SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Data Engineering and Analytics

# Probabilistic Sales Data Forecasting using Stochastic Differential Equations

Ali Elbadry

SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Data Engineering and Analytics

# Probabilistic Sales Data Forecasting using Stochastic Differential Equations

# Probabilistische Verkaufsdatenprognose mithilfe stochastischer Differentialgleichungen

| | |
|---|---|
| Author: | Ali Elbadry |
| Supervisor: | Prof. Dr. Felix Dietrich |
| Advisor: | Dr. Thomas Stecher (Carl Zeiss AG) |
| Submission Date: | October 14th, 2024 |

I confirm that this master's thesis is my own work and I have documented all sources and material used.

Munich, October 14th, 2024 Ali Elbadry

# Acknowledgments

First and foremost, I would like to thank Prof. Dr. Felix Dietrich for his support throughout the thesis and his periodic monitoring which made sure I remained on the right path. Next, I am truly thankful to Dr. Lydia Nemic and Dr. Kersten Clauss for giving me the opportinity to work on this thesis within ZEISS. I would also like to extend my sincere thanks to Dr. Thomas Stecher for his unwavering day to day assistance and guidance throughout the duration of the thesis. The support and assistance have been invaluable without which I could have never been able to reach my endeavor. Furthermore, I would like to thank my friends and colleagues at ZEISS for the mental support. Finally, I would like to express my heartfelt appreciation to my parents for their constant support and belief in me.

# Abstract

Financial forecasting of sales is vital for companies to make appropriate and informative decisions. Unlike point forecasting, probabilistic forecasting provides realistic uncertainty estimates, which result in more informed decisions. Additionally, interpretable solutions allow for understanding the rationale behind a model's decision. Stochastic differential equations (SDE)-based models offer probabilistic and more interpretable solutions. This thesis presents an SDE model capable of providing probabilistic forecasting for the weekly sales of 45 Walmart stores. A neural network implementation is used to learn the parameters of the SDE. Extracted features, such as the month and the day of the weekly sale, are also investigated to further improve the SDE parameters estimation. More context of the sales in the previous and recent weeks to a specific date is additionally provided to the model in the form of time-delay embedding to further boost the estimation. Using the estimated SDE parameters, multiple trajectories of the time series of each Walmart store are simulated from which the confidence intervals are computed. Different feature scaling and transformations are experimented with and applied to the data. Two baseline models, a Prophet model and a SDE-based one were additionally built to which all the experimented models were compared to. Two models were found to be superior to the rest, surpassing the SDE-based baseline and matching the Prophet-based one. The first model, which used a 4-week time-delay embedding of the weekly sales, produced the most accurate results with realistic uncertainty estimates. However, the mean of the confidence intervals was roughly constant or featureless. The second model, which utilized a standard scaled 52-week time-delay embedding of the weekly sales, grasped the structures of the original time series extremely well and provided accurate predictions with realistic uncertainty estimates. Yet, the second model was less accurate than the first and generates wider confidence intervals.

**Keywords:** Stochastic Differential Equations (SDE), probabilistic forecasting, time series forecasting, time delay embedding

# Contents

# 1 Introduction

Financial forecasting refers to predicting a company's future financial performance [Boy22]. Historical data indicative of the company's previous performance, such as cash flow statements, is used to predict performance in the future. Forecasting can either be short or long-term as demonstrated by the literature [DNP14; CPW21]. By predicting a company's future financial performance, future trends and outcomes can be anticipated. Appropriate and informed financial decisions can subsequently be taken. Additionally, an organization can avoid challenges, or at least mitigate their impact, by being informed about the potential outcome.
Financial forecasting can be conducted via machine learning. Machine learning approaches to financial forecasting are of interest at companies like ZEISS. Developed financial forecasting solutions can often be point-based, providing a single predicted value of the company's financial performance metric of interest [Tag22]. In contrast, probabilistic forecasting approaches provide either a predictive density of future outcomes or a confidence interval to which the actual value belongs with a certain probability [VWM18]. By providing uncertainty estimates, probabilistic forecasting allows making informed business decisions.
In addition to that, some of the already existing models are black-box solutions. In contrast to interpretable and explainable solutions, black box solutions do not offer the means to understand how the model reached its decision [Cas16]. Nowadays, there is a growing demand for more interpretable and explainable forecasting solutions, especially in situations where a wrong prediction could have severe consequences [Cas16]. An interpretable solution, which allows one to understand the financial forecasts, would increase trust in the model's decision-making.
Therefore, there is a need for more probabilistic and explainable solutions. This will enable companies and organizations to make more informed decisions based on more understandable forecasts that factor in uncertainty.
SDEs offer a solution to model complex processes, where change over time can occur both in a deterministic and random or noisey manner [Van76]. Since financial use cases are often of this type, SDEs can be used to model financial processes and data. Machine and deep learning solutions have been developed to learn the parameters of SDEs [Die+23]. By modeling financial data in terms of SDEs and using deep learning models to learn its parameters, one can generate multiple trajectories from which a

probabilistic solution can be computed. In addition to that, SDEs are interpretable in their formulation. Consequently, an interpretable probabilistic forecasting solution can be constructed via SDEs.
In this thesis, the use of SDEs to develop an interpretable probabilistic forecasting solution that can provide reliable forecasts with realistic uncertainty estimates is investigated. Historical financial data is used to learn the parameters of the SDE model that are then used to generate confidence intervals. The investigated approach is additionally compared in performance to state of the art solutions used in literature, namely Prophet [TL18]. Two models achieved the best performance. The usage of 4-week time delay embedding of the weekly sales produced the most accurate results with realistic uncertainty estimations. The use of standard scaled 52-week time delay embedding of the weekly sales, grasped the structures of the original time series extremely well and provided accurate predictions.
The research questions can be formulated and summarized as follows:

1. Can a more interpretable probabilistic solution generate accurate forecasts with realistic uncertainty estimates?

2. How does our approach compare to already existing solutions from literature?

The remaining part of the thesis is organized as follows:

1. **Section 2 Background and State of the Art:** This section commences with a general background of SDEs. Next, a background of probabilistic forecasting and its various approaches used in financial forecasting is outlined. The section concludes with a thorough summary of studies related to probabilistic forecasting with SDEs.

2. **Section 3 Probabilistic sales data forecasting using stochastic differential equations:** This section comprises the main part of the thesis and details the work done. It begins with a description of the data used along with the data preparation procedure. An explanation of how the chosen SDE tool was used in this work is detailed. Subsequently, an explanation of the various experiments is given. This section additionally includes the definition of the used evaluation metrics and a presentation and discussion of the obtained results.

3. **Section 4 Conclusions:** A summary of the thesis is provided in the final section. Additionally, a discussion and outlook are detailed.

# 2 State of the Art

In this section, all relevant topics related to the approaches discussed in Section 3 will be introduced. In Section 2.1, an introduction to SDE on a mathematical level will be given. Additionally, various SDE models used to represent processes and systems will be presented. The section concludes with an overview of the deep learning-based SDE frameworks that are used to develop the models in this work. In Section 2.2, the Prophet framework which is used as the baseline of this thesis is presented. Additionally, state-of-the-art probabilistic forecasting approaches in financial applications will be outlined. Finally, the section concludes with a survey of similar approaches to the one presented in this thesis, where SDEs are used for probabilistic explainable forecasting.

## 2.1 Stochastic Differential Equations

### 2.1.1 Definition and Mathematical Representation

SDEs are equations used to model a process where the behavior involves both randomness and deterministic change over time [Klo+92]. While regular differential equations are solely used to describe deterministic changes, they fail to represent the occurrence of noise in that system.
In order to represent stochastic processes, a SDE is composed of drift and diffusion components [De 06]. On the one hand, drift represents the deterministic change of the system, capturing the average change over time excluding noise. On the other hand, diffusion represents the noise, quantifing the likeliness of a system to deviate from the norm as a result of random factors. A simple form of SDEs can be represented as

$$dX_t = a(X_t, t)dt + b(X_t, t)dW_t, \tag{2.1}$$

where $X_t$ is a stochastic process, $a(X_t, t)$ represents the drift, $b(X, t)$ represents the diffusion, $dt$ is an infinitesimal time step, and $W_t$ is the Wiener process [Ive+16].
In Equation 2.1, the Wiener process $W_t$, also commonly known as the Brownian motion, is a mathematical model that is used in SDEs to model the randomness in a system [Oks13]. It is a continuous-time stochastic process or in simpler terms a continuous-time random walk, where the next step is determined by a normally distributed random variable.

In general, Equation 2.1 is not well defined and must be integrated via an SDE integration scheme. One class of stochastic calculus, is the Itô integral, using which Equation 2.1 can be represented as

$$X_t = X_0 + \int_0^t a(X_s, s)ds + \int_0^t b(X_s, s)dW_s. \tag{2.2}$$

There exists several integration schemes that are based on Itô calculus. Two of these schemes are Euler–Maruyama and Milstein [Die+23]. The Euler-Maruyama scheme is defined as

$$X_{n+1} = X_n + a(X_n, t_n)\Delta t + b(X_n, t_n)\Delta W_n. \tag{2.3}$$

The Milstein scheme is defined as

$$X_{n+1} = X_n + a(X_n, t_n)\Delta t + b(X_n, t_n)\Delta W_n + \frac{1}{2}b(X_n, t_n)b'(X_n, t_n)((\Delta W_n)^2 - \Delta t). \tag{2.4}$$

### 2.1.2 Existing SDE Models in Literature

Several variants of SDE models are used in financial applications such as stock market price prediction. One vanilla SDE model is the Geometric Brownian Motion (GBM) [NP11]. GBM is a model that has drift and diffusion terms that are proportional to the state variable. The GBM model is represented as

$$dX_t = \mu X_t dt + \sigma X_t dW_t, \tag{2.5}$$

where $\mu$ is the expected mean and $\sigma$ is the volatility or the magnitude of the diffusion. Apart from the simple GBM model, there exists a class of SDE models integrated with a jump component [NP11]. SDE models with a jump component are useful to convey sudden unexpected jumps that might occur to the value of the state variable over time. These models are particularly useful in various financial forecasting use cases. This can especially be seen in sales predictions, where unexpected increases and decreases in sales could occur as a result of unexpected events. Jump-diffusion is one of the variants that represents a stochastic process where random jumps occur alongside the usual continuous Brownian motion [Run03]. Jump Diffusion models are often used to model situations such as the occurrence of unpredictable movements in asset prices and interest rates. The jump-diffusion model is described as

$$dX_t = \mu X_t dt + \sigma X_t dW_t + \sum_{i=1}^{N_t} J_i, \tag{2.6}$$

where $J_i$ is the $i$-th jump at time $t_i$. The term $\sum_{i=1}^{N_t} J_i$ represents the sum of all the jumps that occur up to time $t$ [NP11].

The equation above is comprised of three components. The first component represents the drift term that is identical to that in the GBM. The second and third components represent the stochastic process of the Jump Diffusion model, offering a more realistic representation of continuous and discontinuous random changes in the prices of assets [Run03]. On the one hand, the second component represents the diffusion or the continuous stochastic changes. It is modeled using Brownian motion and, similar to the drift component, is identical to the diffusion term of the GBM. On the other hand, the third component represents the discontinuity in the stochastic process. Unlike diffusion, it is not modeled via Brownian motion. Instead, it is modeled using a Poisson process, where the time of occurrence of the ith jump event as well as its magnitude or size $J_i$ are stochastic.
In all previous examples, the values were assumed to be Gaussian distributed. However, there are some cases where volatility is itself stochastic [NP11]. In that case, stochastic volatility models would work best to model these systems and processes. Many stochastic volatility models exist that include the Heston model and Generalized AutoRegressive Conditional Heteroskedasticity (GARCH) models [WW05; Pal96].
Being one of the SDE models used in financial forecasting, the Heston model is used in stock markets to model option pricing [WW05]. It can be used by investors to determine and estimate the price of an option for a related security. The prices of options and their securities change throughout the day. Therefore, as an option pricing model, the Heston model attempts to identify the ideal price for investors to invest. The Heston model is different from other stochasticity models in several manners. First, the model allows for the possible existence of a correlation between the stock price and its volatility, a volatility that is expressed as reverting to the mean. Additionally, in the Heston model, the stock price being modeled does not have to be log-normal distributed. The Heston model is defined as

$$dX_t = rX_t dt + \sqrt{v_t} X_t dW_t^X, \quad dv_t = \kappa(\theta - v_t)dt + \sigma\sqrt{v_t} dW_t^v, \tag{2.7}$$

where $X_t$ is the stock or asset price, $r$ is the interest rate on the asset without any risk, $v_t$ is the stochastic variance of the asset price at time $t$, $\kappa$ is the rate of mean reversion of the variance $\theta$, $\theta$ is the long-term price variance, $\sigma$ is the volatility of the variance $v_t$, and $W_t^X$ and $W_t^v$ are the Wiener processes of the asset price and the variance of the asset price, respectively [NP11].
Another stochastic volatility model is the GARCH diffusion model [Rup04]. While the GARCH model can be used in various financial use cases and with different financial data, its primary usage is in predicting the volatility of financial assets such as stocks [Inv24; Rup04]. By predicting the volatility, the risk and return of the assets can be determined. By balancing the risk and return, the best assets for investment can be calculated. The GARCH model is useful in the case of heteroskedastic variance

of the error term, where the variance is not constant and changes over time [Rup04]. Additionally, the heteroskedastic error terms are assumed to be correlated to one another, following an autoregressive moving average process. The GARCH model is defined as

$$dX_t = \sqrt{h_t} dW_t, \quad dh_t = \alpha_0 + \alpha_1 (X_{t-1} - \mu)^2 + \beta h_{t-1}, \tag{2.8}$$

where $h_t$ is the stochastic variance at time $t$, $\alpha_0$, $\alpha_1$, $\beta$ are constants, and $\mu$ is the long-term mean of the process [NP11].
In order to estimate the parameters of SDE models using a specific dataset, there are various methods to do so. Three possible methods include maximum likelihood estimation, Bayesian inference, and Kalman filtering. The maximum likelihood involves maximizing the likelihood of the observed data [Pan+02]. In Bayesian inference, prior distribution and knowledge are assumed about the parameters. Subsequently, the posterior distribution is computed using the prior knowledge and observed data [BT11]. Finally, Kalman filtering is a method that updates the estimates of the parameters recursively. The interested reader is referred to [Sim01].

### 2.1.3 Neural Network-based Stochastic Differential Equations tools

In each of the following subsections, a specific SDE tool based on underlying deep learning frameworks will be presented. These tools are used to identify and approximate the drift and diffusion functions of the SDE. The tools discussed here are used in the methodology described in Chapter 3.

#### A Tool developed by Dietrich et al.

In their study, Dietrich et al. [Die+23] developed a deep learning-based method to approximate drift and diffusion from discrete data observations for SDEs of the form

$$dX_t = f(X_t)dt + \sigma(X_t)dW_t \tag{2.9}$$

[Die+23]. The drift and diffusion are approximated using two neural networks. As seen in Equation 2.9, the study constrained itself to SDEs with simple Brownian motion. Jumps components and stochastic volatility are not handled within this method.
The tool does not require long time series or trajectories to train the two neural networks. Instead, the tool accepts snapshot data at different time stamps, as will be discussed in detail in Section 3.2. The time step between two pair points can vary and differ from one snapshot to another.
To train and learn the weights of the drift and diffusion neural networks, Dietrich et al. develop specific loss functions based on stochastic integral schemes [Die+23]. Specifically, the tool provides two loss functions using the Itô-based Euler Maruyama

and Milstein schemes. For each of these schemes, a probability density function is defined on the data, based on which the loss function is subsequently formulated.
Dietrich et al. then proceed to demonstrate how numerical integration schemes can be reformulated for Langevin type and partial differential SDEs to be able to use the same loss functions [Die+23]. The study concludes with a demonstration of the tool using coarse-grain particle and lattice simulations.
The code of the tool that was created based on the methods described above is publicly available. Specifics about how to use the tool and the specific format of the data that is expected by it are explained in Section 3.2.

**A Modification of the Tool**

Hazrika modified the tool developed by Dietrich et al. [Die+23] to identify SDEs that are characterized by alpha-stable Lévy process noise [Haz23]. The modified tool is supposed to tackle limitations in the tool developed by Dietrich et al. [Die+23] The original tool performs well in situations and scenarios where the underlying distribution is Gaussian. This is because the algorithms used in the tool are log-likelihood-based. The tool was modified such that, along with identifying Gaussian systems and scenarios, it can additionally be used to identify stochastic systems with any alpha-stable Lévy noise. The tool was then tested on several examples with SDEs of Lévy noise.
In the modified tool, Hazrika altered Equation 2.9 to incorporate Lévy motion. The resultant equation is

$$dX_t = f(X_t)dt + \sigma(X_t)dL_t^{\alpha}. \tag{2.10}$$

Similar to Equation 2.9, Equation 2.10 constitutes a drift and a diffusion term. However, the diffusion term is divided into two components. The first component is a Brownian motion term $\sigma(X_t)$. The second component is a vector $dL_t^{\alpha}$ consisting of independent one-dimensional $\alpha$-stable Lévy motions. Each one-dimensional $\alpha$-stable Lévy motion consists of a triplet that is determined by three parameters. Two of the mentioned parameters, the drift vector $b$ and the covariance matrix $Q$, are not relevant to the tool and are set to zero. For this particular reason, no further details will be given on the drift vector and covariance matrix. The third parameter $v_{\alpha}$ is the jump measure, given as

$$v_{\alpha}(dx) = C(1,\alpha)|x|^{-1-\alpha}dx, \tag{2.11}$$

where $x \in \mathbb{R} \setminus \{0\}$ and $C(1,\alpha)$ is the intensity function of the jump measure. Thus, by setting the first two parameters to zero and only giving value to the jump measure, the alpha-based Lévy motion represents pure jumps.
To learn the drift and diffusion terms for SDEs with Lévy motion, pure jump, noise, the tool distinguishes between two cases. One case is the Cauchy distribution, where

the $\alpha$ term in the $\alpha$-stable Lévy motion equals 1. The second case comprises all other alphas in the range (0,2). For the Cauchy distribution case, the algorithm is composed of two steps. The first step involves defining a loss function for the drift component. By optimizing the Least Absolute Deviation (LAD) function, the drift function can be approximated. Once the drift function is approximated, the diffusion can easily be computed by substituting the approximaed drift into closed formed solution. For more details on the loss function used to approximate the drift term and the closed form solution determining the diffusion, the interested reader is referred to [Haz23].
For the more general case, the Euler Murayama method is initiated to train the drift and diffusion networks. Given the initial point $x_0$ and the time step $h$, $x_1$ is interpreted as a point drawn from an alpha-stable distribution. The algorithm is composed of two steps. First, an MSE loss function is used, where the error is the difference between $x_1$ and $x_0 + h * f(x_0)$. This loss is minimized to estimate the drift term. In the second step of the algorithm, a log-likelihood function is defined. There are three available $\alpha$-stable density functions that can be used in the log-likelihood function. Being seen as the best of the three options, the Zolotarev form was selected. The loss function of the diffusion function is then a combination of the log likelihood and the four parameters, $\alpha$, $b$, $Q$ and $v_\alpha$, of the alpha-stable distribution. The diffusion is subsequently estimated using the loss with the already estimated dr ase. The experiments obtained mixed results, with some leading to good estimations of the actual drift and diffusion, while others led to bad estimation. As to the best of our knowledge, the bad results are still under ongoing analysis.

## 2.2 Probabilistic Forecasting Approaches

In this section, various probabilistic forecasting approaches will be discussed, with a specific focus on financial use cases and applications. In Subsection 2.2.1, the Prophet model, which was selected as the baseline of this thesis, will be reviewed [TL18]. In Subsection 2.2.2, recent financial probabilstic forecasting research will be surveyed. However, it should be noted that the survey of approaches is not limited to sales data.

### 2.2.1 The Prophet Model

Prophet is a framework that offers time series forecasting for large scale data with the aim to produce highly accurate results even with the absence of expert analysts [TL18]. The Prophet framework offers configurable models with the ability to have a human in the loop analyst. The analyst can conduct performance analysis, where forecasting methods are compared and evaluated, and forecasts are flagged for review and modification.

Underneath the hood of the framework is a modular regression model capable of forecasting given data that contains trends and seasonality as well as data that exhibits holidays effects [TL18]. The parameters of the model are both interpretable, adjustable and configurable. Regarding trend in the data, the Prophet model comprises piecewise linear functions [TL18]. Each piecewise function models a specific trend. Therefore, a series of piecewise functions would allow to model changes in the trend and, hence, capture non-linear patterns. The model further uses a Fourier series to capture and represent seasonal effects. Using Fourier series, it can detect weekly and yearly patterns. Moreover, the model allows for the specification of holiday effects. Hence, this accounts for data where holidays have a significant effect on it, such as retail data.
In terms of probabilistic forecasting, the Prophet model can account for uncertainty by generating confidence intervals [TL18]. There are several sources of uncertainty that the model accounts for. By default, the Prophet model accounts for uncertainty in the trend and observation noise. The model assumes that the trends in the past would change on average at the same magnitude and frequency in the future. It extrapolates changes in trends and produces a distribution from which the confidence interval with degree of confidence $\alpha$. The parameter $\alpha$ is by default set to 80% and can be adjusted by the user. Apart from uncertainty in trends and observation noise, the model can also factor in uncertainty in the seasonality component. By default, the Prophet model does not factor in this source of uncertainty and full Bayesian sampling must be specified by the user for it to included. The fact that Prophet accounts for uncertainty makes it particularly useful for probabilistic forecasting use cases such as the one investigated in this thesis.
Due to its ability to produce probabilistic forecasts and to account for trends, seasonality, and holiday effects, all of which are expected to be exhibited in the sales and financial data of a corporation, Prophet was chosen as the baseline model in this thesis. The aim is to determine whether it is possible to reach the boundaries with SDEs and produce a probabilistic interpretable model that outperforms Prophet.

### 2.2.2 Previous Research in Probabilistic Financial Forecasting

Several studies investigated probabilistic forecasting in the financial field, where a variety of models have been used. Some studies were concerned with sales and retail data, while others conducted investigations on other financial applications such as cryptocurrency prices prediction.
Like the work of this thesis, Jung et al. conducted a study on probabilistic forecasting on sales data [Jun+20]. The data was self-obtained from the e-commerce website of the authors of this study. Jung et al. trained three different models, namely Prophet, DeepState and an auto-regressive recurrent network-based model consisting of Long

Short Term Memory (LSTM) cells known as DeepAR [Ran+18; Sal+20]. Moreover, four non-probabilistic baselines, namely Seasonal Auto-Regressive Integrated Moving Average (SARIMA), linear regression, moving average and multilayer perceptron, were constructed to compare the performance of the probabilistic ones. Different evaluation metrics, such as Root Mean Squared Error (RMSE) and Mean Absolute Percentage Error (MAPE), were used to compare the performance of the models. The study concluded that the used probabilistic models achieved inferior point forecast results compared to the non-probabilistic models.
Similarly, Topallar investigated probabilistic forecasting on retail data obtained from a chain of apparel stores all over Turkey [TOP22]. The data comprised 120 time series from 2018 to 2022. Similar to Jung et al., this study constructed a DeepAR model. DeepAR was used to learn a probabilistic distribution, which was subsequently used to generate point forecast trajectories for the data of the stores. The constructed DeepAR probabilistic model was compared to classical probabilistic forecasting approaches. Specifically, the DeepAR model was compared to Autoregressive Integrated Moving Average (ARIMA), Exponential Smoothing (ETS) and Seasonal-Trend decomposition using LOESS (STL). By utilizing Mean Absolute Error (MAE) and MAPE, the performance of the DeepAR was compared to classical probabilistic approaches. The DeepAR model outperformed both ETS and ARIMA.
Barry et al. also conducted probabilistic forecasting using retail sales data [BHW20]. The focus of the study was on consumer data, obtained from supermarkets and containing daily sales of items sold there. The study developed a new Bayesian methodology that is capable of forecasting counts per customer transaction via dynamic mixture count models. The model uses various features as predictors that include time-varying trends, seasonality, prices, promotions, and random effects. The study concluded that probabilistic sales forecasting on supermarket sales resulted in more accurate forecasts.
Spiliotis et al. also proposed solutions for probabilistic forecasting on sales data [Spi+21]. They utilized 30,490 time series of products from stores, obtained from the M5 dataset, to compare the performance of different forecasting techniques. The methods compared include both linear and non-linear approaches, as well as parametric and non-parametric models. Some of these methods include ARIMA, a random walk model known as Naive, Seasonal Naive (sNaive), Quantile Empirical Estimation (QEE), Poisson Theoretical Estimation (Pois), Negative Binomial (NB), Random Forest (RF) and Gradient Boosting Trees (GBT). The findings suggest that the choice of method should depend on the desired quantile and the characteristics of the series being forecasted.
Ferenczi and Bădica investigated probabilistic forecasting on a financial use case other than the sales forecasting [FB24]. They investigated probabilistic forecasting of gas prices of cryptocurrency prices, specifically Etherium. Similar to previous studies, Ferenczi and Bădica used DeepAR. The results obtained by the DeepAR model were

compared to the results obtained by a Prophet model using the same exact data. The DeepAR model was found to outperform the Prophet model.
Karanikola et al. compared several forecasting models [KLK22]. One of these models was a probabilistic deep learning-based model. The model was compared to other models such as ARIMA and Neural Basis Expansion Analysis (NBEATS). The performance of the models were compared using specifically three metrics, namely RMSE, MAE and MAPE. The focus was specifically on univariate time series. forty one year period univariate financial series were used in this study.
While many of these studies attempted to do probabilistic forecasting on sales data and many models were found to achieve good performance on the task, none investigated the use of SDEs. Thus, a main a question that this thesis aims to answer is whether SDEs can be as performant or even exceed already existing approaches. The results of this work are specifically compared to a Prophet baseline.

## 2.3 Probabilistic Forecasting via Stochastic Differential Equations

In this subsection, we discuss various studies in which SDE models were used in probabilistic forecasting. It should be noted that, to the best of our knowledge, no other study attempted to use SDEs in the probabilistic financial forecasting of sales data. However, some studies investigate other financial data, specifically stock market data. Moreover, some of the studies to be discussed modeled processes that were completely outside the financial domain. Nonetheless, the approaches in these studies are still relevant, since they demonstrate the effectiveness of SDE models for forecasting tasks in the presence of stochasticity.
Iversen et al. investigated the use of SDEs for probabilistic forecasting in the domain of wind power production [Ive+16]. Specifically, it was concluded that to carry wind power production economically, probabilistic forecasting of wind speed is necessary. To begin with, a simple SDE was proposed. To learn the SDE parameters, the SDE was trained on wind speed measurements coming from a meteorological center. Additionally, 48-hour ahead forecasts are provided via numerical weather predictions. Using this data, the model is then trained on 1-hour ahead data. To learn the parameters of the SDE, extended Kalman filtering is used. Upon training the basic SDE model, three limitations were observed. Primarily, the model predicted negative wind speed values. Secondly, when the prediction horizon exceeded 1 hour, there was a constant shift or lag between the weather predictions and the wind speed values predicted by the basic SDE model. Finally, there were some structures in the wind speed process that were not being captured by the model. In order, to fix these issues, the SDE was adjusted as

follows. First, a state-dependent diffusion term was introduced to counter the negative speed values. Second, the time derivative of the numerical weather predictions was added to the model to address the lag. Finally, diurnal variation and a stochastic trend were allowed to make sure more structures were captured by the model.
To evaluate the SDE model, two evaluation metrics were used. The likelihood was computed for both the training and test sets. Continuous rank probability score was also calculated on the test set. Additionally, to aid with the evaluation of the SDE, several autoregressive-based baselines were evaluated. The SDE outperformed all baselines in terms of the likelihood and the continuous rank probability score. The solution presented by this study is capable of producing point forecasts, predictive densities as well as multiple trajectories generated by Monte Carlo simulations.
Similarly, Moller et al. studied wind power forecasting via probabilistic means [MZM16]. A logistic-type SDE was used to account for both conditional uncertainties dependent on the state in addition to correlation structure. Subsequently, model parameters were estimated via maximum likelihood. Maximum likelihood was computed on the SDE-based correlation structures as well as a multi-dimension random vector.
Badosa et al. also investigated the use of SDE-based models on a non-financial use case [Bad+17]. Instead of wind speed and power forecasting, the study investigated the forecasting of solar irradiance. The focus of this study was to generate probabilistic forecasting for a day ahead. Therefore, given the AROME numerical weather predictions in a particular location and day, the model generates confidence intervals for the solar irradiance of the following day. Results showed and concluded that the confidence intervals generated by the model were adequate.
Similarly, Iversen et al. also conducted a study on the use of probabilistic forecasting for solar irradiance [Ive+14]. Three initial SDE models were fitted on training data. Then, they were evaluated by generating a one year period forecasts. After the three initial models, a final model was developed. The final model defines a specific bounded state space that contains all the possible events that occur in the use case. Events that almost surely will not occur are given a probability of zero.
Moreover, Elkantassi additionally conducted a study on probabilistic wind power forecasting [Elk17]. Data obtained from Uruguayan wind power production was used to learn parameters of an SDE. Two approaches were used. The first model used inference based on state-dependent diffusion, while the second model transformed the SDE into a state independent diffusion via the Lamperti transform. Like other studies, maximum likelihood was used to infer and estimated the model parameters. The study concluded that second approach or model resulted in more accurate and stable numerical predictions.
In all of the previously outlined studies in this subsection, none of the applications were financial-related. However, in the study conducted by Shah, a financial application of

SDEs was tackled [Sha21]. A comparison between five different stochastic approaches was undergone for probabilistic forecasting on stock market data. Two of these approaches involved SDEs, namely a GBM with no time-dependent drift and diffusion components and a GBM with nonlinear and time-dependent components. Additionally, ARIMA was investigated. The two final approaches are Kalman and Bayesian filters. The stock market data used comprises three specific stocks, namely Microsoft, Target, and Tesla. 199 days are used for training, while the remaining 54 days are used for testing. In order to evaluate the results and compare the models, MAE is used along with the correlation scores between the actual and predicted values. Moreover, the confidence interval is computed. Regarding the results, all three stocks show a high correlation. Additionally, mixed results are achieved based on the computed MAE. Some models perform well on some of the stocks while poorly on others.
Stavis investigated whether the complexity of cryptocurrency exchanges can be modeled via SDE and whether such models can be used in the probabilistic forecasting of cryptocurrencies prices over a short period of time [Sta23]. In an iterative approach, they constructed an SDE model capable of forecasting cryptocurrency prices. They commenced with a linear regression model fitted on the previous week data, where the slope or mean of the regression line was used as the drift parameter $\mu$, while the compute standard deviation was used as drift parameter. Next, a GARCH model was used to further improve the performance. Finally, peak performance was reached using a simple feedforward neural network. With each of the iterations of the developed SDE model, the model was tested on its ability to forecast cryptocurrency prices fifteen minutes into the future, regardless of the used data. 100 trajectories were simulated for the forecasted time series using the drift and diffusion parameters computed by the SDE model. Using the 100 simulated trajectories, the confidence interval is computed via bootstrapping. The performance of the developed SDE models is compared to traditional and existing forecasting approaches, specifically a SARIMA model, both in terms of runtime performance and prediction accuracy via RMSE. The paper concluded that creating a mathematical SDE model that could model and capture the closing prices to a certain degree is possible. However, it does not surpass the accuracy of the SARIMA model. However, unlike the SARIMA, the runtime of the constructed SDE models is much faster, which is important in the context of predicting coin price real-time predictions.
While the work of Stavis presents a recent and interesting outlook of the capabilities of SDE models in the realm of financial forecasting, the problem that was being solved does not completely overlap with the problem being investigated within this thesis. While this thesis is attempting to predict financial values for a span of several weeks, the work of Stavis focuses on generating a single, specifically the next, predicted value. Thus, the created model is attempting to predict the coin prices within the next few

moments. Once, the forecast is generated, the data is updated with the prediction and another real-time forecast is given as output. The work of Stavis is not attempting to predict how the coin prices will vary over a long duration using the given data, unlike the work of this thesis.

# 3 Probabilistic Sales Data Forecasting using Stochastic Differential Equations

In this section, the details of the approach of generating more realistic uncertainty estimates and accurate forecasts for financial, specifically sales, data is presented. In Section 3.1, the sales dataset that is used for the task at hand is introduced. All data processing and data preparation steps, that are necessary to bring the data into the needed shape for the various machine learning approaches that were attempted, are outlined. In Section 3.2, the means by which the available tools generate probabilistic forecasts via SDE are explained. Additionally, any specific requirements needed by the tool when giving the data as input are detailed. In Section 3.3, the undergone experiments, including the Prophet baseline, the initial basic SDE models along with all refinement attempts made, are explained. In Section 3.4, the metrics, both pointwise and probabilistic ones, used to evaluate the results of the undergone experiments are presented. Finally, in Section 3.5, the results of the experiments are highlighted and discussed.

## 3.1 Dataset and Data Preparation

The dataset used in this work is the Walmart Sales Dataset [Kag14]. The Walmart Sales Dataset is a large collection of weekly sales data for various departments of forty five Walmart stores over a period of two years ranging from 2010 to 2012. The weekly sales values are in the range of hundreds of thousands to millions, with some stores having sales values in the range of 500,000 USD while other stores have sales values of 1 or 2 million USD. In addition to the series sales data, the dataset includes other features such as store size and store type. Moreover, CPI, unemployment rate, and fuel Prices in the region of each store are reported. Metadata additionally includes an indicative flag of whether the week of the sales value was a holiday week as well as anonymized data related to five different promotional markdowns. Table 3.1 illustrates example data from the Walmart dataset, highlighting the different mentioned features.
The Walmart Sales Dataset is a publicly available dataset, which makes it an ideal choice for this work. This contrasts with the usage of proprietary data, where one can run into data confidentiality issues, which could hinder the work of a thesis.

Table 3.1: Example data is presented from the Walmart dataset.

| Store | Dept | Date | Weekly_Sales | IsHoliday |
|---|---|---|---|---|
| 1 | 1 | 2010-02-05 | 24924.50 | False |
| 1 | 1 | 2010-02-12 | 46039.49 | True |
| 1 | 1 | 2010-02-19 | 41595.55 | False |
| 1 | 1 | 2010-02-26 | 19403.54 | False |
| 1 | 1 | 2010-03-05 | 21827.90 | False |

| Store | Date | Temp | FuelPrice | MarkDown1 | CPI | Unemployment |
|---|---|---|---|---|---|---|
| 1 | 2010-02-05 | 42.31 | 2.572 | NaN | 211.096358 | 8.106 |
| 1 | 2010-02-12 | 38.51 | 2.548 | NaN | 211.242170 | 8.106 |
| 1 | 2010-02-19 | 39.93 | 2.514 | NaN | 211.289143 | 8.106 |
| 1 | 2010-02-26 | 46.63 | 2.561 | NaN | 211.319643 | 8.106 |
| 1 | 2010-03-05 | 46.50 | 2.625 | NaN | 211.350143 | 8.106 |

| Store | Type | Size |
|---|---|---|
| 1 | A | 151315 |
| 2 | A | 202307 |
| 3 | B | 37392 |
| 4 | A | 205863 |
| 5 | B | 34875 |

In preparation for the forecasting task, the sales of all the departments in a single store were summed up to obtain a single time series for each store, rather than one for each department. Additionally, feature extraction was applied to the data. The month in which the weekly sales occurred was extracted and the sine and cosine values of the months were computed to make the values periodic. The sine and cosine values were then normalized by dividing by the number of months in a single year. Moreover, the day of the month in which the weekly sales occurred along with the week number, such as the first or second week of the month, was extracted. Jump meta data was also generated to indicate where jumps in the sales occur. A sales value was not considered a jump if it lied between 0.9 and 1.1 times the median of the time series of the specific store. It was considered a jump otherwise. In case the weekly sales value was considered a jump, the actual sales value was reported in the jump field. Otherwise, the sales value divided by 100,000 was reported. An alternative to that would have been to use a flag feature to indicate whether a sales value is considered a jump. However, due to limitations in the tool, the former approach was selected.
To create the training, validation, and test sets, the time series data was split both temporally and by store, as illustrated in Figure 3.1. For 38 of the available Walmart stores, the first 121 weeks were used for training, the subsequent 7 weeks were used for validation, and the remaining 15 weeks were kept for testing. For two of the stores, 128 weeks were used for validation, while the remaining was used in testing. The final five stores were then kept solely for the test dataset to determine how well the model behaves and generalizes on data from stores not seen during training. As a result of the split, the training, validation, and test sets comprised 71.45%, 8.11%, and 20.44% of the data, respectively.
Following the splitting of the data, further preprocessing was done to bring the train, validation, and test sets into the shape expected by the forecasting tool, which will be discussed in the following section. The data was transformed into tuples of ($X_{t-h}$, $X_t$, $h$), where $X_t$ are the features (weekly sales, month, etc.) of the current week $t$, and $X_{t-h}$ are the same features $h$ weeks previous to the current week $t$. In other words, the tool is passed features from a pair of distinct weeks along with an indication of how far apart these weeks are.

## 3.2 SDE Tool

The probabilistic prediction was done mainly using the SDE tool developed by Dietrich et al. [Die+23]. As discussed in Subsubsection 2.1.3, the tool provides a neural network approach to estimate the drift and diffusion components from the provided data. Training data is used to learn the parameters of a neural network model which

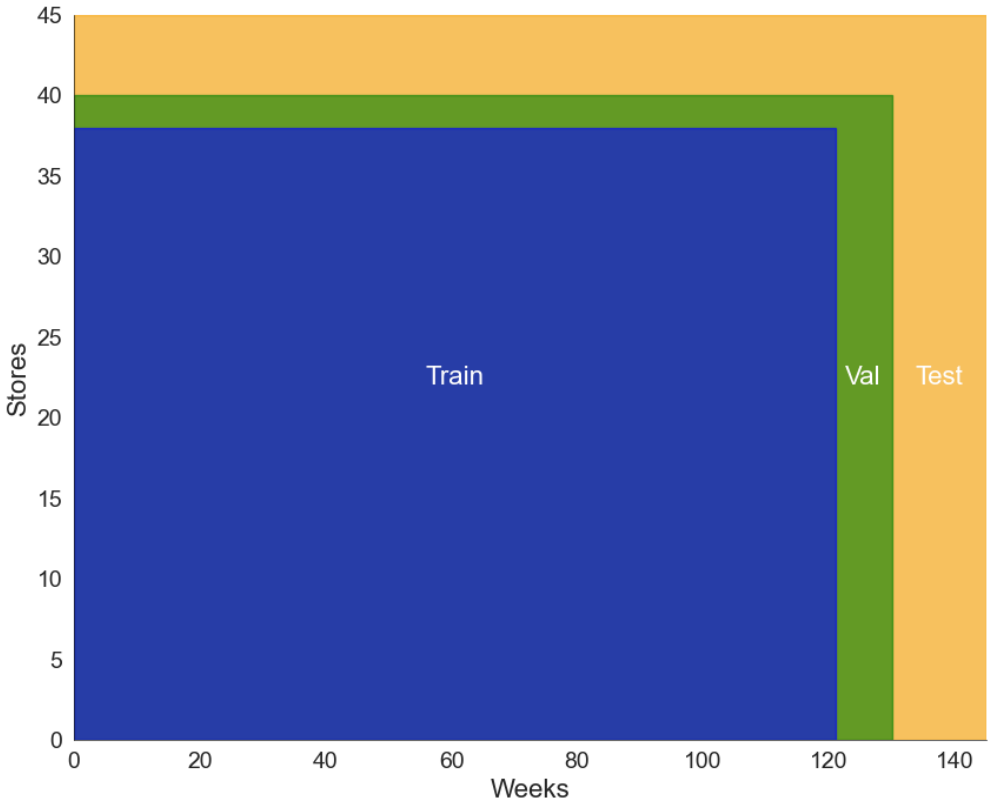


Figure 3.1: The time series data is split both store-wise and temporally to create training, validation, and test sets.

can subsequently be used to output the drift and diffusion parameters of each data point in a dataset. Using the drift and diffusion parameters produced by the trained model, the tool allows the generation of simulated trajectories that reflects the change of the sales value over time.
The framework provided by Dietrich et al. [Die+23] is Python-based[1]. In this thesis, the codebase was run using Python 3.8.7. Apart from NumPy, the software relies on several Python libraries. In terms of the deep learning encompassed in the tool, the framework depends on Keras[2] which, in turn, is built on TensorFlow[3]. Probabilistic logic is provided via TensorFlow Probability[4], a library that is built on top of TensorFlow, allowing the training of probabilistic deep learning models. In terms of plotting features provided by the framework, the tool depends on Matplotlib[5] and Seaborn[6].
As mentioned previously, the main functionality provided by the tool is to train neural networks to approximate the drift and diffusion parameters. Regarding model selection options, the framework provides several models constructed with several dense and fully connected layers. Namely, the user is given a choice between a residual model, a regular feed forward model and a gaussian process. For each of these model options, the user of the tool can specify the dimension of the input layers, the dimension of the hidden layers, the dimension of the output layer, the number of hidden layers and the activation functions to be used. In the case of the residual model, the user of the tool can additionally specify the scale term which is used in determining the output layer of the model. In the case of the Gaussian Process model, the tool creates a model that comprises dense layers for both the mean and the variance. The output layer of the variance can be customized by the user of the tool. Specifically, the user can determine the diffusivity type used for the output layer of the variance. Three diffusivity types exist, namely diagonal diffusivity, triangular diffusivity and SPD. In addition, to these three model options, the framework offers the possibility of training an autoencoder and variational autoencoder. The autoencoders can be used in conjunction with the Gaussian process model. The introduction of autoencoders by the tool allows the enhancement of the Gaussian process model by reducing the dimensionality of the input data in case a large number of datapoints is given.
Regardless of the selected model, the framework in not constructed to handle entire time series. Instead, as outlined in Subsubsection 2.1.3, the tool is built to handle snapshots of a time series. Hence, specific requirements, regarding the input to the

[1]https://www.python.org/.
[2]https://keras.io/.
[3]https://www.tensorflow.org/.
[4]https://www.tensorflow.org/probability.
[5]https://matplotlib.org/
[6]https://seaborn.pydata.org/.

models, is expected. The snapshots, expected by the given model, are given in terms of pairs ($X_{t-h}$, $X_t$) and the duration $h$ between the pair of points. Since the Walmart dataset provides entire time series rather than snapshots from given time series, the input had to be reformatted. Consequently, the data preparation steps, in which the time series were transformed to tuple form, outlined in the previous section were necessary.
In addition to neural network-based drift and diffusion estimators, the tool additionally provides a function to generate multiple trajectories for the time series. The function takes as input the computed drift and diffusion parameters in addition to the initial point of the time series for which a trajectory is to be estimated. Additionally, the user should specify the number of trajectories that need to be generated. The number of steps within the generated trajectory along with the step size between two time steps should also be detailed. With the specified initial starting point of the trajectory in addition to the approximated drift and diffusion terms, the subsequent values in the trajectory are determined via the stochastic integration strategies. As mentioned in Subsubsection 2.1.3, the SDE tool, provided by [Die+23], offers several integration strategies. In addition to the Euler Maruyama and Milstein, already detailed in in Subsubsection 2.1.3, the tool additionally provides the Milstein approximate scheme [Die+23].
In addition to estimating SDE parameters and simulating a number of trajectories, the tool provides means to plot time series and trajectories. However, the plot functions are built to handle one dimensional data. While this was sufficient at first, later experiments required more dimensions with the introduction of time-delay embeddings, as will be detailed later. Consequently, the plotting functionality of the framework was modified to better suit the requirements of the experiments carried out in this thesis. Finally, the framework provides means to calculate and generate the probability density functions. This functionality of the tool is not used in the work carried out by this thesis. Hence, no further details will be provided on the probability density.
With the neural network-based tool, the work of this thesis can be facilitated. The sales time series data is given as input to the neural network estimators to compute the drift and diffusion parameters for the individual data points. With the computed drift and diffusion terms, multiple trajectories for each time series are generated. The mean and standard deviation of the trajectories are subsequently computed, with which one can generate a confidence interval for the time series of each store.
Although the work mainly relied on the tool developed by Dietrich et al. [Die+23], later experiments depended on the modified version of the tool, discussed in Subsubsection 2.1.3 that incorporated alpha-stable Lévy process noise [Haz23]. The modified version offers the same functionality as the original framework. However, the stochastic integration schemes and probability density functions incorporate Lévy noise in addition to the regular Gaussian case. Moreover, the tool additionally provides two fully

connected neural network models in addition to the models available in the original tool. These fully connected networks are used to estimate the drift and diffusion in case the Lévy noise is not based on a Cauchy distribution. In this thesis, the tool was used in the scope Cauchy distribution Lévy noise. Consequently, details of the usage of the tool for general Lévy noise is not discussed.

## 3.3 Experiments

Several different experiments were carried out to investigate the use of SDE for probabilistic forecasting. To begin with, two baselines were created to be able to have a reference model for comparison. One of these baseline models was an SDE using the tool described in Section 3.2, while the other comprised a Prophet model. Next, a basic SDE model based only on the weekly sales value was trained. As an attempt to build on the basic model, time-delay embedding was introduced as a means to introduce more history and context to the model for better estimation of the drift and diffusion parameters. Subsequently, several variations of this SDE model were experimented with by combining the weekly sales values with the extracted features. To improve results further, different scaling methods, to counter the difference in feature scales and the non-normality of the data, were used. Additionally, varying the embedding length was attempted. This section concludes with an experiment where the modified version of the SDE tool, that introduces jump-based SDEs, was used.

### 3.3.1 Prophet Baseline

Our experiments began with the construction of a Prophet baseline model. Since no hyperparameter tuning or model selection was carried out, there was no use for the validation dataset in that case. Therefore, the Prophet model was trained on a combination of the training and validation, while the test set was used for forecasting. Following the combination of the training and validation sets, a series of Prophet models were trained. A general model was trained using the entire training and validation sets. The general model was used to generate forecasts for stores not seen during training. Additionally, an individual model was created for each of the 28 stores available in the combined training and validation set. Hence, each of the 28 individual models were used to generate forecasts for the store with which its data was trained on. This prediction task involved generating point forecasts along with a a 95% confidence interval for a 15-week period kept solely for the test dataset. In case a particular store has a period longer than 15 weeks in the test set, the prediction is done for the first 15 weeks.

### 3.3.2 SDE Baseline

As a second baseline, an SDE model based on the "same month one year back" concept was created. The reasoning behind this is to take advantage of seasonality that is found in the sales data. Since seasonality is evident in the data as seen in Figure 3.2, the model is expected to perform well and its performance could be a target to aim for when the full data is used. Instead of using the training, validation, and test sets described in Section 3.1, the series data was restricted to the months of June and July. The training set included the sales values of this two-month period from the year 2010. Sales values from 2011 were used in the validation dataset, while sales from the year 2012 were used in the test set. Months and other extracted features were excluded from this experiment. This resulted in the training, validation, and test sets containing 9 sales values for each of the 45 stores. It should be noted that the procedure, described in Section 3.1 to bring the training, validation, and test sets in the tuple format expected by the used SDE tool, was also applied for the baseline. Each tuple includes the sales of the specific week, the sales of the previous week and the difference between the weeks, which in this experiment was always one.

In order to estimate the drift and diffusion components from the weekly sales data

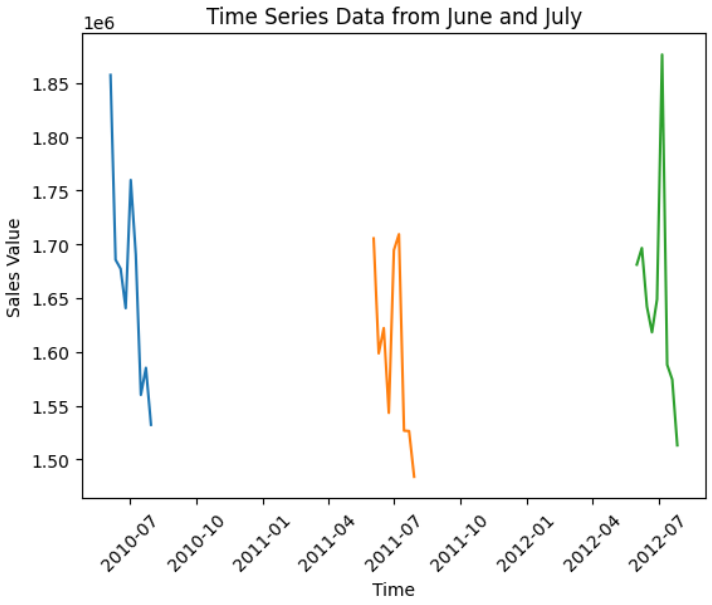


Figure 3.2: Seasonality of the data is evident from plotting the June and July data.

from the months of June and July, a Gaussian process model was defined using the SDE tool. The model had two hidden layers. The input layer and output layers had a

Table 3.3: Description of the layers in the Gaussian process model is presented.

| Layer (type) | Output Shape | Param # | Connected to |
|---|---|---|---|
| GP_inputs (InputLayer) | (None, 2) | 0 | |
| GP_mean_hidden_0 (Dense) | (None, 50) | 150 | GP_inputs[0][0] |
| GP_std_hidden_0 (Dense) | (None, 50) | 150 | GP_inputs[0][0] |
| GP_mean_hidden_1 (Dense) | (None, 50) | 2550 | GP_mean_hidden_0[0][0] |
| GP_std_hidden_1 (Dense) | (None, 50) | 2550 | GP_std_hidden_0[0][0] |
| GP_output_mean (Dense) | (None, 2) | 102 | GP_mean_hidden_1[0][0] |
| GP_output_std (Dense) | (None, 2) | 102 | GP_std_hidden_1[0][0] |

dimension of one, while the two hidden layers had a dimension of fifty. Exponential Linear Unit (ELU) activation functions and a diagonal diffusivity are additionally used. The full architecture of the model is described in Table 3.3. For the training of the model, a batch size of 32 and a learning rate of 0.001 were specified. In terms of SDE integration schemes, the Euler–Maruyama scheme was used. The model was trained for one thousand epochs. Hyperparameter tuning was done over the number of epochs to obtain an optimal value.

After the training of the model, the computed drift and diffusion were used to generate one hundred different trajectories for each of the forty five stores for nine selected weeks in 2012 kept for the test dataset. To generate the trajectories for each time series, the model uses, as a starting point, the last sales value prior to the period kept for prediction. Once the one hundred trajectories are generated, the mean and standard deviation of the trajectories are calculated. The mean and standard deviation is then used to compute the 95% confidence interval as follows:

$$CI = \bar{x} \pm z \cdot s, \tag{3.1}$$

where $CI$ is the confidence interval, $\bar{x}$ is the sample mean, $z$ is the z-value for the confidence level and $s$ is the sample standard deviation.

### 3.3.3 Basic SDE Model with Weekly Sales Values

Following the baseline, the same training setting was used with the full data. Similar to the baseline, each tuple includes the sales of a specific week, the sales of the previous week, and the difference between the weeks. Additionally, to generate the trajectories for each time series, the model uses, as a starting point, the last sales value prior to the period kept for prediction. In the case of the stores not seen during training, and hence do not have points prior to the prediction period, the first point in the prediction

period is used as a starting point. Hyperparameter tuning was done over the number of epochs. As a result, the number of epochs was adjusted to 400.

### 3.3.4 Time-delay Embedding

While input to the basic SDE consists of tuples of data comprising of weekly sales values from two different weeks along with how many weeks apart these sales values are, an alternative would be to include more history to the model via time-delay embedding. Once again, the model would be given tuples of data. However, rather than inputting weekly sales values from two different weeks, one would give it a vector consisting of sales values from $n$ consecutive weeks along with another vector of equal length to the first, comprising the weekly sales values shifted by $t$ weeks. Therefore, if a time series consists of the following values [800, 750, 600, 900, 925, 300, 300] and one uses an embedding length of n =4 and shift of one week, the first tuple would consist of the following form ([800,750,600,900], [750, 600, 900, 925], t= 1 week).
The shift $t$ was always set to one week. Initially, an embedding length of four weeks ($\approx 1 months$) was set. This embedding length was then varied such that lengths of 2, 3, 5, 6, 7 and 8 were experimented with. Additionally, to generate the trajectories for each time series, the model uses, as a starting point, the last embedding prior to the period kept for prediction. In the case of the stores not seen during training, and hence do not have embeddings prior to the prediction period, the first embedding in the prediction period is used as a starting point. Once again, hyperparameter tuning was done and the number of epochs was set to 300.

### 3.3.5 Combining Extracted Features with Time-delay Embedding

In all previously mentioned experiments, only the sales values are passed as input to the SDE and Prophet models from which point forecast and a confidence interval for the store time series data are generated. In an attempt to further improve the performance of the time-delay embedding model outlined in the previous section, extracted features were given as input to the model in combination with the time-embedding of weekly sales values.
As outlined in Section 3.1, the month in which the sales occurred was extracted from the date. Different variations of representing the month were experimented with. One variation comprised just a raw value of the month, where it was represented as an integer ranging from one to tweleve. Another variation comprised the normalized sine and cosine values to make the values periodic.
In addition to combining the extracted months with the weekly sales values, other combinations with the sales values were tried out. It was observed, when visualizing

the data, that there is an increase in the sales in the beginning of the months. Therefore, in order for the model to observe this, both the week number and day of month are used. Both features were used in raw form, meaning the sine and cosine of these values are not computed.
Finally, the generated jump meta data was used. As indicated previously, the purpose of this feature was to indicate extreme jumps in the time series, whether they are sharp increases or decreases in the weekly sales values. The reason behind the use of this feature is due to the fact the used SDE tool, unlike its modified counterpart that will be used in a later experiment, does not capture jumps on its own.
It is worth noting that other features were also used in addition to the month, day of month, week number and the indicator of a jump in sales. These used features came as metadata with the Walmart Sales Dataset. However, all these features were only experimented only on the basic SDE model and not with the time-delay embedding. First, the flag that indicates whether the weekly sales occurred during a holiday week was used. This flag was zero for all weeks except for four specific occasions. Namely, these specific occasions are Super Bowl, Labor Day, Thanksgiving and Christmas. Therefore, for each of the 45 stores, only 4 of the 143 weeks were marked as holiday weeks, which is quite sparse. Still, the holiday flag was combined in addition to the weekly sales.
Additionally, anonymized data from five markdown events that preceded prominent holidays were combined with the weekly sales values [Kag14]. Since the markdown values are only reported starting November 2011, most values were NaN. NaN values were replaced by 0.
The same model and training setting used in the previous experiments were used here. Again, hyperparameter tuning was carried out and the number of epochs was set to that was deemed optimal for the different combination of features.

### 3.3.6 Applying Scaling and Transformations to Time-delay Embedding Data

In attempt to further improve the performance of the model, various feature scaling and transformations were applied to the time-delay embedding format of the data. First, since the underlying model assumes a normal distribution of the data and such an assumption may be violated, the attempt to scale the data using box-cox transform was carried out [Sak92].
Additionally, it was observed that the model struggles to correctly estimate the drift and diffusion parameters when features of different scales are used. The SDE model obtains a greater loss when estimating the equation parameters. Consequently, the generated mean of trajectories is less accurate, and the confidence intervals are wider and less confident. The loss of the model also increases when using extremely long

time-delay embeddings. As a result, min-max and standard scaling were experimented with to deal with the different feature scalings and long-time-delay embeddings [KK16]. Ultimately, standard scaling was concluded to be superior to min-max scaling with this specific dataset and was, hence, selected. Scaling was applied to both time-delay embedding where other extracted features were combined with the weekly sales value and in cases where an extremely long time-delay embedding was used. Embeddings of lengths 5, 6, 7, 8, 16 ($\approx 4months$), 24 ($\approx 6months$), 52 ($\approx 1year$) and 78 ($\approx 1.5years$) were tried. Once again hyperparemeter tuning is applied on the epochs parameter. It should also be noted, to visualize the plots, the inverse scaling was applied to obtain the original values.

### 3.3.7 Using the Modified Tool for Lévy Process

As mentioned previously, the SDE tool used so far does not incorporate jumps in its computation of the drift and diffusion. Thus, the tool does not factor in the sharp jumps that occur in the weekly sales data when computing the parameters. One way of countering this was to generate jump meta data and use them as additional features to the existing model as discussed.
Another solution would be to use a jump-based SDE model. As previously mentioned, the currently used tool was modified to allow for jumps by incorporating alpha-based Lévy noise in addition to regular Brownian motion. In the experiments, the $\alpha$ was set to one. Hence the focus was specifically on a Cauchy distribution and not on general Lévy motion. The tool was applied to the basic model and the time-delay embedding containing 4 weeks of sales values.

## 3.4 Evaluation Metrics

To account for the evaluation of both point and probabilistic forecasts, different evaluation metrics are utilized. For point forecast, the distance between the actual sales values in the test data and those generated by the model is measured. In the case of the Prophet baseline model, the relative distance is measured between the original sales values and those generated by the Prophet model. In the case of the SDE models, the distance between the original values and the mean of the 100 generated trajectories is computed. To measure the relative distance, several different metrics are computed namely Mean Squared Error (MSE), RMSE, and MAE. The metrics are defined as

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \tag{3.2}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}, \tag{3.3}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|, \tag{3.4}$$

where $n$ is the number of samples, $y_i$ is the true value of the $i$th sample , and $\hat{y}_i$ is the predicted value of the $i$th sample.
To evaluate the probabilistic side of the forecast, the correctness of the confidence interval is evaluated, by measuring how well the confidence interval covers the original sales forecast. Specifically, the log-likelihood is used as an indication of that, which is defined as

$$\frac{1}{n}\mathcal{L}(\theta|y_1, y_2, ..., y_n) = \frac{1}{n}\sum_{t=1}^{n}\log f(y_t|\theta), \tag{3.5}$$

where $\theta$ is the estimated parameters, $y_1, ..., y_n$ are the observed weekly sales value in the time series at, $f(y_t|\theta)$ is the likelihood of observing given the parameters $\theta$. The average is taken over all the weekly sales values in the time series.
The percentage of weekly sales values that fall between the lower and upper limit of the confidence interval is also computed as another indication of how well the confidence interval covers the data. Finally, the average standard deviation of the confidence interval of each time series is additionally reported. This indicates how confident the model is. A small average standard deviation is an indication of a highly confident model, while a large average standard deviation is an indication of the opposite.

## 3.5 Results

In this section, the results of the previously outlined experiments will be presented. Additionally, a thorough discussion and comparison of the results of the various experiments will be given. Table 3.4 presents the average results over all the stores for the significant experiments.
Additionally, Table 3.5, Table 3.6 and Table 3.7 represent the average results on the stores seen in all three sets, seen only in the validation and test sets and seen only in the test set, respectively. By comparing results of stores seen during training with those not seen, one can assess how well the model generalizes on unseen data.

Table 3.4: A summary of the overall metric results of the significant experiments is presented. The best results are highlighted in bold.

| Experiment | MSE | RMSE | MAE | log-LL | % covered | STD |
|---|---|---|---|---|---|---|
| Prophet | 20,608,437,501.53 | 81,804.77 | 72,978.99 | -12.45 | 94.07 | 94135.84 |
| SDE Baseline | 28,719,237,265 | 148,708.50 | 140,187.70 | -15.55 | 52.50 | 74,731.07 |
| Basic SDE | 16,524,931,287 | 102,285.12 | 86,379.34 | -13.52 | **100.00** | 353,774.81 |
| 4w Embedding | **5,820,000,000.00** | **64,131.58** | **51,305.31** | **-11.68** | 92.74 | **67,128.35** |
| 4w + months | 11,000,000,000.00 | 90,680.08 | 78,164.28 | -12.10 | 99.85 | 158,959.50 |
| Box-cox | 7,680,000,000.00 | 74,068.24 | 60,859.34 | -11.99 | 99.85 | 169,730.40 |
| Scaled + month | 10,400,000,000.00 | 86,053.40 | 73,451.65 | -12.20 | 100 | 191,103.40 |
| 52w + month | 55,020,000,000 | 192,332.2 | 175,431.72 | -13.55 | 93.90 | 245,515.20 |
| 52w | 20,100,000,000 | 130,073.20 | 109734.26 | -13.18 | 86.41 | 105,143.97 |
| Jump tool | 27,097,314,173 | 143,943.59 | 121586.10 | -16.37 | 65.19 | 172177.08 |

Table 3.5: A summary of the overall metric results of the significant experiments for only the stores found in all three sets is presented. The best results are highlighted in bold.

| Experiment | MSE | RMSE | MAE | log-LL | % covered | STD |
|---|---|---|---|---|---|---|
| Prophet | **393,638,8125** | **51,153.52** | **42,193.74** | **-12.22** | 93.33 | **59,247.55** |
| Basic SDE | 12,466,460,386.00 | 94,869.30 | 78,867.38 | -13.49 | **100.00** | 344,673.36 |
| 4w Embedding | 5,200,000,000.00 | 60,998.31 | 49,550.85 | -12.41 | 93.86 | 68,449.30 |
| 4w + months | 11,800,000,000.00 | 93,883.87 | 82,376.57 | -12.93 | **100.00** | 162,806.40 |
| Box-cox | 6,530,000,000.00 | 69,508.76 | 57,733.15 | -12.93 | **100.00** | 173,711.30 |
| Scaled + month | 10,100,000,000.00 | 85,943.46 | 74,868.80 | -13.04 | **100.00** | 194,149.30 |
| 52w + month | **5,040,000,000** | 61,840.42 | 49,779.34 | -12.866 | 99.47 | 176,653.30 |
| 52w | 10,714,085,830 | 88,012.11 | 73,096.43 | -12.70 | 92.81 | 91,286.62 |
| Jump tool | 30,261,959,216 | 153,987.99 | 131,432.84 | -17.50 | 63.33 | 181,251.15 |

Table 3.6: A summary of the overall metric results of the significant experiments for only the stores found just in validation and test sets is presented. The best results are highlighted in bold.

| Experiment | MSE | RMSE | MAE | log-LL | % covered | STD |
|---|---|---|---|---|---|---|
| Basic SDE | 117,378,000,000.00 | 298,026.19 | 277,182.00 | -14.31 | **100.00** | 649,945.79 |
| 4w Embedding | 22,200,000,000.00 | 139,916.00 | 111,209.70 | -14.13 | 83.33 | **113,182.30** |
| 4w + months | **16,800,000,000.00** | **127,398.70** | **102,082.90** | **-13.55** | 97.00 | 256,854.90 |
| Box-cox | 36,800,000,000.00 | 188,128.70 | 163,434.10 | -13.72 | 96.67 | 269,819.60 |
| Scaled + month | **36,500,000,000.00** | 184,838.10 | 146,312.60 | -13.75 | **100.00** | 349,370.60 |
| 52w + month | 105,000,000,000 | 322,822.20 | 301,084.10 | -14.23 | 93.33 | 314,377.10 |
| 52w | 29,658,391,283 | 172,134.29 | 146,372.08 | -13.66 | 80.00 | 119,001.32 |
| Jump tool | 24,262,780,008 | 154,701.27 | 124,583.61 | -15.47 | 83.33 | 334,614.00 |

Table 3.7: A summary of the overall metric results of the significant experiments for only the stores found just in test sets is presented. The best results are highlighted in bold.

| Experiment | MSE | RMSE | MAE | log-LL | % covered | STD |
|---|---|---|---|---|---|---|
| Prophet | 153,984,832,516.00 | 327,014.75 | 319,261.07 | -14.31 | **100.00** | 373242.20 |
| Basic SDE | 7,028,126,102.00 | 80,348.91 | 67,149.14 | -13.45 | **100.00** | 304,477.50 |
| 4w Embedding | 3,940,000,000.00 | 57,630.63 | 40,677.43 | -5.16 | 88.00 | 38,667.57 |
| 4w + months | 2,850,000,000.00 | 51,643.86 | 36,583.42 | -5.17 | **100.00** | 90,565.14 |
| Box-cox | 4,780,000,000.00 | 63,096.10 | 43,588.45 | **-4.14** | **100.00** | 99,439.95 |
| Scaled + month | **2,390,000,000.00** | **47,375.02** | **33,536.90** | -5.22 | 99.00 | 104,647.60 |
| Jump tool | 4,179,825,516 | 63,303.04 | 45,551.89 | -8.17 | 72.00 | **38.239.34** |

### 3.5.1 Prophet Baseline

As illustrated by Table 3.4, the baseline resulted in a MSE of 20,608,437,501.53, a RMSE of 81,804.77, a MAE of 72978.99, an average log likelihood of -12.4538 and a percentage covered of 94.07%. Additionally, a standard deviation of 94135.84 is reported, which indicates a confident model.

Figure 3.3 illustrate generated trajectories and confidence intervals. The confidence interval covers the majority of the original weekly sales values, which is also validated by the coverage probability. The confidence interval is not extremely wide and maintains the same width across all time intervals. This signifies that the model is highly confident of its predictions and it does not lose confidence the more it progresses into future time stamps, at least within a 15-week period that is used for generation. Additionally, the mean of the trajectories and the confidence interval represent structures present in the original data. Thus, the confidence interval is following the time series of original values as depicted in Figure 3.3.

Considering that the weekly sales values are generally in order of 1 or 2 million dollars, with some stores having slightly lower sales values, the resulted in a MSE, RMSE, and MAE represent a reasonable point forecast error. It should also be noted that the difference in the order of magnitude of the sales values between different stores and the fact that the mean is not robust to outliers causes the metric values to be slightly large. All three point forecast metrics use absolute errors in the computation and, consequently, the absolute errors will be dominated by the stores with the large values. In addition to the overall point and probabilistic forecast metrics, Table 3.5 and Table 3.7 report the average metrics for the stores found in all sets and in the test set only, respectively. As depicted in Table 3.5, the stores found in all sets resulted in a MSE of 3,936,388,124.72, a RMSE of 51153.52, a MAE of 42193.74, an average log likelihood of -12.2218 and a percentage covered of 93.33%. A standard deviation of 59247.55 is obtained. As illustrated in Table 3.7, the stores found in the test set only resulted in a MSE of 153,984,832,516.00, a RMSE of 327,014.75, a MAE of 319,261.07, an average log likelihood of -14.31 and a percentage covered of probability of 100%. A standard deviation of 373242.20 is obtained. As seen by the contrast in the results, the model is more accurate,point-wise, with the stores seen during training. It also produces more realistic uncertainty estimates.

### 3.5.2 SDE Baseline

As depicted by Table 3.4, the baseline attained a MSE of 28,719,237,265, a RMSE 0f 148,708.50, a MAE 0f 140,187.70, a log-likelihood of -15.55 a coverage percentage of 52.50% and a standard deviation of 74,731.07. In terms of point forecast performance,


Figure 3.3: Confidence intervals are generated by the Prophet model for stores seen during training with both original sales values (blue) and mean of the one hundred trajectories (orange) plotted.

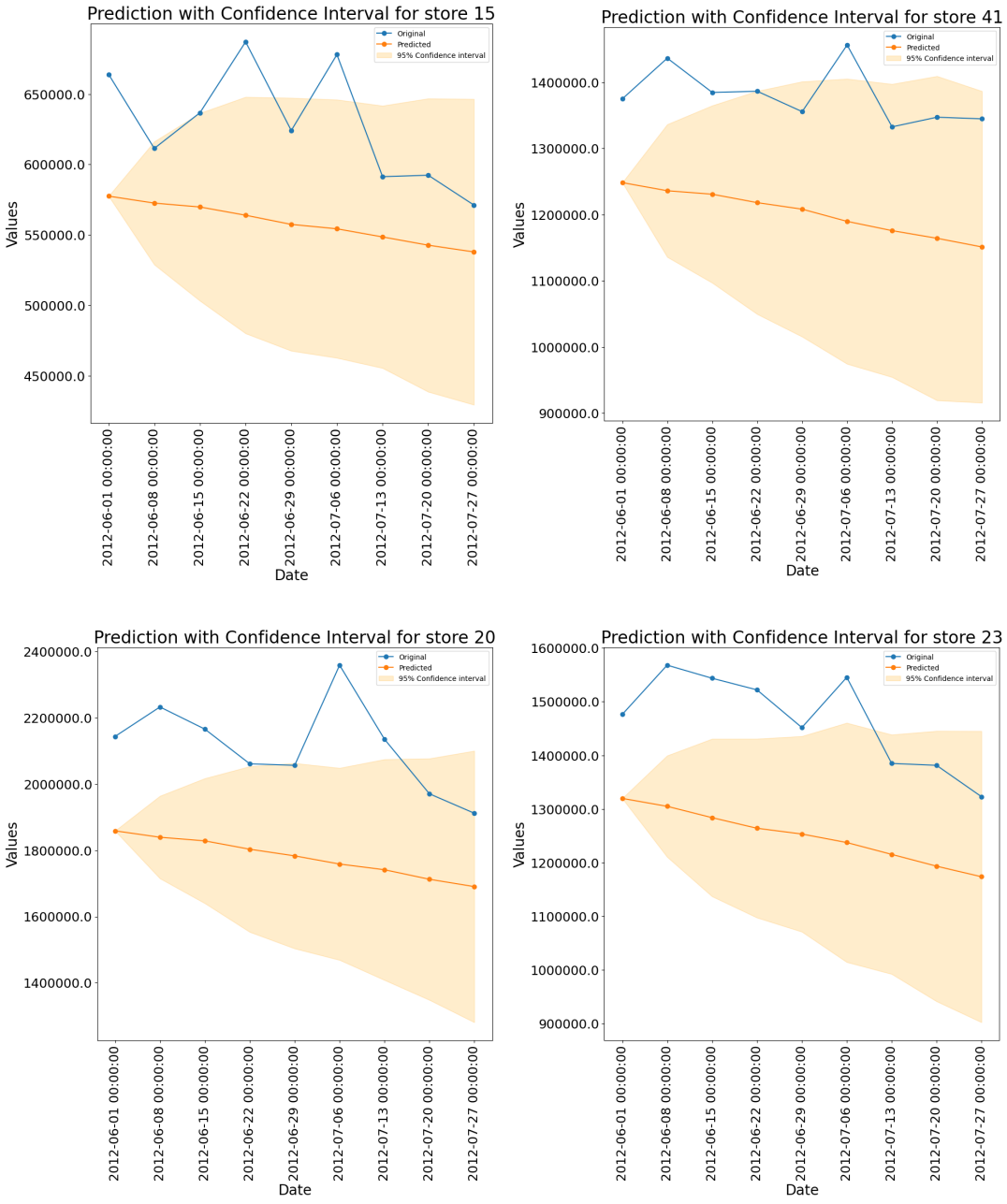


Figure 3.4: Confidence intervals are generated by the SDE baseline with both original sales values (blue) and mean of the one hundred trajectories (orange) plotted.

the error resulting from the difference between the original values and the mean of the trajectories is high, as can be deduced from the obtained RMSE and MAE. Probabilistically, the percentage of original weekly sales values covered by the confidence interval is low, as also indicated by Figure 3.4. Additionally, a low likelihood is reported since the majority of the sales values lies on the edge or outside the confidence interval as evident by the plots in the figure. However the average standard deviation of the trajectories is small, which indicates a confident model, regardless of whether it is faulty or not.
Unlike the Prophet baseline, the generated mean of the trajectories and the confidence intervals, illustrated in the figure, do not reflect any structure of the original weekly sales values. As seen in the figure, the mean of the trajectories is approximately a slowly decreasing straight line. The confidence interval does not reflect structures and trends in the original data. The confidence interval additionally starts narrow and gets wider towards the end, indicating that the model is getting less confident as the weeks progress.
At a first glance, the reported results might look to be counterintuitive. While the goal was to take advantage of the seasonality that is evidently there by using data from the months of June and July, the model performed poorly. However, the poor performance can be attributed to the low number of datapoints, unlike what is expected by the underlying deep learning-based SDE models.

### 3.5.3 Basic SDE Model with Weekly Sales Values

The basic SDE model on the full data achieved a MSE of 16,524,931,287, a RMSE of 102,285.12, a MAE of 86,379.34, a log-likelihood of -13.52, a coverage percentage of 100.00%, and a standard deviation of 353,774.81. These results are reported in Table 3.4. In addition to that, Table 3.5 and Table 3.7 report the average metrics for the stores found in all sets and in the test set only, respectively. Figure 3.5, Figure 3.6 and Figure 3.7 illustrate generated trajectories and confidence intervals for stores found in all three sets, stores found solely in the validation and test sets, and stores seen only in the test set, respectively. In comparison to the SDE baseline, there is an improvement in the overall metrics. Specifically, point-wise error is lower as indicated by the RMSE among other metrics. The log likelihood and the percentage of the original values covered by the confidence interval demonstrate an improvement in the probabilistic performance. The mentioned probabilistic improvement reflects an improved point-forecast, unlike in the SDE baseline case, as indicated by the figures.
However, one major downside to the current model is that the confidence interval is too wide, which is indicated statistically, by the large standard deviation, and visually in

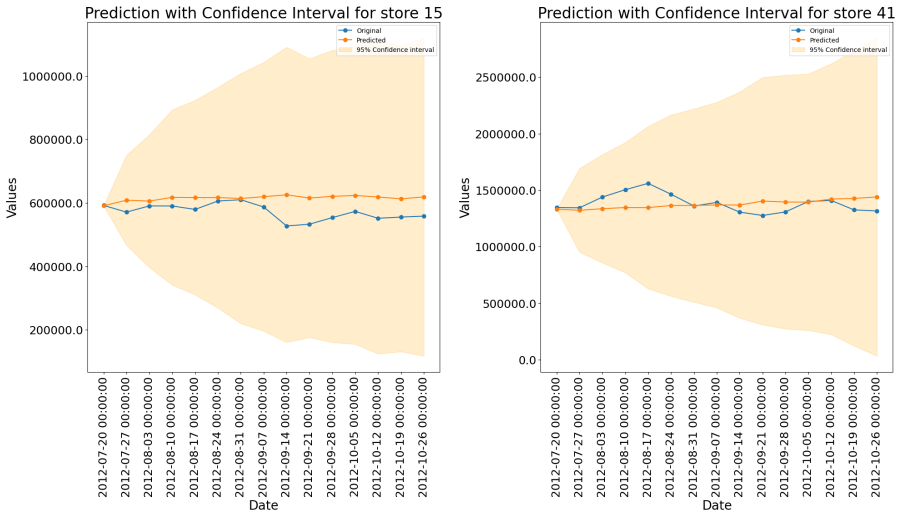


Figure 3.5: Confidence intervals are generated by the SDE basic model for stores found only in all three sets with both original sales values (blue) and mean of the one hundred trajectories (orange) plotted.

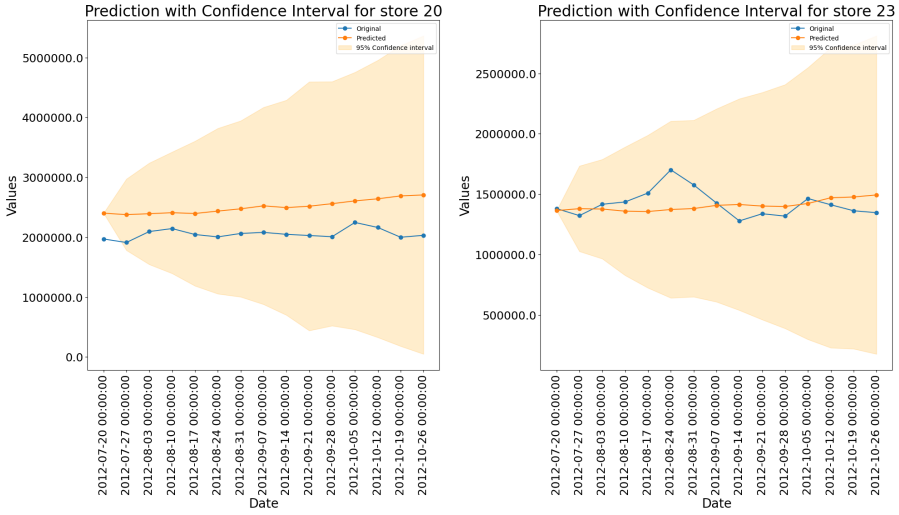


Figure 3.6: Confidence intervals are generated by the SDE basic model for stores found only in the validation and test sets with both original sales values (blue) and mean of the one hundred trajectories (orange) plotted.

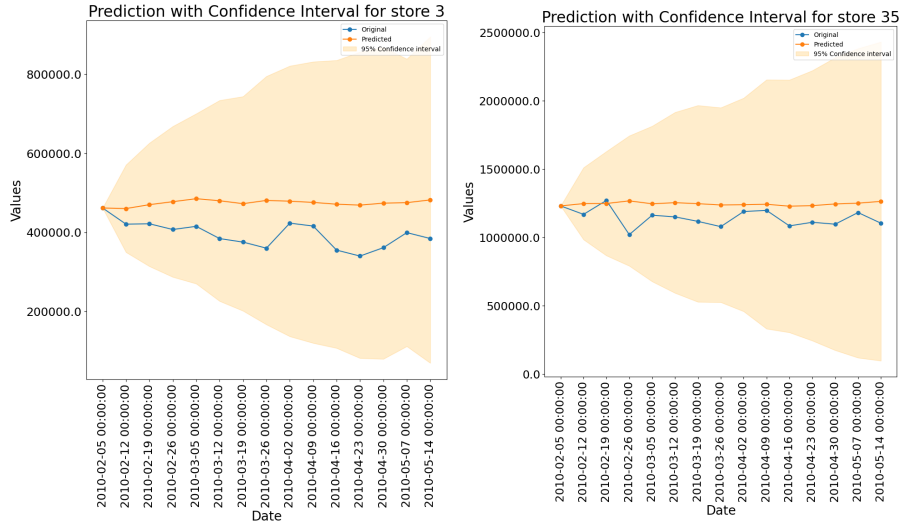


Figure 3.7: Confidence intervals are generated by the SDE basic model for stores found only in the test set with both original sales values (blue) and mean of the 100 trajectories (orange) plotted.

the plots. As seen in the figure, for some stores, the confidence interval includes most realistic positive sales values starting with zero dollars. Such a model, that predicts that most reasonable positive value are plausible, is not particularly useful despite the improvement in the point forecast errors and the log likelihood.
Once again, the confidence interval does not reflect the structures in the original data. The generated mean of the trajectories is no longer constant nor featureless. Thus, this problem persists even with the use of the full data.
The representation of the data given to the model is one explanation of why the model is not confident in its predictions. The current representation consists of the sales data of a specific week and the sales data of the week before. Such a representation only gives a narrow glance into the history of the weekly sales before a specific week. Consequently, the model is not confident about its predictions.

### 3.5.4 Time-delay Embedding

Since the basic SDE model is not confident in its predictions, time-delay embeddings was introduced to offer more history. With many experiments carried out using time-delay embeddings, the length of the vector was varied from two to eight weeks. Out of all the variations, the four weeks produced the best results. Consequently, most of

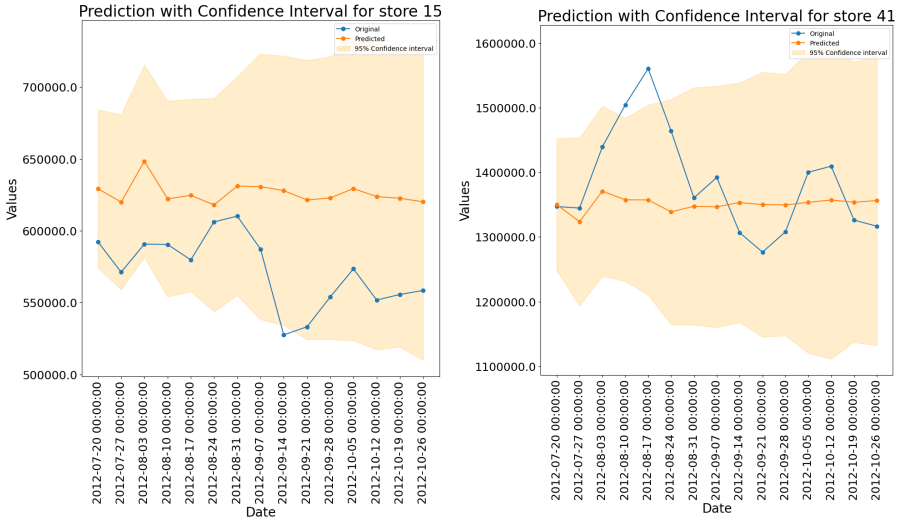


Figure 3.8: Confidence intervals are generated by the 4-week time-delay embedding model for stores found only in all three sets with both original sales values (blue) and mean of the one hundred trajectories (orange) plotted.

the focus of this subsection will be on the results of the 4-week time-delay embedding. Towards the end of the subsection, results of the other length variations will be briefly presented.

As presented by Table 3.4, the 4-week time-delay embedding resulted in a MSE of 5,820,000,000.00, a RMSE of 64,131.58, a MAE of 51,305.31, a log-likelihood of -11.68, a coverage percentage of 92.74%, and a standard deviation of 67,128.35. In addition to the overall point and probabilistic forecast metrics, Table 3.5 and Table 3.7 report the average metrics for the stores found in all sets and in the test set only, respectively. Figure 3.8, Figure 3.9 and Figure 3.10 illustrate generated trajectories and confidence intervals for stores found in all three sets, stores found only in the validation and test sets and stores seen only in the test set, respectively. As can be seen from the plots, an improvement in the accuracy and confidence in the generated results can be noticed. Furthermore, the reported metrics are superior to those of the previous SDE experiments. Similar to the Prophet model and unlike the previous SDE models, this model maintains it confidence, with the width of the confidence interval remaining the same throughout the time series.

In comparison to the Prophet model, the best performing baseline model so far, the model performs relatively well. The current SDE model with 4-week time-delay embedding produces less point forecast errors and is more confident. However, when

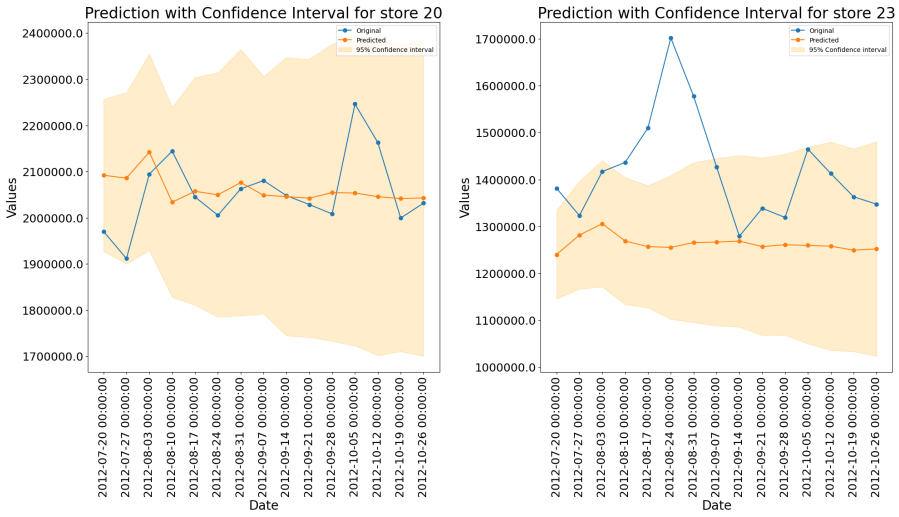


Figure 3.9: Confidence intervals are generated by the 4-week time-delay embedding model for stores found only in the validation and test sets with both original sales values (blue) and mean of the one hundred trajectories (orange) plotted.

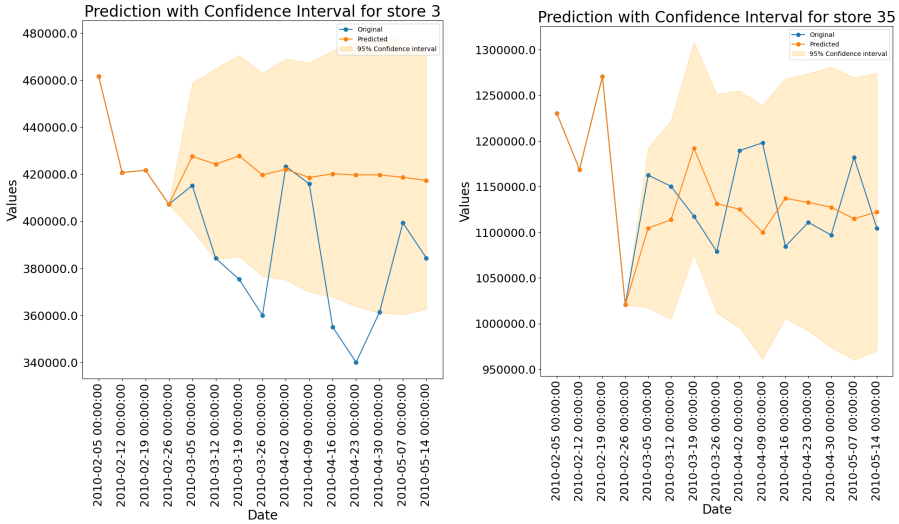


Figure 3.10: Confidence intervals are generated by the 4-week time-delay embedding model for stores found only in the test set with both original sales values (blue) and mean of the 100 trajectories (orange) plotted.

it comes to the generated results exhibiting structures found in the data, this current SDE model still fails to achieve that. Thus, the mean of the trajectories is still fairly constant and featureless. Consequently, there is still room for improvement.
Experiments, where the length of the embedding was either greater or smaller than four weeks, resulted in inferior performance. Increasing the length of the embedding increased the model loss when estimating the drift and diffusion parameters. Hence the generated confidence interval and mean of the trajectories were less accurate and confident. This was especially the case with 7 and 8-week embeddings, where the generated confidence interval is extremely narrow and does not cover the original sales values wells. As it will be demonstrated in later sections, standard scaling mitigates this problem and allows increasing the size of the embedding to sizes much larger than eight weeks, such as 52 and 78 weeks. Likewise, decreasing the size of the embedding below four weeks resulted in inferior results and hence a 4-week period was concluded to be ideal.

### 3.5.5 Combining Extracted Features with Time-delay Embedding

In this subsection, the results of attempting to improve on the previous time-delay embedding model are presented. Specifically, the results of using the months of the sale with a 4-week embedding is outlined. Results of using other features will be reported in the following subsection, when the result of using standard scaling is presented.

As illustrated in Table 3.4, the use of months with a 4-week time-delay embedding resulted in a MSE of 11,000,000,000.00, a RMSE of 90,680.08, a MAE of 78,164.28, a log-likelihood of -12.10, a coverage percentage of 99.85%, and a standard deviation of 158,959.50. Figure 3.11, Figure 3.12 and Figure 3.13 illustrate generated trajectories and confidence intervals for stores found in all three sets, stores found only in the validation and test sets and stores seen only in the test set, respectively. Visually, we can observe that the model is less confident than when using a 4-week time-delay embedding without any extracted features. This is additionally confirmed when comparing the probabilistic and point forecast metrics of the two approaches. However, it is worth noting that the results are still improvement to both the SDE baseline and the basic SDE model.
One explanation of why using extracted features did not result in an improved model is the difference between the scales of the different features. While the weekly sales are on an order of hundred thousands and millions, the values of the month and days are in the form of one or two digits. In the next subsection, a solution to this problem is illustrated, when the results of applying standard scaling on the time-delay embeddings is presented.
The results of using the extracted features in conjunction with the basic model are not

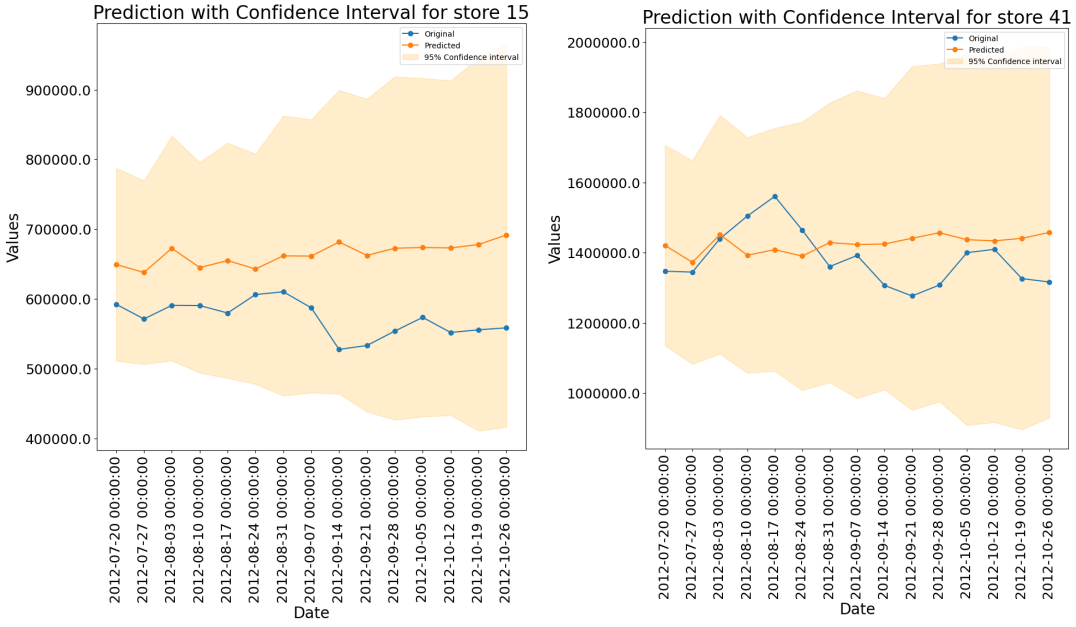


Figure 3.11: Confidence intervals are generated by the 4-week time-delay embedding model with the months feature for stores found only in all three sets with both original sales values (blue) and mean of the one hundred trajectories (orange) plotted.

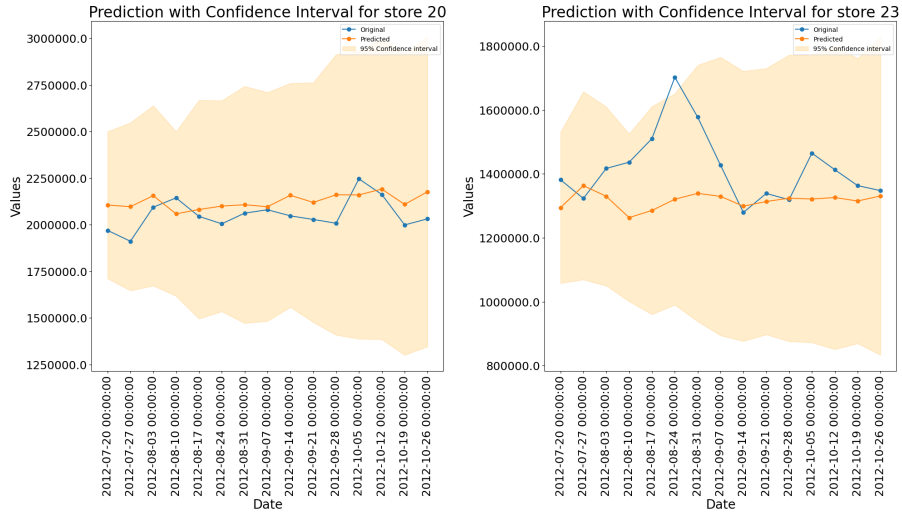


Figure 3.12: Confidence intervals are generated by the 4-week time-delay embedding model with the months feature for stores found only in the validation and test sets with both original sales values (blue) and mean of the one hundred trajectories (orange) plotted.

shown from this section. Combining these features with the basic model resulted in extremely large confidence intervals that cover a huge range of sales values, making them not useful for probabilistic forecasting. For similar reasons, results of utilizing other features such as holidays and markdown events are also not shown.

### 3.5.6 Applying Scaling and Transformations to Time-delay Embedding Data

In this subsection, the results of the experiments, where various transformations were applied to time-delay embedding, are reported. First, the results of applying Box Cox transformation, to make the possibly non-normal data more normally distributed, are presented. The results of using both extracted features and weekly sales values along with standard scaling are reported. Next, the results of increasing the size of the time-delay embedding with standard scaling applied, are reported. The results of increasing the size embedding with the best performing extracted feature are first reported followed by that of increasing the time-delay embedding with weekly sales only.

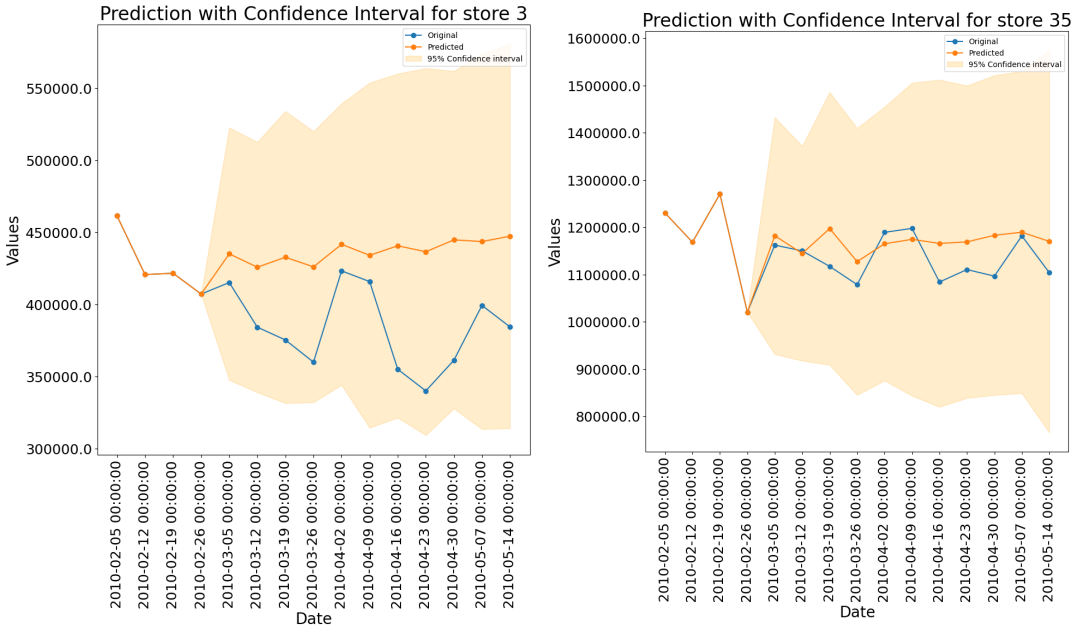


Figure 3.13: Confidence intervals are generated by the 4-week time-delay embedding model with the months feature for stores found only in the test set with both original sales values (blue) and mean of the one hundred trajectories (orange) plotted.

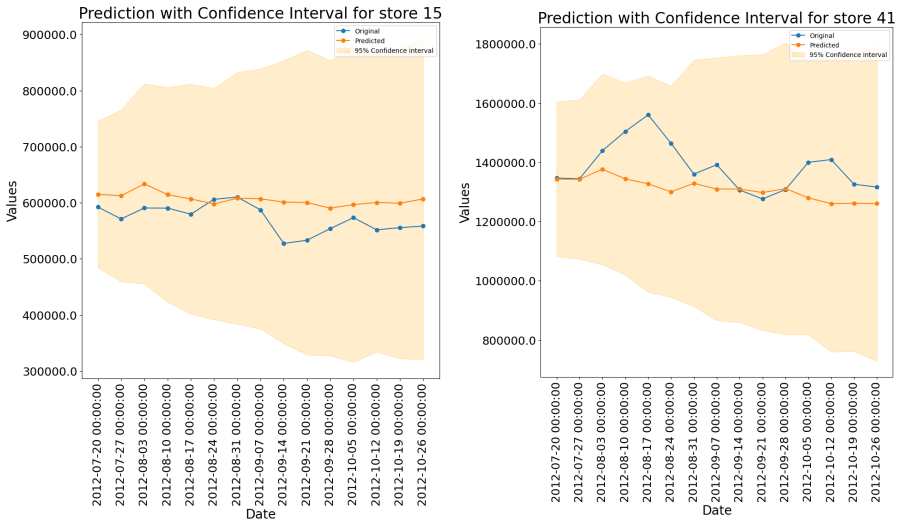


Figure 3.14: Confidence intervals are generated by the 4-week time-delay embedding model where Box Cox transformation is applied for stores found only in all three sets with both original sales values (blue) and mean of the one hundred trajectories (orange) plotted.

### Applying Box Cox Transformation

As illustrated by Table 3.4, applying Box Cox transformation to a 4-week time-delay embedding resulted in a MSE of 7,680,000,000.00, a RMSE of 74,068.24, a MAE of 60,859.34, a log-likelihood of -11.99, a coverage percentage of 99.85%, and a standard deviation of 169,730.40. Figure 3.14, Figure 3.15 and Figure 3.16 illustrate generated trajectories and confidence intervals for stores found in all three sets, stores found only in the validation and test sets and stores seen only in the test set, respectively.
Based on the plots and the computed metrics, it can be observed that, while the results are satisfactory, the model did not surpass the Prophet baseline, which is our main target. Furthermore, the constructed model did not surpass the best performing SDE model so far. Therefore, it was concluded that applying Box Cox transform is not beneficial in improving the performance.

### Using the Months, Day, Number of Week and Jump Meta Data Features with Standard Scaling

As illustrated in Table 3.4, applying standard scaling to a 4-week embedding of months and weekly sales resulted in a MSE of 10,400,000,000.00, a RMSE of 86,053.40, a MAE

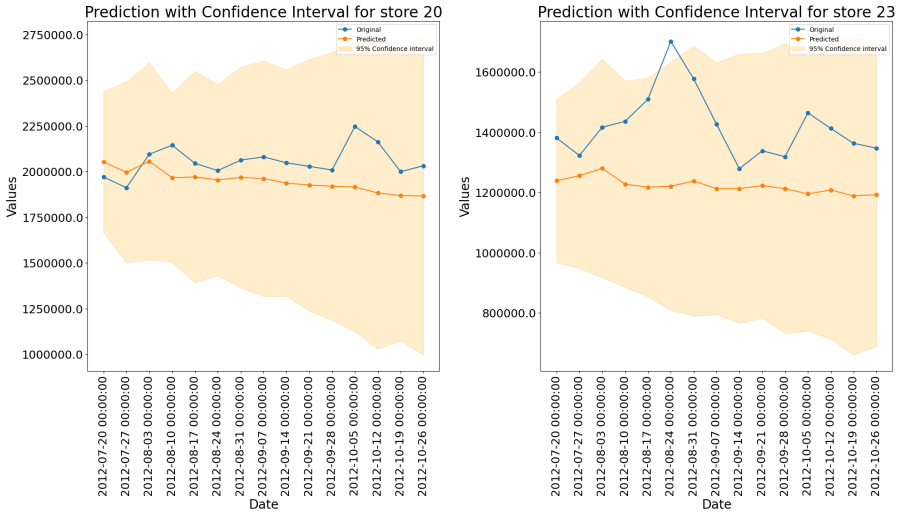


Figure 3.15: Confidence intervals are generated by the 4-week time-delay embedding model, where the Box Cox transformation is applied, for stores found only in the validation and test sets with both original sales values (blue) and mean of the one hundred trajectories (orange) plotted.

of 73,451.65, a log-likelihood of -12.20, a coverage percentage of 100%, and a standard deviation of 191,103.40. Figure 3.17, Figure 3.18 and Figure 3.19 illustrate generated trajectories and confidence intervals for stores found in all three sets, stores found only in the validation and test sets and stores seen only in the test set, respectively.
There is a noticeable improvement when applying standard scaling. Scaling allowed for the introduction of additional features to the model without an extreme deterioration in the generated confidence intervals. However, it is worth noting that, despite showing an improved performance in comparison to not applying it, standard scaling does not outperform the best performing SDE model.
Other experiments were performed, where the day of the month, number of the week and the jump meta data were used. Additionally, combinations of the extracted features were used together. However, the results of the remaining experiments are not discussed in this subsection since they do not exceed those of using the month of the sales.

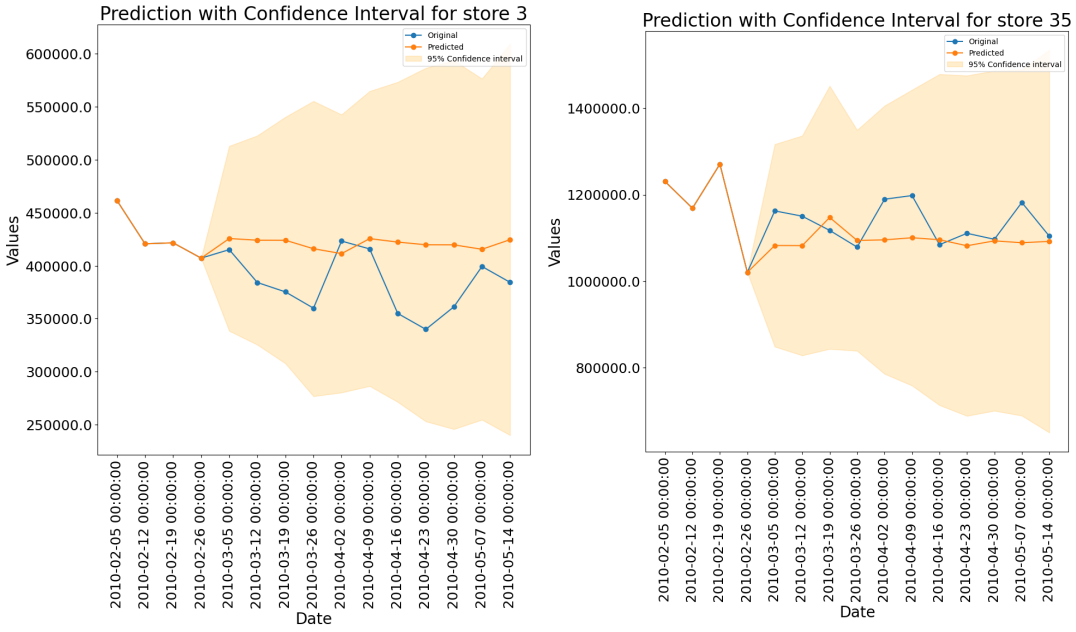


Figure 3.16: Confidence intervals are generated by the 4-week time-delay embedding model, where the Box Cox transformation is applied, for stores found only in the test set with both original sales values (blue) and mean of the one hundred trajectories (orange) plotted.

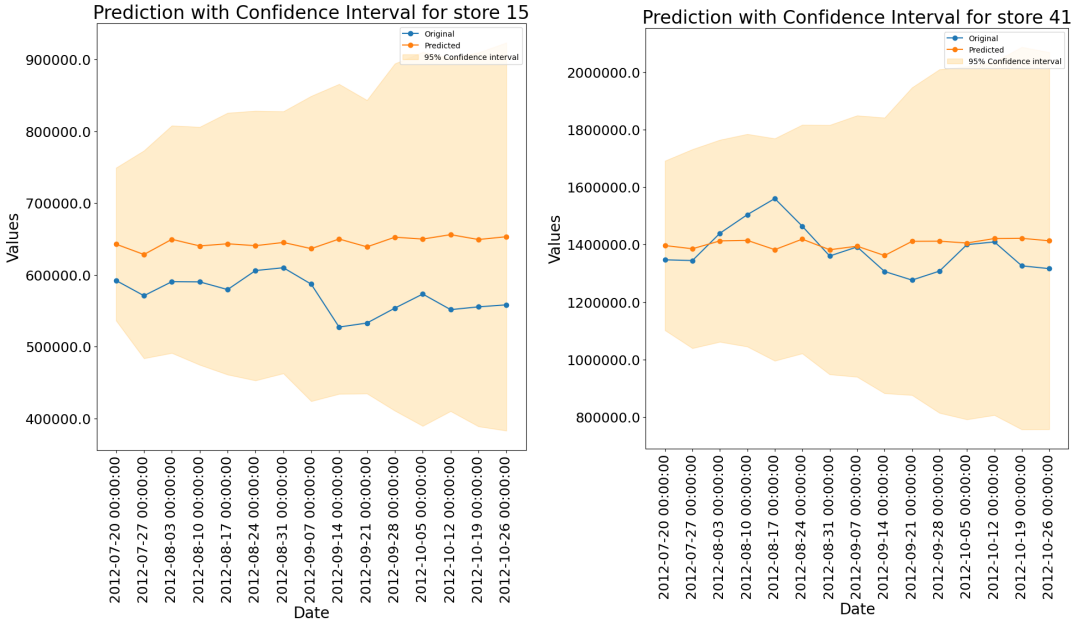


Figure 3.17: Confidence intervals are generated by the 4-week time-delay embedding model, where standard scaling is applied, for stores found only in all three sets with both original sales values (blue) and mean of the one hundred trajectories (orange) plotted.

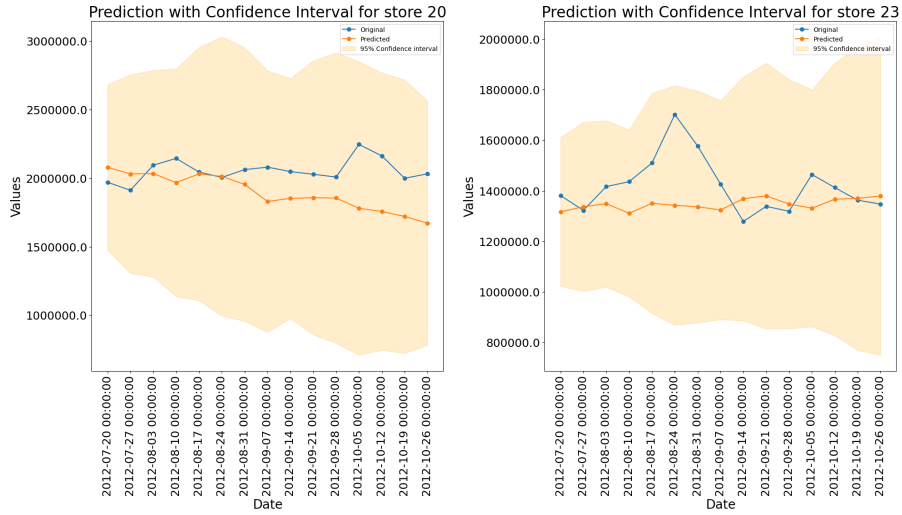


Figure 3.18: Confidence intervals are generated by the 4-week time-delay embedding model, where standard scaling is applied, for stores found only in the validation and test sets with both original sales values (blue) and mean of the one hundred trajectories (orange) plotted.

### Increasing the Length of the Time-delay Embedding with the Month Feature with Standard Scaling

Since the usage of scaling did not surpass the best performing model, one possibility was to investigate how the increasing of the time-delay embedding and incorporating more history would affect the performance. The time-delay embedding, whose length was increased, contained both weekly sales and months features.
The experiments were performed based on the hypothesis that standard scaling would allow increasing the window size without a deterioration of the results unlike increasing it without applying any feature scaling. This hypothesis was confirmed with the results of these experiments. When experimenting with five to eight weeks, there was still a slight deterioration in the results. However, unlike before, the deterioration did not progress with the increasing of the window size.
This conclusion encouraged a larger increase in the embedding size. An attempt to use 16,24 and 52-week time-delay embedding was made. One key conclusion that was increasing the embedding size results in a mean of the trajectories that is no longer constant. Thus, scaling allowed for increase of the embedding which in turn allowed the mean of the trajectories to represent the structures of the original time series, as it

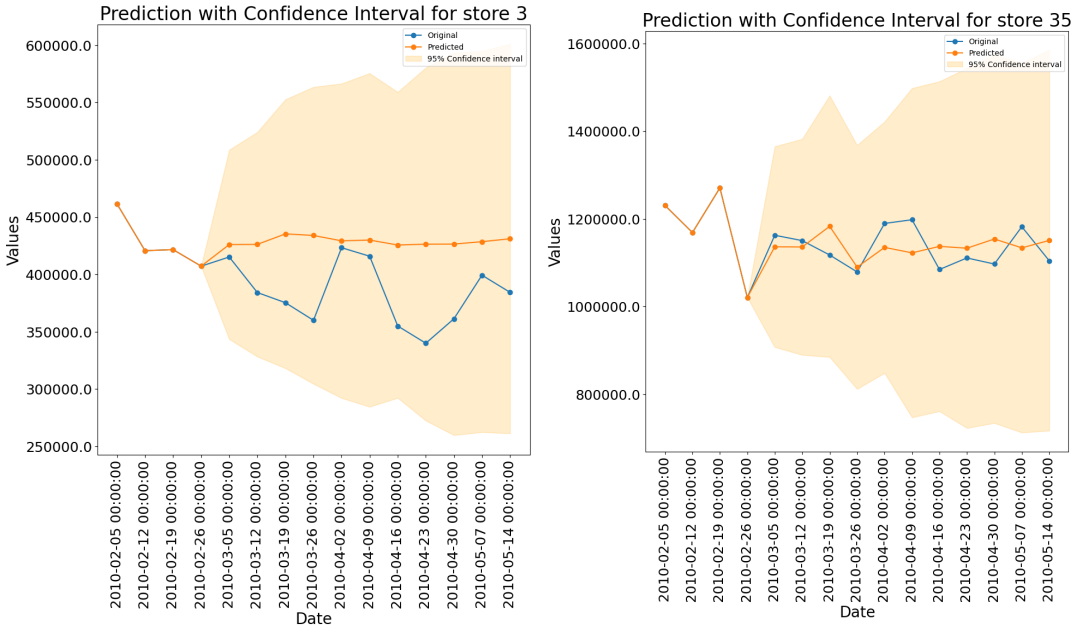


Figure 3.19: Confidence intervals are generated by the 4-week time-delay embedding model, where the standard scaling is applied, for stores found only in the test set with both original sales values (blue) and mean of the one hundred trajectories (orange) plotted.

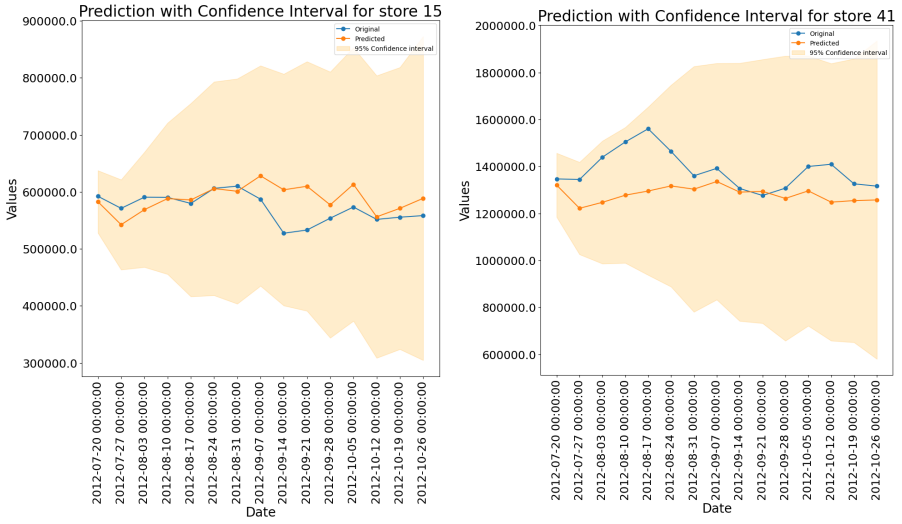


Figure 3.20: Confidence intervals are generated by the 52-week time-delay embedding model, where the months feature is used, for stores found only in all three sets with both original sales values (blue) and mean of the one hundred trajectories (orange) plotted.

is the case with Prophet. The more the embedding was increased, the more structures were evident.

The focus will be on the best results. Therefore, only the results of applying standard scaling to 52-week embedding of weekly sales and months are reported. As illustrated in Table 3.4, the approach reported a MSE of 55,020,000,000, a RMSE of 192,332.2, a MAE of 175,431.72, a log-likelihood of -13.55, a coverage percentage of 93.90%, and a standard deviation of 245,515.20. Figure 3.20 and Figure 3.21 illustrate generated trajectories and confidence intervals for stores found in all three sets and stores found only in the validation and test sets, respectively.

### Increasing the Length of Embedding using Only Sales Values with Standard Scaling

With the observation that increasing the time-delay embedding to include approximately one year of data resulted in better representation of the structure of the original time series, standard scaling was applied to a 52-week embedding of weekly sales only. Since using weekly sales without any extracted features always resulted in improved performance, the hypothesis was that removing the months from the scaled 52-week embedding might result in more confident and accurate confidence intervals that better

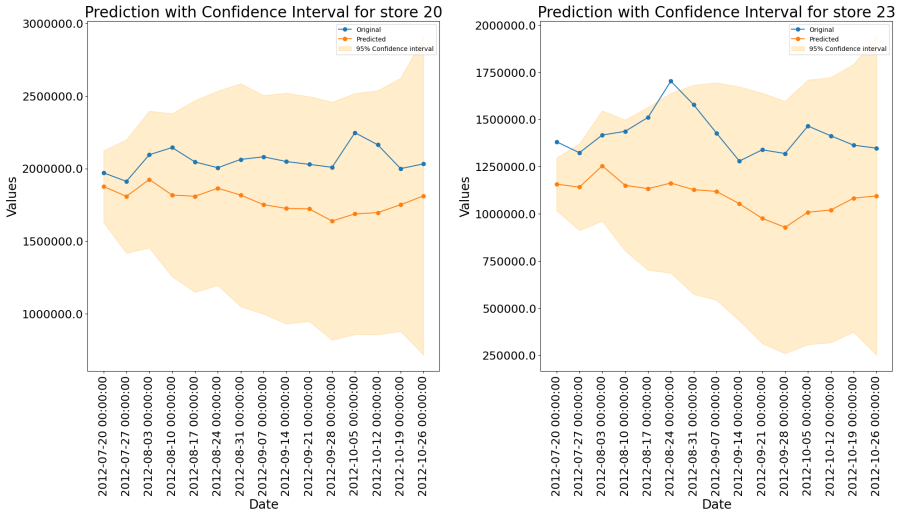


Figure 3.21: Confidence intervals are generated by the 52-week time-delay embedding model, where the months feature is used, for stores found only in the validation and test sets with both original sales values (blue) and mean of the one hundred trajectories (orange) plotted.

grasp the structure of the original weekly sales values.

This hypothesis was confirmed with the results obtained from the experiment. As illustrated by Table 3.4, the use of months with a 52-week time-delay embedding resulted in a MSE of 20,100,000,000, a RMSE of 130,073.20, a MAE of 109,734.26, a log-likelihood of -13.18, a coverage percentage of 86.41%, and a standard deviation of 105,143.97. Figure 3.22 and Figure 3.23 illustrate generated trajectories and confidence intervals for stores found in all three sets and stores found only in the validation and test sets, respectively. Visually, most of the main structures in the original time series are reflected in the generated plots. The mean of the trajectories is no longer constant. The amount of structures are present in the confidence interval depends on which store data is used. Additionally, it is also observed that, sometimes, the correct structure is predicted by the model however during a slightly wrong week. The model would correctly predict an increase in sales and correctly predict the magnitude of the increase. However, the increase of sales would be predicted a few weeks before or after its actual occurrence. Figure 3.24 illustrates some best case examples observed by the model. These plots illustrate the best confidence intervals generated by the model and demonstrate its peak performance. Therefore, these results are now comparable to the Prophet model, which represents the baseline model that we are attempting to surpass

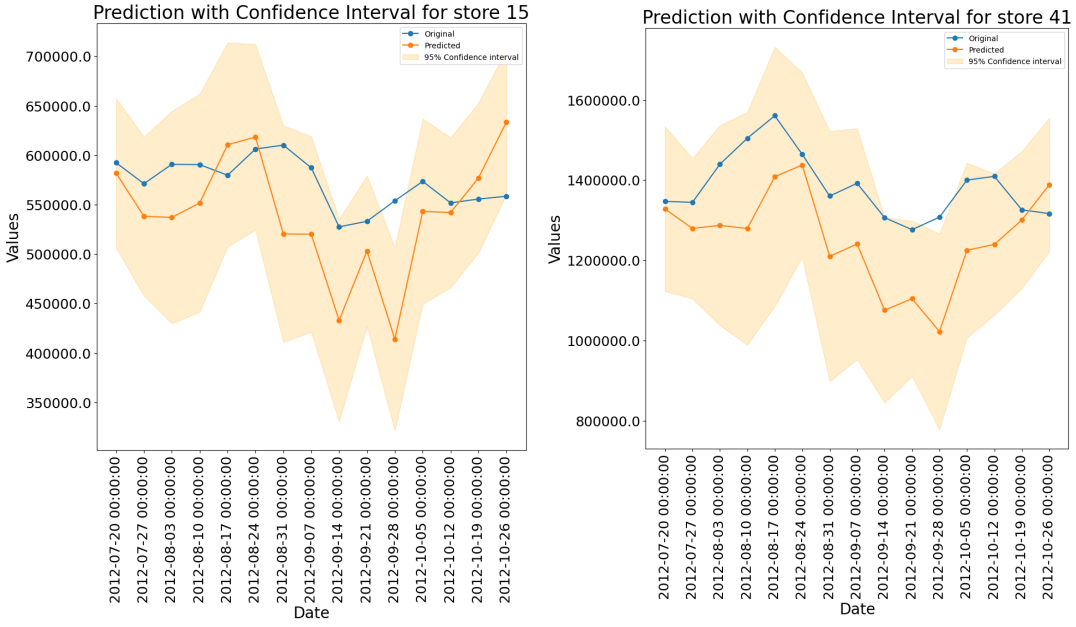


Figure 3.22: Confidence intervals are generated by the 52-week time-delay embedding model, where the months feature is not used, for stores found only in all three sets with both original sales values (blue) and mean of the one hundred trajectories (orange) plotted.

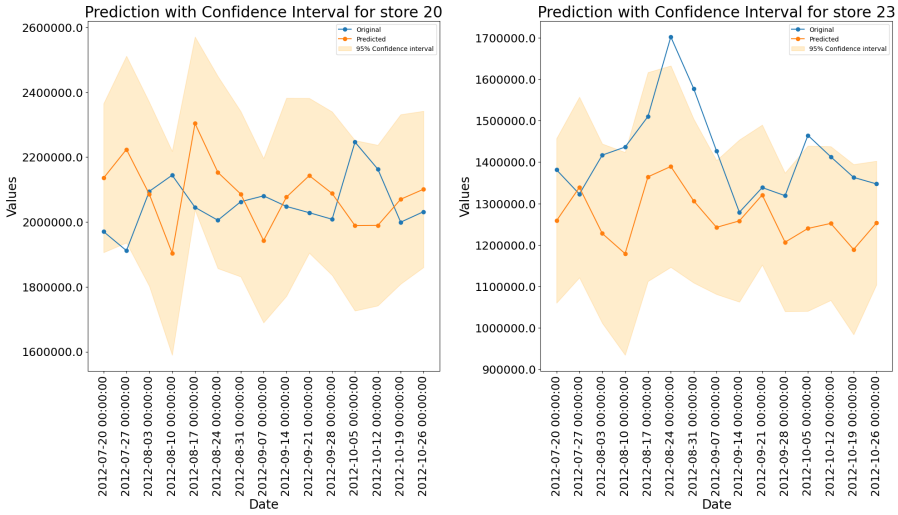


Figure 3.23: Confidence intervals are generated by the 52-week time-delay embedding model, where the months feature is not used, for stores found only in the validation and test sets with both original sales values (blue) and mean of the one hundred trajectories (orange) plotted.

or at least match.
When one observes the reported point and probabilistic forecast metrics, one can conclude that the model is performing well in terms of accuracy, in terms of RMSE and MAE, and probabilistically, in terms of width of the confidence interval and log likelihood. However, it is worth noting that the performance metrics are still a bit inferior to those of the 4-week weekly sales embedding without any scaling applied. This might seem counter-intuitive at first glance considering that the 4-week embedding failed to grasp structures in the original time series. However, two explanations can be given to this. First, regarding the deterioration of the point-wise forecasts, the shift in the structures between the mean of the trajectories and the original values, outlined above, increases the error. Regarding the probabilistic performance, the model could be less confident due to the large number of features introduced by the embedding.
It is worth noting that an attempt to use a 78-week embedding was also made. However, while they still pertain a lot of the structure in the original time series, results were inferior to those of the 52 weeks. This could be a result of using less training data due to having longer embeddings.

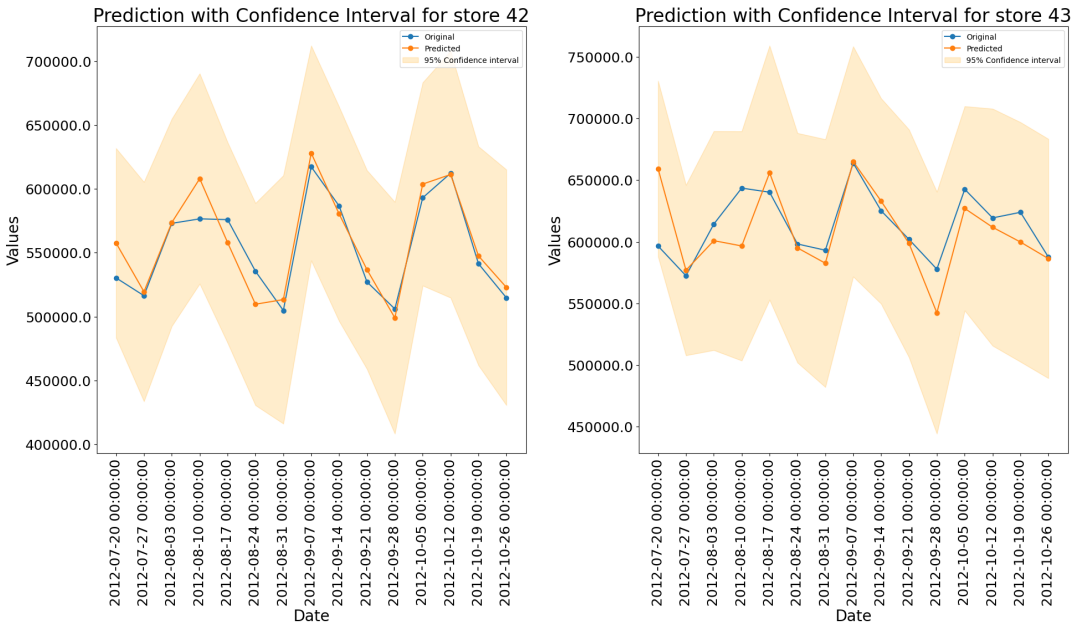


Figure 3.24: Best case examples of confidence intervals by the 52-week time-delay embedding model, where the months feature is not used, are presented with both original sales values (blue) and mean of the one hundred trajectories (orange) plotted.

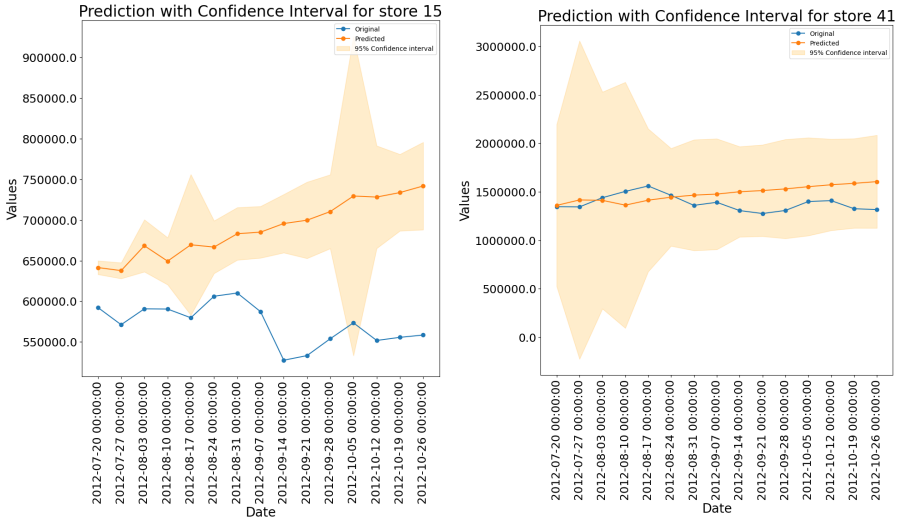


Figure 3.25: Confidence intervals are generated by the model, where the modified tool is used, for stores found only in all three sets with both original sales values (blue) and mean of the one hundred trajectories (orange) plotted.

### 3.5.7 Using the Modified Tool for Lévy Process

As illustrated by Table 3.4, the model created using the modified tool via a Cauchy distribution resulted in a MSE of 27,097,314,173, a RMSE of 143,943.59, a MAE of 121,586.10, a log-likelihood of -16.37, a coverage percentage of 65.19%, and a standard deviation of 172,177.08. Figure 3.25, Figure 3.26 and Figure 3.27 illustrate generated trajectories and confidence intervals for stores found in all three sets, stores found only in the validation and test sets and stores seen only in the test set, respectively. Visually, the generated results look much inferior to the previous experiments.
In terms of point forecast metrics, the RMSE and MAE indicate a significant error between the original sales values and the mean of the trajectories. Probabilistically, the model reports the worst log-likelihood of all the experiments. Moreover, the percentage of original weekly sales covered by the confidence interval is particularly low. Since the overall results are particularly poor, it was concluded that the modified SDE tool does not help in this particular use case.

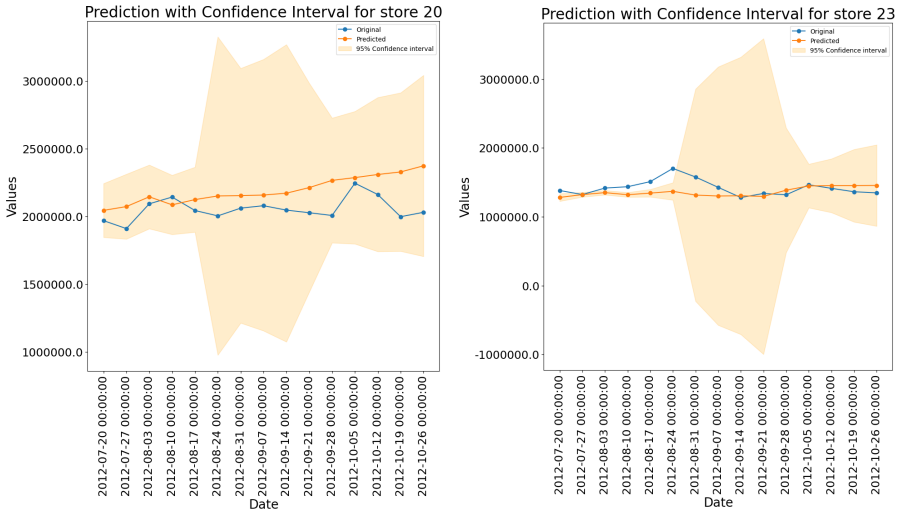


Figure 3.26: Confidence intervals are generated by the model, where the modified tool is used, for stores found only in the validation and test sets with both original sales values (blue) and mean of the one hundred trajectories (orange) plotted.

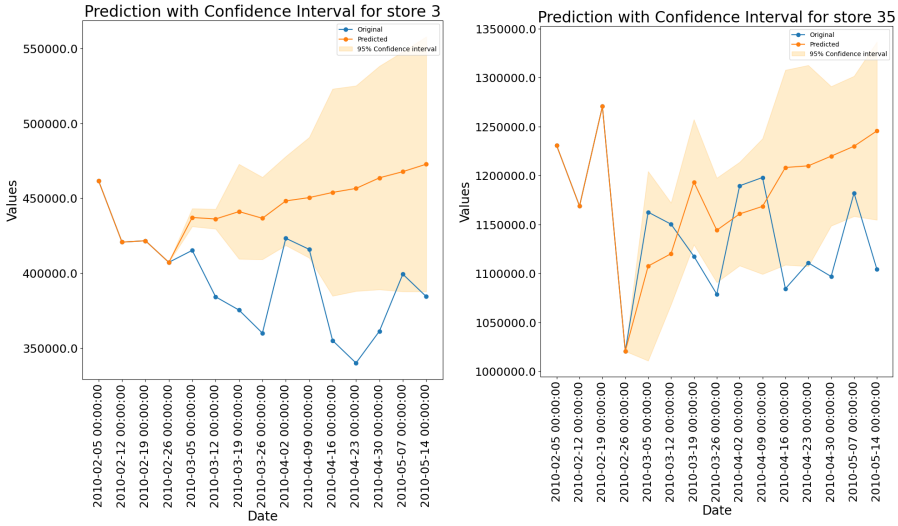


Figure 3.27: Confidence intervals are generated by the model, where the modified tool is used, for stores found only in the test set with both original sales values (blue) and mean of the one hundred trajectories (orange) plotted.

# 4 Conclusions

In conclusion, this thesis investigated the use of SDEs in probabilistic forecasting. In Section 1, the underlying motivation behind probabilistic and interpretable forecasting is presented. The reasoning behind the use of SDE in financial forecasting is additionally clarified. In Section 2, all relevant background to the subsequent investigation is introduced. First, a mathematical introduction of SDEs along with a review of some commonly used models in various financial forecasting application is given. Furthermore, deep learning tools to approximate SDEs are presented. Subsequently, probabilistic forecasting in financial applications is discussed with a focus on the Prophet model and previous studies. Section 2 is concluded with a survey of studies investigating the use of SDEs in probabilistic forecasting. Section 3 presents the work contributed by this thesis. The section begins with the dataset used in addition to data preparation steps carried out on the data. The neural network-based tool used to approximate SDEs is subsequently discussed thoroughly. Additionally an explanation of all significant experiments conducted is presented. The section is concluded with the point-forecast and probabilstic metrics used to evaluate the experiments along with the results.
While many models with different variations were experimented with, only two models stand out as plausible solutions for the weekly sales use case when compared to the constructed baselines. While the two models have their strengths and weaknesses, both surpass the performance of the SDE baseline and provide promising results. On the one hand, the first model, which uses a 4-week time delay embedding of the weekly sales, represents an accurate model with a confidence interval that is not extremely wide. However, like most of the experimented models, the mean of the confidence interval is constant and does not capture increases nor decreases that occur in the actual sales values. On the other hand, the second model, which uses standard scaled 52-week time delay embedding of the weekly sales, showed extremely good performance at grasping the structures of the original time series. The model is also highly accurate when compared to the many other variations. Additionally, it has a realistic uncertainty estimate. However, it is slightly less accurate, when compared to the 4-week time delay embedding model, and also has a slightly wider confidence interval.
The thesis was attempting to answer two research questions. The first was whether an interpretable probabilistic solution can generate accurate forecasts with realistic

uncertainty estimates. The second research question was interested in comparing the performance of the SDE solution to state of the art solutions. As an answer to the first question, the resultant model is accurate and has a realistic uncertainty estimate. The solution is also able to capture structures in the time series even if that entails a slight sacrifice in the accuracy and confidence. The minor sacrifice in accuracy can be attributed to too many features that are a result of extremely large embeddings. Therefore, the probabilistic solution is usable, regardless of which of the two models is used. However, whether the solution is interpretable is questionable. Considering that neural networks are being used to represent the drift and diffusion components, the underlying model can be argued to be a black box even if the parameters computed are of an interpretable equation. Therefore, if interpretability is a hard requirement, the use of a linear regression model or other forms of interpretable models, to approximate the SDE parameters, would be more ideal. Regarding the second question, the overall solution is comparable to Prophet and has a narrower confidence interval. However, as previously stated, the two models have their strengths and weaknesses. The first model is more accurate than the Prophet models, although it does not grasp important structure in the data. The second model achieves comparable performance to Prophet in terms of capturing structure. It makes more mistakes than the first model as a compromise.

Regarding future work and outlook, there are few directions in which the work of this thesis can be built on and expanded. First, since large embeddings result in a greater loss by the model, one direction would be to decrease the number of features used by the model by using a more condensed form of the 52-week time delay embedding. The condensed form would include the most recent 4 weeks, skip the next few weeks, include the subsequent week to those skipped and so forth. Another direction, in which future work could go, is to modify the underlying SDE tool to better suit the use case. One modification would be to integrate the training of models other than the neural network-based Gaussian process. By inspecting the losses of the model, it seems plausible that simpler, and more interpretable, models such as linear regression could be sufficient for the weekly sales forecast use case. Therefore, modifying the tool to allow for the option of training of linear regression and other simpler models would be beneficial. Moreover, the way the simulated trajectories are generated can be modified to better suit the use case. With the current implementation, the model learns drift and diffusion values for all the features provided in the time delay embedding. Using these learned parameters, one generates trajectories for all the features. However, in the weekly sales use case, generating trajectories solely for the weekly sales values is sufficient. All of the remaining features should merely be control variables used to obtain more accurate trajectories. This modification could decrease the loss obtained when predicting the sales values, since the model is learning the drift and diffusion

terms for less features. In summary, making these modifications to the tool in addition to using condensed time delay embeddings constitute a good starting point for future work.

# Abbreviations

**ARIMA** Autoregressive Integrated Moving Average

**ELU** Exponential Linear Unit

**ETS** Exponential Smoothing

**GARCH** Generalized AutoRegressive Conditional Heteroskedasticity

**GBM** Geometric Brownian Motion

**GBT** Gradient Boosting Trees

**LSTM** Long Short Term Memory

**MAE** Mean Absolute Error

**MAPE** Mean Absolute Percentage Error

**MSE** Mean Squared Error

**NB** Negative Binomial

**NBEATS** Neural Basis Expansion Analysis

**Pois** Poisson Theoretical Estimation

**QEE** Quantile Empirical Estimation

**RF** Random Forest

**RMSE** Root Mean Squared Error

**SARIMA** Seasonal Auto-Regressive Integrated Moving Average

**SDE** Stochastic differential equations

**sNaive** Seasonal Naive

**STL** Seasonal-Trend decomposition using LOESS

# List of Figures

# List of Tables

# Bibliography


[Bad+17] J. Badosa, E. Gobet, M. Grangereau, and D. Kim. "Day-ahead probabilistic forecast of solar irradiance: a Stochastic Differential Equation approach." In: *Forecasting and risk management for renewable energy*. Springer, 2017, pp. 73–93.

[BHW20] L. R. Berry, P. Helman, and M. West. "Probabilistic forecasting of heterogeneous consumer transaction–sales time series." In: *International Journal of Forecasting* 36.2 (2020), pp. 552–569.

[Boy22] M. Boyles. *7 Financial Forecasting Methods to Predict Business Performance*. `https://online.hbs.edu/blog/post/financial-forecasting-methods`. Accessed on June 1, 2024. 2022.

[BT11] G. E. Box and G. C. Tiao. *Bayesian inference in statistical analysis*. John Wiley & Sons, 2011.

[Cas16] D. Castelvecchi. "Can we open the black box of AI?" In: *Nature News* 538.7623 (2016), p. 20.

[CPW21] P. Chatigny, J.-M. Patenaude, and S. Wang. "Spatiotemporal adaptive neural network for long-term forecasting of financial time series." In: *International Journal of Approximate Reasoning* 132 (2021), pp. 70–85.

[De 06] M. De Lara. "On drift, diffusion and geometry." In: *Journal of Geometry and Physics* 56.8 (2006), pp. 1215–1234.

[Die+23] F. Dietrich, A. Makeev, G. Kevrekidis, N. Evangelou, T. Bertalan, S. Reich, and I. G. Kevrekidis. "Learning effective stochastic differential equations from microscopic simulations: Linking stochastic numerics to deep learning." In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 33.2 (2023).

[DNP14] E. Dobrescu, D.-I. Nastac, and E. Pelinescu. "Short-term financial forecasting using ANN adaptive predictors in cascade." In: *International Journal of Process Management and Benchmarking* 4.4 (2014), pp. 376–405.

[Elk17] S. Elkantassi. "Probabilistic forecast of wind power generation by stochastic differential equation models." PhD thesis. 2017.

[FB24] A. Ferenczi and C. Bădică. "Prediction of Ethereum gas prices using DeepAR and probabilistic forecasting." In: *Journal of Information and Telecommunication* 8.1 (2024), pp. 18–32.

[Haz23] J. Hazarika. *Identification of Stochastic Differential Equation with Lévy Noise*. Guided research conducted at the Technical University in Munich. 2023.

[Inv24] Investopedia. *GARCH Model: Definition and Uses in Statistics*. `https://www.investopedia.com/terms/g/garch.asp`. Updated August 07, 2024. Reviewed by Erika Rasure. Fact checked by Vikki Velasquez. Aug. 2024.

[Ive+14] E. B. Iversen, J. M. Morales, J. K. Møller, and H. Madsen. "Probabilistic forecasts of solar irradiance using stochastic differential equations." In: *Environmetrics* 25.3 (2014), pp. 152–164.

[Ive+16] E. B. Iversen, J. M. Morales, J. K. Møller, and H. Madsen. "Short-term probabilistic forecasting of wind speed using stochastic differential equations." In: *International Journal of Forecasting* 32.3 (2016), pp. 981–990.

[Jun+20] S. Jung, K.-M. Kim, H. Kwak, and Y.-J. Park. "A Worrying Analysis of Probabilistic Time-series Models for Sales Forecasting." In: *Proceedings on "I Can't Believe It's Not Better!" at NeurIPS Workshops*. Ed. by J. Zosa Forde, F. Ruiz, M. F. Pradier, and A. Schein. Vol. 137. Proceedings of Machine Learning Research. PMLR, Dec. 2020, pp. 98–105.

[Kag14] Kaggle. *Walmart Recruiting - Store Sales Forecasting*. `https://www.kaggle.com/c/walmart-recruiting-store-sales-forecasting/data`. Accessed: June 7, 2024. 2014.

[KK16] O. Kramer and O. Kramer. "Scikit-learn." In: *Machine learning for evolution strategies* (2016), pp. 45–53.

[KLK22] A. Karanikola, C. M. Liapis, and S. Kotsiantis. "A comparison of contemporary methods on univariate time series forecasting." In: *Advances in Machine Learning/Deep Learning-based Technologies: Selected Papers in Honour of Professor Nikolaos G. Bourbakis–Vol. 2* (2022), pp. 143–168.

[Klo+92] P. E. Kloeden, E. Platen, P. E. Kloeden, and E. Platen. *Stochastic differential equations*. Springer, 1992.

[MZM16] J. K. Møller, M. Zugno, and H. Madsen. "Probabilistic forecasts of wind power generation by stochastic differential equation models." In: *Journal of Forecasting* 35.3 (2016), pp. 189–205.

[NP11] A. Neisy and M. Peymany. "Financial modeling by ordinary and stochastic differential equations." In: *World Applied Sciences Journal* 13.11 (2011), pp. 2288–2295.

[Oks13] B. Oksendal. *Stochastic differential equations: an introduction with applications*. Springer Science & Business Media, 2013.

[Pal96] F. C. Palm. "7 GARCH models of volatility." In: *Handbook of statistics* 14 (1996), pp. 209–240.

[Pan+02] J.-X. Pan, K.-T. Fang, J.-X. Pan, and K.-T. Fang. "Maximum likelihood estimation." In: *Growth curve models and statistical diagnostics* (2002), pp. 77–158.

[Ran+18] S. S. Rangapuram, M. W. Seeger, J. Gasthaus, L. Stella, Y. Wang, and T. Januschowski. "Deep state space models for time series forecasting." In: *Advances in neural information processing systems* 31 (2018).

[Run03] W. J. Runggaldier. "Jump-diffusion models." In: *Handbook of heavy tailed distributions in finance*. Elsevier, 2003, pp. 169–209.

[Rup04] D. Ruppert. "GARCH Models." In: *Statistics and Finance: An Introduction*. New York, NY: Springer New York, 2004, pp. 363–395. ISBN: 978-1-4419-6876-0. DOI: 10.1007/978-1-4419-6876-0_12.

[Sak92] R. M. Sakia. "The Box-Cox transformation technique: a review." In: *Journal of the Royal Statistical Society Series D: The Statistician* 41.2 (1992), pp. 169–178.

[Sal+20] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski. "DeepAR: Probabilistic forecasting with autoregressive recurrent networks." In: *International journal of forecasting* 36.3 (2020), pp. 1181–1191.

[Sha21] S. Shah. "Comparison of Stochastic Forecasting Models." In: (2021).

[Sim01] D. Simon. "Kalman filtering." In: *Embedded systems programming* 14.6 (2001), pp. 72–79.

[Spi+21] E. Spiliotis, S. Makridakis, A. Kaltsounis, and V. Assimakopoulos. "Product sales probabilistic forecasting: An empirical evaluation using the M5 competition data." In: *International Journal of Production Economics* 240 (2021), p. 108237.

[Sta23] A. Stavis. *SARIMA predictions versus SDEpredictions of a stock market*. 2023.

[Tag22] R. J. Taggart. "Point forecasting and forecast evaluation with generalized Huber loss." In: *Electronic Journal of Statistics* 16.1 (2022), pp. 201–231.

[TL18] S. J. Taylor and B. Letham. "Forecasting at scale." In: *The American Statistician* 72.1 (2018), pp. 37–45.

[TOP22] S. T. TOPALLAR. "Probabilistic Forecasting of Multiple Time Series with Single Recurrent Neural Network." MA thesis. Middle East Technical University, 2022.

[Van76] N. G. Van Kampen. "Stochastic differential equations." In: *Physics reports* 24.3 (1976), pp. 171–228.

[VWM18] D. W. Van der Meer, J. Widén, and J. Munkhammar. "Review on probabilistic forecasting of photovoltaic power production and electricity consumption." In: *Renewable and Sustainable Energy Reviews* 81 (2018), pp. 1484–1512.

[WW05] R. l Weron and U. Wystup. "7 Heston's Model and the Smile." In: *Statistical tools for finance and insurance* (2005), p. 161.