a thorough comprehension of demand patterns for enhancing inventory control.

The chapter also explores the field of demand forecasting, highlighting the application of time series forecasting techniques and the difficulties in determining their correctness. Additionally, it dives into the unique field of intermittent demand forecasting, analyzing traditional statistical methods, advanced machine learning techniques, and measures for prediction accuracy. This chapter provides a thorough examination of the latest strategies and approaches that shape the field of effective forecasting and inventory control for intermittent parts from the most relevant studies.

2.1 Classification of demand

The process of demand classification plays a crucial role in the operational efficiency of spare parts inventory management. This procedure is fundamental because it forms the basis for making judgments about forecasting and stock control that demands more accurate to keep the operational cost down. The level of precision attained in these projections is intimately correlated with the careful characterization of demand patterns based on their underlying distribution features (Bacchetti & Saccani, 2012b; Lengu et al., 2014). Demand classification serves multiple purposes beyond accurate forecasting, including enhancing inventory management, optimizing revenue generation, and ensuring the efficient lifecycle management of spare parts. The process is not standardized and differs between businesses influenced by various factors. Price, demand volume, frequency, and possible threshold levels are some of the variables that affect it.

The earliest paper that classified the demand was devised to calculate the reorder points and to reduce the overall inventory cost of the items (Williams, 1984). The method made use of the mean arrival rate as well as the mean lead time, which were then multiplied and assumed to have a Poisson distribution, in which the parts were classified as smooth, sporadic, or slow moving. This method paved the way for further improvement over the year which resulted in the next approach which categorizes the demand based on the disparities between the frequency of demand occurrences and the magnitude of each demand, and it has found extensive application in research for classifying the demand of the relevant components. To assess the typical period between the requests, a new term known as average inter-demand interval (ADI) was devised (Syntetos et al., 2005). The authors categorized the parts by assessing their irregularity and intermittency using the squared coefficient of variation and the average of inter-demand intervals. Subsequently, they grouped the entire demand into four categories: smooth, erratic, intermittent, and lumpy. Forecasting techniques were then selected according to this classification. There are other methods that have been used in literature like the ABC approach, which is frequently employed in managing spare parts inventories, is the most well-liked categorization technique (Braglia et al., 2004). The components are often based on the parts demand and unit price and are divided into three

categories: most important, relative significance, and unimportant (Hatefi et al., 2014). This approach to stock classification is the simplest available, and it has demonstrated acceptable levels of customer satisfaction and inventory management. The following widely used strategy is founded on experience and knowledge about the items under issue. Despite being straightforward, if done wrong, this procedure might produce wildly skewed results.

2.2 Time series forecasting

Time series forecasting, a pivotal analytical approach, is extensively employed across domains, including sales, finance, and stocks. Its enduring relevance is most pronounced in the realms of inventory management and production optimization, geared towards profit maximization and resource allocation efficiency. Moreover, it fulfills vital functions in pinpointing periods of both subdued and heightened demand while also facilitating the detection of product obsolescence trends in specific categories. Many academics have studied various forecasting techniques to make better predictions. However, there has been disagreement about how accurate these techniques are. Researchers have assessed them both within the data they used to develop the models (in-sample) and with new data (out-of-sample). Interestingly, it has been observed that in-sample evaluations often lead to overly optimistic results for the forecasting models (Makridakis et al., 1982). As a result, many researchers, especially in the field of inventory demand forecasting, prefer to rely on out-of-sample evaluations for more realistic assessments (Januschowski et al., 2020; Spithourakis et al., 2015). Because of its importance and usefulness, a large amount of literature has been written about it using innovative techniques in a variety of sectors. This domain is dynamic and constantly seeking breakthroughs, as seen by the continued attention and research efforts in it.

2.3 Intermittent demand forecasting

Limited research has specifically addressed intermittent time series demand data. Due to its intermittent nature, this demand differs from smooth demand in several important ways. This oddity shows itself as periods of low demand intermingled with erratic demand patterns at other times (Eaves & Kingsman, 2017). Different approaches were proposed to determine the intermittency of a time series. The first stage entails calculating the average time between successive demand occurrences, which is an essential indicator for determining how intermittent the demand pattern is. The degree of fluctuation in demand regarding order magnitude is then measured using the coefficient of variation, which is represented by the standard deviation normalized by the mean. To assess the level of statistical independence between the size of demand and the frequency of demand occurrences, the study also includes the computation of autocorrelations and cross-correlations (Willemain et al., 1994). Several methods have been determined specifically to deal with the problem of intermittency.

The principal technique employed is Croston's method, which is among the initial models that gained recognition for tackling the difficulties related to intermittent demand patterns. (Croston, 1972a). This approach has garnered substantial acclaim and acknowledgment, receiving recognition from a broad spectrum of researchers and industry professionals and despite the advent of newer forecasting techniques, the Croston method continues to be employed as the standard of comparison in various studies (Babai et al., 2019; Zhu et al., 2017). This method showed higher performance and efficacy compared to conventional statistical time series forecasting techniques like the simple moving average and simple exponential smoothing (Syntetos et al., 2005) by

exhibiting a less variation in errors. But those methods are still in use and have been shown to give plausible results in some specific cases of intermittency (Wallström & Segerstedt, 2010). In Croston's method, non-zero demand occurrences and non-zero demand interval data are divided into separate time series. To estimate the anticipated future demand, an exponential smoothing approach is then independently applied to both time series. This forecasting technique's effectiveness is inextricably linked to the choice of the smoothing parameter alpha since the results of the forecasting depend on its value. Since its inception, the fundamental Croston model has undergone significant enhancements and refinements through the dedicated efforts of various researchers. The Croston method was empirically shown to possess a bias in its predictions, particularly favouring positive demand occurrences. This observation prompted the development of an alternative formulation by Syntetos and Boylan, which aimed to rectify the inherent shortcomings of the original method and provide a more accurate forecasting approach for intermittent demand patterns (Syntetos & Boylan, 2005). Empirical evidence suggests that both Croston's method and the Syntetos and Boylan approach may exhibit limitations when confronted with abrupt declines in demand, indicating suboptimal performance in scenarios involving items approaching obsolescence (Romeijnders et al., 2012).

Further improvement was done by Teunter, Syntetos, and Babai to accommodate situations involving obsolescence, particularly when demand dwindles to zero. To achieve this, the methodology involved continuous updates of demand estimates in each period, as opposed to updating only when demand events occurred, thereby addressing the unique challenges associated with intermittent demand forecasting more comprehensively and it was coined as the TSB method (Teunter et al., 2011a). Croston's method has a single smoothing constant that is uniformly applied to both demand size and demand intervals. However, in the case of the SBA and TSB methods, a departure from this approach is observed, as these methods utilize two distinct smoothing constants—one specifically tailored for demand size and another exclusively for demand intervals. Subsequent research efforts have yielded a multitude of alternative methods tailored specifically for handling intermittent data, with notable examples being the application of bootstrapping techniques (Porras & Dekker, 2008), Integer–valued Auto– Regressive Moving Average briefly known as the INARMA model(Engelmeyer, 2016), predictive count data distributions (Kolassa, 2016)and aggregation of the time series (Kourentzes et al., 2014; C. Li & Lim, 2018).

2.4 Intermittent demand forecasting with advanced methods

Apart from the basic statistical methods, the more advanced machine learning, and its deep learning sunset methods have also been applied to the specific problem of intermittent demand forecasting and are a viable alternative because of their ability to address the non-linear patterns in the data (Gutierrez et al., 2008). As computing power increased, several machine-learning techniques were created and used to anticipate demand in both theoretical studies and practical implementations.

Numerous research papers have applied various machine-learning methods to address this type of demand pattern. A selection of these papers includes the use of a support vector machine and an artificial neural network model and compares it with a basic Croston model (Assaghir et al., 2017). It is noteworthy that even the most fundamental Support Vector Machine (SVM) model has demonstrated superior performance compared to both the Croston model and the ARIMA model (Hansen et al., 2006). This phenomenon can be attributed to the model's ability to generalize

effectively and create unique solutions that surpass the local minima, allowing it to outperform the basic methods (Bao et al., 2004). In another study, a paper employed a combined approach that integrated Support Vector Regression (SVR) with logistic regression (Hua & Zhang, 2006). But incidentally, when compared to other machine learning forecasting methodologies, the precision obtained using Support Vector Regression using the linear kernel was among the least favourable (Makridakis et al., 2018a). The subsequent technique employed was the k-nearest neighbour method, which demonstrated its highest utility when integrated with other statistical approaches (Petropoulos et al., 2016). Other methods such as the XGBoost method and random forest method (Assaghir et al., 2017) were also studied. However, it is worth noting that the most widely favoured models in this study were distinct types of deep learning models.

The increased availability of extensive datasets has made deep learning models, particularly neural network models, a subject of great interest among researchers. These techniques are renowned for their adaptability and capacity to successfully manage non-linearities that may exist in the data (G. Zhang et al., 1998) and they can approximate a wide range of continuous functional relationships (Gutierrez et al., 2008b). Numerous modifications and variations of the foundational model have been experimented with, yielding diverse outcomes. An intricate multi-layered LSTM model was designed to predict highly fluctuating demand, with a focus on optimizing model parameters for improved accuracy (Abbasimehr et al., 2020) and a neural network model has been developed that uses ensemble input to improve the precision of demand forecasting by using the median of forecasts from many neural networks (Kourentzes, 2013).

Considerable debate has arisen concerning the utilization of machine learning models for forecasting this data type, with observations suggesting that they frequently yield subpar results when contrasted with traditional, less resource-intensive statistical models (Makridakis et al., 2018b). Another potential drawback faced by these models is their demand for extensive training data to effectively capture hidden patterns within time series, which can be particularly challenging in the context of intermittent data characterized by numerous zero-demand periods interspersed between active periods(Gutierrez et al., 2008b). A counterargument has been made, however, that these models, when compared to statistical models, can offer much more accurate and less skewed forecasts in some cases (Semenoglou et al., 2021).

2.5 Error metrics

To determine the most effective forecasting method for practical implementation, it is imperative to quantify the outcomes of each employed method and select the most appropriate error metric for this purpose. This aspect gains particular significance when dealing with intermittent demand due to the presence of multiple zeros in the time series, as certain metrics have the potential to bias or distort the results (Hyndman & Koehler, 2006). Forecast evaluation metrics can be broadly divided into two categories: intrinsic and extrinsic measures. Intrinsic metrics assess forecast accuracy solely based on the generated forecast and the actual ground truth data. Conversely, extrinsic metrics incorporate an external reference forecast in conjunction with the generated forecast and the ground truth data to evaluate forecasting performance. This classification helps in comprehensively assessing the quality of forecasting models and methods.

Research has shown a wide range of performance indicators are available in the current literature, which may be used to evaluate and compare the precision of different forecasting

methodologies for predicting the demand for spare parts. Additionally, research has indicated that relying solely on traditional methods may not be the most suitable approach, potentially leading to misleading or inaccurate results (Kourentzes, 2013). Specific metrics like the mean absolute percentage error may yield inconclusive outcomes due to the significant presence of zero values in our dataset, rendering them less informative (Kim & Kim, 2016) with this being the case for most measures that consider the time series data's natural patterns and properties when assessing predicting accuracy. In those cases, an improvement is made to those models so that they can give out meaningful results, e.g., Symmetric Mean Absolute Percentage Error is proposed to negate the presence of zeros in the actual series (Makridakis & Hibon, 2000). The percentage improvement statistic, which evaluates how the model performs in comparison to a baseline model—in this case, often a naive technique or the Croston model—is a widely used metric. An additional point of concern pertains not only to the precision of the forecasts but also to the accuracy of inventory management based on these forecasts. This becomes crucial when there is a need to prioritize one over the other depending on the specific context. The best performance in real-world stock control situations cannot be guaranteed by just demonstrating higher accuracy on theoretical grounds. In these circumstances, the installation of models that provide greater stock control measures is preferred (Kourentzes, 2013).

2.6 Summary

A comprehensive summary tables containing the studies conducted in the field provides a concise overview of the most recent findings. This tables methodically displays the many approaches, metrics used, data processing procedures, and hyperparameter tuning strategies used in these investigations which improves comprehension of the body of current research.

Table 2.1: Literatures utilizing basic statistical models

STUDY	TECHNIQUES	METRICS	TUNING / DATA PROCESSING
Cheng et al., 2016	ARIMA & CRO	MAPE, RMSE, RGRMSE	No data preprocessing or model tuning
Kilimci et al., 2019	MA, ES, ARIMA, MLR, SVR, Holts Trend and winter methods	MAPE, MAD	Stock-related features of products
L. Li et al., 2023	Naïve, seasonal naïve, SES, MA, ARIMA, ETS, CRO, SBA, TSB, ADIDA, IMAPA, FIDE, DIVIDE	RMSSE	Using initial non-zero demand and nine time series features input.
Luochen & Hasachoo, 2021	CRO, TSB, SBA, MA, WMA, ES, Kalaya et al' approach	MSE	No data preprocessing or model tuning
Petropoulos & Kourentzes, 2015	Naive, CRO, SBA, ES, MA: Combination of different methods and different frequencies	SME, SMAE, SMSE, SMPIS, SMAPIS	Non-overlapping temporal aggregation process

Rožanec et al., 2022	Naïve, SES, MA (3), MFV, Random sampling from past values with jittering, LightGBM, Catboost	AUC/ROC, MASE, SPEC	Different combinations of models
Willemain et al., 2004	ES, CRO, Bootstrap	MASE, MAD	No data preprocessing or model tuning

With a few exceptions involving sophisticated models, Table 2.1 lists a variety of models that are mainly based on statistical models and their variations. Numerous studies primarily used an aggregation process in conjunction with various model combinations to improve results. Furthermore, in most of the papers, little to no effort was put into data processing or tuning.

Table 2.2: Literatures utilizing machine learning models

STUDY	**TECHNIQUES**	**METRICS**	**TUNING / DATA PROCESSING**
Hoffmann et al., 2022	MA, Linear regression, ES, CRO, SBA, ANN	MAPE	Model tuning using varying alphas
Jeon & Seong, 2022	DeepAR, Rolled DeepAR	WRMSSE	Time series, price and calendar features
Jiang et al., 2021	SES, ARIMA, SBA, MSBA, WSS, SWB, FFNN, RNN, SVM, AUSVM	MASE, AMAPE, SME	Adaptive tuning for SVM models
Kourentzes, 2013	Naïve, MA, SES, CRO, Dual-NN, Rate-NN	MAE, ME	Regularization
Lolli et al., 2017	FFNN, Time-Delay NN, RNN	MAPE, ME	Last Non-Zero Demand, Demand Separation, Successive Zero Demand Periods.
Sousa et al., 2022	TBATS, Prophet, MLP, LSTM, HistGB	Win Ratio, RelMAE	Normalization, Outlier Treatment

Research studies that used deep learning models for experimentation are listed in Table 2.2. While it serves as a commendable starting point, all the papers predominantly opted for simpler versions of neural networks, such as recurrent neural networks and feedforward neural networks. Notably, there was minimal emphasis on the utilization of convolutional neural network models.

Table 2.3: Literatures with their various error metrics

STUDY	**TECHNIQUES**	**METRICS**	**TUNING / DATA PROCESSING**
Chaudhuri & Alkan, 2022	ELM with HHO, GRU, ARIMA, SARIMAX, ELM - BO	MAPE, MPE, RMSE	Normalization, tuning of nodes and activation function

Pennings et al., 2017	SES, CRO, SBA, SY, TSB, DLP, Bootstrap, Bootstrap DLP	MASE, GMAE, Service Level	No data preprocessing or model tuning
Montero-Manso et al., 2020	Naïve, seasonal naïve, RAND, ARIMA, ES, TBATA, STLM-AR, NN	OWA	Forty-two-time series features input
Güven et al., 2021	KNN & RF	RMSE	Twenty-nine product features include colour, price, and tourist count.
Gutierrez et al., 2008b	CRO, SBA, ES, NN	MAPE, RGRMSE, PB	Last Non-Zero Demand, Demand Separation Period, Top of Form
Mukhopadhyay et al., 2012	WMA (5), SBA, NN, SES: Combination of different data splits	MAPE, RAE, RGRMSE, PB	Adjusted for varying constants, Features as Last Period Demand, Cumulative Zero Demand Periods

The literature presented in Table 2.3 summarizes various error metrics, predominantly measured in absolutes or percentages through statistical computations. Notably, limited attention is given to assessing forecast bias or considering associated costs. Despite extensive exploration of intermittent time series predictions in the literature, existing studies exhibit limitations. These include a focus on known approaches with inadequate exploration of advanced methodologies, insufficient insights into data transformation challenges, and neural network models lacking detailed architectural and hyperparameter information. Additionally, the study lacks a thorough examination of parameter modifications in neural network models. These shortcomings underscore the necessity for further research and methodological enhancements in the field of intermittent parts management. Consequently, this work distinguishes itself from prior studies by incorporating advanced sequence learning techniques, introducing novel convolutional architectures, and fine-tuning them for optimal results, thereby contributing valuable insights to the existing literature.

CHAPTER 3: Data

To assess and examine models for intermittent data, we utilize a real-world dataset that includes sales of aircraft spare parts for two and a half years. This chapter introduces the dataset employed and we describe how we gathered, cleaned, and categorized the data related to spare parts in the study.

3.1 Data collection

The data for this research was provided by aviation aftermarket leader in Mirabel, Quebec who specializes in the sales of aftermarket parts to a variety of aircraft, which primarily consists of regional jets. The dataset includes daily sales records for a large inventory of 30,000 distinct SKUs that were rigorously recorded over a two-and-a-half-year period. All additional information related to the components, including cost and inventory levels, was maintained as confidential in adherence to regulatory requirements. Moreover, to ensure confidentiality, all part numbers were substituted with anonymized placeholders.

A sample of the raw data obtained for the parts is given below.

Table 3.1: Example of raw data

Index	Part Number	2020-06-01	2020-06-02	2020-06-03	...
1611	1448OM	NaN	3.0	NaN	...
13166	4675TB	NaN	NaN	NaN	...
20016	6626DI	NaN	NaN	1.0	...
22445	7317BG	NaN	5.0	23.0	...
25512	8177LN	NaN	20.0	30.0	...
...	...	..	...	...	...

As evident, there are many missing data, so it was necessary to clean and better prepare the data. This essential decision was made to guarantee the dataset's stability and dependability for further investigation.

3.2 Data cleaning and examination

The raw data, as observed in Table 3.1, contained numerous missing values, requiring thorough cleaning before any models could be developed from it. To enhance understanding and visualization, all data points with missing values were replaced with zeros. A line plot of one of the parts under examination is shown in Figure 3.1 with the date index being represented by the x-axis, and the sales values for that date being represented by the y-axis. The plot visibly demonstrates the irregular character of this specific part's demand pattern.

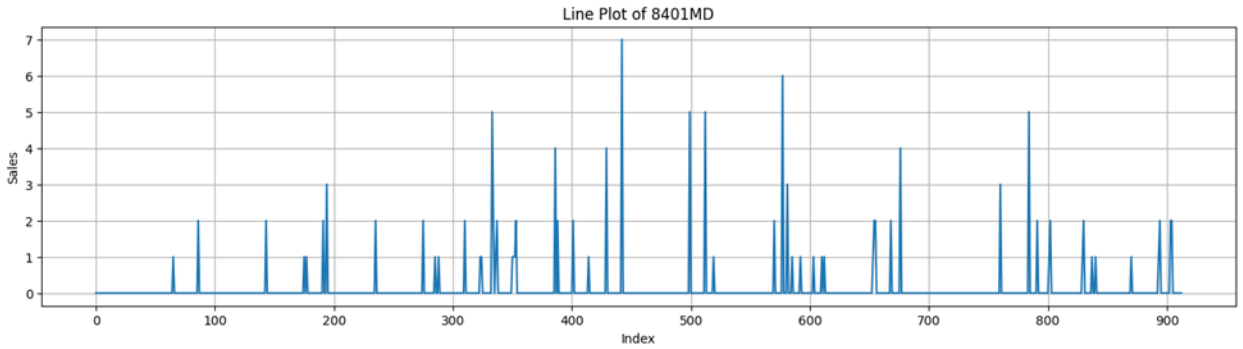

Figure 3.1: Sales plot of part 8401MD

The percentage breakdown of unique values within the sample data, which comprises the first 1000 parts in the dataset, is shown as a histogram in Figure 3.2. The height of each bar in the histogram, which represents a unique value, reveals how frequently that value occurs in the dataset. The histogram highlights a noteworthy difficulty in modelling this data: more than 94% of the days in the time series show no sales, a marked imbalance that complicates the modelling procedure.

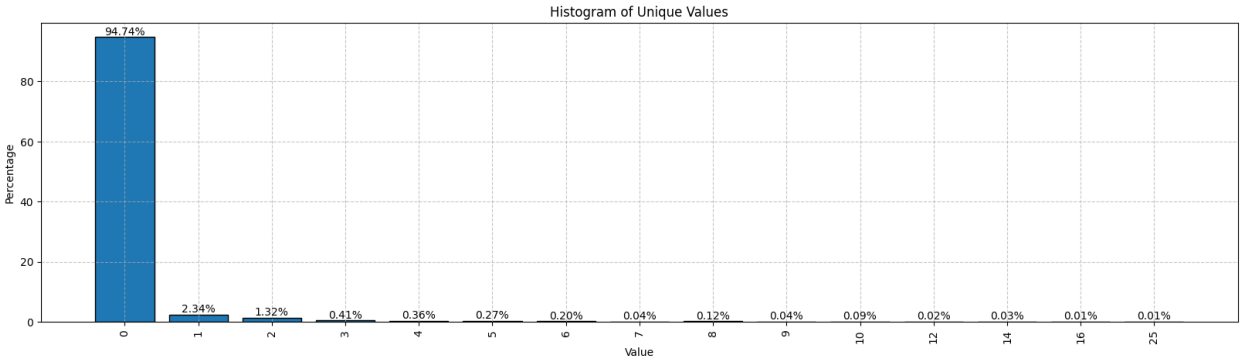

Figure 3.2: Histogram of unique sales values

Every daily consignment total that was recorded represented the daily demand. Like many other businesses, there was also occasional observation of negative numbers in the databases, which indicated that certain parts had been returned to the warehouse. Since it was hard to link the returns to previous shipments in these cases due to a lack of information, we chose to replace these negative values with zeros. The replacement strategy with zeros was carefully considered because the other option would have included treating returns as if they were random replenishments.

3.3 Data categorization

The sales data we obtained underwent categorization according to their distribution patterns, intending to discern which components display intermittent characteristics. The categorization of the components falls into four overarching groups: smooth, intermittent, lumpy, and erratic. This categorization, commonly referred to as the SBC classification method, was based on the utilization of Average Demand Interval (ADI) and squared coefficient of variance (CV^2) values, as recommended by established scholarly works (Syntetos et al., 2009, 2012). The ADI, as the name implies, provides the average time gap between two consecutive instances of demand,

which evaluates the consistency of demand patterns. The CV^2 serves as a metric that gauges the level of variation in data while excluding instances of zero demand.

$$\text{ADI} = \frac{Total\ number\ of\ observation}{Number\ of\ non-zero\ demand} \tag{3.1}$$

$$CV^2 = \frac{standard\ deviation}{mean\ value} \tag{3.2}$$

The demand classification matrix is defined as follows.

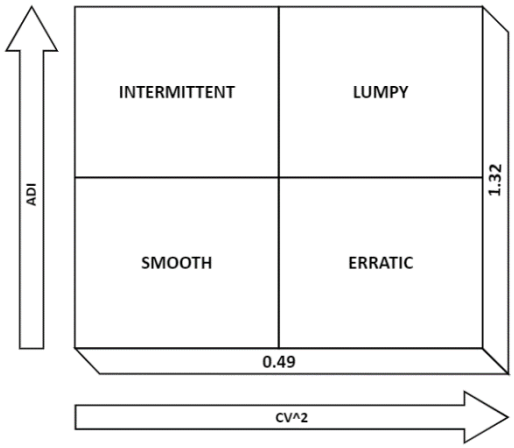

Figure 3.3: Demand-based data categorization matrix

The threshold values for ADI and CV^2 were taken as 1.32 and 0.49, respectively. Intermittent data is classified as any time series having ADI equal to or greater than 1.32, but CV^2 remains below 0.49. In such cases, client's transactions remain steady, but the time intervals between them might fluctuate greatly. This makes making very precise projections about future sales more difficult. Smooth demand is distinguished by regularity in timing and consistency in amount, with both ADI and CV^2 values falling below the threshold which is easier to predict. Erratic demand occurs when the ADI is less than 1.32 but the CV^2 is 0.49 or higher, showing the consistent time but considerable changes in demand quantity. The last category is lumpy, which has an ADI of 1.32 or higher and a CV^2 of 0.49 or higher, indicating significant changes in demand quantity and time. These metrics are defined in the matrix in Figure 3.3. The Figure 3.4 provides a general representation of the various demand distributions discussed above.

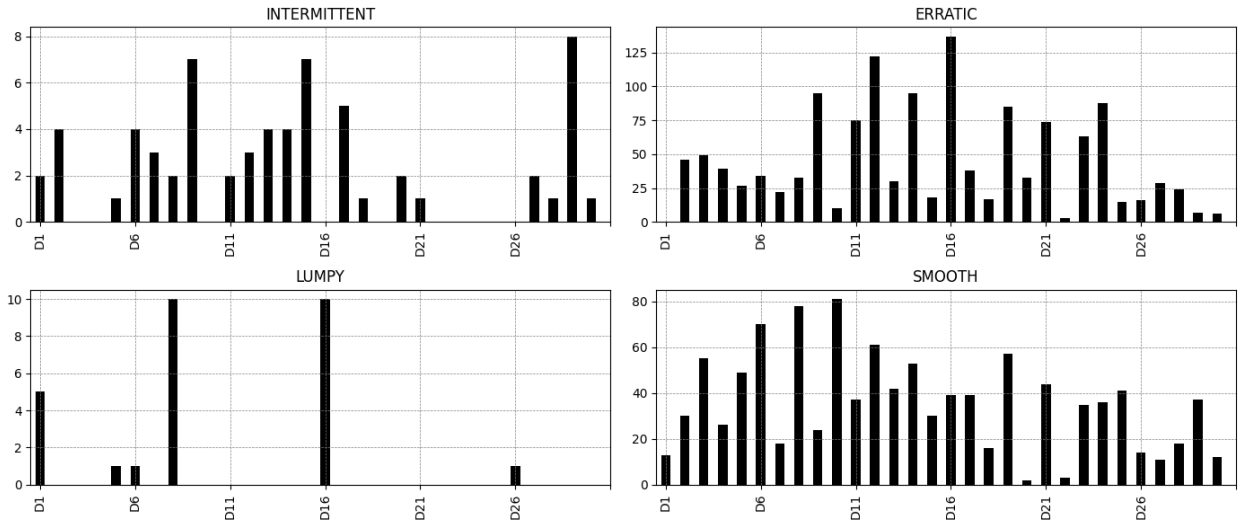

Figure 3.4: Demand type classification example

With these metrics equipped, we classify our dataset into the four categories defined. First, the ADI and CV² values for all the time series were found. From the statistics of the values found in Table 3.2, it shows that the mean of the both the ADI and CV² are far off from the threshold values for our classification which shows intermittency.

Table 3.2: ADI and CV² statistics

	ADI	CV^2
count	31874.000	31874.000
mean	423.846	0.386
std	347.608	0.689
min	4.058	0.000
25%	114.125	0.000
50%	304.333	0.111
75%	913.000	0.531
max	913.000	27.052

The following plot (Figure 3.5) is obtained for the classification using ADI and CV² values for all the time series in question.

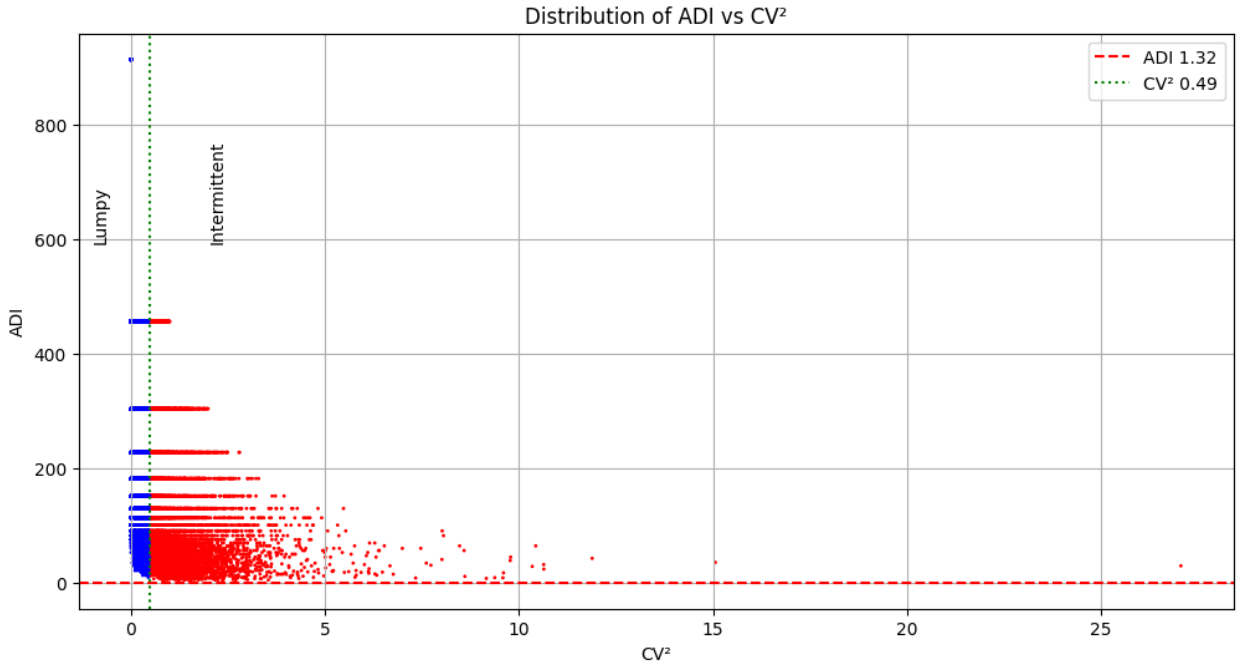

Figure 3.6: ADI vs CV^2 scatter plot

The statistical evidence is confirmed by the plot and most of the time series are lumpy or intermittent according to Figure 3.5. Table 3.3 below further explains it by showing us that more than 73 percent of the data is intermittent, and much of the remaining data is lumpy. A small number of time series are indicated as 'No demand' since they had no sales for the entire period. A portion of the time series was chosen for additional analysis using our models to accommodate time and computational resource constraints.

Table 3.3: Demand classification statistics of data

Demand	Parts	Percentage(%)
Intermittent	23324	73.176%
Lumpy	8550	26.824%
No Demand	14	0.044%
Erratic	0	0%
Smooth	0	0%

CHAPTER 4: Methodology

In this chapter, we will provide a comprehensive and detailed overview of the various models employed in our research. We will delve into the structures of each model and the parameters that were tuned for optimization. Our study covers a total of fifteen distinct models, categorized into four statistical models, followed by three elementary machine learning models, and finally, a set of advanced deep learning models. Furthermore, we will expound upon the software tools and libraries that were instrumental in the process of model tuning and data preparation.

4.1 Croston model

The widely used exponential smoothing method has biases and restrictions, which led to the development of the Croston model (Croston, 1972b). The exponential smoothing strategy places more emphasis on current demand data, leading to estimates for future demand that are more optimistic immediately after periods of demand and less optimistic after periods of zero demand. This approach is inappropriate for products with sporadic demand patterns. So, the Croston method splits the demand probability and demand size and predicts them separately which results in a more overall accurate forecast. The procedure requires creating two separate time series, one for instances of non-zero demand and another for the intervals between successive non-zero demand events and using exponential smoothing to independently compute each of them. Within this model, updates are exclusively made to the demand size and demand interval parameters when a non-zero demand event takes place. The formula for Croston model is given as below,

When a demand occurs, i.e., $d_t > 0$,

$$l_{t+1} = \alpha d_t + (1- \alpha)\, l_t \quad (4.1)$$

$$p_{t+1} = \alpha q + (1- \alpha)\, p_t \quad (4.2)$$

$$f_{t+1} = \frac{l_{t+1}}{p_{t+1}} \quad (4.3)$$

$$q = 1 \quad (4.4)$$

And when the demand is zero, i.e., $d_t = 0$,

$$l_{t+1} = l_t \quad (4.5)$$

$$p_{t+1} = p_t \quad (4.6)$$

$$f_{t+1} = f_t \quad (4.7)$$

$$q = q + 1 \quad (4.8)$$

Where,

d_t = Demand at time t

l_t = level estimate at time t

p_t = periodicity at time t

f_t = Forecast at time *t*

q = time interval between two nonzero demands

α = smoothing factor, $0 < \alpha < 1$

The baseline model used a smoothing factor (α) value of 0.5. After forecast and error metrics were obtained from the baseline models, hyperparameter tuning was done using an array of smoothing values for each time series individually.

4.2 Syntetos-boylan approximation model

Although this approach segregates the demand interval and demand size into distinct time series, its application revealed that the enhancements introduced by the model were marginal in significance. One major flow found in the basic Croston model is that the forecast does not change when there is zero demand and it results in an overestimation of the forecast, and this overestimation demonstrates a positive correlation with the smoothing factor α (Syntetos & Boylan, 2001). To counteract this positive bias, Syntetos and Boylan made a minor adjustment is introduced into the forecasting formula.

$$f_{t+1} = (1 - \frac{\propto}{2}) \frac{l_{t+1}}{p_{t+1}} \quad 4.9$$

The approach for determining the level and periodicity remains unchanged; however, the forecast now incorporates the smoothing constant to mitigate the bias induced by the basic Croston formulation.

Just like the basic Croston model, the baseline model used had a smoothing factor (α) value of 0.5 again. After the basic models' forecast and accuracy metrics were determined, hyperparameter tuning was conducted using a range of smoothing values for each time series separately.

4.3 Teunter-syntetos-babai model

Despite the improvements made by Syntetos and Boylan, the model still could not account for products that would become obsolete. To improve upon the model, a formulation was employed where individually exponentially smoothed estimates for both the probability of demand occurrence and the magnitude of demand (Teunter et al., 2011b). In practical application, the level estimate would remain consistent with the standard Croston model, while the periodicity would be regularly updated at each time interval, even if there is zero demand, and expressed as the probability of a demand event occurring. Following periods of zero demand, the periodicity diminishes, while after periods of positive demand, it increases, consequently influencing the final forecast. This enhancement would enable the model to respond more effectively in scenarios of product obsolescence, in contrast to the Croston model, where adjustments are made solely when a demand event occurs.

When a demand occurs, i.e., $d_t > 0$,

$$l_{t+1} = \alpha d_t + (1- \alpha)\, l_t \quad 4.10$$

$$p_{t+1} = \beta + (1- \beta)\, p_t \quad 4.11$$

$$f_{t+1} = l_{t+1}p_{t+1} \quad 4.12$$

And when the demand is zero, i.e., $d_t = 0$,

$$l_{t+1} = l_t \quad 4.13$$

$$p_{t+1} = (1-\beta)\, p_t \quad 4.14$$

$$f_{t+1} = l_{t+1}p_{t+1} \quad 4.15$$

$$\text{If } d_t = 0 \text{ then } \begin{cases} l_{t+1} = l_t \\ p_{t+1} = (1-\beta)p_t \\ f_{t+1} = l_{t+1}p_{t+1} \end{cases} \quad 4.16$$

The baseline model in this case had two smoothing parameter values, α and β both used a value of 0.5. Once we had obtained forecasts and error metrics from the initial baseline models, we proceeded to fine-tune the model's hyperparameters. This involved adjusting a range of smoothing values for both parameters and for each time series within our dataset.

4.4 Exponential smoothing

Simple exponential smoothing is one of the simplest and oldest methods of forecasting demand that is straightforward to construct due to its basic recursive computation technique. It has been demonstrated to be competitive with more complex forecasting approaches and it is still widely used today. The idea behind exponential smoothing is to add a smoothing procedure to the source data series, like what moving averages do. This smoothing produces an altered series, which is then used to forecast future values of the variable under discussion.

The simple exponential smoothing method is given as follows,

$$F_t = \alpha \cdot Y_t + (1-\alpha) \cdot F_{t-1} \quad 4.17$$

Where F_t is the forecasted value for the next period, Y_t represents the actual observation for the current period. F_{t-1} corresponds to the forecast made for the current period, which is derived from the previous step and α serves as the smoothing parameter or factor, which is a value between 0 and 1. The value of α influences the outcome of the forecast to a great extent; the higher the value, the higher is weightage given to the most recent values. As with the prior models, the basic model used a smoothing factor value of 0.5 and was later adjusted for each forecasted part using a range of alternative values.

4.5 Random forest

Random forest is a machine learning method that is widely used for various tasks including classification and regression, encompassing multiple decision trees whose average is taken as the output (Breiman, 2001). The model undergoes training on multiple decision trees and subsequently employs these trees to make predictions about individual samples (K. Zhang et al., 2022). Bagging is the primary premise, and it creates unpruned decision trees from varied training data fits, picking optimal split features based on impurity criteria for superior ensemble learning this process enhances the model's stability and accuracy, lowers variance, and aids in preventing overfitting.

Individually, every tree is built using a bootstrap sample from the training data, and at each node, a random subset of features is selected with the trees in the forest grown to their maximum.

Algorithm: Random Forest for Regression (Hastie et al., 2009)

1. For $b = 1$ to B:

    (a) Draw a bootstrap sample Z^* of size N from the training data.

    (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.

        i. Select m variables at random from the p variables.

        ii. Pick the best variable/split-point among the m.

        iii. Split the node into two daughter nodes.

2. Output the ensemble of trees

To make a prediction at a new point x Regression:

$$f_{rf}^{B}(\mathrm{x}) = \frac{1}{B}\sum_{b=1}^{B} T_b(\mathrm{x}) \qquad 4.18$$

Due to the inclusion of random sampling and the increased properties of ensemble techniques, the Random Forest methodology offers greater generalization and trustworthy estimations (Qi, 2012).

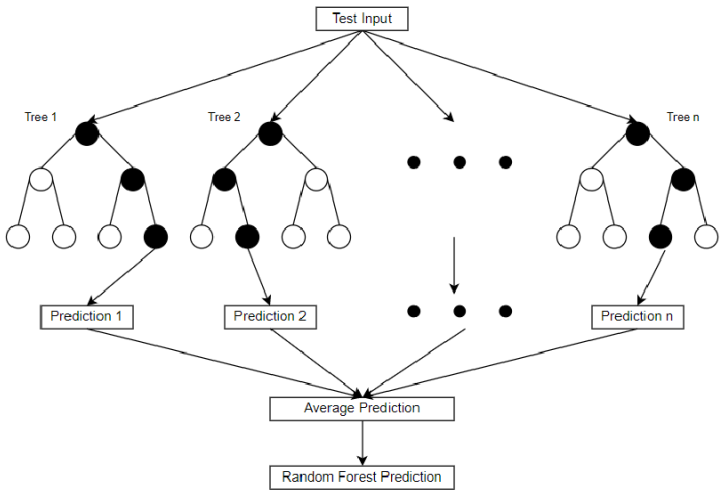

Figure 4.1: Random Forest algorithm for regression (Son & Yang, 2022)

The scikit library and the built-in parameters were used to build the basic random forest regressor model. The least number of samples needed to split an internal node is two, the minimum number of samples needed to be at a leaf node is one, and the minimum number of samples needed to be at the forest's one hundred trees maximum depth is zero. A grid search was used to fine-tune each of these parameters using an array of values for each time series.

4.6 Support vector machines

SVMs, which were originally used for classification tasks in 1992, were later modified for regression challenges. SVR, which is specially built to adapt to existing data based on performance benchmarks, is then used to forecast unfamiliar data points using the trained model (Boser et al., 1992). They are a powerful supervised method whose main objective is to find a hyperplane in an N-dimensional space, where N is the quantity of features, on the surface. The objective of this hyperplane is to efficiently segregate and classify the data points in a distinct and comprehensible manner (Cortes et al., 1995). Our goal is to locate the plane with the largest margin—basically, the biggest distance between data points from distinct classes. The widest radius around a classification border that is devoid of any data points is referred to as the maximum margin, while the nearest data points are known as support vectors. These support vectors, which are the hardest to categorize, are crucial for establishing the decision boundary and creating the classification model. Kernel functions, which define the shape of both the hyperplane and the decision boundary, are another crucial component. The non-linear radial basis function (RBF) kernel is used to convert the original input space and identify the regression hyperplane in a higher-dimensional feature space (Smola & Schölkopf, 2004).

$$\text{MIN}\,\frac{1}{2}||w||^2 + C\sum_{\varepsilon=1}^{N}|\xi| \qquad 4.19$$

In this context, w denotes the object value, C represents the hyperparameter governing model behaviour, and ξ signifies the margin of error between support vectors and observed data points. The error minimization function is employed to reduce the gap between the support vector and the hyperplane.

The formula for the margin of error is given as,

$$|yi - wixi| \leq \varepsilon \qquad 4.20$$

In this context, y_i corresponds to the target value, w_i represents the coefficient, x_i stands for the predictor value, and ε denotes the margin of error between the hyperparameter and the support vector. This formula acts as a constraint within the minimization function.

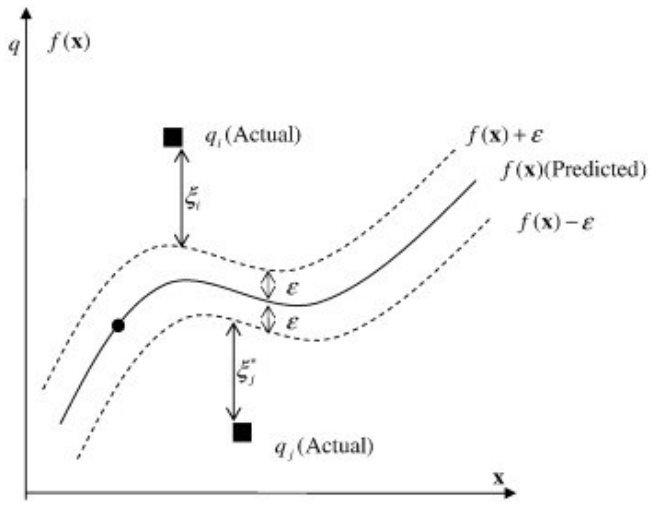

Figure 4.2: Support vector regression description (Lu et al., 2009)

The baseline model was produced using the scikit learn python library's default parameters, the same as the prior model. In this model, the regularisation parameter is set to one by default and the kernel type is "RBF". Later, these parameters were fine-tuned using grid search and various values. For the "RBF" kernel, the regularisation parameter value ranged from 0.1 to 100 and the kernel coefficient gamma value from 0.01 to 10.

4.7 K nearest neighbors

A well-known nonparametric method used for both classification and regression applications is the k-nearest neighbours (KNN) algorithm. This model has undergone a series of studies and improvements over the years with a solid foundation being laid in 1967 with the proposed and developed idea of "nearest neighbour rules," which formed the basis of the KNN algorithm as it is currently known (Cover & Hart, 1967). When using the k-nearest neighbours' method, the choice of the parameter k-which denotes the number of nearest neighbours considered is crucial. By choosing a lower value of k, a model with increased sensitivity to minute differences in the data may be produced. A model produces a more continuous and smoother decision border or prediction surface when k is set to a higher value. To use this model for a univariate time series problem, the data is fitted in the form of its own lagged values. By locating identical historical patterns and extrapolating from their subsequent behaviour, KNN for time series forecasting makes use of repeating patterns in time series data to estimate future trends. The KNN time series formulation is given as the sum of the product of the weight of the i-th neighbour and the i-th neighbour (Sinta et al., 2014).

$$y_l = \sum_{i=1}^{k} w_i \, y_i \qquad 4.21$$

Where w_i is the weight of the i-th neighbour. The weights can be custom and be based on the distance or could be equal for all the k neighbours.

The baseline k closest neighbour model was created using the Tslearn Python module. The default value for the most crucial base parameter, the number of nearest neighbours to be

considered when making a regression decision, is five. Using an array of values, this was further adjusted for all the components that were predicted.

4.8 Long short-term memory model

The Long short-term memory (LSTM) models are a special type of recurrent neural network model that learns the correlation of consecutive data points in a time series data. These models are intended to manage sequential data with long-term dependencies and their design solves the vanishing gradient problem and is ideally suited for simulating complicated time series patterns as in our case. LSTMs assume that previous sales of components include sequential patterns and dependencies that may be used to accurately anticipate future demand.

This model enables a persistent flow of errors through self-connected units to prevent gradient decay. An LSTM cell unit, proposed first in 1997, is structured with a memory cell and three primary gates, forget, input and output gates. The forget gate controls whether previous information is retained or discarded, the input gate controls how much added information is absorbed into memory, and the output gate controls whether the current cell value contributes to the final output (Hochreiter & Schmidhuber, 1997). The formulation of the various gates inside the model is as follows,

$$f_t = \sigma(W_{fh} \cdot h_{t-1} + W_{fx} \cdot x_t + b_f) \quad 4.22$$

$$i_t = \sigma(W_{ih} \cdot h_{t-1} + W_{ix} \cdot x_t + b_i) \quad 4.23$$

$$o_t = \sigma(W_{oh} \cdot h_{t-1} + W_{ox} \cdot x_t + b_o) \quad 4.24$$

$$c_t = f_t \times c_{t-1} + i_t \times tanh(W_o \times [h_{t-1}, x_t] + b_o) \quad 4.25$$

$$h_t = o_t \times tanh(c_t) \quad 4.26$$

Where, f_t is the forget gate, i_t is the input gate, o_t is the output gate, c_t holds the information from the start-up to the present time and h_t regulates the extent to which data information, preserved from the beginning to the current moment, can be transmitted to the subsequent moment. x_t is the Vector representing the input at time step "t", and they are multiplied with the weight matrix W. The b represents the bias that is added to the gate and is multiplied with an activation function σ. The LSTM cell architecture is presented in Figure 4.3.

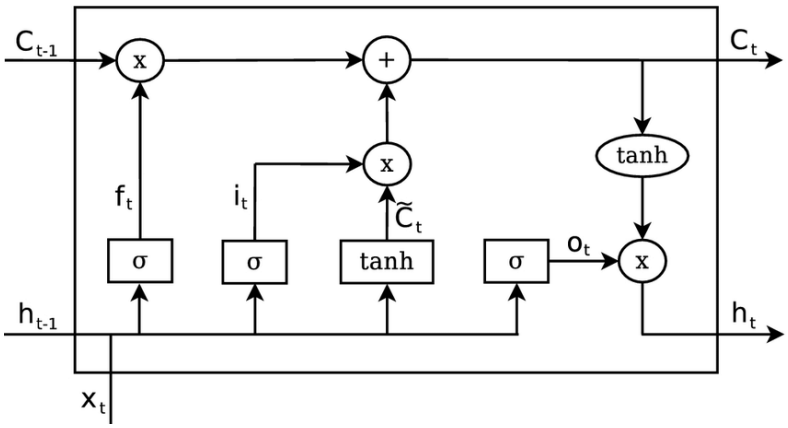

Figure 4.3: Long Short-Term Memory cell architecture (Fernandes et al., 2020)

The following Figure 4.4 describes the LSTM model architecture in accordance with the methods outlined in the literature previously cited.

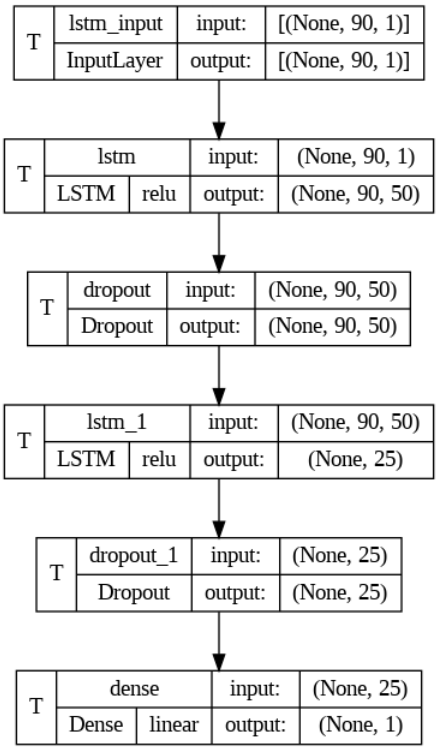

Figure 4.4: Structure of the LSTM model used.

The base model begins with a 50-unit LSTM layer that uses the "relu" activation function. Since our data's step-back value is ninety, this LSTM layer is designed specifically for handling sequential data with an input shape of (None, 90, 1). After the LSTM layer, a Dropout layer with a dropout rate of 0.2 is added to lessen the likelihood of overfitting. With twenty-five units, the second LSTM layer employs the "relu" activation function. The last element of the model is a Dense (fully connected) layer with a single unit that uses the "linear" activation function. This layer is the output layer for regression tasks, allowing the model to continuously generate numerical forecasts.

The LSTM units, a crucial component of every LSTM model, were the focus of hyperparameter tweaking. A wide range of unit values, including 32, 64, 128, and 256, were considered during our extensive analysis. We were able to thoroughly adjust the model for each time series because of our exhaustive research.

4.9 Gated recurrent unit model

Introduced in 2014, the gated recurrent unit is a significant development of the simple recurrent neural network model and that is like the LSTM model in that it combines the input and the forget gate into a single entity (Cho et al., 2014). GRU is capable of grasping correlations across time intervals and successfully using the intrinsic properties of time series data. The formulation of the components of a GRU cell are as follows,

$$z_t = \sigma(W_{zh} \cdot h_{t-1} + W_{zx} \cdot x_t + b_z) \quad 4.27$$

$$r_t = \sigma(W_{rh} \cdot h_{t-1} + W_{rx} \cdot x_t + b_r) \quad 4.28$$

$$h'_t = tanh(W_{xh}x_t + r_t W_{hh} h_{t-1}) \quad 4.29$$

$$h_t = z_t h_{t-1} + (1 - z_t)h'_t \quad 4.30$$

Here the z_t represents the update gate, r_t is the reset gate. The update gate controls how much earlier memory information is retained and brought through to the present time while the reset gate determines how much prior knowledge should be purged or forgotten. h'_t, which is the current memory content, is obtained by multiplying the weight matrix with the input info and the information from the previous step and finally passed through an activation function. In the last step, the update gate information is multiplied by the previous step information and added with the data preserved from the present memory state to the ultimate memory state and given as output by the final gated loop unit. The GRU cell architecture is presented in Figure 4.5.

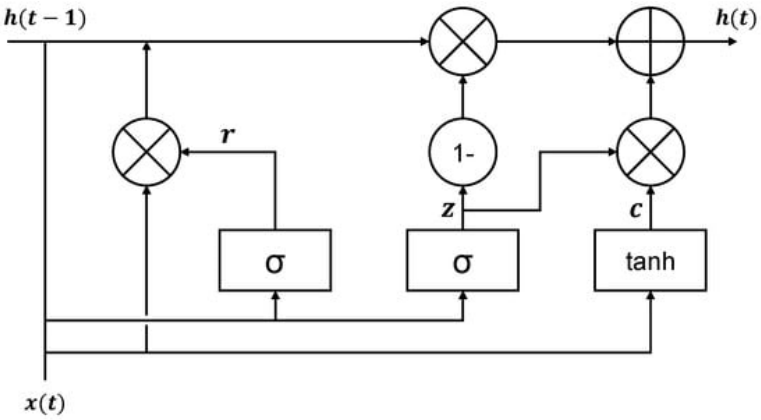

Figure 4.5: Gated Recurrent Unit cell architecture (Jung et al., 2021)

The following Figure 4.6 describes the GRU model architecture in accordance with the methods outlined in the works of literature previously cited.

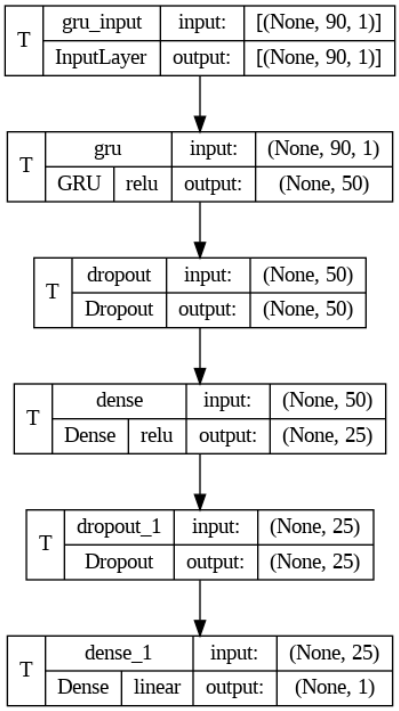

Figure 4.6: Structure of the GRU model used.

The architecture is a sequential neural network architecture with a GRU layer of fifty units, two dropout layers with a dropout rate of 0.2 to prevent overfitting, and two dense (completely connected) layers with one unit for regression tasks and twenty-five units each in the other. Just like the LSTM model, the ‘relu’ activation function was used for the GRU dense layer and ‘linear’ for the final layer which outputs our predictions.

For the models using GRU units, additional hyperparameter tweaking was performed using a variety of values, including 32, 64, 128, and 256. The GRU-based models were optimized via a

grid search tuning procedure, which allowed us to investigate a wide range of combinations and pinpoint the ideal architecture for different applications.

4.10 Convolutional neural network model

The Convolutional neural networks are a subset of deep learning models designed primarily for efficient analysis of grid-like data, including tasks like image analysis and spatial data processing. CNN differs from a traditional neural network in that it employs the notion of weight sharing. CNN's success in image processing jobs has spurred their use in time series (Wang et al., 2019). Convolutions applied over time intervals which have a one-dimensional grid topology, enable CNNs to efficiently detect time-based patterns within sequential data, making them suitable for predictive forecasting.

Four steps form the foundation of the basic CNN architecture, they are: Convolution, Bias Addition, Non-Linear Activation Function, and Pooling. To succinctly describe, the model starts by multiplying a matrix (filters) with the input values that were supplied in the form of a vector where each filter recognizes distinctive features inside the input data. The second step, which adds a bias vector after the matrix multiplication, enables the model to consider variances that the convolution step by itself does not capture. The result of this step is fed through an activation function that detects intricate patterns and associations within the data. The final phase, pooling, improves computing efficiency, prioritizes crucial characteristics, and reduces the spatial dimensions of the data while keeping crucial information. Figure 4.7 shows a basic Convolutional Neural Network model that gives an illustration of the layers that were previously addressed.

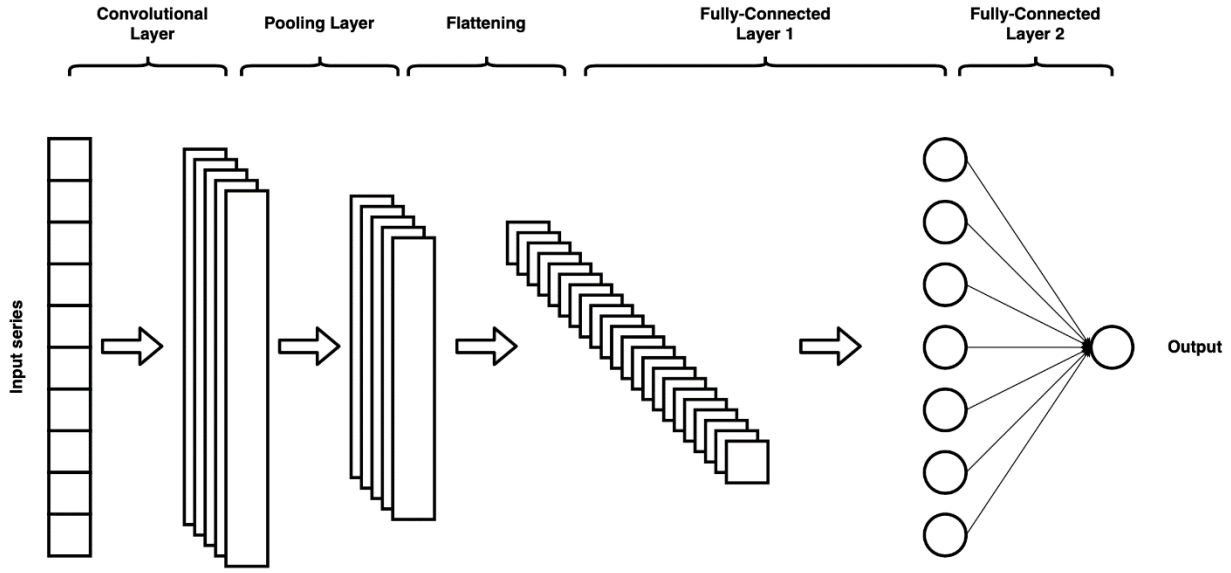

Figure 4.7: Basic structure of convolutional neural network (Huang et al., 2015)

The CNN model architecture is described below in accordance with the techniques mentioned in the previously referred literature.

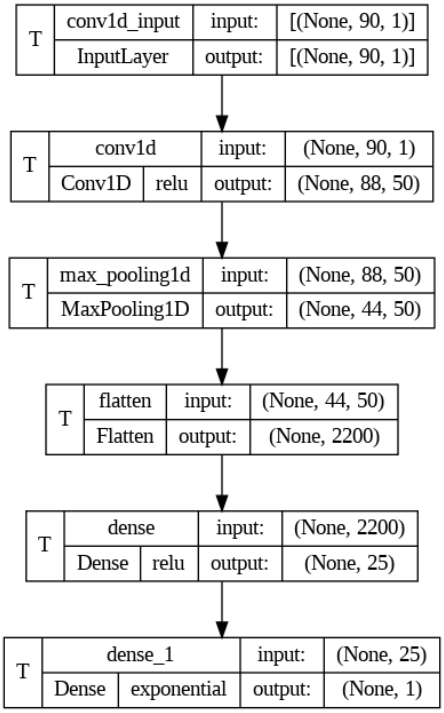

Figure 4.8: Structure of the CNN model used.

The baseline model consists of an input layer, a 1D Convolutional layer with fifty filters, a kernel size of three, and activated by the rectified linear unit (ReLU). The next layer is a MaxPooling1D layer with a pool size of two, which helps in reducing the spatial dimensions of the model, with the next layer being a flatten layer to transform the 1D feature maps into a 1D vector. A dense layer of twenty-five units, activated by "relu," adds a level of nonlinearity. The "exponential" activation function is used by the output layer which is the final layer giving out the prediction. Convolutional and dense layers are mixed in the architecture to accommodate various data patterns.

Grid search was employed to fine-tune hyperparameters, particularly focusing on the "filters" parameter, representing the number of convolutional filters applied within the 1D convolutional layer. The grid search explored a range of filter values, including 16, 32, 64, and 128, to identify the most effective setting for individual time series. These filters act like small inspection windows that traverse the input data to detect patterns or characteristics. The number of filters determines the variety of features the model can extract at this convolutional layer, directly influencing its ability to capture distinct data characteristics.

4.11 CNN-LSTM hybrid model

The combination of multiple deep learning models in hybrid time series models has sparked considerable interest in revealing patterns across both temporal and spatial dimensions. Because time series data inherently exhibit high temporal correlations, LSTM, which specializes in learning sequence dependencies, is an appropriate candidate. CNN, on the other hand, excels in feature extraction and capturing spatial interaction patterns, making it a powerful tool for investigation. There have been meaningful results shown by this ensemble when used with regards to predicting

the short-term power load making it more energy efficient (Wan et al., 2023) and in predictive maintenance planning (Dehghan Shoorkand et al., 2024).

The model architecture used is as follows,

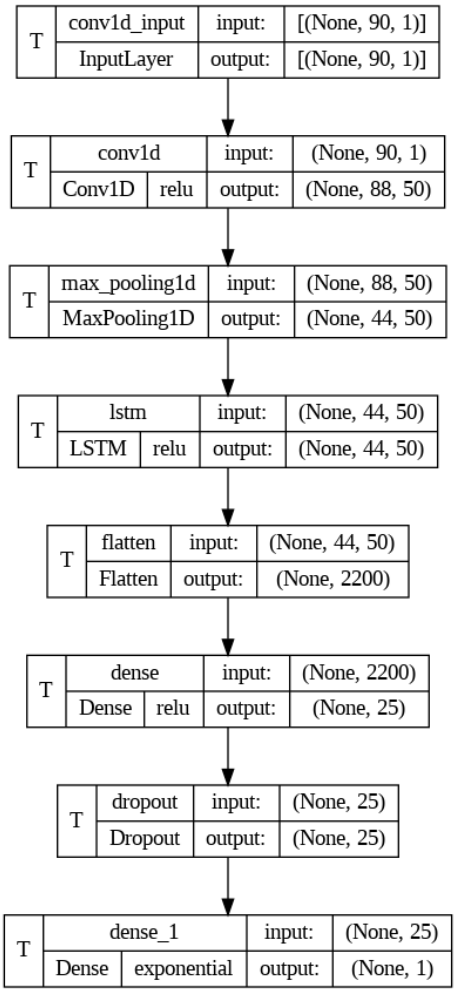

Figure 4.9: Structure of the CNN-LSTM hybrid model used.

The adaptive input layer of the design is the first layer to deal with variable-length sequences. After that, a 1D convolutional layer with filters and a kernel size activated by the rectified linear unit (ReLU) collects the key characteristics. Reduced spatial dimensions are achieved by using a max pooling layer with a pool size of two. The next layer is an LSTM layer, whose number of units are configured is set up to produce output sequences for every input sequence. The output is changed into a 1D vector via a flattening layer. A 20 percent dropout layer then prevents overfitting while a dense layer with twenty-five units and ReLU activation further refines feature representation. The output is obtained using the final dense layer. This architecture combines the benefits of CNN and LSTM to provide results that are superior to those of either model used alone.

Just like the previous model which included CNN architecture, this model also tuned the filters in the CNN part of the model using the Grid search using the same array of filters to find

the optimal filter value. The capacity of the model to extract useful information from the time series data depends heavily on the number of filters used, which are the most crucial component of the model.

4.12 CNN-GRU hybrid model

The CNN-GRU is a hybrid model that integrates various deep learning techniques to harness the distinct strengths of each technique. This model takes advantage of both the GRU (Gated Recurrent Unit) and CNN capabilities (Convolutional Neural Network) as it uses CNN's powerful feature extraction skills to find the underlying links inside the data, the max pooling layer reducing the output's dimensionality aiding in mitigating overfitting, investigate how data points are linked and showing the data's internal dynamics. Simultaneously, the GRU component captures the intricate nonlinear interactions between input and output and finds any extended connections that may exist inside the CNN layer-encoded features resulting in a deep knowledge of these relationships.

Following are the details of the model architecture used to train this model,

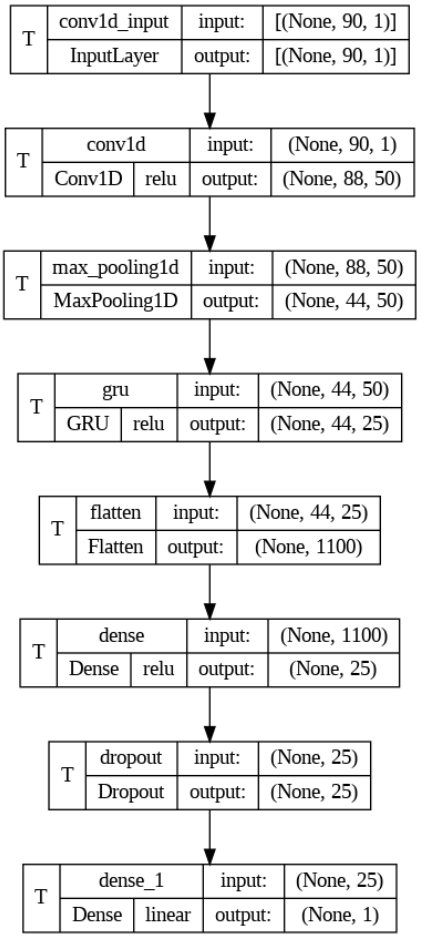

Figure 4.10: Structure of the CNN-GRU hybrid model used.

This model also starts with an input layer that is tuned to the structure of the input sequences and is followed by a 1D convolutional layer, just as the CNN-LSTM model previously

described. The ReLU activation function is used in this layer to draw out pertinent characteristics from the input data. A max pooling layer with a pool size of two follows the convolutional layer. While maintaining the most crucial data points, max pooling reduces spatial dimensions. The GRU's recurrent layer, which produces an output sequence for each input sequence, is the next layer to appear. Then comes the flatten layer, which turns the 2D output into a 1D vector so it may be processed further. A dropout layer comes after the dense layer, which has twenty-five units and ReLU activation, and further refines the feature representation. The output layer, which outputs our forecast, is the last.

The filters on the CNN portion of the model were modified for higher-performing models, just as in the prior models. As filter parameters, the following array [32, 64, 128, 256] was utilized. Particularly in the context of filters or units in layers, the choice of values for hyperparameter tuning for neural networks is frequently influenced by a combination of practical considerations and empirical testing.

4.13 Recurrent convolutional neural network model

The Recurrent convolutional neural network is again a hybrid neural network design that combines the benefits of Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) to manage and analyze sequential input. The model constitutes the convolution layer and pooling layer from the CNN architecture, with a recurrent layer and dense layers from the RNN architecture. Recurrent neural networks (RNNs) are a type of neural network in which the results of one phase are used as inputs for the next and can be considered a simpler version of the LSTM and GRU models discussed before. In contrast to standard feedforward networks, recurrent neural networks employ shared weight parameters across nodes within each layer of the network. The SimpleRNN layer is skilled in capturing far-off associations within the sequential data, whereas the Conv1D layers are capable of autonomously identifying significant characteristics in the input data. This makes it useful for applications like natural language processing and time series prediction.

A thorough visual representation of the structural components used in the model's training is given in Figure 4.11.

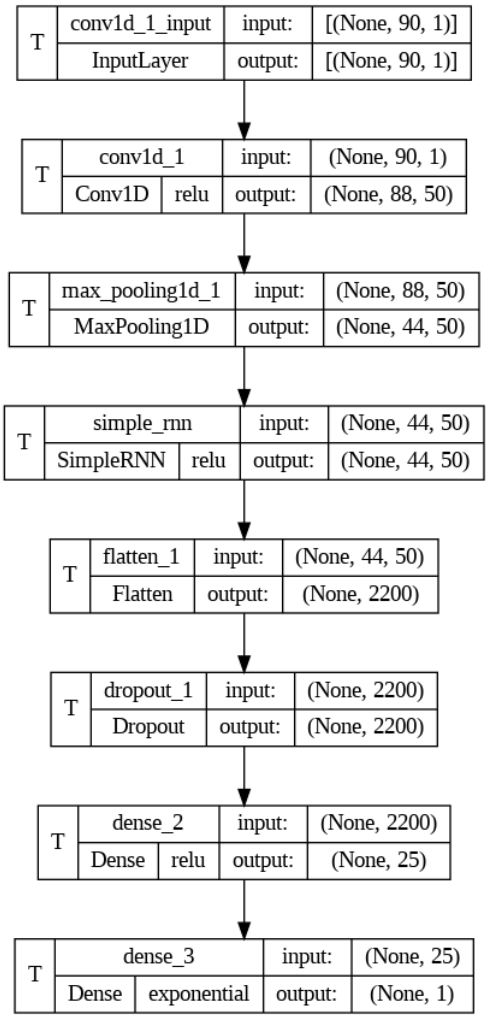

Figure 4.11: Structure of the RCNN model used.

The model starts with an input layer defined by the shape of the input sequences. The first layer is a 1D convolutional layer with fifty filters and a kernel size of three. It applies the “Rectified Linear Unit” activation function following which is the max pooling layer with a pool size of two. Max pooling helps reduce spatial dimensions and focuses on the most valuable information. The model then incorporates a recurrent layer using a SimpleRNN with fifty units and ReLU activation which is followed by a flatten layer that transforms the 2D output into a 1D vector. To prevent overfitting, a dropout layer with a rate of 0.2 is applied. Dropout randomly deactivates a portion of the neurons during training, improving the model's generalization. It is followed by two dense layers; The first layer has twenty-five units and ReLU activation and the second layer serves as the output layer.

Grid search was used to adjust the model's hyperparameters, and the filters were adjusted to produce better-performing models. Each filter oversees spotting textures, edges, or patterns in the data. The model can be adjusted to the complexity of the dataset by adjusting this parameter. Additionally, adjusting for filters aids in finding a balance between model generalization and complexity.

4.14 Gated CNN model

The gated CNN model is a novel customized neural network architecture that incorporates the convolution and pooling layers from the convolution neural network with a gating mechanism which are a staple in the recurrent neural network models such as Long Short-Term Memory (LSTM) network and its variations. These information flow-controlling gating mechanisms provide the model the freedom to update itself selectively and decide what to keep and what to throw away as time goes on. By combining convolutional and gating features, this architecture was designed to recognize certain qualities or patterns in one-dimensional data. The gate introduced, enhances feature extraction, and improves sequential modelling. One major advantage of this model is that when compared to traditional sequential RNNs, gated CNNs are faster in training and inference because they can manage data concurrently.

The specifics of the model architecture that was utilized to train this model are listed below.

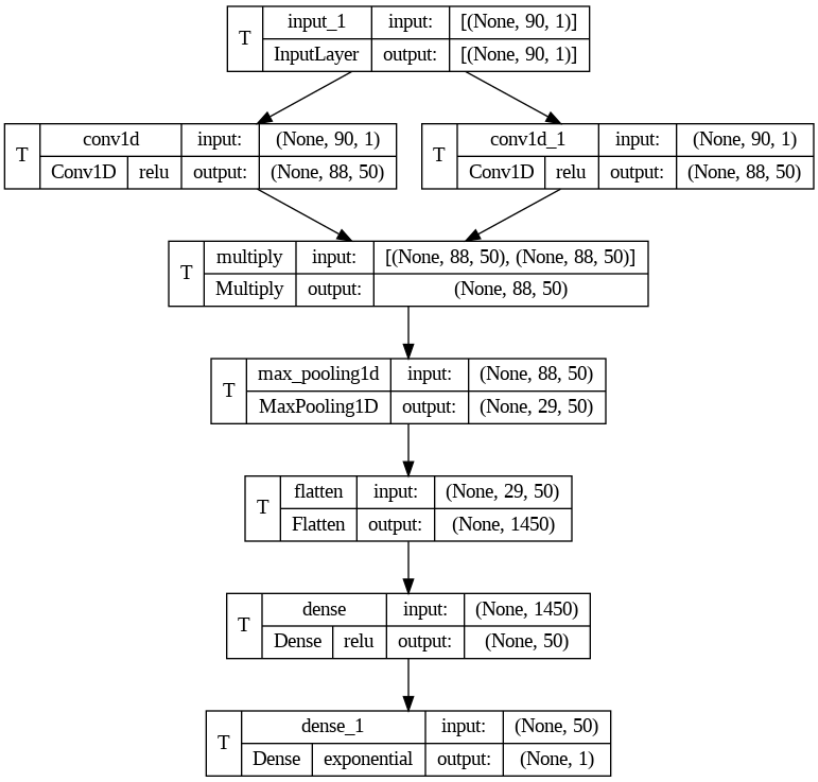

Figure 4.12: Structure of the Gated CNN model used.

The model starts with an input layer which inputs a shape of (90,1) like all other models. The next layers are the two convolution layers which is 1D convolutional layer with both being used for the gating mechanism. These two layers are outputted to multiply layer which performs elementwise multiplication. As a result, a "gated" convolutional layer is created by combining the data from the two convolutional layers with gating. Depending on the significance of a feature, this gating mechanism can selectively emphasize or suppress it. Following the gated convolutional layer, three-pool max pooling is used. By reducing the spatial dimensions, max pooling concentrates on the most crucial information. The next layer, the flattened layer transforms the 2D

output from the previous layer into a 1D vector. Finally, a fully connected layer with fifty units and ReLU activation processes the flattened data, and the data is outputted using an output layer.

Again, the filters in the two 1D convolutional layers at the start of the model are tuned using the grid search Python module. The network may learn a wider range of features when a variety of filters are used, but this also makes the model more complex. The risk of overfitting must be balanced against the benefits of collecting additional data.

4.15 Inception CNN model

Inception networks represent a significant step forward in CNN's evolution which challenges the conventional wisdom that just adding additional convolution layers is the best approach to improve CNN's performance (Zhou et al., 2023). Inception entails expanding the network's coverage as well as its depth, and this larger setup makes it possible to capture minute information more efficiently. Along with maintaining the network topology, this approach makes use of a dense matrix's potent processing capacity. The inception modules were created with a specific goal in mind, considering issues like computational complexity and the danger of overfitting, among other things. The basic structure of the inception module is given in Figure 4.13. This technique, which was first developed for picture classification and object recognition, can be used in time series forecasting to effectively capture complex temporal trends.

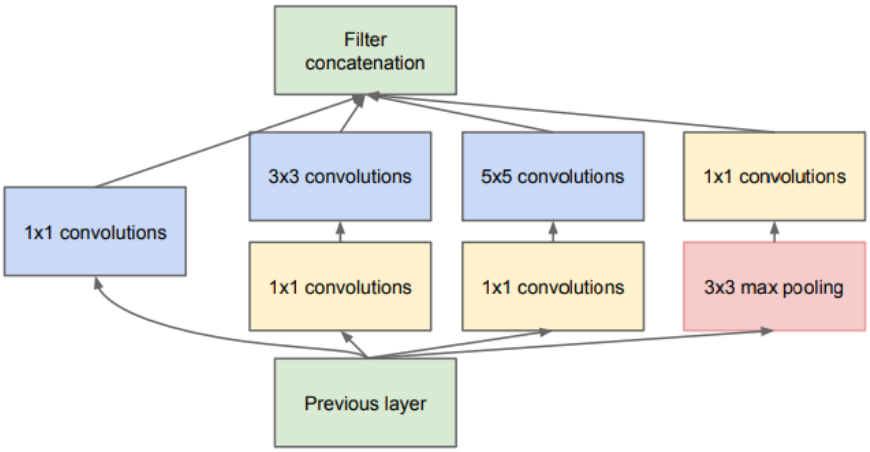

Figure 4.13: Inception module with dimension reductions (Szegedy et al., 2014)

The following is a description of the Inception CNN model's architecture used for training:

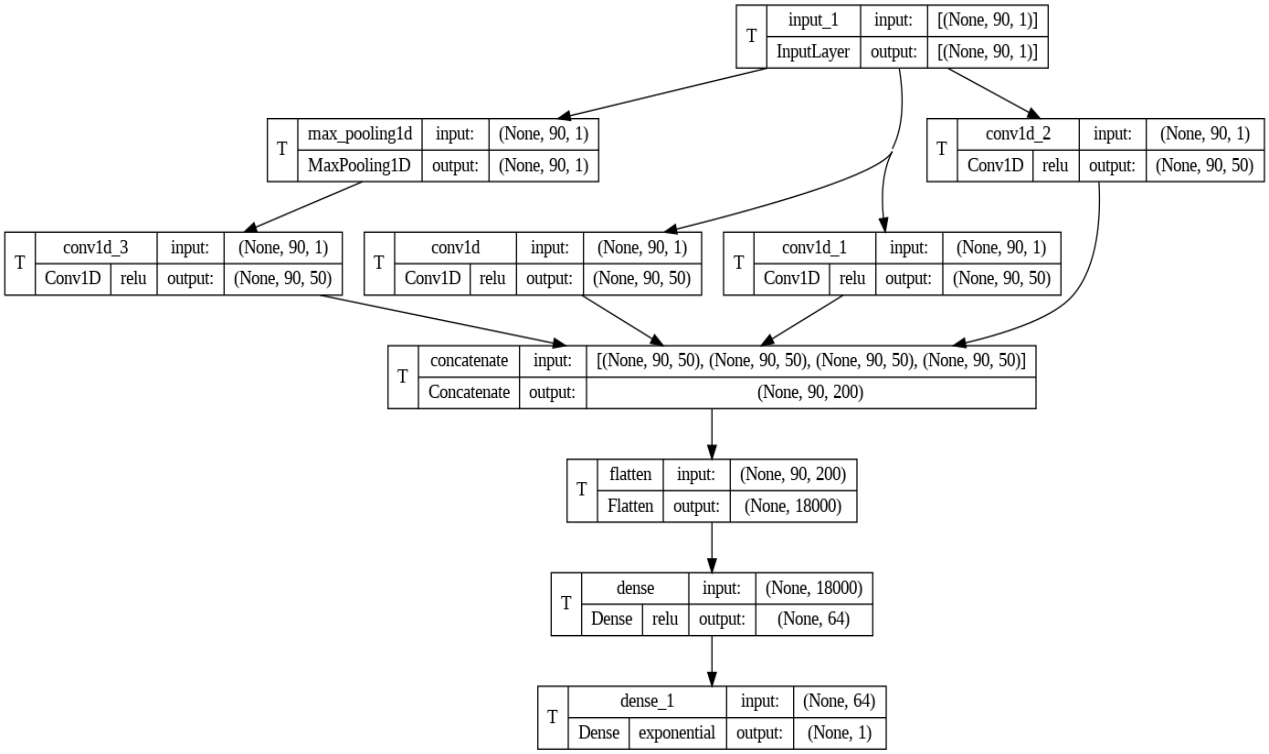

Figure 4.14: Structure of the Inception CNN model used.

The model has a branching architecture like that of inception and is built to process input sequences with 90-time steps and one feature. Convolutional layers make up the four branches; they each have a different kernel size and ReLU activation function, resulting in a different feature map. The outputs of these branches are combined to create a single feature map of form (90, 200). The model then uses a flattening layer to convert the 2D feature map into a 1D vector with 1,800 elements. The next layer has sixty-four units with ReLU activation and is fully connected. The output layer is made up of a single unit with an exponential activation function and there are 115,029 trainable parameters in the model.

Just like the other models, the final model was tuned with a grid search to tune for the best filters in the four convolutional layers in the inception model. A feature hierarchy is established by filters in Inception networks. The network may capture characteristics at multiple scales, from trivial details to bigger patterns, by employing filters of varied sizes. Also, the network may learn to extract a variety of features while avoiding the computational complexity associated with utilizing very deep networks by employing numerous filter sizes in parallel.

4.16 Metrics used

Three primary metrics are used to measure the performance of the models. They are root mean squared error (RMSE), cumulative forecast error (CFE) and Stock-keeping-oriented Prediction Error Costs (SPEC). The Percentage Better measure was used as the last criteria for comparing models, allowing for a thorough evaluation based on the gathered error metrics.

4.16.1 Root mean squared error

RMSE computes the square root of the average of the squared errors, providing a measure of the forecasting model's overall accuracy. The RMSE is given by the following,

$$\text{RMSE} = \sqrt{\frac{\sum_{j=1}^{n} (y_{p,j} - y_{a,j})^2}{n}} \tag{4.31}$$

Where, $y_{p,j}$ is the predicted value, $y_{a,j}$ is the actual value and n is the data points.

4.16.2 Cumulative forecast error metric

The next error metric used is the cumulative forecast error metric (CFE). The CFE is given by the following,

$$\text{CFE} = \sum_{i=1}^{n} (Forecast - Demand) \tag{4.32}$$

Where the forecast error is summed over all the data points on the given time series. It is calculated straightforwardly by adding up the forecast errors with both positive and negative faults canceling each other out. This involves the process of summing the differences between predicted values and actual values to obtain the CFE score. This metric is frequently used to assess forecasting bias, to minimize its value, ideally approaching zero. A CFE score of zero signifies an ideal forecast in terms of predicted units, whereas a negative CFE value indicating an overestimation and a positive CFE value indicating an underestimation from the actual value.

4.16.3 Stock-keeping oriented prediction error cost

The next metric in use is the stock-keeping oriented prediction error cost (SPEC) which was specifically created to evaluate models when working with intermittent data. The SPEC metric is given as follows,

$$SPEC_{\alpha_1,\alpha_2} = \frac{1}{n} \sum_{t=1}^{n} \sum_{i=1}^{t} \left(max\left[\begin{matrix} 0; \\ min[y_i; \sum_{k=1}^{i} y_k - \sum_{j=1}^{t} f_j\ ] \cdot \alpha_1; \\ min[f_i; \sum_{k=1}^{i} f_k - \sum_{j=1}^{t} y_j\ ] \cdot \alpha_2 \end{matrix} \right] \cdot (t - i + 1) \right) \tag{4.33}$$

Where, n is the length of the time series y_t is the actual value of the time series at time t, f_t is the forecasted value at time t. α_1 represents the opportunity costs and the α_2 represents the inventory holding costs. These values range from one to infinity, with a total of one typically preferred to assure consistency. The metric works this way, after each prediction a cost is associated with that prediction, an opportunity cost, or an inventory holding cost and it is summed over the prediction time steps. Underpredicting future forecasts can result in higher opportunity costs while

overpredicting can lead to increased inventory expenses. This metric prompts the model to discover the ideal point at which both costs are kept to a minimum (Martin et al., 2020). For this research, more weightage was given to inventory holding cost with α_2 value of 0.9 and an opportunity cost α_1 as 0.1 to penalize longer inventory holding times. These values represent assumed parameters for research purposes, as actual cost data could not be obtained due to confidentiality constraints.

4.16.4 Percentage better

The final metric that was used to compare the baseline model with the other models was the percentage better metric. The formula is given bellow,

$$PB = \left(\frac{|a-b|}{(a+b)\div 2}\right) \times 100 \qquad 4.34$$

The error metric values of the model are represented by the parameters designated as "a" and "b". The output that results is a percentage that shows how well or poorly the model performed in comparison to the other model and quantifies the difference in performance. This is a useful metric that provides information about how well the model performs relative to the given benchmark model. Croston model was chosen as the baseline mode to compare with all the other models in the study. This option was chosen because of its historical relevance in addressing intermittent demand patterns and its extensive usage as a benchmark in comparable forecasting research (Pinçe et al., 2021; Willemain"' et al., 1994). Furthermore, the Croston model's simplicity makes it an excellent reference point for evaluating the performance of more complicated and advanced forecasting methodologies used in this research.

4.17 Data preparation

The received raw data must be processed and reshaped into the proper dimensions for the models to properly capture the patterns before being input into the models to be trained. The scaling and step-back procedures used in data preparation are described in depth in the sections that follow this.

4.17.1 Scaling

Before training the different machine learning models, the dataset is standardized using the "TimeSeriesScalerMinMax" function, which guarantees that the rescaled dataset remains within the default range of 0 to 1. This stage is crucial to our research approach since it offers two major advantages which are accelerating the training process, where many optimization algorithms, such as gradient descent, converge faster when features are within a similar scale, and it also helps avoid outliers from influencing how our model learns. This scaling procedure not only improves the performance of our models but also guarantees that regularisation is implemented uniformly to all features to prevent overfitting by penalizing large coefficients because we use models like KNN and SVR, which are sensitive to how data is scaled.

4.17.2 Step back

Time series data is often broken down into smaller units for easier comprehension and prediction when analyzed and modelled. Using the step-back value, the time series sequence is

translated into input-output pairs or feature vectors with associated goal values. A time series sequence's step-back value is the number of time steps backward that are utilized to create these input-output pairs. It is an important parameter that controls the amount of previous data that is considered when constructing these pairs. The input for each time step corresponds to the data seen in the preceding time steps, and the output represents the data at the current time step. This strategic approach guarantees that these combinations are methodically generated while conforming to the data limits. Furthermore, this approach is an important stage in the data pre-treatment pipeline, particularly for training machine learning models such as the Support Vector Regression model. The properties of the time series data and the patterns that one wants to capture influence the step-back value selection and a step-back value of 3 months was devised for all the models.

4.18 Computational Environment and Toolset Utilized

The major tool for forecasting all the models, together with data cleaning and preparation, was the Python programming language. Python version 3.10.12 was used in Google Colab for the entirety of the thesis. It's crucial to remember that neither a dedicated Graphics Processing Unit nor a dedicated Tensor Processing Unit were used in any of the research's experiments. The hosting environment's standard Central Processing Unit resources were used to carry out the computations. With two cores for concurrent computation and 64-bit addressing support, the CPU in use is an x86_64 architecture. Pandas (Mckinney, 2010) and NumPy (Harris et al., 2020), two adaptable libraries, managed complex data manipulation and processing. The fundamental statistical and machine learning models were created using the Scikit-Learn (Pedregosa Fabianpedregosa et al., 2011), StatsModels (Seabold & Perktold, 2010), and tslearn (Tavenard et al., 2020) libraries. By incorporating TensorFlow (Abadi et al., 2016) and Keras (Chollet F et al., 2015) into our research, we were able to examine deep learning methods like forecasting neural networks. To evaluate how well our models were performing, we used a variety of libraries, including spec metric (Martin et al., 2020.) and permetrics (Van Thieu & Mirjalili, 2023). These tools helped us get a complete picture of how our models were performing. The comprehensive list of the libraires used is given in table 4.1.

Table 4.1: Python Libraries Utilized in the Research

Library	**Functions**
pandas	Manipulating data using DataFrame.
NumPy	Numerical operations on multi-dimensional arrays
time	Python module for measuring execution time.
matplotlib	A 2D plotting library for visual aid creation
spec_metric	Library for automated SPEC calculations
permetrics	Library for RMSE metric calculations
TimeSeriesScalerMinMax	Time series scaling for Min-Max normalization

SimpleExpSmoothing	Exponential smoothing for time series forecasting
RandomForestRegressor	Regression model using random forest algorithms
SVR	Support Vector Regression for regression tasks
KNeighborsTimeSeriesRegressor	Time series regression with k-nearest neighbors
GridSearchCV	Optimizing hyperparameters in machine learning models
Keras	Deep learning model construction and training using a high-level API
tensorflow	Open-source deep learning library for building ML models.

4.19 Hyperparameter tuning

Every model and every time series in our research underwent hyperparameter tuning, which was a crucial step. To get the best results from statistical models, this required fine-tuning the alpha and beta parameters. A grid search methodology was used to carefully adjust the hyperparameters of more complex machine learning and deep learning models. This meticulous approach was conducted to make sure that each model performed to its fullest ability, improving its resilience and forecast accuracy.

The ADAM Optimizer was employed (Kingma & Ba, 2014), a stochastic objective function improvement technique that makes effective use of adaptive estimations of lower-order moments. This helps it work well with problems that have missing or noisy data.

The optimization formulas are as follows,

$$g = (h_\theta(x\hat{\imath} - y\hat{\imath}))x^i \quad 4.34$$

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) * g \quad 4.35$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) * g^2 \quad 4.36$$

$$m_t^- = \frac{m_t}{1-\beta_1^t} \quad 4.37$$

$$v_t^- = \frac{v_t}{1-\beta_2^t} \quad 4.38$$

$$\theta_j = \theta_{j-1} - m_t^- * \frac{\propto}{\sqrt{v_t^-}+\in} \quad 4.39$$

Where,

" g " represents the gradient that has been computed.

"m_t" represents the initial moment of the gradient "g"

"v_t" represents the secondary moment of the gradient "g "

"β_1" refers to the coefficient for attenuating the first order moment.

"β_2" refers to the coefficient for attenuating the second-order moment.

"θ" represents the parameter that requires a solution.

m_t^- and v_t^- explains the process of offset correction for "m_t" and "v_t"

We established the initial values for the parameter vector, the first and second-moment vectors, and the time step in this optimizer. Up until the parameter θ converges, the loop keeps adjusting the various parts iteratively. Adam was utilized to enhance the performance of all the neural network models used, whose primary aim is to identify a set of parameters that will minimize the associated error function.

CHAPTER 5: Empirical results

In this section, we assess the performance of our models across the machine learning algorithms, ranging from classical statistical models to more sophisticated deep learning algorithms. We commence by introducing the optimized models and demonstrating the improvements achieved through hyperparameter tuning. Subsequently, we evaluate and rank the models based on their performance metrics to provide a comprehensive assessment of their effectiveness.

5.1 RMSE metric analysis

Initially, all models were executed with default parameters derived from Python libraries, and their respective error metrics were computed. Subsequently, an extensive phase of hyperparameter tuning was conducted for both machine learning and deep learning models, employing grid search techniques. Notably, the statistical models underwent a manual tuning process, focusing on optimizing the various smoothing parameters. Following this tuning process, error metrics were recalculated for the tuned models, utilizing the most effective parameter configurations. As seen in table 5.1, which shows the baseline and tuned metrics, this approach was conducted to increase the models' predictive accuracy and guarantee their optimal performance.

In our analysis, we observed significant improvements in the performance accuracy of all the forecasting methods after optimization. These enhancements are evident when we compare the Root Mean Square Error values before and after optimization. As an example, consider the simple exponential smoothing method, which, following hyperparameter optimization, demonstrated an impressive 35.766 percent decrease in its RMSE value. It should be noted that these reductions are in relation to the pre-optimization RMSE values. Certain approaches, such as Long Short-Term Memory and Gated Recurrent Unit, on the other hand, exhibited only slight improvements in RMSE. This implies that their initial configurations were already near optimal. Our analysis's RMSE results are rounded to the third decimal place, giving a clear indication of the increases in forecasting accuracy brought about by optimization.

Table 5.1: RMSE values before and after optimization

Models	RMSE Baseline	RMSE Optimized
Croston	0.623	0.595
Croston SBA	0.602	0.591
Croston TSB	0.778	0.629
SES	0.804	0.592
RF	0.677	0.603
SVR	0.597	0.592
KNN	0.691	0.664
Inception CNN	0.625	0.606
Gated CNN	0.591	0.587
RCNN	0.596	0.587
CNN GRU	0.590	0.589
CNN LSTM	0.593	0.591
CNN	0.627	0.590
GRU	0.589	0.588
LSTM	0.589	0.589

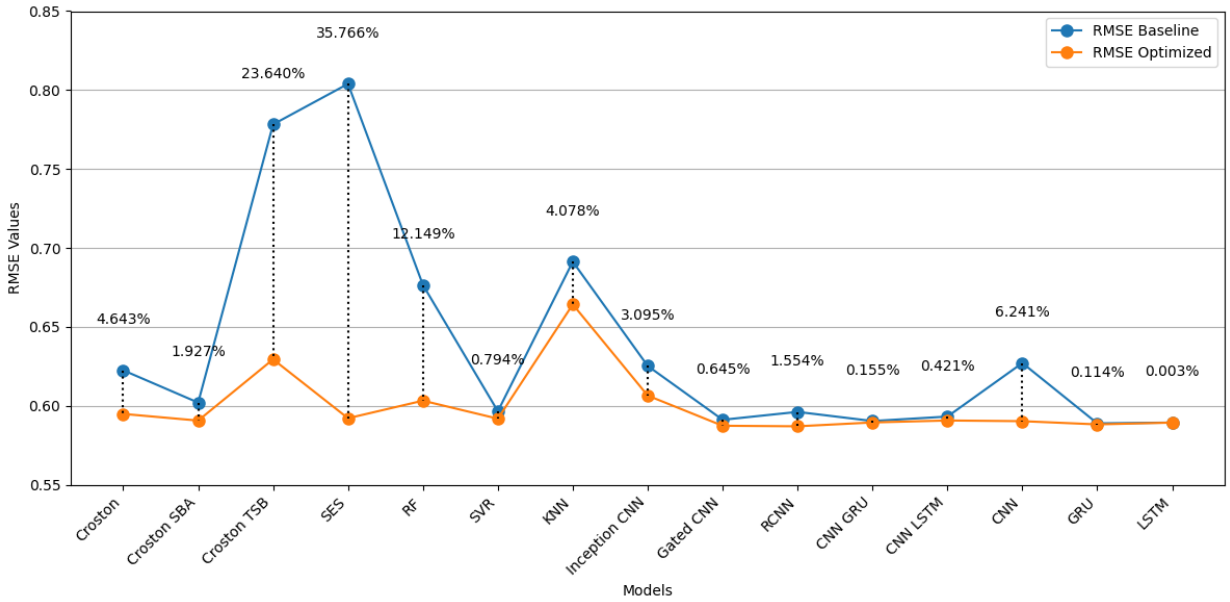

Figure 5.1: Performance improvement (Baseline vs Tuned)

After each model was fine-tuned, its performance was ranked; the results are shown in Table 5.2. With a rank of '1' designating the best-performing model and '15' the least effective, a

lower rank in this table corresponds to higher performance. Notably, with nearly coinciding performance values, the top-performing models are GRU, RCNN, and Gated CNN. Table 5.2 displays a notable variation in RMSE scores between 0.587 and 0.664, indicating notable deviations in model accuracy. Furthermore, it is evident that deep learning models are preferred over other researched approaches. Table 5.3 provides additional support for this trend, grouping models into three general categories: "Basic Statistical," "Deep Learning," and "Machine Learning (Non-Deep Learning). Deep learning models have an average RMSE value of 0.591, whereas non-deep learning models have an RMSE value of 0.620 and basic statistical models have an RMSE value of 0.602. It is to be noted that the mean RMSE values calculated included all the hybrid models used for the deep learning models such as Gated CNN and Inception CNN.

Table 5.2: Performance Evaluation with RMSE and Final Rank.

Model	RMSE	Rank
Croston	0.595	11.0
Croston SBA	0.591	7.0
Croston TSB	0.629	14.0
SES	0.592	10.0
RF	0.603	12.0
SVR	0.592	9.0
KNN	0.664	15.0
Inception CNN	0.606	13.0
Gated CNN	0.587	2.0
RCNN	0.587	1.0
CNN GRU	0.589	5.0
CNN LSTM	0.591	8.0
CNN	0.590	6.0
GRU	0.588	3.0
LSTM	0.589	4.0

Table 5.3: Comparison of RMSE values by model type.

Type	Mean RMSE
Advanced Deep Learning	0.591
Basic Machine Learning	0.620
Traditional Statistical	0.602

5.2 Percentage better (PB) metric analysis

In our analysis, we compared the performance of different forecasting models to the Croston baseline model using the 'percentage better' metric. The 'percentage better' metric is used to quantify the improvement in forecasting accuracy over the benchmark Croston model and this metric quantifies the extent to which the forecasting accuracy of the other models exceeds or falls short of the Croston model. In cases where the 'percentage better' is positive, it signifies that the model performs better than the Croston model; the percentage expresses the percentage difference and when the model under consideration appears to perform worse than the Croston model when the 'percentage better' is negative; the percentage indicates how much this degradation occurs. With a PB value of 1.336, the "RCNN" model stands out as the best model in this situation. It's worth noting that the models that excel in performance, including "RCNN," "Gated CNN," "GRU," and "LSTM," are all part of the deep learning category. Conversely, models like "KNN," "Croston TSB," and "Inception CNN" exhibit negative percentage better values, suggesting they underperform compared to the reference model, Croston. Notably, "KNN" lags behind Croston by a significant margin, with a 10.441% worse performance.

Table 5.4: Model Performance Relative to Croston Model (%)

Model	Percentage Better(%)
RCNN	1.336%
Gated CNN	1.277%
GRU	1.128%
LSTM	0.934%
CNN GRU	0.917%
CNN	0.779%
Croston SBA	0.714%
CNN LSTM	0.705%
SVR	0.511%
SES	0.462%
Croston	-
RF	-1.391%
Inception CNN	-1.915%
Croston TSB	-5.492%
KNN	-10.441%

5.3 Cumulative forecast error (CFE) analysis

The next metric that was obtained from the models was the cumulative forecast error which gives the overall forecast bias and accuracy for the cumulative period. Table 5.5 presents all the forecasting models, each associated with a CFE value, and the ranking is based on the absolute deviation of the CFE value from a perfect prediction error of zero. The CFE values shows a wide range where models like "Croston," "SES," and "KNN" have negative cumulative forecast error

values have overestimated the target variable. On the other hand, models like "RCNN," "LSTM," and "CNN LSTM" that have positive CFE values have undervalued the target variable. It's crucial to remember that forecasting models ideally strive for a balanced performance in both directions, thus interpreting overestimation or underestimation should be done with caution. It's important to note, nevertheless, that the CNN LSTM hybrid model performs the best overall, underestimating just by 0.183 units. This almost flawless prediction shows how well the model works to reduce predicting mistakes over the whole dataset.

Table 5.5: Performance Evaluation with CFE and Final Rank.

Model	CFE	Rank
Croston	13.326	14.0
Croston SBA	8.164	12.0
Croston TSB	7.404	11.0
SES	10.799	13.0
RF	6.165	10.0
SVR	6.056	9.0
KNN	16.412	15.0
Inception CNN	3.279	4.0
Gated CNN	2.798	3.0
RCNN	5.128	8.0
CNN GRU	3.299	5.0
CNN LSTM	0.183	1.0
CNN	3.444	6.0
GRU	3.463	7.0
LSTM	2.558	2.0

Upon observation, like the previous error metric, the advanced deep learning models once again demonstrate notably superior performance in relation to the other two categories, as evidenced in Table 5.6. Deep learning models tend to overestimate by around 0.992 units which is significantly more accurate than the overestimation of 9.544 and 9.924 by machine learning and statistical models. But it is to be noted that all the models on average tend to overestimate the forecast.

Table 5.6: Comparison of CFE values by model type

Type	Mean CFE
Advanced Deep Learning	-0.992
Basic Machine Learning	-9.544
Traditional Statistical	-9.924

5.4 SPEC metric analysis

The metric in deliberation is the spec metric, which takes into consideration the inventory cost and the opportunity cost associated with an estimate cumulatively. Table 5.7 lists the models and their corresponding SPEC values associated with their predictions where lower SPEC value indicates better performance and ranked based on it. As before, the numbers span a wide range, resulting in a notable difference in how well each model performs. The best performing model is the Gated CNN model with a SPEC value of just 100.477 and the poorest performing model is the TSB model with a value of 2075.442.

Table 5.7: Performance Evaluation with SPEC and Final Rank.

Model	SPEC	Rank
Croston	1202.041	14.0
Croston SBA	565.923	13.0
Croston TSB	2075.442	15.0
SES	213.711	11.0
RF	145.560	9.0
SVR	162.690	10.0
KNN	344.622	12.0
Inception CNN	144.147	8.0
Gated CNN	100.477	1.0
RCNN	114.176	4.0
CNN GRU	125.066	7.0
CNN LSTM	118.610	5.0
CNN	120.879	6.0
GRU	109.326	2.0
LSTM	109.509	3.0

The SPEC ranking obtained, aligns closely with the existing literature on the subject (Azizi & Wibowo, 2022; Kiefer et al., 2021). These findings verify our results' consistency and dependability, emphasizing the resilience of advanced deep learning models and being consistent with past studies in the field of spare parts demand forecasting. When computing the mean SPEC value for the several types of models in question, yet again the advanced deep learning models perform significantly better than the other two methods. With the best SPEC Optimized result of 117.774, the deep Learning category stands out because it demonstrates the strong effectiveness of deep learning models in drastically lowering prediction error costs in stockkeeping.

Table 5.8: Comparison of SPEC values by model type

Type	Mean SPEC
Advanced Deep Learning	117.774
Basic Machine Learning	217.624
Traditional Statistical	1014.280

5.5 Computational time

The Table 5.9 provides a breakdown of the processing time required for each model. Most remarkably, the support vector regression model performed computations in a mere 0.88 minutes, which is a little less than a minute for tuning over all time series. Reasonable computation durations were also demonstrated by the Croston and SBA models, which took 2.14 and 2.16 minutes, respectively. The convolutional neural network with gated recurrent units and the recurrent convolutional neural network demonstrated reasonable computing times in the context of deep learning models, with tuning times of 21.54 and 12.11 minutes, respectively. The gated recurrent unit and long short-term memory models, which are well-known for their usefulness in sequence modeling, recorded longer computing times of 46.61 and 58.31 minutes, respectively. The primary cause of this is that they handle data in a sequential manner while taking interdependence between time steps into account. It is more difficult to parallelize computations due to this sequential processing, which could result in more prolonged training times. This isn't the case for CNN models, which have a significant degree of parallelization, particularly in the early convolutional layers. However, the 8.59-minute TSB model was notable due to the need to adjust two distinct alpha values for that model among the statical models. These findings highlight the various computational needs of different forecasting models, highlighting the significance of taking accuracy and computational efficiency into account when choosing a model for time series forecasting applications.

Table 5.9 Computational Time for Various Forecasting Models

Model	Time (Minutes)
Croston	2.14
Croston SBA	2.16
Croston TSB	8.59
SES	2.18
RF	10.83
SVR	0.88
KNN	38.67
Inception CNN	8.05
Gated CNN	14.30
RCNN	12.11
CNN GRU	21.54
CNN LSTM	29.70
CNN	7.22
GRU	46.61
LSTM	58.31

5.6 Comprehensive performance evaluation

The Table 5.9 compiles all the previously discussed metrics, making it easy for you to examine the findings. The table provides a thorough performance evaluation of different forecasting models across the variables, and it can be noted that with a mean rank of two, the Gated CNN model regularly holds a spot in the top three when compared to the other models. With a mean rank of 3.000, the LSTM emerges as the next best-performing model, underscoring its consistent performance across all metrics. The worst performing model considering all three metrics is the KNN model indication its not very well suited for intermittent demand forecasting. The mean rank is presented as a scatter plot, and it shows a linear trend through the different models indicating that certain models do perform better for such sporadic data compared to others across different metrics. Additionally, all deep learning models outperform all other models and assume positions 1 through 8.

Table 5.10: Comprehensive Performance Evaluation and Rankings of Forecasting Models.

Model	RMSE	Rank RMSE	CFE	Rank CFE	SPEC	Rank SPEC
Croston	0.595	11.0	-13.326	14.0	1202.041	14.0
Croston SBA	0.591	7.0	-8.164	12.0	565.923	13.0
Croston TSB	0.629	14.0	-7.404	11.0	2075.442	15.0
SES	0.592	10.0	-10.799	13.0	213.711	11.0
RF	0.603	12.0	-6.165	10.0	145.560	9.0
SVR	0.592	9.0	-6.056	9.0	162.690	10.0
KNN	0.664	15.0	-16.412	15.0	344.622	12.0
Inception CNN	0.606	13.0	-3.279	4.0	144.147	8.0
Gated CNN	0.587	2.0	2.798	3.0	100.477	1.0
RCNN	0.587	1.0	5.128	8.0	114.176	4.0
CNN GRU	0.589	5.0	-3.299	5.0	125.066	7.0
CNN LSTM	0.591	8.0	0.183	1.0	118.610	5.0
CNN	0.590	6.0	-3.444	6.0	120.879	6.0
GRU	0.588	3.0	-3.463	7.0	109.326	2.0
LSTM	0.589	4.0	-2.558	2.0	109.509	3.0

5.7 Discussion

The gated CNN, LSTM, and GRU are the best-performing (top three) models in terms of all the metrics in question. And all the hybrids of CNN are performing better than the basic statistical and classic machine learning algorithms. The potential of CNNs to automatically identify crucial data properties is well known. In our example, the model is given a set of grids that reflect the data from the previous three months and are punctuated by intermittent patterns. The convolution layers of gated CNNs excel at extracting and interpreting these traits, providing a useful method for data comprehension. Additionally, they are well-suited to accurately capture time-dependent patterns in the data because of their capacity for hierarchical feature collection and use of non-linear activation functions. Additionally, the presence of the gating mechanism like in the case of the next best-performing LSTM and GRU, enables the model to regulate the information flow and retain prior data, a valuable feature for modeling intermittent patterns. The industry standard and often used smoothing methods produced mediocre results in all three criteria, highlighting their shortcomings. Overall, Figure 5.2 conclusively shows that deep learning models outperform statistical and machine learning models. It is noteworthy that the KNN model performs the worst, which can be linked to several issues, especially its sensitivity to data sparsity. A common symptom of sparsity in intermittent time series data is the presence of zeros or missing values at several time steps. Less precise predictions may result from KNN's inability to locate meaningful neighbors in such minimal data. One more probable reason could be KNN’s nature of ignoring the temporal component of time series data. In time series forecasting, when temporal connections are essential, it frequently interprets all data points as independent, which is not appropriate. In summary, while statistical learning methods are only partially effective at capturing

the variability in our time series data, deep learning models, particularly those based on sequence learning and gating, demonstrate impressive proficiency in simultaneously learning and parameterizing multiple time series in our dataset.

Figure 5.2: Model performance scatter plot sorted by mean rank.

CHAPTER 6: Conclusion

This chapter concludes our research by providing critical insights from our analysis, addressing the limitations that affected the study's depth and accuracy, and indicating potential directions for future research.

6.1 Concluding remarks

This study was motivated by the necessity for our aviation aftermarket organization to improve the capability of foreseeing intermittent and unpredictable demand for essential aircraft replacement parts. To answer this problem, we investigated an array of techniques, some tried and tested methods, and some new ones to achieve the precision and accuracy required by our sector. Each model was tuned with a variety of parameters after each time series underwent separate evaluation, training, and tuning to be suited to each distinct demand pattern. Our study has shown that the gated CNN, LSTM, and GRU models perform better than other models in all areas of examination. In particular, the set of CNN variations has repeatedly outperformed traditional statistical and machine learning models, indicating the promising potential of deep learning techniques. In particular, the gated CNN models have proven to be remarkably adept at automatically recognizing important data properties. As different metrics produce different models as the best performing, this project has demonstrated how challenging it is to find a good forecasting model for all the spare parts. Before choosing a model for forecasting such data, this work also demonstrated the need for careful consideration and understanding of the metric used.

The study addressed literature gaps by integrating advanced sequence learning techniques and innovative convolutional architectures. Additionally, a detailed architectural definition was complemented by fine-tuning the model, showcasing enhanced performance through meticulous parameter tuning. The findings of this study have major implications for this specific research area as well as the larger subject of demand forecasting. The aviation aftermarket is the focus of this study, where it is crucial to accurately forecast the intermittent and irregular demand for replacement parts, but this study can be extended to all areas of demand forecasting where such patterns are observed. According to the study, we can estimate intermittent demand much more accurately by utilizing sophisticated deep-learning models. To effectively manage unpredictable demand patterns, industries should think about switching from employing old approaches to new innovative ones such as the one discussed in this work.

6.2 Research limitations

The analysis's depth and accuracy are considerably impeded by limitations that place major constraints on the total scope of the investigation. The inability to execute fine-grained forecasting and optimization is hampered using the company's raw sales data and the lack of more information about individual items, such as their specifications, technological features, and usage patterns. Additional details regarding the parts such as lead time and unit cost could have significantly improved the accuracy of the forecast. The requirement to uphold strong confidentiality and privacy agreements gives rise to these restrictions. The company is prohibited from disclosing comprehensive item-specific data and cost-related information due to the proprietary nature of some data and its commitment to protecting sensitive information.

The study's absence of outside variables that could affect the availability and demand for spare parts is another important drawback. The research does not consider managerial choices or modifications in business strategy, as well as natural occurrences like weather changes or market variations. Although difficult to predict, these outside factors can significantly affect inventory management outcomes and, if considered, could heighten the study's realism.

6.3 Future work

Future studies should concentrate on a few crucial areas of optimization and feature engineering to improve the performance and efficacy of the suggested model. First, a thorough investigation of the optimization of the step-back value, a crucial time series analysis parameter, is required. This parameter's fine modification can result in forecasts that are more precise and efficient. Further efforts should be made to collect and include thorough data pertaining to the time series data, such as specific value trends, aircraft details, and repair data making the whole analysis a multivariate time series analysis instead of a univariate as like now. This added context can provide insightful information for predictive modelling. Exploring new features produced from these data sources has the potential to improve the model's predictive ability while feature engineering, which is still a crucial part of model refining, is still important. Additionally, lag values should be considered when modelling since they can capture temporal dependencies and trends in time series data, resulting in more precise forecasting and decision-making. Adding historical data from a longer period to the dataset would be the easiest and the most feasible avenue of improvement.

REFERENCES

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., … Zheng, X. (2016). TensorFlow: A system for large-scale machine learning. *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016*, 265–283. https://arxiv.org/abs/1605.08695v2

Abbasimehr, H., Shabani, M., & Yousefi, M. (2020). An optimized model using LSTM network for demand forecasting. *Computers & Industrial Engineering*, *143*, 106435. https://doi.org/10.1016/J.CIE.2020.106435

Aircraft Aftermarket Parts Market Size, Share & Forecast [2028]. (n.d.). Retrieved September 6, 2023, from https://www.fortunebusinessinsights.com/aircraft-aftermarket-parts-market-105451

Assaghir, Z., Makki, S., & Zeineddine, H. (2017). *Machine Learning For Intermittent Demand Forecasting*.

Azizi, F., & Wibowo, W. C. (2022). Intermittent Demand Forecasting Using LSTM With Single and Multiple Aggregation. *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, *6*(5), 855–859. https://doi.org/10.29207/RESTI.V6I5.4435

Babai, M. Z., Dallery, Y., Boubaker, S., & Kalai, R. (2019). A new method to forecast intermittent demand in the presence of inventory obsolescence. *International Journal of Production Economics*, *209*, 30–41. https://doi.org/10.1016/J.IJPE.2018.01.026

Bacchetti, A., & Saccani, N. (2012a). Spare parts classification and demand forecasting for stock control: Investigating the gap between research and practice. *Omega*, *40*(6), 722–737. https://doi.org/10.1016/J.OMEGA.2011.06.008

Bacchetti, A., & Saccani, N. (2012b). Spare parts classification and demand forecasting for stock control: Investigating the gap between research and practice. *Omega*, *40*(6), 722–737. https://doi.org/10.1016/J.OMEGA.2011.06.008

Bao, Y., Wang, W., & Zhang, J. (2004). Forecasting intermittent demand by SVMs regression. *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, *1*, 461–466. https://doi.org/10.1109/ICSMC.2004.1398341

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). Training algorithm for optimal margin classifiers. *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, 144–152. https://doi.org/10.1145/130385.130401

Braglia, M., Grassi, A., & Montanari, R. (2004). Multi-attribute classification method for spare parts inventory management. *Journal of Quality in Maintenance Engineering*, *10*(1), 55–65. https://doi.org/10.1108/13552510410526875

Breiman, L. (2001). *Random Forests*. *45*, 5–32.

Chaudhuri, K. D., & Alkan, B. (2022). A hybrid extreme learning machine model with harris hawks optimisation algorithm: an optimised model for product demand forecasting applications.

Applied Intelligence, *52*(10), 11489–11505. https://doi.org/10.1007/S10489-022-03251-7/TABLES/5

Cheng, C. Y., Chiang, K. L., & Chen, M. Y. (2016). Intermittent Demand Forecasting in a Tertiary Pediatric Intensive Care Unit. *Journal of Medical Systems*, *40*(10), 1–12. https://doi.org/10.1007/S10916-016-0571-9/TABLES/6

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 1724–1734. https://doi.org/10.3115/v1/d14-1179

Cortes, C., Vapnik, V., & Saitta, L. (1995). Support-vector networks. *Machine Learning 1995 20:3*, *20*(3), 273–297. https://doi.org/10.1007/BF00994018

Cover, T. M., & Hart, P. E. (1967). Nearest Neighbor Pattern Classification. *IEEE Transactions on Information Theory*, *13*(1), 21–27. https://doi.org/10.1109/TIT.1967.1053964

Croston, J. D. (1972a). Forecasting and Stock Control for Intermittent Demands. *Operational Research Quarterly (1970-1977)*, *23*(3), 289. https://doi.org/10.2307/3007885

Croston, J. D. (1972b). Forecasting and Stock Control for Intermittent Demands. *Operational Research Quarterly (1970-1977)*, *23*(3), 289. https://doi.org/10.2307/3007885

Dehghan Shoorkand, H., Nourelfath, M., & Hajji, A. (2024). A hybrid CNN-LSTM model for joint optimization of production and imperfect predictive maintenance planning. *Reliability Engineering & System Safety*, *241*, 109707. https://doi.org/10.1016/J.RESS.2023.109707

Eaves, A. H. C., & Kingsman, B. G. (2017). Forecasting for the ordering and stock-holding of spare parts. *Https://Doi.Org/10.1057/Palgrave.Jors.2601697*, *55*(4), 431–437. https://doi.org/10.1057/PALGRAVE.JORS.2601697

Engelmeyer, T. (2016). Managing intermittent demand. *Managing Intermittent Demand*, 1–157. https://doi.org/10.1007/978-3-658-14062-5

Fernandes, W. P. D., Silva, L. J. S., Frajhof, I. Z., de Almeida, G. da F. C. F., Konder, C. N., Nasser, R. B., de Carvalho, G. R., Barbosa, S. D. J., & Lopes, H. C. V. (2020). Appellate Court Modifications Extraction for Portuguese. *Artificial Intelligence and Law*, *28*(3), 327–360. https://doi.org/10.1007/S10506-019-09256-X

Fildes, R., Goodwin, P., Lawrence, M., & Nikolopoulos, K. (2009). Effective forecasting and judgmental adjustments: An empirical evaluation and strategies for improvement in supply-chain planning. *International Journal of Forecasting*, *25*(1), 3–23. https://doi.org/10.1016/J.IJFORECAST.2008.11.010

GitHub - keras-team/keras: Deep Learning for humans. (n.d.). Retrieved December 6, 2023, from https://github.com/keras-team/keras

Gutierrez, R. S., Solis, A. O., & Mukhopadhyay, S. (2008a). Lumpy demand forecasting using neural networks. *International Journal of Production Economics*, *111*(2), 409–420. https://doi.org/10.1016/J.IJPE.2007.01.007

Gutierrez, R. S., Solis, A. O., & Mukhopadhyay, S. (2008b). Lumpy demand forecasting using neural networks. *International Journal of Production Economics*, *111*(2), 409–420. https://doi.org/10.1016/J.IJPE.2007.01.007

Güven, İ., Uygun, Ö., & Şimşir, F. (2021). Machine Learning Algorithms with Intermittent Demand Forecasting: An Application in Retail Apparel with Plenty of Predictors. *Textile and Apparel*, *31*(2), 99–110. https://doi.org/10.32710/TEKSTILVEKONFEKSIYON.809867

Hansen, J. V., McDonald, J. B., & Nelson, R. D. (2006). Some evidence on forecasting time-series with support vector machines. *Journal of the Operational Research Society*, *57*(9), 1053–1063. https://doi.org/10.1057/PALGRAVE.JORS.2602073

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array Programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning*. https://doi.org/10.1007/978-0-387-84858-7

Hatefi, S. M., Torabi, S. A., & Bagheri, P. (2014). Multi-criteria ABC inventory classification with mixed quantitative and qualitative criteria. *International Journal of Production Research*, *52*(3), 776–786. https://doi.org/10.1080/00207543.2013.838328

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/NECO.1997.9.8.1735

Hoffmann, M. A., Lasch, R., & Meinig, J. (2022). Forecasting Irregular Demand Using Single Hidden Layer Neural Networks. *Logistics Research*, *15*(1). https://doi.org/10.23773/2022_6

Hua, Z., & Zhang, B. (2006). A hybrid support vector machines and logistic regression approach for forecasting intermittent demand of spare parts. *Applied Mathematics and Computation*, *181*(2), 1035–1048. https://doi.org/10.1016/J.AMC.2006.01.064

Huang, Z., Xu, W., & Yu, K. (2015). *Bidirectional LSTM-CRF Models for Sequence Tagging*. http://arxiv.org/abs/1508.01991

Hyndman, R. J., & Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting*, *22*(4), 679–688. https://doi.org/10.1016/J.IJFORECAST.2006.03.001

Januschowski, T., Gasthaus, J., Wang, Y., Salinas, D., Flunkert, V., Bohlke-Schneider, M., & Callot, L. (2020). Criteria for classifying forecasting methods. *International Journal of Forecasting*, *36*(1), 167–177. https://doi.org/10.1016/J.IJFORECAST.2019.05.008

Jeon, Y., & Seong, S. (2022). Robust recurrent network model for intermittent time-series forecasting. *International Journal of Forecasting*, *38*(4), 1415–1425. https://doi.org/10.1016/J.IJFORECAST.2021.07.004

Jiang, P., Huang, Y., & Liu, X. (2021). Intermittent demand forecasting for spare parts in the heavy-duty vehicle industry: a support vector machine model. *International Journal of Production Research*, *59*(24), 7423–7440. https://doi.org/10.1080/00207543.2020.1842936

Jung, S., Moon, J., Park, S., & Hwang, E. (2021). An Attention-Based Multilayer GRU Model for Multistep-Ahead Short-Term Load Forecasting. *Sensors 2021, Vol. 21, Page 1639*, *21*(5), 1639. https://doi.org/10.3390/S21051639

Kiefer, D., Grimm, F., Bauer, M., & Van Dinther, C. (n.d.). *Demand Forecasting Intermittent and Lumpy Time Series: Comparing Statistical, Machine Learning and Deep Learning Methods*. Retrieved November 15, 2023, from https://hdl.handle.net/10125/70784

Kilimci, Z. H., Akyuz, A. O., Uysal, M., Akyokus, S., Uysal, M. O., Atak Bulbul, B., Ekmis, M. A., & Silva, T. C. (2019). An improved demand forecasting model using deep learning approach and proposed decision integration strategy for supply chain. *Complexity*, *2019*. https://doi.org/10.1155/2019/9067367

Kim, S., & Kim, H. (2016). A new metric of absolute percentage error for intermittent demand forecasts. *International Journal of Forecasting*, *32*(3), 669–679. https://doi.org/10.1016/J.IJFORECAST.2015.12.003

Kingma, D. P., & Ba, J. L. (2014). Adam: A Method for Stochastic Optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. https://arxiv.org/abs/1412.6980v9

Kolassa, S. (2016). Evaluating predictive count data distributions in retail sales forecasting. *International Journal of Forecasting*, *32*(3), 788–803. https://doi.org/10.1016/J.IJFORECAST.2015.12.004

Kourentzes, N. (2013). Intermittent demand forecasts with neural networks. *International Journal of Production Economics*, *143*(1), 198–206. https://doi.org/10.1016/J.IJPE.2013.01.009

Kourentzes, N., Petropoulos, F., & Trapero, J. R. (2014). Improving forecasting by estimating time series structural components across multiple frequencies. *International Journal of Forecasting*, *30*(2), 291–302. https://doi.org/10.1016/J.IJFORECAST.2013.09.006

Lengu, D., Syntetos, A. A., & Babai, M. Z. (2014). Spare parts management: Linking distributional assumptions to demand classification. *European Journal of Operational Research*, *235*(3), 624–635. https://doi.org/10.1016/J.EJOR.2013.12.043

Li, C., & Lim, A. (2018). A greedy aggregation–decomposition method for intermittent demand forecasting in fashion retailing. *European Journal of Operational Research*, *269*(3), 860–869. https://doi.org/10.1016/J.EJOR.2018.02.029

Li, L., Kang, Y., Petropoulos, F., & Li, F. (2023). Feature-based intermittent demand forecast combinations: accuracy and inventory implications. *International Journal of Production Research*, *2023*(22), 7557–7572. https://doi.org/10.1080/00207543.2022.2153941

Lolli, F., Gamberini, R., Regattieri, A., Balugani, E., Gatos, T., & Gucci, S. (2017). Single-hidden layer neural networks for forecasting intermittent demand. *International Journal of Production Economics*, *183*, 116–128. https://doi.org/10.1016/J.IJPE.2016.10.021

Lu, C. J., Lee, T. S., & Chiu, C. C. (2009). Financial time series forecasting using independent component analysis and support vector regression. *Decision Support Systems*, *47*(2), 115–125. https://doi.org/10.1016/J.DSS.2009.02.001

Luochen, X., & Hasachoo, N. (2021). The Study of Irregular Demand Forecasting for Medicines: The Case Study of ABC Medical Center Hospital. *Proceedings - 2021 10th International Conference on Industrial Technology and Management, ICITM 2021*, 115–120. https://doi.org/10.1109/ICITM52822.2021.00028

Makridakis, S., Andersen, A., Carbone, R., Fildes, R., Hibon, M., Lewandowski, R., Newton, J., Parzen, E., & Winkler, R. (1982). The accuracy of extrapolation (time series) methods: Results of a forecasting competition. *Journal of Forecasting*, *1*(2), 111–153. https://doi.org/10.1002/FOR.3980010202

Makridakis, S., & Hibon, M. (2000). The M3-competition: Results, conclusions and implications. *International Journal of Forecasting*, *16*(4), 451–476. https://doi.org/10.1016/S0169-2070(00)00057-1

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018a). Statistical and Machine Learning forecasting methods: Concerns and ways forward. *PLoS ONE*, *13*(3). https://doi.org/10.1371/JOURNAL.PONE.0194889

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018b). *Statistical and Machine Learning forecasting methods: Concerns and ways forward*. https://doi.org/10.1371/journal.pone.0194889

Martin, D., Spitzer, P., & Kühl, N. (n.d.). *A New Metric for Lumpy and Intermittent Demand Forecasts: Stock-keeping-oriented Prediction Error Costs*.

Mckinney, W. (2010). *Data Structures for Statistical Computing in Python*.

Montero-Manso, P., Athanasopoulos, G., Hyndman, R. J., & Talagala, T. S. (2020). FFORMA: Feature-based forecast model averaging. *International Journal of Forecasting*, *36*(1), 86–92. https://doi.org/10.1016/J.IJFORECAST.2019.02.011

Mukhopadhyay, S., Solis, A. O., & Gutierrez, R. S. (2012). The Accuracy of Non-traditional versus Traditional Methods of Forecasting Lumpy Demand. *Journal of Forecasting*, *31*(8), 721–735. https://doi.org/10.1002/FOR.1242

Pedregosa FABIANPEDREGOSA, F., Michel, V., Grisel OLIVIERGRISEL, O., Blondel, M., Prettenhofer, P., Weiss, R., Vanderplas, J., Cournapeau, D., Pedregosa, F., Varoquaux, G., Gramfort, A., Thirion, B., Grisel, O., Dubourg, V., Passos, A., Brucher, M., Perrot andÉdouardand, M., Duchesnay, andÉdouard, & Duchesnay EDOUARDDUCHESNAY, Fré. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*(85), 2825–2830. http://jmlr.org/papers/v12/pedregosa11a.html

Pennings, C. L. P., van Dalen, J., & van der Laan, E. A. (2017). Exploiting elapsed time for managing intermittent demand for spare parts. *European Journal of Operational Research*, *258*(3), 958–969. https://doi.org/10.1016/J.EJOR.2016.09.017

Petropoulos, F., & Kourentzes, N. (2015). Forecast combinations for intermittent demand. *Journal of the Operational Research Society*, *66*(6), 914–924. https://doi.org/10.1057/JORS.2014.62

Petropoulos, F., Kourentzes, N., & Nikolopoulos, K. (2016). Another look at estimators for intermittent demand. *International Journal of Production Economics*, *181*, 154–161. https://doi.org/10.1016/J.IJPE.2016.04.017

Pinçe, Ç., Turrini, L., & Meissner, J. (2021). Intermittent demand forecasting for spare parts: A Critical review. *Omega*, *105*, 102513. https://doi.org/10.1016/J.OMEGA.2021.102513

Porras, E., & Dekker, R. (2008). An inventory control system for spare parts at a refinery: An empirical comparison of different re-order point methods. *European Journal of Operational Research*, *184*(1), 101–132. https://doi.org/10.1016/J.EJOR.2006.11.008

Qi, Y. (2012). Random Forest for Bioinformatics. *Ensemble Machine Learning*, 307–323. https://doi.org/10.1007/978-1-4419-9326-7_11

Romeijnders, W., Teunter, R., & Van Jaarsveld, W. (2012). A two-step method for forecasting spare parts demand using information on component repairs. *Eur. J. Oper. Res.*, *220*(2), 386–393. https://doi.org/10.1016/J.EJOR.2012.01.019

Rožanec, J. M., Fortuna, B., & Mladenić, D. (2022). Reframing Demand Forecasting: A Two-Fold Approach for Lumpy and Intermittent Demand. *Sustainability 2022, Vol. 14, Page 9295*, *14*(15), 9295. https://doi.org/10.3390/SU14159295

Seabold, S., & Perktold, J. (2010). Statsmodels: Econometric and Statistical Modeling with Python. *PROC. OF THE 9th PYTHON IN SCIENCE CONF*. http://statsmodels.sourceforge.net/

Semenoglou, A. A., Spiliotis, E., Makridakis, S., & Assimakopoulos, V. (2021). Investigating the accuracy of cross-learning time series forecasting methods. *International Journal of Forecasting*, *37*(3), 1072–1084. https://doi.org/10.1016/J.IJFORECAST.2020.11.009

Sinta, D., Wijayanto, H., & Sartono, B. (2014). Ensemble K-Nearest Neighbors Method to Predict Rice Price in Indonesia. *Applied Mathematical Sciences*, *8*(160), 7993–8005. https://doi.org/10.12988/ams.2014.49721

Smola, A. J., & Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing*, *14*(3), 199–222. https://doi.org/10.1023/B:STCO.0000035301.49549.88/METRICS

Son, J., & Yang, S. (2022). A New Approach to Machine Learning Model Development for Prediction of Concrete Fatigue Life under Uniaxial Compression. *Applied Sciences (Switzerland)*, *12*(19). https://doi.org/10.3390/APP12199766/S1

Sousa, M., Tomé, A. M., & Moreira, J. (2022). Long-term forecasting of hourly retail customer flow on intermittent time series with multiple seasonality. *Data Science and Management*, *5*(3), 137–148. https://doi.org/10.1016/J.DSM.2022.07.002

Spithourakis, G. P., Petropoulos, F., Babai, M. Z., Nikolopoulos, K., & Assimakopoulos, V. (2015). Improving the Performance of Popular Supply Chain Forecasting Techniques. *Http://Dx.Doi.Org/10.1080/16258312.2011.11517277*, *12*(4), 16–25. https://doi.org/10.1080/16258312.2011.11517277

Syntetos, A. A., Babai, M. Z., & Altay, N. (2012). On the demand distributions of spare parts. *International Journal of Production Research*, *50*(8), 2101–2117. https://doi.org/10.1080/00207543.2011.562561

Syntetos, A. A., & Boylan, J. E. (2001). On the bias of intermittent demand estimates. *International Journal of Production Economics*, *71*(1–3), 457–466. https://doi.org/10.1016/S0925-5273(00)00143-2

Syntetos, A. A., & Boylan, J. E. (2005). The accuracy of intermittent demand estimates. *International Journal of Forecasting*, *21*(2), 303–314. https://doi.org/10.1016/J.IJFORECAST.2004.10.001

Syntetos, A. A., & Boylan, J. E. (2010). On the variance of intermittent demand estimates. *International Journal of Production Economics*, *128*(2), 546–555. https://doi.org/10.1016/J.IJPE.2010.07.005

Syntetos, A. A., Boylan, J. E., & Croston, J. D. (2005). On the categorization of demand patterns. *Journal of the Operational Research Society*, *56*(5), 495–503. https://doi.org/10.1057/PALGRAVE.JORS.2601841

Syntetos, A. A., Keyes, M., & Babai, M. Z. (2009). Demand categorisation in a European spare parts logistics network. *International Journal of Operations and Production Management*, *29*(3), 292–316. https://doi.org/10.1108/01443570910939005/FULL/PDF

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2014). Going Deeper with Convolutions. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *07-12-June-2015*, 1–9. https://doi.org/10.1109/CVPR.2015.7298594

Tavenard, R., Faouzi, J., Vandewiele, G., Divo, F., Androz, G., Holtz, C., Payne, M., Yurchak, R., Rußwurm, M., Kolar, K., & Woods, E. (2020). Tslearn, A Machine Learning Toolkit for Time Series Data. *Journal of Machine Learning Research*, *21*(118), 1–6. http://jmlr.org/papers/v21/20-091.html

Teunter, R. H., Syntetos, A. A., & Babai, M. Z. (2011a). Intermittent demand: Linking forecasting to inventory obsolescence. *European Journal of Operational Research*, *214*(3), 606–615. https://doi.org/10.1016/J.EJOR.2011.05.018

Teunter, R. H., Syntetos, A. A., & Babai, M. Z. (2011b). Intermittent demand: Linking forecasting to inventory obsolescence. *European Journal of Operational Research*, *214*(3), 606–615. https://doi.org/10.1016/J.EJOR.2011.05.018

Van Thieu, N., & Mirjalili, S. (2023). MEALPY: An open-source library for latest meta-heuristic algorithms in Python. *Journal of Systems Architecture*, *139*, 102871. https://doi.org/10.1016/J.SYSARC.2023.102871

Wallström, P., & Segerstedt, A. (2010). Evaluation of forecasting error measurements and techniques for intermittent demand. *International Journal of Production Economics*, *128*(2), 625–636. https://doi.org/10.1016/J.IJPE.2010.07.013

Wan, A., Chang, Q., AL-Bukhaiti, K., & He, J. (2023). Short-term power load forecasting for combined heat and power using CNN-LSTM enhanced by attention mechanism. *Energy*, *282*, 128274. https://doi.org/10.1016/J.ENERGY.2023.128274

Wang, K., Qi, X., & Liu, H. (2019). Photovoltaic power forecasting based LSTM-Convolutional Network. *Energy*, *189*. https://doi.org/10.1016/J.ENERGY.2019.116225

Willemain, T. R., Smart, C. N., & Schwarz, H. F. (2004). A new approach to forecasting intermittent demand for service parts inventories. *International Journal of Forecasting*, *20*(3), 375–387. https://doi.org/10.1016/S0169-2070(03)00013-X

Willemain, T. R., Smart, C. N., Shockor, J. H., & DeSautels, P. A. (1994). Forecasting intermittent demand in manufacturing: a comparative evaluation of Croston's method. *International Journal of Forecasting*, *10*(4), 529–538. https://doi.org/10.1016/0169-2070(94)90021-3

Willemain"', T. R., Smarta, C. N., Shockor, J. H., & Desautels, P. A. (1994). Forecasting intermittent demand in manufacturing: a comparative evaluation of Croston's method. *International Journal of Forecasting*, *10*, 529–538.

Williams, T. M. (1984). Stock Control with Sporadic and Slow-Moving Demand. *The Journal of the Operational Research Society*, *35*(10), 939. https://doi.org/10.2307/2582137

Zhang, G., Eddy Patuwo, B., & Y. Hu, M. (1998). Forecasting with artificial neural networks:: The state of the art. *International Journal of Forecasting*, *14*(1), 35–62. https://doi.org/10.1016/S0169-2070(97)00044-7

Zhang, K., Yang, J., Sha, J., & Liu, H. (2022). Dynamic slow feature analysis and random forest for subway indoor air quality modeling. *Building and Environment*, *213*, 108876. https://doi.org/10.1016/J.BUILDENV.2022.108876

Zhou, E., Xu, X., Xu, B., & Wu, H. (2023). An enhancement model based on dense atrous and inception convolution for image semantic segmentation. *Applied Intelligence*, *53*(5), 5519–5531. https://doi.org/10.1007/S10489-022-03448-W/TABLES/6

Zhu, S., Dekker, R., van Jaarsveld, W., Renjie, R. W., & Koning, A. J. (2017). An improved method for forecasting spare parts demand using extreme value theory. *European Journal of Operational Research*, *261*(1), 169–181. https://doi.org/10.1016/J.EJOR.2017.01.053