SHORT-TERM FORECAST METHODOLOGIES AND CASE STUDIES IN TRAFFIC FLOW

by

Elif Yılmaz

B.S., Primary Mathematics Education, Boğaziçi University, 2019

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computational Science and Engineering

Boğaziçi University

2022

ACKNOWLEDGEMENTS

First, I would like to thank my thesis supervisor, Ümit Işlak. I feel extremely lucky to have such a supervisor that is excited to share his enthusiasm for research with students. In the time we have known each other, he taught me many things and always helped me to improve my vision. He has significantly impacted my life, opened my way into academic studies, and guided me on every occasion.

I would like to give special thanks to my thesis co-supervisor, İlker Arslan. His instructions have had significant contributions towards the completion of this study. It has been a great chance to study with him. We are a perfect team, and I think we are a very lovely traffic family. Their energy motivated me throughout this period.

Moreover, I want to thank Ayhan Günaydın, Mustafa Gökçe Baydoğan and Burak Gürel for being a part of my thesis commitee.

I would like to express my gratitude to my friends for their support during my thesis.

Most importantly, I am grateful to my mother, Hacer Yılmaz, and my father, Dursun Yılmaz, for their support and patience throughout my education. On the other hand, I thank Tuğçe Yılmaz, Sudenur Yılmaz, Muhammet Yılmaz and Özgür Yılmaz to always motivate me during my study.

ABSTRACT

SHORT-TERM FORECAST METHODOLOGIES AND CASE STUDIES IN TRAFFIC FLOW

This thesis gives case studies on short-term traffic flow forecasting strategies within a time series framework. After discussing the traditional, machine learning and deep learning methods, one of main goals is to experiment on the uses of hybrid methods. Besides analyzing approaches that were already used in the traffic flow literature, we also introduce and test distinct strategies. Further, we supplement our point forecast results with interval forecasts. In particular, quantiles regression based intervals such as quantile regression averaging and quantile regression neural network are implemented. Both point and interval forecasts are evaluated via several evaluation metrics, and an extensive comparison is provided among the methodologies studied.

ÖZET

TRAFİK AKIŞINDA KISA DÖNEM TAHMİN METODOLOJİLERİ VE VAKA ÇALIŞMALARI

Bu tez, zaman serileri çerçevesinde kısa vadeli trafik akışı tahmin stratejileri üzerine vaka çalışmaları sunmaktadır. Geleneksel, makine öğrenimi ve derin öğrenme yöntemlerini tartıştıktan sonra, ana hedeflerden biri hibrit yöntemlerin kullanımları üzerinde de testler gerçekleştirmektir. Trafik akışı literatüründe halihazırda kullanılan yaklaşımları analiz etmenin yanı sıra, farklı yöntemleri de tanıtıyor ve test ediyoruz. Ayrıca, nokta tahmin sonuçlarımızı aralık tahminleri de kullanarak destekliyoruz. Tez içerisinde, kantil regresyon ortalaması ve kantil regresyon sinir ağı gibi kantil regresyona dayalı aralıklarla özel olarak ilgilenilmektedir. Hem nokta hem de aralık tahminleri, çeşitli değerlendirme ölçütleri aracılığıyla değerlendirilmektedir ve incelenen metodolojiler arasında kapsamlı bir karşılaştırma sağlanmaktadır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS/ABBREVIATIONS

AIC	Akaike's Information Criterion
ANN	Artificial Neural Networks
AR	Autoregressive
ARMA	Autoregressive Moving Average
ARIMA	Autoregressive Integrated Moving Average
ARIMAX	ARIMA with Exogenous Variables
BIC	Bayesian Information Criterion
CNN	Convolutional Neural Network
DNN	Deep Neural Network
GBDT	Gradient-Based Decision Tree
GRU	Gated Recurrent Units
k-NN	k-Nearest Neighbours
LSTM	Long Short-Term Memory
MA	Moving Average
MAE	Mean Absolute Error
MLE	Maximum Likelihood Estimation
MSE	Mean Squared Error
PI	Prediction Interval
QR	Quantile Regression
QRA	Quantile Regression Averaging
QRNN	Quantile Regression Neural Network
QRLSTM	Quantile Regression Long Short-Term Memory
QRSLSTM	Quantile Regression Seasonal Long Short-Term Memory
QRSNN	Quantile Regression Seasonal Neural Network
RBF	Radial Basis Function
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
S-ARIMA	ARIMA with Seasonal Exogenous Variables

SARIMA	Seasonal ARIMA
SGD	Stochastic Gradient Descent
SLSTM	Seasonal Long Short-Term Memory
SVR	Support Vector Regression
SSVR	Seasonal Support Vector Regression
SXGBoost	Seasonal eXtreme Gradient Boosting
XGBoost	eXtreme Gradient Boosting

1. INTRODUCTION

1.1. Motivation and Problem Description

A time series is a sequence of data points listed with respect to time. The time interval between two consecutive data points is constant, and these times are usually indexed by natural numbers. Time series emerge in various fields such as economy, energy, transportation, and environment. Since the conclusions drawn from time series analysis may have several benefits, forecasting approaches on this subject are of fundamental importance these days.

A particular example of a time series is the traffic flow, defined as the number of vehicles in a specific area over time. The number of vehicles in a given region varies depending on time, and it has seasonal characteristics concerning hours, days, weeks, and so on. Traffic flow prediction methods have become essential for transportation systems since they have a significant role in planning, managing, controlling, and improving them. The current study aims to examine the time series models for the traffic flow and to do case studies for California and İstanbul data. Our contributions include developments on currently available methodologies. Further, the current thesis contains one of the few attempts to analyze the traffic flow using the İstanbul traffic data set.

Before continuing, let us note that various factors affect the traffic flow in general, but the models in this study will be solely based on previous observations, which, as we shall see below, are significant, determining features for doing forecasts. That said, let also note that in a further study, one could incorporate exogenous data such as possible accidents, or presence of rain, and so on in order to improve our current results.

1.2. Related Work

It is no surprise that there is a vast literature on the topic of this thesis since traffic forecasting has an essential role in transportation systems. Most studies focus on short-term or long-term traffic flow forecasting by reasoning about future traffic conditions based on previous observations. In this section, we discuss time series methods used for traffic flow predictions in the literature. We may summarize the work on forecasting methodologies under three main headings for traffic flow context. These are traditional time series and variations, nonlinear approaches such as machine learning and deep learning, and hybrid models combining the previous two.

1.2.1. Traditional Methodologies

Traditional methods in time series forecasting refer to variations based on autoregressive and moving average models, which take a linear approach to time series predictions. These models are usually treated as base models in the development of time series analysis, and still they yield successful results in several cases [2–7]. Although these models, which only provide linear techniques to time series problems such as traffic flow, are usually not as effective as the more recent deep learning methods, they provide a base ground for evaluating forecasting results.

It is suggested that the Autoregressive Integrated Moving Average (ARIMA) model works better if the model parameters are estimated by considering different periods in a day of the week and hour of the day by [2]. For example, they use four different categories for days (one category for Monday, one for Tuesday, Wednesday, Thursday, one for Friday, and one for Saturday and Sunday) and three different categories (morning peak hour, night rush hours, and off-peak hours) for hours in a day.

On the other hand, the seasonal ARIMA (SARIMA) model has shown to be relatively more successful for traffic flow forecasting, see [3,5–7]. However, these studies differ in some aspects regarding the SARIMA model dynamics. While the weekly and

daily seasonality effects of the SARIMA model with the Kalman filter are compared by using only the traffic flow data on workdays in [5, 6] supports that the SARIMA model works better by removing outliers from the data. Moreover, the SARIMA model with weekly seasonality is more successful in traffic flow forecasting in [5, 7]. The study of [4] estimates the capabilities of the autoregressive space-time model, which includes spatial and temporal correlations.

1.2.2. Nonlinear Approaches

Besides modeling linear features for time series prediction, many nonlinear approaches exist. We may try to gather these under the machine learning and deep learning titles. Here, we review the Support Vector Regression (SVR), k-Nearest Neighbours (k-NN), and gradient boosting algorithms used in time series predictions under the machine learning frame, and discuss some general deep learning approaches as well as Recurrent Neural Network (RNN), especially Long Short-Term Memory (LSTM).

The SVR model is one of the widely used models to capture nonlinear features in time series forecasting. In [5, 8], SVR is used to predict traffic flow, and they include comparisons to other models such as ARIMA. While Hong [8] asserts that the seasonal SVR (SSVR) model works better for nonlinear and peak periods in traffic flow changes, SARIMA with Kalman filter turns out to yield better results in comparison to SVR in 15-minute traffic flow data in [5]. Another machine learning approach used in traffic models is k-NN which is based on a grouping of similar cases [9, 10]. These studies present the nonparametric regression model by treating the k-nearest neighborhood as a dynamic clustering model. These claim that this similarity based approach gives better results than various others such as ARIMA and Neural Network.

On the other hand, the eXtreme Gradient Boosting (XGBoost) algorithm has very successful results in traffic flow forecasting [11–14], although it mainly is used to classify imbalanced data [15, 16]. In particular, it turns out that the Support Vector Machine (SVM) can be outperformed when XGBoost discrete wavelet denoising algorithm is

applied using traffic speed, occupancy, and previous traffic flow data [12]. Similarly, holiday, temperature, rain, weather, and previous traffic volume data are used in [13] to get predictions for traffic volume by using XGBoost.

Our treatment of the deep learning method applications involve both using existing methods in the literature [17–21], along with some new suggestions. The main common focus in these is LSTM, and they indicate that LSTM leads to very successful results in time series predictions. The studies [18, 20, 21] compare the LSTM with Support Vector Machine (SVM), Artificial Neural Network (ANN), Recurrent Neural Network (RNN), and Gated Recurrent Units (GRU) while [17] uses ARIMA for comparison purposes.

Beyond the LSTM model, the seasonal LSTM, also called sequenced LSTM (SLSTM), is proposed adding seasonal components to model the runoff-sediment process by [19]. SLSTM allows determining the autoregressive components and utilizing the seasonality dependence in the time series. Furthermore, the fuzzy seasonal LSTM (FSLSTM) model is applied to forecast monthly wind power in [22]. In FSLSTM, the lower and upper bounds and mode values in data are also crucial since they are used during training process.

In addition to these deep learning approaches, the language model called n-gram, proposed by [23], can also be used in time series analysis. The n-gram model is applied mainly in classification problems, and it is based on the probabilities corresponding to each possible observation. It has been used for forecasting in time series analysis earlier in [24], but not in a traffic context. Although we did experimentations on the use of n-gram model using our data, we will not include the results below since they did not perform well enough compared to the other methods we discuss.

1.2.3. Hybrid Models

Hybrid models consist of model combinations. The first such approach focuses on the straightforward way of combining the predictions obtained from results of distinct models using a weighted sum [25], [26]. A second approach, which we call additive models, is based on using the predictions of a selected model, and the residuals obtained from this model are then modeled with a different approach. The overall result is then the sum of the forecasts from these two models [27–30]. In yet another approach, overall results are obtained by inserting the predictions of a model into some other model. That is, the models are applied one after the other. See, for example, [31] in which one-dimensional CNN, GRU, and Attention modules are used for traffic flow forecasting.

Let us next provide some pointers to the literature towards the hybrid methods. In order to form a hybrid model as a weighted sum of distinct model results, it is essential to achieve the best balance among the models. LSTM and XGBoost are used to predict network traffic by [25]. After calculating the LSTM and XGBoost model results separately using the hourly device traffic data, these results are collected as a weighted sum to capture the traffic pattern. Furthermore, k-NN and LSTM models are applied for traffic flow prediction by [26]. Compared to SVR, ARIMA, k-NN and LSTM, the proposed approach gives better forecast results.

Regarding the additive models, the traffic flow is forecasted by combining the SVM [29] and Radial Basis Function Artificial Neural Network (RBFANN) with the ARIMA model [30]. A similar study is done for electricity price forecasting in [27, 28]. Lastly, let us not that short-term traffic flow is obtained by applying Gated Recurrent Unit (GRU) with Attention modules for long temporal features to Convolutional Neural Network (CNN) forecasts to capture local trend features in [31].

1.3. Overview

Many of the time series approaches included in the related work section above will be discussed in detail in Chapter 2. We also investigate the main challenges in the time series analysis and discuss how to improve the existing approaches. We gather these methods under the titles of traditional, machine learning, deep learning, and hybrid approaches.

In Chapter 3, we briefly discuss optimization techniques for traditional models and deep learning models. The optimization of the models means optimizing both the model parameters and hyperparameters. The models introduced in Chapter 2 allow obtaining interval estimates along with the point estimates. Chapter 4 is devoted to prediction intervals, and the related evaluation metrics.

Chapter 5 discusses the experiments and results of the case studies for California and İstanbul. We investigate the contents of the data and data processing in detail. Moreover, we include notes on how to decide the features of the data. We also focus on the techniques to complete missing data points. We conclude the thesis with some possible future directions in Chapter 6.

2. METHODOLOGY

In this chapter, we will discuss the approaches mentioned in the previous chapter to predict traffic flow. We group the methods into four categories: autoregressive and moving average model variations (AR, ARMA, ARIMA, SARIMA, and ARIMAX), machine learning models (SVR, XGBoost), deep learning models (ANN, LSTM) and hybrid approaches (SVRARIMA, LSTMARIMA).

A basic approach to predict the traffic flow is the naive method. We expect the traffic flow at time t to be the traffic flow at time $t-1$. So, if $\tilde{X}_t$ is the prediction at time t, then

$$\tilde{X}_t = X_{t-1}, \tag{2.1}$$

where X_{t-1} is the previous observation.

Another one is the historical average method. The historical average method predicts the current situation using the mean of the previous observations. It has the following form:

$$\tilde{X}_t = \frac{1}{t-1}\sum_{i=1}^{t-1} X_i. \tag{2.2}$$

There are also seasonal historical average approaches, which enable us to catch seasonal trend in time series. For example, when we have hourly data and want to apply a daily seasonality trend, the average of the same hours in the previous days is used as a forecast. The naive method, the historical average method and the seasonal historical average method are benchmark methods. They are used to check whether an arbitrary model is promising, or not.

2.1. Autoregressive and Moving Average Model Variations

The models AR, ARMA, ARIMA, SARIMA, and ARIMAX, which can be classified as autoregressive and moving average model variations, are to be discussed in this section. We can define autoregressive and moving average models as regression approaches defining the following variable with previous variables and errors. To explain briefly, the AR model allows us to predict the next time by linearly combining traffic flows from previous times, depending on how much historical data we are looking for.

The ARMA model also includes a linear combination of errors from previous times. Moreover, the forecasting results of ARIMA are obtained by making calculations after the time series data is integrated. While the SARIMA model allows us to add seasonal components, ARIMAX provides forecasting using external features. The studies of [7, 32–35] can be cited as an auxiliary sources for the explanations in this section.

First, we discuss the concept of stationarity in time series before presenting variations of the ARIMA model. A stationary time series has properties independent of the time at which the series is observed. Hence, the time series with trends or seasonality, which affect the distribution of the time series at different times, is not stationary. Conversely, a white noise series, which may be defined as the error at each time step, is stationary since the observation time does not matter, and it appears almost the same at any time.

Let us start with an autoregressive model for a time series $\{X_t\}$, where X_t is a random variable at time t. We define the AR model with order p; AR(p), in the following form:

$$X_t = \sum_{i=1}^{p} \phi_i X_{t-i} + \varepsilon_t, \tag{2.3}$$

where ϕ_i's are real numbers and ε_t's refer to white noise. In addition, the value of p indicates the number of time steps in the past are involved to be considered in the

model, and each ϕ_i value represents the model parameters, that is, weights.

As an example, let us choose p equal to 2 and have 15-minute traffic flow data. Suppose that the traffic flow value is 100 at 12:15 and is 90 at 12:30. Therefore, the traffic flow forecast at 12.45 is expressed as the following:

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} = 100\phi_1 + 90\phi_2.$$

We discuss how to optimize the parameters ϕ_i's in Chapter 3.

We obtain the ARMA model when we add a part including the linear combination of previous errors. ARMA(p, q) has the form:

$$X_t = \varepsilon_t + \sum_{i=1}^{p} \phi_i X_{t-i} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j}, \tag{2.4}$$

where q is the number of time steps to consider the previous errors, and θ_j's are real numbers representing weights for the errors.

The ARIMA(p, d, q) model, which is en extension of ARMA, has an integrated part, which is d that refers to *differencing*. Let us explain a particular case, where $d = 1$. If $d = 1$ (first order differencing), then we define y_t as

$$y_t = X_t - X_{t-1}. \tag{2.5}$$

We can write the equations for X_t and X_{t-1} as the following:

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \cdots + \phi_p X_{t-p} + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t, \tag{2.6}$$

$$X_{t-1} = \phi_1 X_{t-2} + \phi_2 X_{t-3} + \cdots + \phi_p X_{t-p-1} + \theta_1 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q-1} + \varepsilon_{t-1}. \tag{2.7}$$

Then, subtracting (2.7) from (2.6), we obtain the following expression:

$$\begin{aligned} y_t &= X_t - X_{t-1} \\ &= \varepsilon_t - \varepsilon_{t-1} + \phi_1(X_{t-1} - X_{t-2}) + ... + \phi_p(X_{t-p} - X_{t-p-1}) \\ &\quad + \theta_1(\varepsilon_{t-1} - \varepsilon_{t-2}) + ... + \theta_q(\varepsilon_{t-q} - \varepsilon(t-q-1)) \\ &= \varepsilon_t - \varepsilon_{t-1} + \sum_{i=1}^{p} \phi_i(X_{t-i} - X_{t-i-1}) + \sum_{j=1}^{q} \theta_j(\varepsilon_{t-j} - \varepsilon_{t-j-1}). \end{aligned} \tag{2.8}$$

When second order differencing ($d = 2$) is utilized, we have a new series as $z_t = y_t - y_{t-1}$ and so on for the case $d > 2$. One of the issues we need to notice is that if we use first order differencing, the predictions belong to the integrated series y_t. In the end, we need to find X_t's using y_t's.

In order to express the SARIMA model clearly we present the lag operator ∇ and the backshift B operator. We define the lag operator as follows:

$$\nabla X_t = X_t - X_{t-1}. \tag{2.9}$$

We also define

$$\nabla^k X_t = \nabla^{k-1} X_t - \nabla^{k-1} X_{t-1} \tag{2.10}$$

for $k \in \mathbb{N}$. Particularly, the second order differencing can be expressed as

$$\nabla^2 X_t = X_t - X_{t-1} - (X_{t-1} - X_{t-2}) = X_t - 2X_{t-1} + X_{t-2}. \tag{2.11}$$

On the other hand, the backshift operator is defined by

$$B^i X_t = X_{t-i}, \tag{2.12}$$

where $i \in \{1, \ldots, t-1\}$. Observe that

$$(1 - B)X_t = X_t - X_{t-1} = \nabla X_t, \tag{2.13}$$

where 1 in the above equation is the identity operator, and

$$(1 - 2B + B^2)X_t = X_t - 2X_{t-1} + X_{t-2}, \tag{2.14}$$

where we can observe that the coefficient of B^i is exactly the coefficient of X_{t-i} for a given i, from the equation above. Another notation is the following:

$$\nabla_i X_t = X_t - X_{t-i} \tag{2.15}$$

which is also called seasonal differencing. We may also express the seasonal differencing by using backshift operator

$$(1 - B^S)X_t = X_t - X_{t-S} = \nabla_S X_t, \tag{2.16}$$

where S is length of seasonal part. We express the differenced series Y_t with seasonal differencing D as

$$Y_t = (1 - B)^d (1 - B^S)^D X_t. \tag{2.17}$$

We use B as the backshift operator $B^S X_t = X_{t-S}$, $\{Z_t\}$ as identically and normally distributed noise series with zero mean and σ^2 variance. We can also express it as $\{Z_t\} \sim \mathcal{N}(0, \sigma^2)$. Thus, we can write the SARIMA process by

$$\phi(B)\Phi(B^S)Y_t = \theta(B)\Theta(B^S)Z_t. \tag{2.18}$$

We write ϕ, Φ, θ and Θ which refer to autoregressive parameter polynomial, seasonal autoregressive parameter polynomial, moving average parameter polynomial and seasonal moving average parameter polynomial, as

$$\begin{aligned}
\phi(z) &= 1 - \phi_1 z - ... - \phi_p z^p, \\
\Phi(z) &= 1 - \Phi_1 z - ... - \Phi_P z^P, \\
\theta(z) &= 1 + \theta_1 z + ... + \theta_q z^q, \\
\Theta(z) &= 1 + \Theta_1 z + ... + \Theta_Q z^Q.
\end{aligned}$$

The SARIMA$(p, d, q)(P, D, Q)_S$ model is equipped with the hyperparameters; seasonal period S, nonnegative integers d and D for nonseasonal differencing and seasonal differencing orders, parameters p and P seasonal and nonseasonal autoregressive orders, and finally, the parameters q and Q for seasonal and nonseasonal moving average orders, respectively. The SARIMA$(p, d, q)(P, D, Q)_S$ model assumes the following process:

$$X_t = \frac{\theta(B)\Theta(B^S)}{\phi(B)\Phi(B^S)(1-B)^d(1-B^S)^D} Z_t. \tag{2.19}$$

The ARIMAX model is obtained by adding an exogenous variable to the ARIMA model. The ARIMAX model is based on the process which has the following form:

$$X_t = \beta x_t + \frac{\theta(B)\Theta(B^S)}{\phi(B)\Phi(B^S)(1-B)^d(1-B^S)^D} Z_t, \tag{2.20}$$

where x_t is a covariate or an exogenous variable at time t, β is its coefficient, and other terms are as mentioned before. It is an extension of (2.19).

Although this expression may seem simple with the added term, it is necessary to interpret the coefficient of this variable correctly. Let us say that when the value

of x_t decreases, the β value cannot be evaluated as an effect of the change on X_t. The non-intuitive interpretation is that the β value affects the expression of previous observations and errors on the right side of (2.20) to balance the equation.

2.2. Machine Learning Approaches

The machine learning approaches mentioned in this section are SVR and XGBoost. Although SVM enables a classification algorithm to obtain appropriate hyperplanes separating the data into a finite number of classes, the SVR model allows a regression algorithm for the time series forecasting. Moreover, XGBoost is used for traffic flow forecasting as mentioned in Section 1.2.

2.2.1. Support Vector Regression

Although Support Vector Machines are suitable for classification problems, Support Vector Regression model can be used for the time series forecasting. By using [36], we write the basic steps of the SVM problem. We generate an SVM by using the ε-insensitive loss function, proposed by [37], expressed by

$$|y-\hat{y}|_\varepsilon = \begin{cases} |y-\hat{y}|-\varepsilon & \text{if } |y-\hat{y}| \geq \varepsilon \\ 0 & \text{otherwise} \end{cases} \tag{2.21}$$

where y and $\hat{y}$ are the actual and the estimated values. Here we discuss the application of the SVM to a linear regression problem given by:

$$\hat{y} = w^T x + b, \tag{2.22}$$

where w and b are unknown parameters referring to the feature vector and bias, respectively. The linear regression problem estimates these parameters with the given independent and identically distributed training data.

The risk function in this problem is expressed by

$$\frac{1}{2}||w||^2 + C\sum_{j=1}^{N} |y_i - \hat{y}_i|_\varepsilon \tag{2.23}$$

where N is the number of training samples, C is a constant for the tradeoff between the training error and the penalizing term $||w||^2$. The constraints of the problem are the following:

$$y_i - \hat{y}_i \leq \varepsilon + \xi_i, \tag{2.24}$$

$$\hat{y}_i - y_i \leq \varepsilon + \xi_i^*, \tag{2.25}$$

$$\xi_i \geq 0, \tag{2.26}$$

$$\xi_i^* \geq 0, \tag{2.27}$$

where $i = 1, \ldots, N$. The optimization for the the SVR problem is done with the Lagrange multipliers with the dual formulation approach [38]. The expression $w^T x$ in (2.22) can also be in a different form in the kernel function approach.

Kernel functions are also called kernel trick, which enable forming a map between the input data points and the features. These functions allow us to analyze the problem in different ways and help us to find an approach that better fits the data. Various kernel functions are used in the literature, two of which are:

$$K(w, x) = (w^T x + 1)^d, \tag{2.28}$$

$$K(w, x) = \exp\Big(- \frac{||x - w||^2}{2\sigma^2} \Big), \tag{2.29}$$

where d is the polynomial degree taking the positive values, and (2.28) and (2.29), respectively, are known to be the polynomial and RBF kernel functions. These kernels are useful in decreasing computational cost, especially in high dimensional problems.

On the other hand, the seasonal SVR (SSVR) method is proposed by [39] to use SVR with the seasonal trends of the time series. The SVR needs the regressors to achieve target variables. However, in time series data, the regressors are usually previous observations and seasonal trends.

2.2.2. eXtreme Gradient Boosting

The eXtreme Gradient Boosting, proposed by [40], is based on the structure of the gradient-based decision tree (GBDT) [41, 42]. We explain briefly the decision tree to understand the basics of the XGBoost approach. First, decision trees are used for both classification and regression problems. It contains the decision nodes and leaf nodes and has the form of a tree. It gets improved by dividing the data into smaller pieces. The first node in the tree is called the root node.

The decision tree algorithm generates the statements using the attributes of the data with the aim of the best possible performance. To receive and optimize the data pattern, it uses uncertainty metrics of the information theory such as Information Gain and Gini Index [43]. When we have the continuous features, the decision tree becomes a regression tree.

Let us briefly explain the regression trees. We have a dataset

$$D = \{(x_t, y_t) : x_t \in \mathbb{R}^k, y_t \in \mathbb{R}, t = 1, ..., T\}, \tag{2.30}$$

where T is the number of observations with k features. Thus, for tth observation, the estimation of the decision trees, $\hat{y}$, is expressed by the following:

$$\hat{y}_t = \sum_{n=1}^{N} f_n(x_t), \tag{2.31}$$

where N is the number of trees $f_n \in F$ and F refers to the space of the regression trees written by

$$F = \{f(x) = w_{g(x)}\}, \text{ where } g(x) : \mathbb{R}^k \to K, w \in \mathbb{R}^K \tag{2.32}$$

and where g is the map which shows the relation between x and the leaf node, w is the weight of the leaf node and K is the number of leaf nodes. The difference between the decision tree and regression tree is that the regression tree allows a continuous score w_i for ith leaf.

The gradient boosting algorithm, also called Gradient Boosting Machine (GBM), enables to optimize the weight using the first-order derivatives [44]. The boosting means that the models are created sequentially by giving more weight on wrong predictions instead of using the random data and attributes. Thus, focusing on the wrong estimates, it tries to improve them. The gradient is used to optimize the weights w of the leaves.

Furthermore, the second-order Taylor expansion of the loss function is also essential, and regularization terms are added to it in the XGBoost algorithm. Therefore, it reduces the model complexity and avoids overfitting. The optimization problem in XGBoost is to minimize the following objective function

$$L = \sum_{t=1}^{T} \ell(y_t, \hat{y}_t) + \sum_{n=1}^{N} \Omega(f_n), \tag{2.33}$$

where y_t and $\hat{y}_t$ are the actual and forecasted values, respectively, ℓ is the loss function and $\Omega(f_n)$ indicates the complexity of nth decision tree which is written by

$$\Omega(f_n) = \gamma K + \frac{1}{2}\lambda ||w||, \tag{2.34}$$

where K is the number of leaf nodes, w is the leaf weight and, γ and λ are the penalties of the number leaves and the leaf node weight, respectively.

In addition to achieve high predictive strength of the model, another most important features of the algorithm are to prevent overfitting and to manage the missing data points. The first step in the XGBoost algorithm is to arbitrarily initialize the values of the leaf nodes, and 0.5 is commonly used for this purpose. Since the model converges to the actual values by the actions to be taken in the next steps, we randomly determine the initial values.

Since the number of leaf nodes in best tree may be large, the γ value is used to remove the necessary parts whose Information Gain score is smaller value than the γ. Thus, the increasing the γ value enables us to avoid overfitting. The decision process continues until the small residuals, or the specified number of trees are reached.

2.3. Deep Learning Approaches

In this section, we mention Artificial Neural Networks (ANN), which may be considered as a simple version of deep neural networks (DNN), and Long-Short Term Memory (LSTM), which is a special case of Recurrent Neural Networks (RNN). ANNs have a structure, which will be discussed in the following subsection, imitating the learning process in a brain. The difference between ANNs and RNNs can be briefly given that ANNs are feed-forward networks and do not have connections between historical and current data, while RNNs have a system that can memorize long-term dependencies in data. The structure of LSTM will be presented in the section just after ANN.

2.3.1. Artificial Neural Networks

Artificial neural networks are proposed by [45]. ANNs enable us to create a mathematical model using the inspiration of biological neural networks. An artificial neuron, also called a node, mimics a real neuron in a brain by performing the mathematical expressions, which comprise multiplication, summation, and activation function. The mathematical analogy will be clear in the following paragraphs.

The main idea is that we make a regression to a given data by creating a function composed, in a particular way, of a sequence of functions. The way we represent this function somehow can be given by simply a graph with nodes and edges (or arrows). It will be explained through an example soon.

Let $f : \mathbb{R} \to \mathbb{R}$ be a function and $x = \langle x_1, x_2, \ldots, x_n \rangle \in \mathbb{R}^n$. Then, the meaning of $f(x)$ is conventionally defined by the (column) vector $f(x) = \langle f(x_1), f(x_2), \ldots f(x_n) \rangle$ for the sake of simplicity.

Let $n_1, n_2, \ldots, n_k$ be positive integers and $\ell_i : \mathbb{R}^{n_i} \to \mathbb{R}^{n_{i+1}}$ be functions so that $\ell_i(x) = W_i x + b_i$ for any $x \in \mathbb{R}^{n_i}$ and $i = 1, 2, \ldots, k-1$, where $W_i \in \mathbb{R}^{n_i \times n_{i+1}}$ and

$b_i \in \mathbb{R}^{n_{i+1}}$ are constants which will be considered as parameters to be optimized in the next chapter. We also call b_i's as bias weights.

In general, an ANN model is defined so that it is used to regress a training data simply by a function of the form

$$R(x) = f_{k-1}(\ell_{k-1}(f_{k-2}(\ell_{k-2}(\ldots(f_1(\ell_1(x))\ldots))))), \tag{2.35}$$

where $f_1, f_2, \ldots, f_{k-1}$ are functions defined on reals (together with their conventional meanings mentioned above) and $x \in \mathbb{R}^{n_1}$.

In this setting, an input is n_1 dimensional (vector) whereas the output is an n_k dimensional (vector). So, an input of the model has n_1 components while the target (output) has n_k components. In the expression of the right side of the equation (2.35), a component of the vector $f_j(\ell_j(x_0))$, where x_0 is in the domain of ℓ_j, is said to be a *neuron (node)* and the collection of these neurons in $f_j(\ell_j(x_0))$ is said to be a *layer*. Clearly, there are k *layers* and the j^{th} layer contains n_j *neurons (nodes)* for $j \in \{1, 2, \ldots, k\}$. The 1^{st} layer is said to be the *input layer* and the k^{th} layer is said to be the *output layer* while the j^{th} layer is said to be a *hidden layer* if $1 < j < k$. Furthermore, we call f_j as an *activation function* of the j^{th} layer for $j \in \{1, 2, \ldots, k-1\}$.

As seen in the definition of a general ANN given above, an ANN may have any finite number of layers equipped with any chosen activation functions together with the flexibility of tuning the number of neurons in the hidden layers. On the other hand, the number of components in an input vector is exactly the number of neurons in the input layer while the number of components in the output layer is the number of neurons in the target vector.

Now, let us present how such a model is represented by a graph by means of specifying particular values to the variables above. Let $k = 3,\ n_1 = 3,\ n_2 = 5,\ n_3 = 1$, $f_1(x) = \tanh(x)$ and $f_2(x) = \frac{1}{1+e^{-x}}$. Then the Figure 2.1, noting that it does not specify the activation functions f_1 and f_2, represents this model.

If a neuron is connected to a group of neurons from a previous layer, that means this group of neurons are multiplied by some weights and added some bias weights and sent to this following connected neuron. This connection is shown by a directed arrow. See the Figure 2.1.

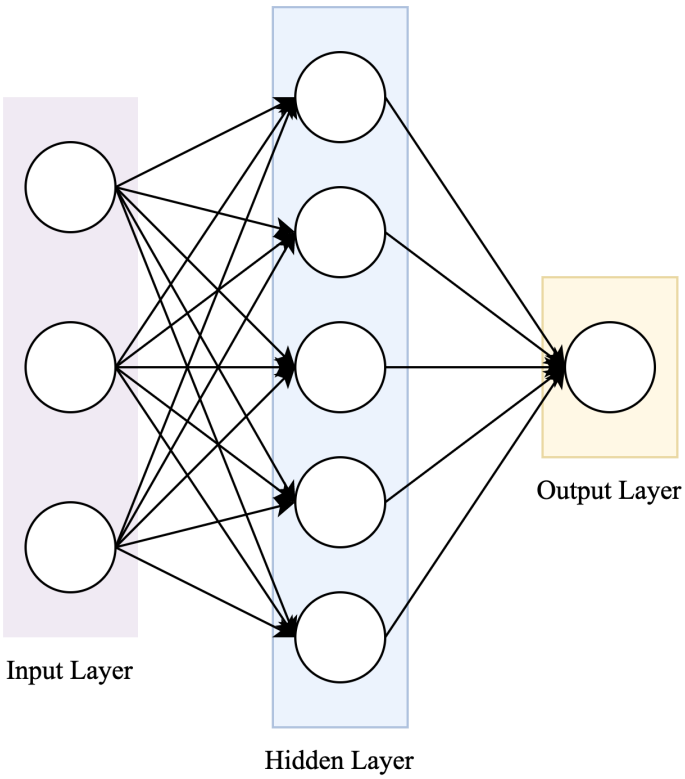

Figure 2.1. Simple ANN Structure.

Observe that if all activation functions are taken to be identity function, then the ANN is turned out to be a *linear regression*. If, further, the output function is taken to be the *sigmoid* function, the ANN is now *logistic regression*.

There may be extended configurations of an ANN so that the number of hidden layers is high and/or the output of layer is added/multiplied by the output of a further layer and/or any other method which leads to fundamentally a different graph. In such cases, the ANN is called as Deep Neural Network (DNN).

The most common activation functions are sigmoid, tanh and ReLU which respectively expressed by following equations:

$$\sigma(x) = \frac{1}{1+e^{-x}}, \tag{2.36}$$

$$\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}, \tag{2.37}$$

$$\mathrm{ReLU}(x) = \max\{0, x\}. \tag{2.38}$$

The components of an output of a layer may be seen as $f(x) = \langle f(x_1), f(x_2), \ldots, f(x_n)\rangle$, where f is the activation function, $x = \langle x_1, \ldots, x_n\rangle$ and f depends on only one component. However, an activation function may depend on other components as well. For example, the *softmax* function defined as follows:

$$\mathrm{softmax}(x) = \Big\langle \frac{e^{x_1}}{\sum_{i=1}^{n} e^{x_i}}, \frac{e^{x_2}}{\sum_{i=1}^{n} e^{x_i}}, \frac{e^{x_3}}{\sum_{i=1}^{n} e^{x_i}}, \ldots, \frac{e^{x_n}}{\sum_{i=1}^{n} e^{x_i}} \Big\rangle, \tag{2.39}$$

which is usually used for the output function for multiclassification.

The parameters in ANNs are updated using the backpropagation algorithm based on the gradients. The details of the process of the optimization of the parameters will be given in the next chapter.

2.3.2. Long Short-Term Memory

LSTM is a more specific structure of RNN which is usually used for NLP (Natural Language Processing) problems as well as time-series problems. An RNN model can be defined as a composition of a sequence of functions. However, the neural network is designed to be recurrent. So, It is hard to present it as a composition of a sequence of functions and so for its corresponding graph. We use another method to represent them as graphs. This method will be explained below through an example given in [1].

For an instance of RNN, as in Figure 2.2, suppose that we have a sequence of data $x_1,\ x_2, \ldots,\ x_n$ and assume that the target variable and the input variable are of the same size. In other words, we predict the next τ targets from given τ terms, where

τ is taken to be depending on the problem. We give the recurrent equations as

$$a_t = b + Wh_{t-1} + Ux_t, \tag{2.40}$$

$$h_t = \tanh(a_t), \tag{2.41}$$

$$o_t = c + Vh_t, \tag{2.42}$$

$$\hat{y} = \text{softmax}(o_t), \tag{2.43}$$

where $b,\ c,\ U,\ V, W$ are parameter vectors and matrices which have appropriately chosen sizes according to the dimension of the input. We apply the above equations from $t = 1$ to $t = \tau$ by choosing an initial value for h_0. In practice, it is 0. If the loss function L_t is defined as the negative log-likelihood of y_t given $x_1, \ldots, x_t$, then

$$\begin{aligned} L(\{x_1, \ldots, x_\tau\}, \{y_1, \ldots, y_\tau\}) &= \sum_t L_t \\ &= -\sum_t \log p_{model}(y_t | \{x_1, \ldots, x_t\}) \end{aligned} \tag{2.44}$$

Then, the parameters of the model given in the equations above are optimized by means of the loss function L. Since the runtime of the optimization process is pretty high, an alternative way is addressed in Section 10.2.2 of [1].

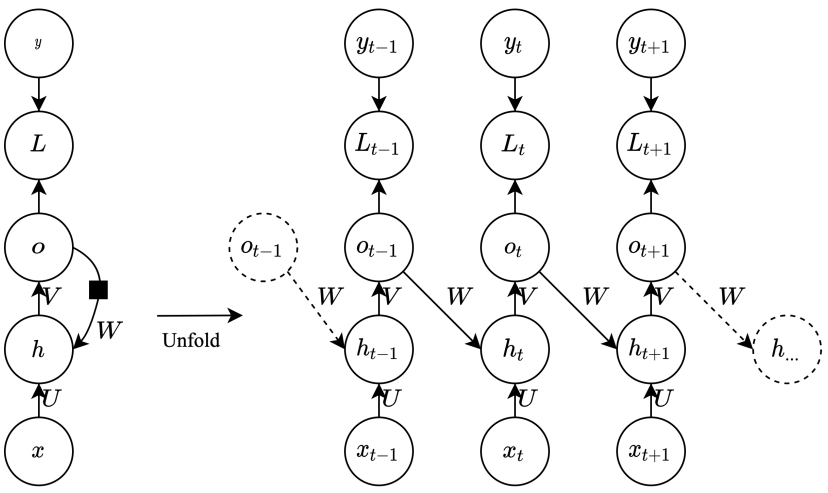

Figure 2.2. RNN Model - reproduced from [1].

To have an idea on how optimization is processed in a general neural network, the global minimum value of the loss function is obtained by choosing a random point in the domain of the loss function, then a second point is selected by moving in the inverse direction of the gradient of the loss function of a very small length and so on till the consecutive loss values evaluated at the last points are ignorable. This process will be presented rigorously in the optimization chapter.

In the setting of such a optimization process, there occurs two main issues which are vanishing gradient and exploding gradient problem. These are basically due to complex graphs of models possibly having high number of connections (edges) and neurons. These problems are treated to be sorted out by creating gated recurrent neural networks (Section 10.10 of [1]). One of those is LSTM. The example explained above can be consecutively connected to itself and so on. Each such connection can be viewed as connections between blocks. An LSTM is designed to have blocks of special RNNs. Each block is also called as an LSTM cell. An LSTM cell is given below in the Figure 2.3.

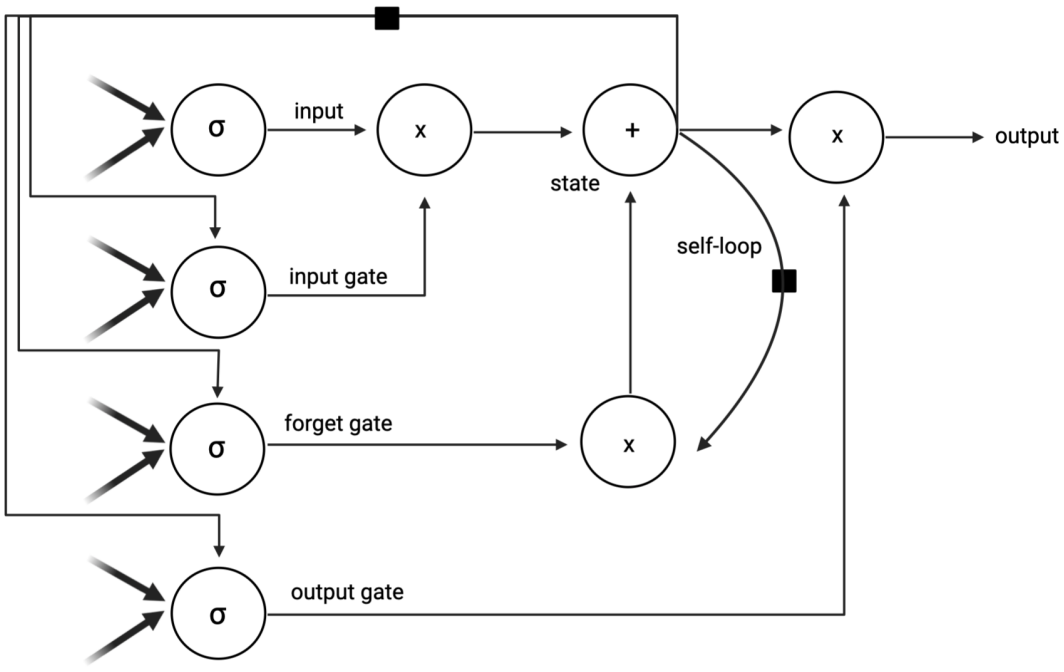

Figure 2.3. LSTM Architecture.

We do not continue for the details of what happens in an LSTM cell owing to the fact that otherwise it would be a deviation from our main goal in this thesis. However, the interpretation of the terminology of the words shown in the Figure 2.3 and basic ideas behind this architecture can be understood from the Section 10.10 of [1].

Here are the recurrent equations of an LSTM cell for equal size of target and input variable. Let us define $X = (x_1, x_2, ..., x_n)$ as time series input, $H = (h_1, h_2, ..., h_n)$ as hidden state of memory cells and $Y = (y_1, y_2, ..., y_n)$ as output. Thus, we write the following expressions:

$$h_t = H(W_{hx}x_t + W_{hh}h_{t-1} + b_h), \tag{2.45}$$

$$p_t = W_{hy}y_{t-1} + b_y, \tag{2.46}$$

where matrices W are weights, b's are bias vectors and p_t is the predicted value of the LSTM with the corresponding actual value y_t.

We need to explain weight matrices in detail. The expression W_{ab} refers to the weight matrix of b and a indicates which gate it exists. We use i for input gate, h for hidden state, f for forget gate, c for memory cell and o for output gate. For example, the weight matrix W_{hx} means the weight of x in the hidden state. Similarly, the b_y represents the bias vector of y.

Using the meaning of the variables above, we can compute hidden state of memory cells by using following equations:

$$i_t = \sigma(W_{ix}x_t + W_{hh}h_{t-1} + W_{ic}c_{t-1} + b_i), \tag{2.47}$$

$$f_t = \sigma(W_{fx}x_t + W_{hh}h_{t-1}W_{fc}c_{t-1} + b_f), \tag{2.48}$$

$$c_t = f_t * c_{t-1} + i_t * g(W_{cx}x_t + W_{hh}h_{t-1} + W_{cc}c_{t-1} + b_c), \tag{2.49}$$

$$o_t = \sigma(W_{ox}x_t + W_{hh}h_{t-1} + W_{oc}c_{t-1} + b_o), \tag{2.50}$$

$$h_t = o_t * h(c_t), \tag{2.51}$$

where the sigmoid function $\sigma(x) = \frac{1}{1+e^x}$, $*$ refers to scalar product matrices or vectors and g and h are the functions extending sigmoid function by changing range [-2,2] and

[-1,1], respectively. Also, we can express these functions with tanh function by using connection between them, which is $\tanh(x) = 2\sigma(2x) - 1$.

We would also like to draw attention to the LSTM network with seasonality. A sequenced LSTM network that detects autoregressive components and realizes seasonal dependency, proposed by [19] and also called seasonal LSTM (SLSTM).

2.4. Hybrid Models

Some models defined in the previous chapter assume linear features of the input variable while some of them do nonlinear ones. A model called SVRARIMA first predicts using the SVR model, which is based on non-linear features by choosing non-linear kernels, and then applies ARIMA, that carries linear assumptions, to the errors of the forecast for electricity prices in [27, 28]. We can see the hybrid model approach in Figure 2.4.

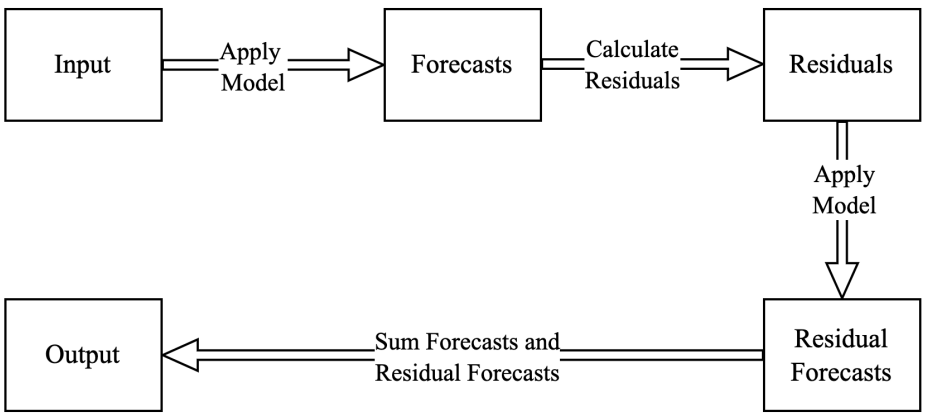

Figure 2.4. The Visualization of the Hybrid Model.

The better we approach the errors after applying any model, the better the hybrid model will work. The hybrid model in [27] has the following form:

$$Y_t = N_t + L_t \tag{2.52}$$

where N_t is the predictions of nonlinear part and L_t is the forecasts of the linear model. The errors is expressed as

$$\varepsilon_t = Y_t - \hat{N}_t \tag{2.53}$$

where ε_t is residual at time t, Y_t is actual values and $\hat{N}_t$ represents predictions of nonlinear model. Therefore, we apply a linear model, ARIMA, to ε_t's. After we estimate the residuals using these parameters, we obtain hybrid model forecasts by adding the nonlinear model predictions to the error predictions of the ARIMA model.

3. OPTIMIZATION

This chapter will discuss optimization techniques for traditional models [34, 46] and deep learning models [1,47,48]. Parameter optimization in machine learning setting is already discussed in previous chapter and so we will not go into it again below.

In general, the optimization of models consist of both the model parameters and hyperparameters. While the model parameters refer to the weights in the structure of the model, the model hyperparameters are the parameters we use when building the model. The hyperparameters are assigned at the beginning of the model setup. Conversely, the model parameters are determined during the learning process of the model. Thus, there are different techniques to determine the optimized model parameters and hyperparameters.

3.1. Maximum Likelihood Estimation

As we mentioned earlier, there are various techniques to optimize model parameters. The purpose of this section is to discuss optimizing the parameters of ARIMA model and its variations [46]. Here, after deciding on the model orders (the values of $p, d,$ and q), one needs to estimate the parameters. Maximum likelihood estimation (MLE), which explores the values of the parameters ϕ_i's and θ_i's that maximize the likelihood of given observed data is used for this purpose. Our treatment below will be for a special case, namely, AR(1).

The AR model we discuss will be causal which we discuss next. Recall that when X_t is expressed in terms of the current and the previous values of Z_t, X_t is called causal of the series of Z_t. Also recall that AR(1) model is given by

$$X_t = \phi X_{t-1} + Z_t. \tag{3.1}$$

MA(q) process is clearly always causal since it is expressed as

$$X_t = \sum_{i=1}^{q} \theta_i Z_{t-i} + Z_t. \tag{3.2}$$

Here and below we assume that the underlying AR(1) process is causal since otherwise the process is not stationary, and it blows up as it evolves in time.

Now, we examine a causal AR(1) process. Using the causal AR(1), we can indicate the steps of Maximum Likelihood Estimation. Let

$$X_t = \mu + \phi(X_{t-1} - \mu) + Z_t, \tag{3.3}$$

where $\{Z_t\} \sim \mathcal{N}(0, \sigma^2)$. Observe that the likelihood of μ, ϕ and σ^2 in AR(1) can be written as

$$L(\mu, \phi, \sigma^2) = f(X_1) f(X_2|X_1) \dots f(X_n|X_{n-1}). \tag{3.4}$$

Since we have $X_t|X_{t-1} \sim N(\mu + \phi(X_{t-1} - \mu), \sigma^2)$, we obtain

$$f(X_t|X_{t-1}) = f_Z((X_t - \mu) - \phi(X_{t-1} - \mu)), \tag{3.5}$$

where f_Z is the density of Z with mean zero and variance σ^2. Furthermore, because of the iid assumption, we may express the likelihood as

$$\begin{aligned} L(\mu, \phi, \sigma^2) &= f(X_1) \prod_{t=2}^{n} f_Z((X_t - \mu) - \phi(X_{t-1} - \mu)) \\ &= (2\pi\sigma^2)^{-n/2} (1 - \phi^2)^{1/2} \exp\Big(- \frac{S(\mu, \phi)}{2\sigma^2} \Big), \end{aligned} \tag{3.6}$$

where

$$S(\mu, \phi) = (1 - \phi^2)(X_1 - \mu)^2 + \sum_{t=2}^{n} ((X_t - \mu) - \phi(X_{t-1} - \mu))^2 \tag{3.7}$$

which is called as the unconditional sum of squares. When we take the derivative of the logarithm of (3.6) with respect to σ^2, and set it equal to zero, we get the maximum likelihood estimator of σ^2 as

$$\hat{\sigma}^2 = \frac{1}{n} S(\hat{\mu}, \hat{\phi}), \tag{3.8}$$

where $\hat{\mu}$ and $\hat{\phi}$ are the maximum likelihood estimators of μ and ϕ. For obtaining the estimators $\hat{\mu}, \hat{\phi}$, replacing σ^2 by $\hat{\sigma}^2$ and ignoring constants in (3.6), we may now work

on

$$\ell(\mu, \phi) = \log\left(\frac{1}{n}S(\mu, \phi)\right) - \frac{1}{n}\log(1-\phi^2), \tag{3.9}$$

which is equivalent to a study of

$$\ell(\mu, \phi) \propto -2\log L(\mu, \phi, \hat{\sigma}^2). \tag{3.10}$$

Now, a little bit of work, which can be found in [46], can be done to conclude that maximum likelihood estimators of mu and ϕ conditional on X_1 is given by

$$\hat{\mu} = \frac{\hat{X}_{(2)} - \hat{\phi}\hat{X}_{(1)}}{1-\hat{\phi}}, \tag{3.11}$$

$$\hat{\phi} = \frac{\sum_{t=2}^{n}(X_t - \hat{X}_{(2)})(X_{t-1} - \hat{X}_{(1)})}{\sum_{t=2}^{n}(X_{t-1} - \hat{X}_{(1)})^2}, \tag{3.12}$$

where

$$\hat{X}_{(1)} = \frac{1}{n-1}\sum_{t=1}^{n-1} X_t,$$

$$\hat{X}_{(2)} = \frac{1}{n-1}\sum_{t=2}^{n} X_t.$$

3.2. Expected Risk and Empirical Risk Minimization

The goal in the optimization in deep learning approaches is to reduce the difference between forecasted values and actual values as much as possible via supervised learning techniques. The loss function, also called the cost function or the error function, is used to evaluate this difference. In particular, the purpose of optimization is to minimize the following cost function:

$$J(w) = \mathbb{E}_{(x,y)\sim p_{data}}[\mathcal{L}(y, f(x; w))], \tag{3.13}$$

where $\mathcal{L}$ means loss function, y is the actual output, $f(x, w)$ is the predicted output with the input x and the parameter w, and p_{data} is the true underlying data distribution. $J(w)$ is called the *expected risk*; however, we cannot immediately minimize it because the true distribution is unknown. For this reason we introduce a different cost function, $J_{data}(w)$, to improve the performance measure over the training set:

$$J_{data}(w) = \mathbb{E}_{(x,y)\sim \hat{p}_{data}}[\mathcal{L}(y, f(x; w))], \tag{3.14}$$

where $\hat{p}_{data}$ is the empirical distribution and other variables are the same as in (3.13). Then the empirical risk minimization problem turns into the minimization of

$$J_{data}(w) = \mathbb{E}_{(x,y)\sim\hat{p}_{data}}[\mathcal{L}(y, f(x; w))] = \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}(y_i, f(x_i, w)) \tag{3.15}$$

where n is the number of samples in the training set. So, since we can not optimize the risk directly, we optimize the empirical risk. Note that empirical risk minimization is likely to cause overfitting and that the models with high capacity can learn the training data straightforwardly.

There are various cost functions, such as mean squared error, hinge loss, and cross-entropy. It is critical to choose a suitable loss function for the problem. Its importance can be explained as the model parameters are determined by minimizing the loss function. The correct loss function for the problem specifies how well the estimators will be.

3.3. Gradient-Based Optimization

The gradient descent method is one of the widely used optimization techniques. The gradient descent algorithm, if it is possible in practice, enables us to find global minimum of the cost function. We consider two different versions of the gradient descent algorithms: batch gradient descent and stochastic gradient descent. They differ in the way of utilizing the data to succeed to find the global minimum. The main idea is that learning process (finding the optimal parameters) moves in a selected search direction, which is in the opposite direction of the gradient of the cost function, i.e. toward the optimal solution.

3.3.1. Batch Gradient Descent

As in the previous section, once the (training) data is fixed, the cost function J depends only on the parameters w. Hence, $J_{data} = J_{data}(w)$. The (batch) gradient descent algorithm start at a randomly chosen point from the domain of J, say w_0. The

next point w_1 is evaluated as follows:

$$w_1 = w_0 - \eta \nabla_w J_{data}(w_0), \tag{3.16}$$

where η is said to be *learning rate* which is uniformly fixed to be a very small positive number. $J_{data}(w_1)$ is now expected to be closer to global or local minimum value of J_{data} because a slightly scaled inverse direction of the gradient at the point w_0 is subtracted from the vector w_0 to obtain w_1 as well known that the inverse direction of the gradient is in the direction of most rapid decrease of J_{data}. Then, we apply this step further using

$$w_n = w_{n-1} - \eta \nabla_w J_{data}(w_{n-1}) \tag{3.17}$$

to find w_n. The iteration proceeds until consecutive difference $|J_{data}(w_n) - J_{data}(w_{n-1})|$ is sufficiently small and less than an appropriately chosen threshold. The process of evaluating $J_{data}(w_j)$ for $j = 1, 2, \ldots$ is said to be *feed-forward propagation* and the process of finding w_n from the equation (3.17) is said to be *back-propagation.*

Notice that w_0 is randomly initialized at the beginning of the learning process. Normal distribution with zero mean and variance of one is widely used to initialize parameters. Too small or too large initialization of parameters may cause the handicap that the sequence w_n diverges so does the sequence $|J_{data}(w_n) - J_{data}(w_{n-1})|$, which may also be a result of a choice of a large learning rate. In fact, if the learning rate is taken to be very small, then the sequence w_n may converge to an undesired point where J_{data} attains its local minimum values [49]. So, it should be determined appropriately.

Another problem is that the speed of the algorithm can be very slow because of high number parameters and a large data makes J_{data} a function which has gradients that are highly time-consuming to calculate. The algorithm given in the next section is more feasible in the aspect of the speed.

3.3.2. Stochastic Gradient Descent

Having said that computational cost in the standard gradient descent algorithm discussed previously, in the Stochastic Gradient Descent algorithm makes calculation of the gradient faster.

Let $D = \{(x_i, y_i) : i \in \{1, 2, \ldots, n\}\}$ be a (training) data, where x_i's are input and y_i's are output (target) and Ω be a subset of D. We define another cost function which is restricted on Ω.

$$J_\Omega(w) = \sum_{(x,y)\in\Omega} \mathcal{L}(y, f(x, w)). \tag{3.18}$$

The procedure of SGD follows the rules below.

1. Pick a batch size, say b.
2. Choose a random subset of the training data of size b. Call that subset Ω.
3. Apply gradient descent to through $w_n = w_{n-1} - \eta\nabla_w(J_\Omega(w))$ as explained in the previous section.
4. Apply the Step 2.
5. Continue to evaluate $w_n = w_{n-1} - \eta\nabla_w(J_\Omega(w))$ with the initial w_0 chosen to be the last w_n.

Once the procedure above is proceeded, the next iteration goes through from step 2 up to 5 as well as the further iterations. The number of iterations are kept sufficiently high to reach appropriate threshold conditions mentioned in the previous section.

We may choose b to be 1 or any other positive integer less than the number of elements in D. If $b = 1$, it is called *Stochastic Gradient Descent* (SGD). If b equals to the size of D, then it is called *Batch Gradient Descent* which is the treated case of the previous section. If we make a partition of D so that each subset (mini-batch) in the partition are of same size and do the procedure above through these mini-batches, it is

called *Mini-Batch Gradient Descent*. When the all training procedure cover all of the elements of D, we say this is one *epoch*. However, we need to note that the rigorous definitions of these three algorithms in the literature are still not that certain to be sure.

Batch gradient descent has computations over the gradients for each parameter update. However, it is costs much in terms of time consuming for large datasets. The stochastic gradient descent algorithm overcomes this redundancy computations by implementing the parameters update with one randomly selected data from the training set. SGD leads to the frequent updates producing the heavy fluctuations of the cost function, unlike the full batch gradient descent converges directly.

The learning rate is an hyperparameter which can be taken to be any small positive number. So, there are large number of possible choice of learning rate. To find an optimum value of the learning rate is troublesome because it is necessary to check whether a value of learning rate gives better results after a training procedure, which means that we need to train the model for each choice of learning rate. That costs time and energy. To overcome this difficulty in optimization of the learning rate, there are several variations of SGD, containing Adagrad [50], Adadelta [51] and Adam [52]. These approaches try to adapt the learning rate to the model parameters with gradient-based optimization. They mostly facilitate the learning rate selection and enable faster convergence.

3.4. Hyperparameter Optimization

Many hyperparameter parameter optimization techniques exist for traditional models, machine learning, and deep learning approaches. The selection of hyperparameters is crucial for model building. Even a single hyperparameter change can significantly change the model. Furthermore, they significantly affect the model parameters since hyperparameters change the structure of the model. Grid search [53,54], random search [55], and Bayesian optimization [56] are widely used methods to decide

about the hyperparameters. The grid search method, which generates combinations among the possible hyperparameter values, will be discussed below. Let us note that in the traditional approaches, AIC, AICc, and BIC, based on the likelihood, are also useful to decide about model hyperparameters [34,46]. These will also be briefly discussed right in the next subsection.

3.4.1. AIC, AICc and BIC

Akaike's Information Criterion (AIC), which helps us to determine the orders of an ARIMA type model, is defined by

$$\text{AIC} = -2\log(L) + 2(p+q+1), \tag{3.19}$$

where L is the likelihood of the data and $(p+q+1)$ is the number of the parameters in the model. The differencing parameter denoted by d is ignored because it is only used to create a differenced series from an existing one. It can be seen in the literature that the value of d is usually at most two. The decision about d is another issue, and we will not go into its details here.

There are two variations of AIC that are often encountered in the literature: AICc and BIC. Using n as the number of observations, corrected AIC (AICc) is defined by

$$\text{AICc} = \text{AIC} + \frac{2(p+q+1)(p+q+2)}{n-p-q-2} \tag{3.20}$$

and Bayesian Information Criteria (BIC) has the following form:

$$\text{BIC} = \text{AIC} + (\log(n) - 2)(p+q+1). \tag{3.21}$$

While building ARIMA and related models, one is willing to to minimize one of AIC, AICc, and BIC. These criteria enable us to select the values p and q that takes the trade-off between the likelihood and the model orders into account.

3.4.2. Grid Search

Grid search enables us to search over a given subset of the hyperparameters space for finding out the most useful hyperparameters towards our task. The method generates all possible combinations of the hyperparameter values in given domain, independent of the impacts on the elements in the optimization process. In Figure 3.1, we see the visualization of the grid search method when we need to decide about two hyperparameters. In theory each tuple on this grid may yield the best performing model in our task. This results with the obvious drawback that it may lead to many computations and time spent when we have a huge number of combinations. However, in practice, the hyperparameter space can be easily limited since the hyperparameter values are usually independent of each other.

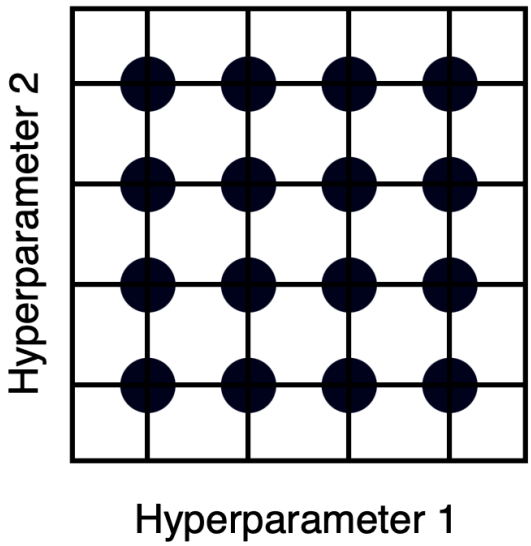

Figure 3.1. The Grid Search Visualization of Two Hyperparameters.

4. PREDICTION INTERVALS BASED ON QUANTILE REGRESSION

The models mentioned so far provide point forecasts, just a single number as a prediction for the next value of the time series. Next we will be looking for an interval forecast for each time in forecast horizon. This is known to be a probabilistic forecast approach, and is getting more and more popular in the literature. Suppose that we have n many observations, and that we want to estimate the $n+1$th observation. The *prediction interval* (PI) approach aims to say that the $n+1$th observation is between the real numbers $a < b$ with a predecided probability.

In the general forecasting literature, confidence intervals and PIs are often confused. The PI is related to a variable not yet observed. On the other hand, the confidence interval is related to a model parameter, which is assumed to be unknown and which is to be estimated from the data [57]. For example, a 90% PI contains the actual value in the interval with a probability of 0.9. A 90% confidence interval includes the actual parameter with a probability of 0.90. In most forecasting applications we look for PIs, that contain the the actual values of future observations.

4.1. Basics on Prediction Intervals

Let us start with the point forecast problem and express the actual value of our time series X_t at time t as the following:

$$X_t = \hat{X}_t + \varepsilon_t, \tag{4.1}$$

where $\hat{X}_t$ is the point forecast at time t and ε_t is corresponding error between the actual value and the estimated value. The PI aims to find a lower bound L_t and an upper bound U_t so that the resulting interval (L_t, U_t) contains the future value with a certain probability. See Figure 4.1 for a demonstration of PIs.

Next, we define the term quantile [58]. For the cumulative distribution function of a random variable X, we write the following expression:

$$F(x) = \mathbb{P}(X \leq x). \tag{4.2}$$

For $0 < \tau < 1$,

$$F^{-1}(\tau) = \inf\{x : F(x) \geq \tau\} \tag{4.3}$$

is called τth *quantile* of a continuous random variable X where F^{-1} is the inverse of the cumulative distribution function.

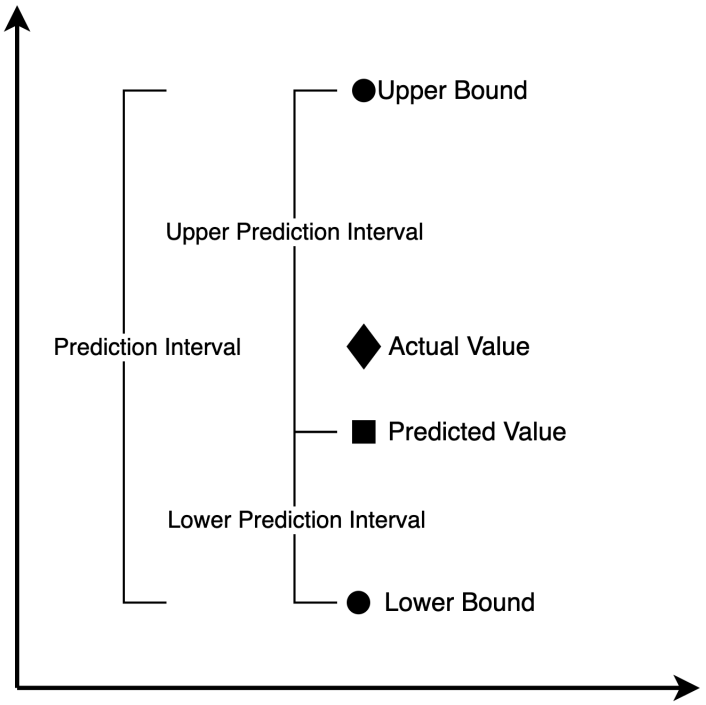

Figure 4.1. The Visualization of the Prediction Interval.

There are various approaches for obtaining prediction intervals. Here we will be focusing on quantile regression averaging (QRA) [59], quantile regression neural network (QRNN) [60] and (QRLSTM) [61] in detail. Furthermore, we will also briefly discuss historical PI, also called historical simulation in [62], and distribution-based PI as benchmark methods.

4.2. Benchmark Models

Let us start with how to obtain historical PI. Suppose we have $t-1$ many observations. We determine some positive integer m as a corridor size, and use m data points along with the corresponding point forecasts along this corridor. According to (4.1), after calculating the corresponding errors as

$$\varepsilon_i = X_i - \hat{X}_i \tag{4.4}$$

for $i = t-m, \ldots, t-2, t-1$, we order ε_i's from smallest to largest as

$$\varepsilon_{(1)} < \varepsilon_{(2)} < \ldots < \varepsilon_{(m)}. \tag{4.5}$$

Let $\vec{\varepsilon} = (\varepsilon_{(1)}, \varepsilon_{(2)}, \ldots, \varepsilon_{(m)})$. Pick τth and $(1-\tau)$th quantiles in $\vec{\varepsilon}$, and call them ε_L and ε_U, respectively. Then for the upper and lower bounds of the PI, we use lower bound error ε_L and upper bound error ε_U, and the resulting $(1-2\tau)$th PI turns out to be

$$(\hat{X}_t + \varepsilon_L, \hat{X}_t + \varepsilon_U). \tag{4.6}$$

Note that the corridor size m above should not be too long as it may cause some problems since otherwise there is the risk of fixing the prediction interval length throughout the prediction period.

Next we briefly go over distribution-based PIs [63]. Since the time series models, such as ARIMA models, are driven by Gaussian noise, the PIs can be obtained by quantiles of this distribution. That is, the lower and upper bounds of PIs are calculated by this distribution. Similarly, in [64], student-t distribution is used for the noise distribution.

Let us explain the distribution-based PIs by using an AR(1) example. As we mentioned before in (3.1), AR(1) is formed via a Gaussian noise Z_t. Our goal is to find an interval (L_t, U_t) such that

$$\mathbb{P}(X_t \in (L_t, U_t)) = \mathbb{P}(L_t < \phi X_{t-1} + Z_t < U_t) = 1 - 2\tau. \tag{4.7}$$

Since we have previous observation X_{t-1} and focus on the distribution of Z_t, we write (4.7) as:

$$\mathbb{P}(L_t - \phi X_{t-1} < Z_t < U_t - \phi X_{t-1}) = 1 - 2\tau.$$

Recalling that $Z_t \sim \mathcal{N}(0, \sigma^2)$, and doing the standardization $z_t = \frac{Z_t}{\sigma}$, we obtain

$$\mathbb{P}\Big(\frac{L_t - \phi X_{t-1}}{\sigma} < z_t < \frac{U_t - \phi X_{t-1}}{\sigma}\Big) = 1 - 2\tau.$$

By using the values in z-table, we set

$$\frac{L_t - \phi X_{t-1}}{\sigma} = z_\tau$$
$$\frac{U_t - \phi X_{t-1}}{\sigma} = z_{1-\tau}$$

and then, we obtain (L_t, U_t) for a $(1 - 2\tau)$ PI where

$$L_t = \phi X_{t-1} + \sigma z_\tau$$
$$U_t = \phi X_{t-1} + \sigma z_{1-\tau}.$$

As mentioned above, the historical PIs and distribution-based PIs are the primary methods to compare with other approaches. Bootstrapped PIs is a another related tool [65, 66], but we will not be going into this here.

4.3. Quantile Regression and Related Methodologies

Before we discuss the Quantile Regression Averaging (QRA) approach, we note that it is based on quantile regression which was first proposed by [58]. Let us discuss the quantile regression first. Below we will be working with the pinball loss function which is defined on real numbers by

$$\rho_\tau(e) = e(\tau - I_{(e<0)}) = \begin{cases} \tau|e| & e \geq 0 \\ (1-\tau)|e| & e < 0 \end{cases} \tag{4.8}$$

where $\tau \in (0, 1)$ and I is the indicator function taking the values 0 and 1. Then the error function of the QR is calculated by

$$E_\tau = \frac{1}{N} \sum_{t=1}^{N} \rho_\tau(X_t - \hat{X}_t) \tag{4.9}$$

where X_t and $\hat{X}_t$ are real and predicted values at time t for $t = 1, \ldots, N$.

The aim of QR problem is to obtain $\hat{x}$ minimizing the expected loss. So we are willing to minimize

$$\mathbb{E}_{\rho_\tau}[X - \hat{x}] = (\tau - 1) \int_{-\infty}^{\hat{x}} (x - \hat{x}) dF(x) + \tau \int_{\hat{x}}^{\infty} (x - \hat{x}) dF(x). \tag{4.10}$$

By differentiating (4.10) and setting it to zero, we obtain:

$$(1 - \tau) \int_{-\infty}^{\hat{x}} dF(x) - \tau \int_{\hat{x}}^{\infty} dF(x) = F(\hat{x}) - \tau = 0. \tag{4.11}$$

This either yields a set of solutions $\{\hat{x} : F(\hat{x}) = \tau\}$ or a unique solution $\hat{x} = F^{-1}(\tau)$. Now, we write the QR problem as

$$\mathcal{Q}_{X_t}(\tau | R_t) = \beta_\tau R_t \tag{4.12}$$

where $\mathcal{Q}_{X_t}(\tau|\cdot)$ means the conditional τth quantile of the random variable X_t, R_t refers to the features or regressors and β_τ is the parameter vector corresponding to the τth quantile. The parameters of τth quantile are then computed by minimizing the loss function as follows:

$$\min_{\beta_\tau} \Big\{ \sum_{\{t: X_t \geq \beta_\tau R_t\}} \tau |X_t - \beta_\tau R_t| + \sum_{\{t: X_t < \beta_\tau R_t\}} (1 - \tau) |X_t - \beta_\tau R_t| \Big\}. \tag{4.13}$$

Note that this is equivalent to the minimization of

$$\min_{\beta_\tau} \Big\{ \sum_{t} (\tau - I_{(X_t < \beta_\tau R_t)}) (X_t - \beta_\tau R_t) \Big\}. \tag{4.14}$$

In quantile regression averaging (QRA) setting proposed by [59], our features are the point forecasts of the models. As an example, if we have k individual forecasting models, then our feature vector is $R_t = [1, \hat{X}_{1,t}, \ldots, \hat{X}_{k,t}]$. The number of individual forecasts can be decided arbitrarily; for example, it can use a certain number of best results [67], or dimensionality reduction techniques can be employed [68]. Let us emphasize that the QRA method does not combine the PIs of the individual forecasting models.

Now, we discuss the QRA problem and the difference between QR and QRA approaches. The problem in QRA can be expressed as

$$F_{X_t}^{-1}(\tau|\hat{X}_t) = w_t\hat{X}_t + F_{\varepsilon_t}^{-1}(\tau|\hat{X}_t) \tag{4.15}$$

where w_t is the weight vector, ε_t is the error or residual, and F_{ε_t} is the cumulative distribution function of ε_t at time t. Thus, the minimization problem in QRA becomes

$$\min_{w_t}\Big\{\sum_t(\tau - I_{(X_t<w_t\hat{X}_t)}(X_t - w_t\hat{X}_t)\Big\}. \tag{4.16}$$

Since τ is arbitrary, the QRA method provides point estimates for any quantile, and the proper combination of quantiles gives the chance to form a PI. We note that strongly correlated exogenous variables should be avoided in the QRA method. In addition the individual point forecasts in QRA should be kept small to obtain a computationally efficient model.

4.4. QRNN and QRLSTM

The QRNN, proposed by [60], is based on an artificial neural network with multilayers, with the given input and outputs. As noted in Chapter 2, the ANN optimizes the parameters using the gradients of the loss function. However, in QRNN, we need another loss function to optimize the model parameters and obtain the point forecasts of the τth quantile. In [69], the loss function is based on the Huber norm, proposed by [70]. On the other hand, we focus on the pinball loss function defined by (4.8) as the loss function of QRNN algorithm to reduce the model complexity.

To generate the QRNN structure, we write the following ANN expression

$$\hat{X}_\tau(t) = f\Big(\sum_{i=1}^{n}\tanh\Big(\sum_{j=1}^{k}X_j(t)w_{ji}^{(h)} + b_i^{(h)}\Big)w_i^{(o)} + b_i^{(o)}\Big) \tag{4.17}$$

with the given input vector $X_j(t)$ for $j = 1, 2, ..., N$ and its output $\hat{X}_\tau(t)$ of τth quantile. Here, f is the output function, $w_{ji}^{(h)}$ and $w_i^{(o)}$ are the weights of the hidden layer and output layer, $b_i^{(h)}$ and $b_i^{(o)}$ are the hidden layer and output layer biases, respectively. The input layer has the linear function with the tanh activation function. Similarly,

in the hidden layer, the linear function is used. Furthermore, the sigmoid function can be preferred as the output function.

On the other hand, Quantile Regression Long Short-Term Memory (QRLSTM), improved by [61,71], is based on QR and LSTM, with a similar approach in QRNN. As in the traditional LSTM, the update of the parameters is achieved by the backpropagation algorithm. However, the loss function is written suitably for the QR approach. With the altered learning process of the model, the parameters are updated with the backpropagation of the new loss function. The model is trained two times using the τth and $(1-\tau)$th quantiles, and the results are the point estimations of τth and $(1-\tau)$th quantiles. Thus, the prediction interval of QRLSTM is obtained.

4.5. Prediction Interval Evaluation Metrics

In this section, we briefly discuss the unconditional coverage and Winkler score [72] (proposed by Winkler [73]) that are used to measure the reliability of prediction intervals.

Let us start with the simple and common approach, unconditional coverage (UC), to evaluate PIs. The empirical coverage focuses on the indicator series written by

$$I_t = \begin{cases} 0 & \text{if } X_t \notin (L_t, U_t) \\ 1 & \text{if } X_t \in (L_t, U_t). \end{cases} \tag{4.18}$$

Unconditional coverage then means that the average of actual values in the forecasted PIs, which can be written as

$$\text{UC} = \frac{1}{n} \sum_{i=1}^{n} I_i. \tag{4.19}$$

A second statistic used to evaluate the quality of the PIs is Winkler score. The goal is to choose the narrowest PIs with large coverage. This is important because it is not enough for PIs to contain the actual values but they should be thin enough in order to be useful in applications. In particular, when the length of PI is quite large,

it may be obvious that the actual value is in this PI. The Winkler score is now defined as the following:

$$W_t = \begin{cases} \delta_t & \text{if } X_t \in (L_t, U_t) \\ \delta_t + \frac{2}{\alpha}(L_t - X_t) & \text{if } X_t < L_t \\ \delta_t + \frac{2}{\alpha}(X_t - U_t) & \text{if } X_t > U_t \end{cases} \tag{4.20}$$

where L_t and U_t are lower and upper bounds of PI, respectively, δ_t is the length of the interval, $U_t - L_t$, and X_t is the actual value for $(1-\alpha)100\%$ PI. Thus, the Winkler score enables a penalty when the actual value is out of PI and a reward when we have a narrow PI. So, a smaller Winkler score indicates we have a better PI.

5. EXPERIMENTS AND RESULTS

5.1. Point Forecast Evaluation Metrics

To compare and evaluate the point forecasts, various loss functions are used. Some of these are the Mean Absolute Percentage Error (MAPE) and its variations, MAPE_{100} and MAPE_{250}, and Mean Absolute Error (MAE).

We give expressions for the evaluation metrics as mentioned above. Let us start with MAPE which is defined by

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\Big|\frac{y_i - p_i}{y_i}\Big|100, \tag{5.1}$$

where y_i's are actual values and p_i's represent predicted values. In MAPE_{100} and MAPE_{250}, we are interested in cases only where actual values are greater than 100 and 250, respectively. In other words, if we have 75 as a real value, we do not use it in calculating MAPE_{100}. Also, we define MAE by setting

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - p_i|, \tag{5.2}$$

where y_i's and p_i's are the same as in (5.1). By comparing the losses of different methods, we may reach at a more meaningful comparison of the models' performances.

5.2. Case Study: California

5.2.1. PeMS Data

We use Caltrans Performance Management System (PeMS) data set. The content of this data, collected with approximately 30,000 sensors and detectors, is quite extensive. In this section we discuss how the data is prepared and which parts of it are used. We examine the content of the data in 17 features which are timestamp, station, district, freeway, the direction of travel, station length, samples, total flow, average

occupancy, average speed, among others.

Timestamp, district, station, total flow, and average occupancy are the most important features among these 17 features for data preparation and modeling in this study. We can define traffic flow as the number of vehicles passing through the selected lane between timestamps. For example, when there is 180 as a value of total flow in PeMS, 180 vehicles have passed through the selected part of the road in five minutes.

Table 5.1. The Station Information in PeMS.

Station	**Min Occupancy**	**Max Occupancy**	**Avg Occupancy**	**Length**
715918	0.0130	0.1744	0.0832	0.885
716933	0.0094	0.5840	0.1388	0.565
717087	0.0000	0.5746	0.0664	0.330
718442	0.0024	0.9989	0.1548	0.300

We have selected District 7 as the region of interest, and then have chosen certain stations which are to be specified below. The data contains many rows collected from many points at the same hour. While modeling the traffic flow, we need one point to focus on spatially. In this study, we have chosen stations 716933, and 717087 and examined the traffic flow at these stations as a time series. While deciding which stations to choose, we paid attention to the stations in the literature so that we have an opportunity to compare the results with them.

As a side note, we also examined the stations 715918 and 718442 other than 716933, and 717087. However, the station 715918 has very small occupancy and thus too similar patterns emerge between days. For the station 718442, the data shows abnormal behaviour on an exploratory data analysis, possible from an undesirable noise. This prevents doing sane forecasts for station 715918. For these reasons we do not include our results for stations 716933, and 717087, and just focus on the other

two below.

Original PeMS data is based on 5-minute periods. We have the data for nine months, from the beginning of January 2021 to the end of September 2021. We use 15-minute data for all stations since 15-minute data is an optimal timestamp in the literature [4–7, 9, 10, 20, 74, 75]. Especially in [9], it has been mentioned that 15 minutes is the best choice for a prediction interval study.

The 15-minute data is obtained by averaging three consecutive 5-minute data that do not intersect. In the end, the data with 15-minute has 26208 rows. In the models, we focus on traffic flow in the data. The length of the training data differs according to the models because some models contain seasonal parts. For this reason, we decided to fix the test size, which includes the last 7863 rows, and the rest of the data is kept for training.

5.2.2. Results of Point Forecasts

Before explaining the point forecast results, we need to tell a little bit about the setup. We previously mentioned that we predict the following traffic behaviors by focusing only on the traffic flow. Traditional models and LSTM are already based on previous observations. On the other hand, we need to determine the features for the SVR and XGBoost models. Since traffic flow has seasonal features, model features are determined using daily and weekly seasonality trends.

Letting X_t be the traffic flow at time t, the model features are summarized in Table 5.2. The features are the same for both stations, 716933 and 717087. Since some of the models in the Table 5.2 are trained with seasonal features, these model names are denoted as seasonal SVR (SSVR), and seasonal LSTM (SLSTM) as mentioned in Chapter 2. Moreover, to the best of our knowledge seasonal XGBoost (SXGBoost) has not been studied in the literature. Inspired by the SSVR and SLSTM methods, we introduce and implement the SXGBoost method below.

Table 5.2. The Traffic Flow Forecasting Models Features in PeMS.

Models	**Features**
S-ARIMA	$X_{t-673}, X_{t-672}, X_{t-97}, X_{t-96}$
SSVR	$X_{t-673}, X_{t-672}, X_{t-671}, X_{t-97}, X_{t-96}, X_{t-95}, X_{t-2}, X_{t-1}$
SLSTM	$X_{t-673}, X_{t-672}, X_{t-671}, X_{t-97}, X_{t-96}, X_{t-95}, X_{t-2}, X_{t-1}$
SXGBoost	$X_{t-673}, X_{t-672}, X_{t-671}, X_{t-97}, X_{t-96}, X_{t-95}, X_{t-2}, X_{t-1}$

The notation of S-ARIMA is used to avoid confusion with SARIMA. Note that in order to form the S-ARIMA model with both weekly and daily components, the features specified in Table 5.2 were added to the ARIMA model as exogenous variables.

Let us briefly discuss how we decide on these features. Weekly and daily seasonality makes sense for traffic flow since the days and hours in which people are active/inactive in general follow a pattern. For example, this may lead to a constant experience of a traffic congestion on a particular road at the same hours every Monday. Moreover, similar situations can be experienced at the same time as the previous day, such as encountering traffic in the morning as many people work on weekdays. However, using only daily seasonality may cause problems with the difference between weekdays and weekends.

Next we will discuss the point forecast results for PeMS data and do the necessary comparisons. We apply ARIMA(1,0,0) to model residuals after applying nonlinear models in hybrid approaches SLSTMARIMA, SSVRARIMA, and SXGBoostARIMA. As mentioned at the beginning of current chapter, we use MAE, MAPE, MAPE_{100} and MAPE_{250} to evaluate the model results.

To train ARIMA variations, AR(5), ARMA(1,1) and ARIMA(4,1,0) are used for station 716933. Similarly, for station 717087, we use AR(5), ARMA(5,4) and ARIMA(3,1,5). To obtain S-ARIMA model, we use ARIMA(1,1,2) with exogenous variables on Table 5.2 for both stations.

On the other hand, after applying a grid search for SSVR model, we obtain the RBF kernel among linear and RBF kernels, and 100 for the relevant C value among and 0.1, 1, 10, 100 for both stations. Also, we utilize the Adam optimizer to optimize SLSTM model, and we recall that the learning rate is already optimized by the Adam as mentioned in Chapter 2. To generate the SLSTM model, the softmax activation function and a linear layer follow the LSTM layer. For SXGBoost approach, we use the squared error as a criterion and 1000 as a number of trees.

We are now ready to state our results. For station 716933, ARIMA type models and naive forecasts seem to be close to each other. Among the seasonal approaches, SSVR has the best model performance for station 716933. Also, since the station 716933 has a minimal flow higher than 100, the values of MAPE and MAPE_{100} are the same.

In hybrid models, we have better results when residuals are well modeled. We apply the ARIMA(1,1,0) model to the errors for both stations. Among the hybrid method attempts, the SSVRARIMA approach shows the best results, as seen in Table 5.5. We present the point forecasts for station 716933 as a plot in Figures 5.1, 5.2, 5.3 below.

Table 5.3. The Evaluation of the Traditional Approaches for Station 716933.

Models	**MAE**	**MAPE**	**MAPE_{100}**	**MAPE_{250}**
Naive Method	31.702	7.366	7.366	6.816
Average Method	125.414	42.494	42.494	18.765
AR	36.613	8.047	8.047	7.692
ARMA	34.706	7.740	7.740	7.334
ARIMA	31.701	7.366	7.366	6.816

Table 5.4. The Evaluation of the Seasonal Models for Station 716933.

Models	**MAE**	**MAPE**	**MAPE$_{100}$**	**MAPE$_{250}$**
S-ARIMA	29.394	6.649	6.649	6.248
SLSTM	26.842	6.253	6.253	5.619
SSVR	26.015	6.001	6.001	5.469
SXGBoost	26.435	6.099	6.099	5.560

Table 5.5. The Evaluation of the Hybrid Models for Station 716933.

Models	**MAE**	**MAPE**	**MAPE$_{100}$**	**MAPE$_{250}$**
SLSTMARIMA	26.842	6.253	6.253	5.619
SSVRARIMA	26.015	6.001	6.001	5.469
SXGBoostARIMA	26.435	6.099	6.099	5.560

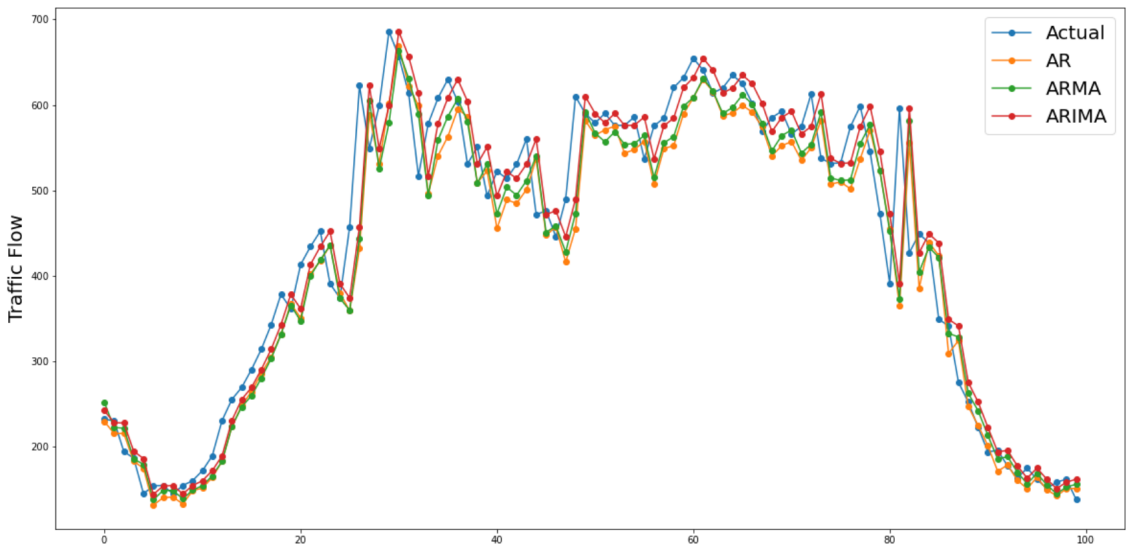

Figure 5.1. The Comparison of the Traditional Approaches for Station 716933.

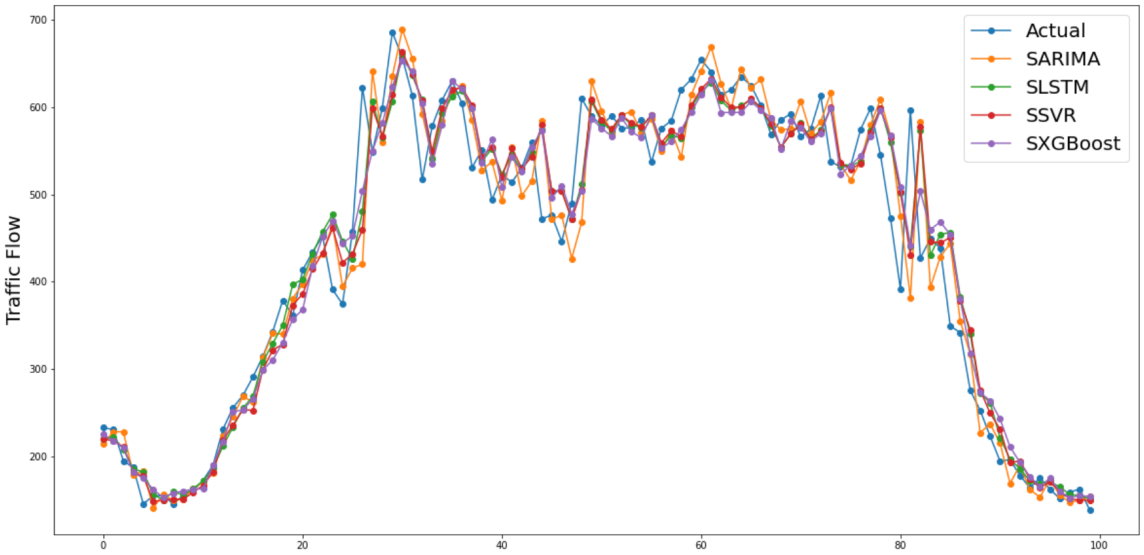

Figure 5.2. The Comparison of the Seasonal Models for Station 716933.

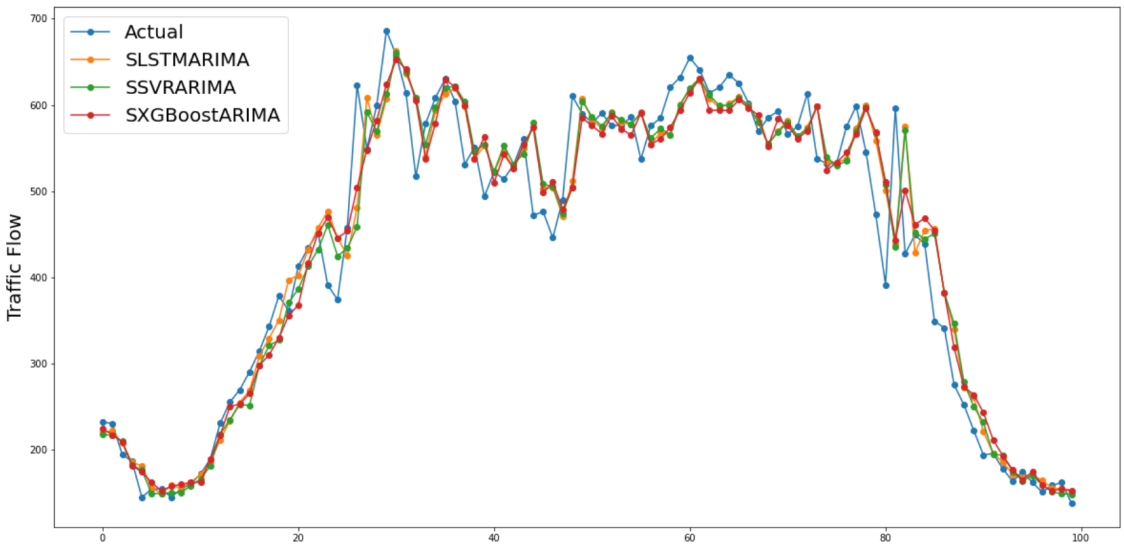

Figure 5.3. The Comparison of the Hybrid Models for Station 716933.

As in Table 5.6, similar to station 716933 results, the ARIMA model has better results for station 717087 among traditional approaches. For seasonal models and hybrid methods, SSVR and SSVRARIMA have best performances on the station 717087, as seen respectively in Table 5.7 and Table 5.8. The point forecasts as a plot for station 717087 can be found in Figures 5.4, 5.5 and 5.6. So, from these we may conclude that SSVR has captured the seasonal and autoregressive features best in our experiments.

Table 5.6. The Evaluation of the Traditional Approaches for Station 717087.

Models	**MAE**	**MAPE**	**$\mathbf{MAPE_{100}}$**	**$\mathbf{MAPE_{250}}$**
Naive Method	21.410	8.254	7.409	5.913
Average Method	125.021	78.546	38.700	21.486
AR	22.241	8.116	7.392	6.267
ARMA	24.499	9.038	8.061	6.900
ARIMA	21.234	8.229	7.314	5.874

Table 5.7. The Evaluation of the Seasonal Models for Station 717087.

Models	**MAE**	**MAPE**	**$\mathbf{MAPE_{100}}$**	**$\mathbf{MAPE_{250}}$**
S-ARIMA	20.024	7.376	6.567	5.781
SLSTM	18.291	7.581	6.118	4.978
SSVR	17.444	7.200	5.775	4.770
SXGBoost	17.499	6.979	5.809	4.851

Table 5.8. The Evaluation of the Hybrid Models for Station 717087.

Models	**MAE**	**MAPE**	**$\mathbf{MAPE_{100}}$**	**$\mathbf{MAPE_{250}}$**
SLSTMARIMA	18.291	7.581	6.118	4.978
SSVRARIMA	17.444	7.200	5.774	4.769
SXGBoostARIMA	17.500	6.979	5.810	4.851

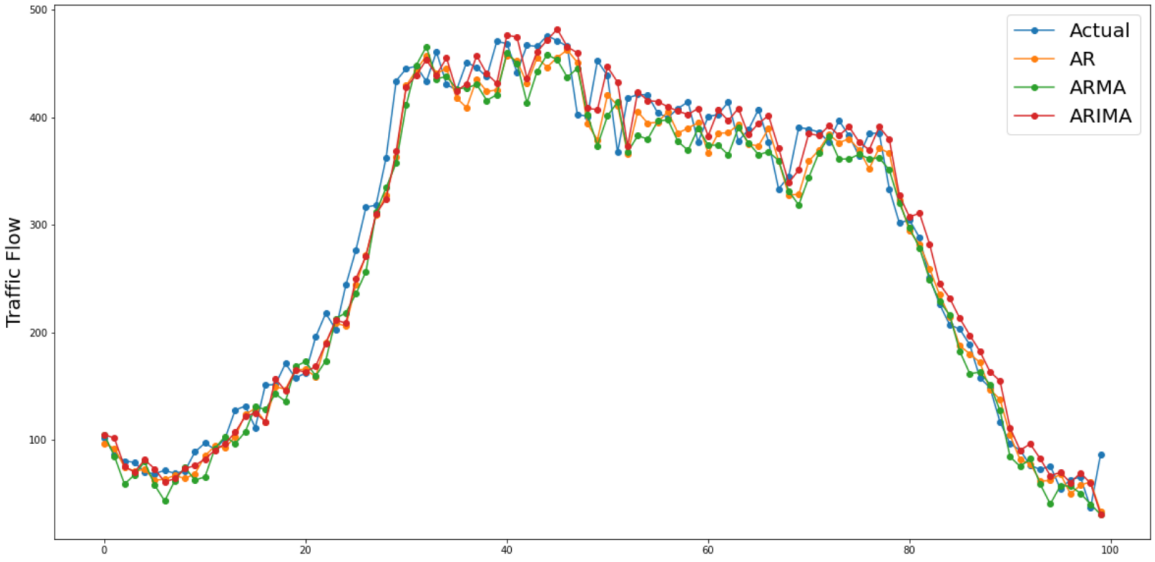

Figure 5.4. The Comparison of the Traditional Approaches for Station 717087.

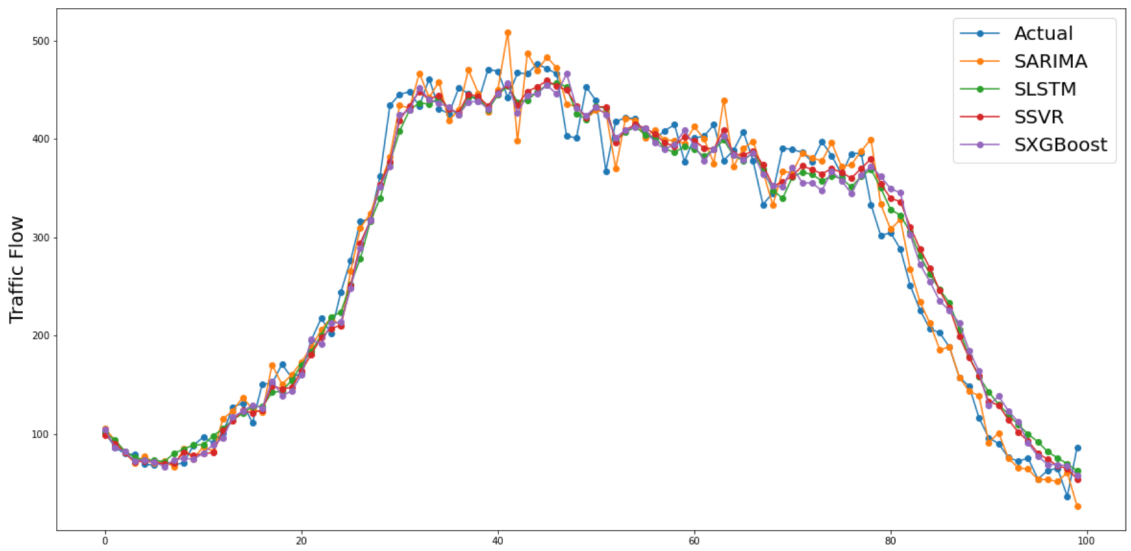

Figure 5.5. The Comparison of the Seasonal Models for Station 717087.

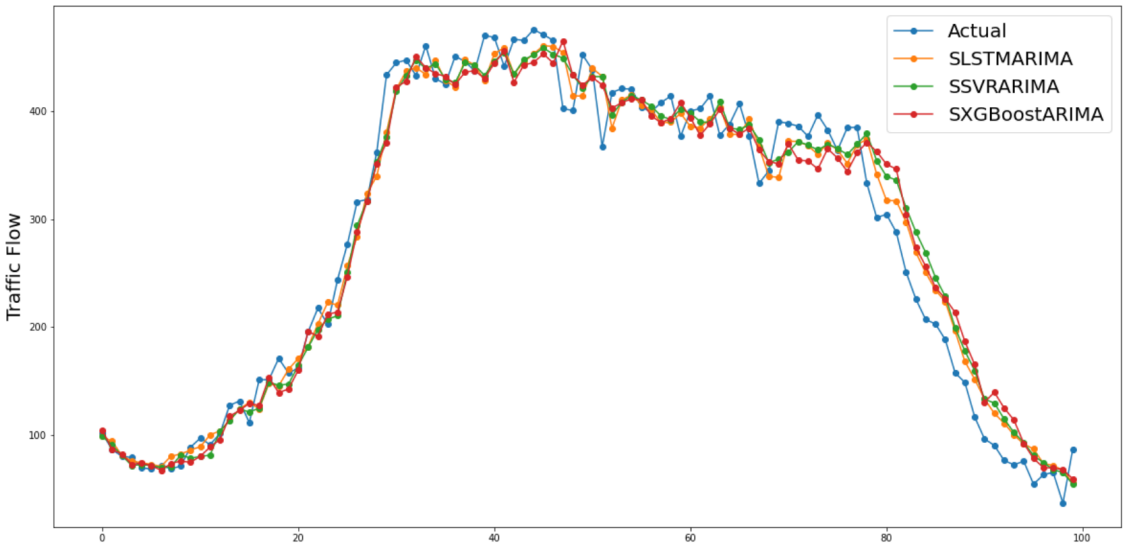

Figure 5.6. The Comparison of the Hybrid Models for Station 717087.

5.3. Case Study: İstanbul

5.3.1. İstanbul Traffic Data

For our study on İstanbul traffic flow, we had access to a traffic data of approximately one and a half years, namely, from January 2020 to the end of April 2021. The traffic flow data has 1-hour timestamps. Our data set consists of traffic flow data in which the locations are defined by geohashes.

A geohash is a geocoding system encoding a location with letters and digits. We see the geohash areas in İstanbul traffic flow data in Figure 5.7. The latitude and longitude values given in the data indicate the midpoint of this geohash area. If the geohash value length is 6, the area consists of 0.0027 latitude error and 0.0055 longitude error. We draw these boundaries using the latitude and longitude errors and the latitude and longitude of the midpoint.

It is impossible to focus on a specific road in some geohash areas since a geohash may contain very complex road systems. Therefore, one should select a suitable geohash which mainly consists of a single road system, preferably whose occupancy is not very low. Keeping this in mind, as sketched in Figure 5.8, we have selected a geohash whose code is sxk9wk. The traffic flow forecasts of this road can be interpreted because the green area around the road in Figure 5.8 refers to the forest.

İstanbul traffic flow data has more missing values compared to PeMS data in our study. Since 1-hour timestamps are larger in short-term approaches, these missing data become more critical. Thus, in the next section, we investigate the effect of missing data in the literature and present the solutions to deal with the missing points in data.

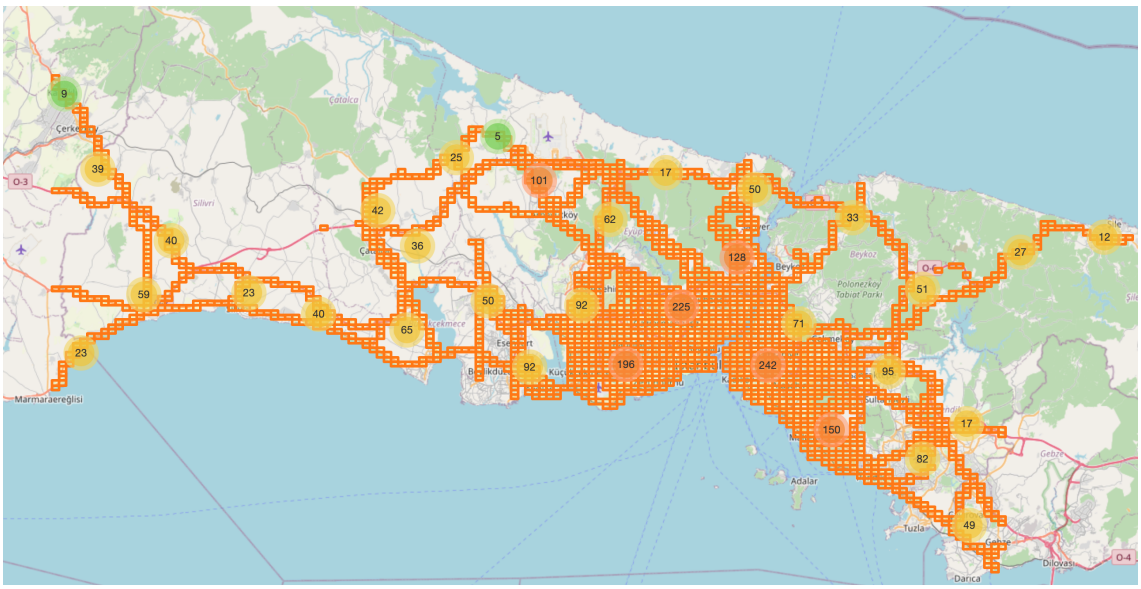

Figure 5.7. The Geohash Boundaries of Traffic Flow Data in İstanbul.

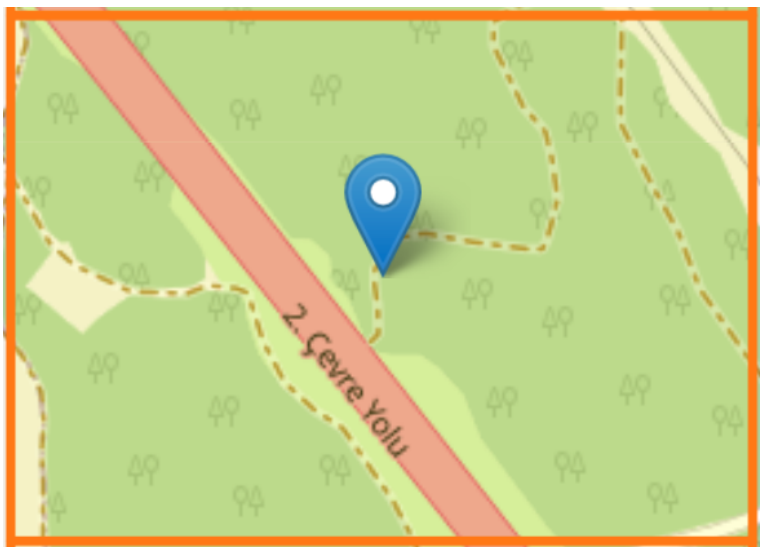

Figure 5.8. The Selected Geohash Boundary: Kavacık Junction.

5.3.2. Effect of Missing Data

In İstanbul traffic data, around 5% of the data is missing, so one needs to pay special attention filling in these properly. More specifically, in our data set, we have 664 missing data points, and further there is significant time difference among them. Also note that since we have hourly traffic flow data for İstanbul, forecasting model results are adversely affected.

Let us give some pointers to the literature on taking care of the missing data problem. The adverse effects of missing data on model performances are indicated, for example, in [76, 77]. A possible completion of the missing data with an AR model is advocated in [78]. All these indicate that it is essential to deal with the missing data for obtaining proper forecasting mechanisms.

Also, [79] uses a technique called the vector autoregressive imputation method (VAR-IM) for dealing with missing data. [80] evaluates the result of ARIMA time series analysis with missing data points by applying four different methods: deletion, mean substitution, mean of adjacent observations, and maximum likelihood estimation.

We use three different completion methods below. The first two of these are based on mean of properly chosen available observations, and the third one is on deletion of the missing values. Regarding the former two, we use two different strategies. In the first one we replace the missing value with the mean of the available data from the same hour. Also,in the second one the missing data is replaced by the mean of the available data from both the same hour and the same day. Also let us note that we also had experimentations on the missing data by completion via an ARIMA model, but we decided not to keep those results in this thesis since we also have forecastings based on ARIMA which could be problematic.

After completing the missing data points, we need train and test sets to learn the models and obtain forecast results. In our case approximately 70 percent of the data is

the training set, and the rest is used as the test set. Since some models have seasonal parts, the training set size differs from model to model. However, we fix the test set to be the last 3500 rows of data.

5.3.3. Results of Point Forecasts

Similar to the PeMS data setting in California, we use methods based on previous observation to obtain point forecast. The features of the seasonal models contain both weekly and daily trends. Noting that the İstanbul traffic flow data has 1-hour time stamps, the components are as in listed in Table 5.9 for S-ARIMA, SLSTM, SSVR and SXGBoost. To apply both weekly and daily seasonality to the S-ARIMA, we add the features to the ARIMA as exogenous variables.

One other goal of ours in İstanbul traffic flow data is to compare the forecast results by filling or deleting missing data points with different techniques. In the model application part, the hyperparameters of the models should remain the same for all three data so that we interpret them correctly. Furthermore, the features in the models should be the same as in Table 5.9 for these three cases.

Table 5.9. The Traffic Flow Forecasting Models Features in İstanbul.

Models	**Features**
S-ARIMA	$X_{t-169}, X_{t-168}, X_{t-25}, X_{t-24}$
SSVR	$X_{t-169}, X_{t-168}, X_{t-167}, X_{t-25}, X_{t-24}, X_{t-23}, X_{t-2}, X_{t-1}$
SLSTM	$X_{t-169}, X_{t-168}, X_{t-167}, X_{t-25}, X_{t-24}, X_{t-23}, X_{t-2}, X_{t-1}$
SXGBoost	$X_{t-169}, X_{t-168}, X_{t-167}, X_{t-25}, X_{t-24}, X_{t-23}, X_{t-2}, X_{t-1}$

Some notes on hyperparameter selection is in order. We used AR(5), ARMA(1,1) and ARIMA(0,1,3). Also, similar to the PeMS case, ARIMA(1,1,2) with seasonal exogenous variables is used to obtain the S-ARIMA model with weekly and daily

components. The SSVR method has the RBF kernel and 100 for the value of C, again by using aforementioned hyperparameter optimization techniques. To train SLSTM model, we utilize 0.01 as the learning rate, MSE as the criterion, Adam optimizer and 2000 as the number of epochs. It has the same structure as the one formed for the PeMS data. Moreover, SXGBoost uses the squared error to update the parameters and 1000 as the number of estimators which refers to the maximum number of trees.

Before giving the evaluation results of the models, we present the loss change during the epochs for the SLSTM model. The Adam optimizer provides us a stochastic gradient approach, and as mentioned Chapter in 3, it gives faster optimization and convergence. Since it is a stochastic approach occurrence of fluctuations as in Figure 5.9 are expected.

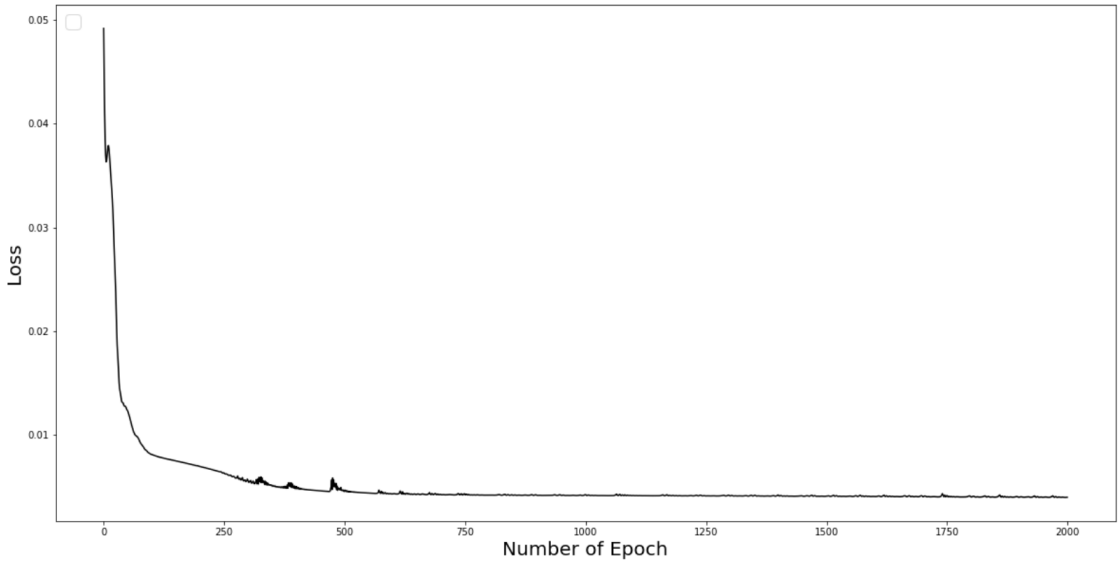

Figure 5.9. The Loss Plot of the SLSTM Model for İstanbul Data with Missing Points Completed with Mean of the Same Days and Hours.

Table 5.10. The Evaluation of the Traditional Approaches for İstanbul Data with Missing Points Completed by Deletion Method.

Models	**MAE**	**MAPE**	**$\mathbf{MAPE_{100}}$**	**$\mathbf{MAPE_{250}}$**
Naive Method	42.684	31.036	22.967	17.581
Average Method	104.516	134.423	34.316	36.851
AR	45.418	28.870	23.713	21.841
ARMA	46.256	29.060	24.188	22.600
ARIMA	42.692	31.046	22.971	17.585

Table 5.11. The Evaluation of the Seasonal Models for İstanbul Data with Missing Points Completed by Deletion Method.

Models	**MAE**	**MAPE**	**$\mathbf{MAPE_{100}}$**	**$\mathbf{MAPE_{250}}$**
S-ARIMA	38.921	27.645	20.601	16.529
SLSTM	31.361	26.084	16.023	11.652
SSVR	30.768	22.780	16.455	11.994
SXGBoost	31.629	24.797	16.783	12.014

Table 5.12. The Evaluation of the Hybrid Models for İstanbul Data with Missing Points Completed by Deletion Method.

Models	**MAE**	**MAPE**	**$\mathbf{MAPE_{100}}$**	**$\mathbf{MAPE_{250}}$**
SLSTMARIMA	31.356	26.030	16.041	11.616
SSVRARIMA	30.723	22.728	16.416	12.061
SXGBoostARIMA	31.695	24.839	16.833	12.001

We present the point forecast results in Tables 5.10 through 5.18. The point forecast results for the İstanbul traffic flow data after deleting the missing values are summarized in Tables 5.10, 5.11 and 5.12. The results indicate that the seasonal approaches outperform the traditional ones. Furthermore, the hybrid approaches enables us to improve the seasonal methods further, and we see that the SSVRARIMA is the best mechanism among the ones we implemented. Next we present Tables 5.13, 5.14 and 5.15, which make it clear that we obtain improved results if we complete the missing data by using the mean of same hours, instead of simply deleting them.

Table 5.13. The Evaluation of the Traditional Approaches for İstanbul Data with Missing Points Completed with Mean of the Same Hours.

Models	**MAE**	**MAPE**	**$\mathbf{MAPE_{100}}$**	**$\mathbf{MAPE_{250}}$**
Naive Method	40.466	29.969	21.690	16.743
Average Method	101.837	128.599	34.837	36.257
AR	50.593	31.640	26.658	25.312
ARMA	45.133	28.572	23.500	22.605
ARIMA	40.465	29.969	21.690	16.743

Table 5.14. The Evaluation of the Seasonal Models for İstanbul Data with Missing Points Completed with Mean of the Same Hours.

Models	**MAE**	**MAPE**	**$\mathbf{MAPE_{100}}$**	**$\mathbf{MAPE_{250}}$**
S-ARIMA	29.047	21.775	15.154	12.364
SLSTM	23.601	17.652	12.373	10.107
SSVR	22.836	17.155	12.024	9.710
SXGBoost	24.710	18.511	12.981	10.599

Table 5.15. The Evaluation of the Hybrid Models for İstanbul Data with Missing Points Completed with Mean of the Same Hours.

Models	**MAE**	**MAPE**	**$\mathbf{MAPE_{100}}$**	**$\mathbf{MAPE_{250}}$**
SLSTMARIMA	23.644	17.670	12.391	10.107
SSVRARIMA	22.850	17.162	12.036	9.706
SXGBoostARIMA	24.732	18.516	12.997	10.614

Table 5.16. The Evaluation of the Traditional Approaches for İstanbul Data with Missing Points Completed with Mean of the Same Days and the Same Hours.

Models	**MAE**	**MAPE**	**$\mathbf{MAPE_{100}}$**	**$\mathbf{MAPE_{250}}$**
Naive Method	40.678	29.919	21.783	16.660
Average Method	103.542	129.963	35.301	36.870
AR	52.294	32.285	27.493	26.359
ARMA	47.101	29.421	24.629	23.556
ARIMA	42.391	31.077	23.077	17.101

Table 5.17. The Evaluation of the Seasonal Models for İstanbul Data with Missing Points Completed with Mean of the Same Days and the Same Hours.

Models	**MAE**	**MAPE**	**$\mathbf{MAPE_{100}}$**	**$\mathbf{MAPE_{250}}$**
S-ARIMA	31.579	23.330	16.908	13.140
SLSTM	23.470	17.936	12.365	9.529
SSVR	22.231	16.751	11.773	9.111
SXGBoost	23.977	18.133	12.610	9.925

Table 5.18. The Evaluation of the Hybrid Models for İstanbul Data with Missing Points Completed with Mean of the Same Days and the Same Hours.

Models	**MAE**	**MAPE**	**MAPE$_{100}$**	**MAPE$_{250}$**
SLSTMARIMA	23.475	17.908	12.382	9.516
SSVRARIMA	22.233	16.752	11.777	9.104
SXGBoostARIMA	23.987	18.138	12.617	9.920

An analysis of the results we provided clearly indicates the effect of the missing data points properly. Filling the missing values using the mean observation in the same days and the hours gives the best forecasting results in our framework. Figures 5.10, 5.11 and 5.12 provide visualizations of the different forecasting results.

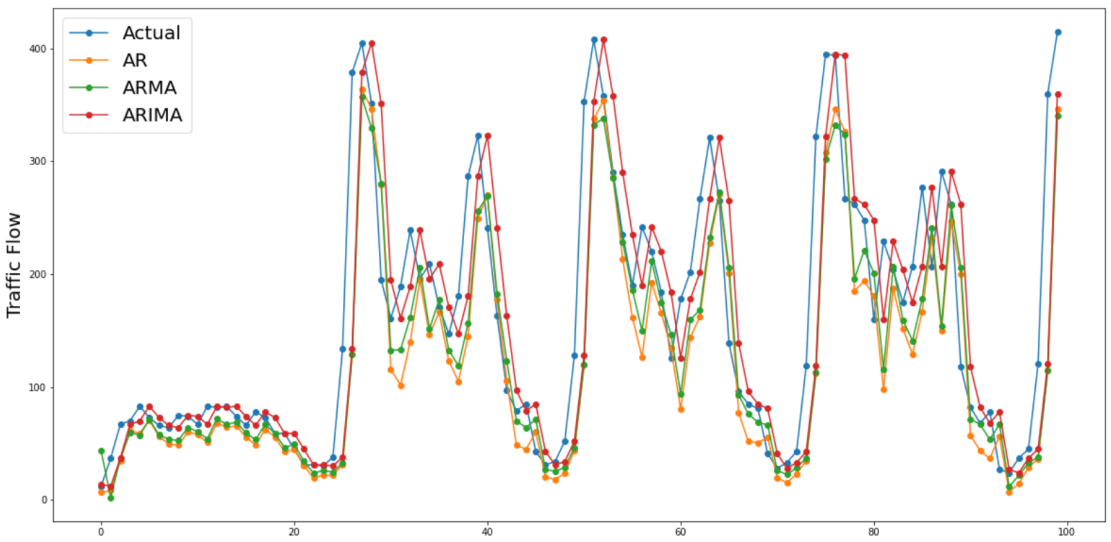

Figure 5.10. The Comparison of the Traditional Approaches for İstanbul Data with Missing Points Completed with Mean of the Same Days and the Same Hours.

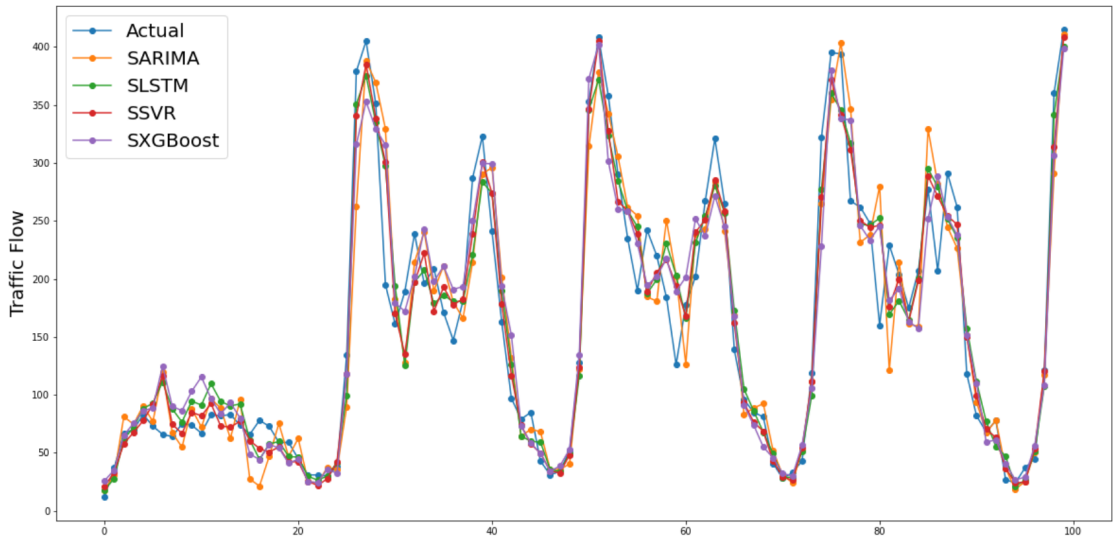

Figure 5.11. The Comparison of the Seasonal Model Approaches for İstanbul Data with Missing Points Completed with Mean of the Same Days and the Same Hours.

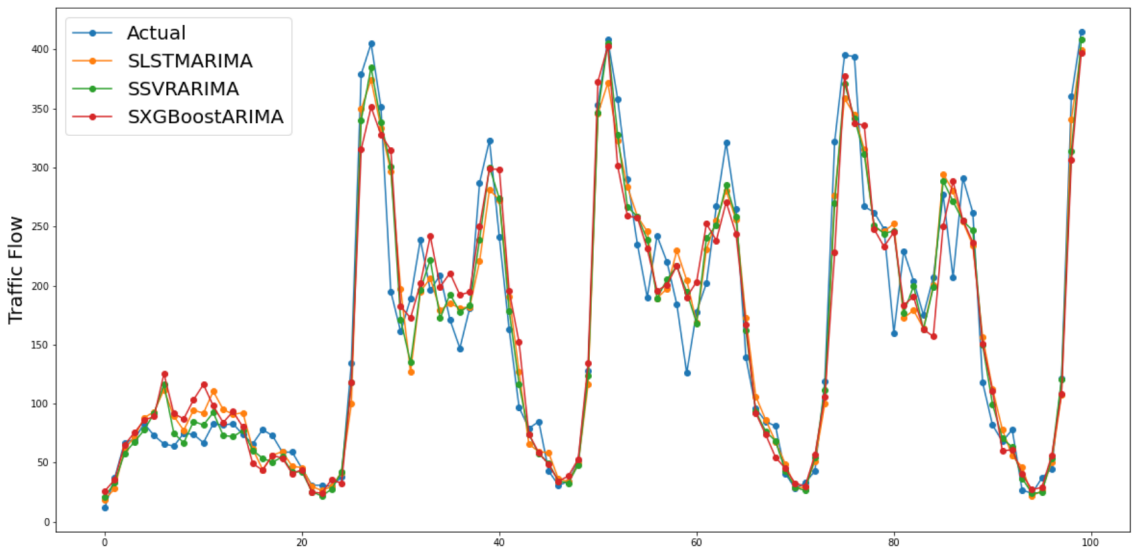

Figure 5.12. The Comparison of the Hybrid Models for İstanbul Data with Missing Points Completed with Mean of the Same Days and the Same Hours.

5.3.4. Results of Prediction Intervals

In this section we provide the results of our experimentations on Istanbul data set for obtaining 90% PIs. Different methods we have are compared in terms of the unconditional coverage and Winkler score.

Since the traffic flow data with the mean observations at the same days and hours has better forecasting results than the other two, our experimentation focused on generating the PIs for this framework. In Tables 5.19 and 5.20, the evaluation results of the PIs are summarized. Since the historical PI approach uses the forecasting results, we present them for all models using to obtain the point forecasts in Table 5.19. These show that the PI corresponding to the SSVR approach has the best performance among the historical ones. On the other hand, the results of distribution-based PIs has 0.52 for the unconditional coverage and 211.537 for Winkler score.

Moreover, Table 5.20 includes the details regarding the approaches QRA, QRSNN and QRSLSTM. The QRSNN refers to Quantile Regression Seasonal Neural Network since it contains seasonal features. In order to obtain the QRA one, we use the prediction results of SLSTM, SSVR and SXGBoost, which are the best three models of the point forecasts. In the QRSLSTM, we use the same architecture with the SLSTM and we obtain the 90 % PIs. The features of the QRSNN and QRSLSTM models are the same. They are given in Table 5.9 and the same components are used as the SLSTM model to obtain QRSNN and QRSLSTM.

We obtain the best PIs with the QRSLSTM approach when compared the unconditional coverage of the results. Moreover, we have the best PIs in the QRSNN when compared to Winkler scores. Since we compare the length of PIs using Winkler score, we conclude that the approaches QRSNN and QRSLSTM has similar results. We present the plot of the PIs of QRA, QRSNN and QRSLSTM results in Figures 5.13, 5.14 and 5.15.

Table 5.19. The Evaluation of the Historical Prediction Intervals.

Models	Unconditional Coverage	Winkler Score
AR	0.83	321.137
ARMA	0.82	324.283
ARIMA	0.80	373.314
S-ARIMA	0.82	237.124
SLSTM	0.80	174.928
SSVR	0.81	171.993
SXGBoost	0.81	179.943

Table 5.20. The Evaluation of the Quantile Regression Based Prediction Intervals.

Models	Unconditional Coverage	Winkler Score
QRA	0.82	178.646
QRSNN	0.89	141.938
QRLSTM	0.92	143.751

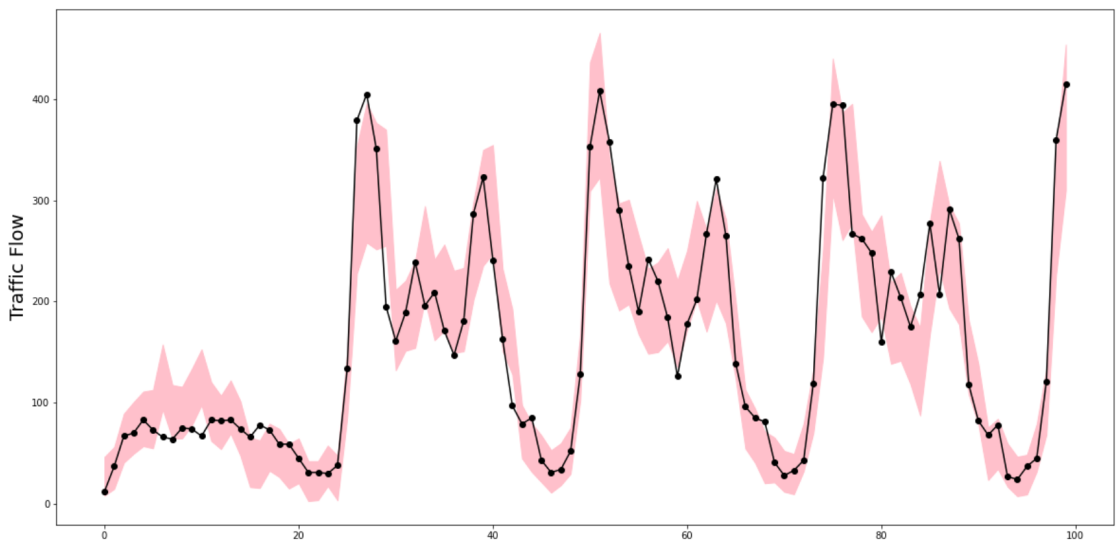

Figure 5.13. The Visualization of QRA Prediction Intervals.

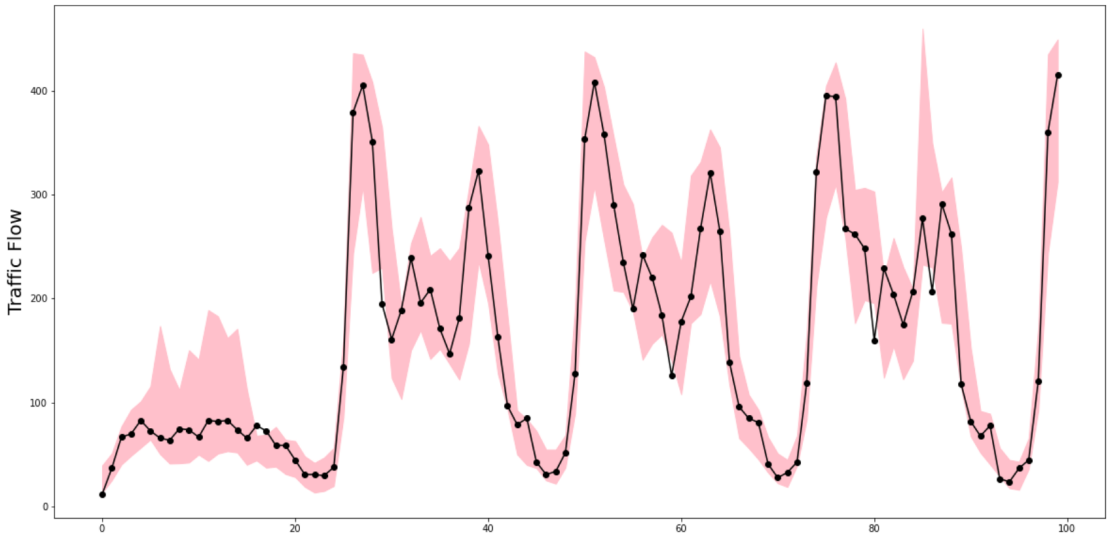

Figure 5.14. The Visualization of QRSNN Prediction Intervals.

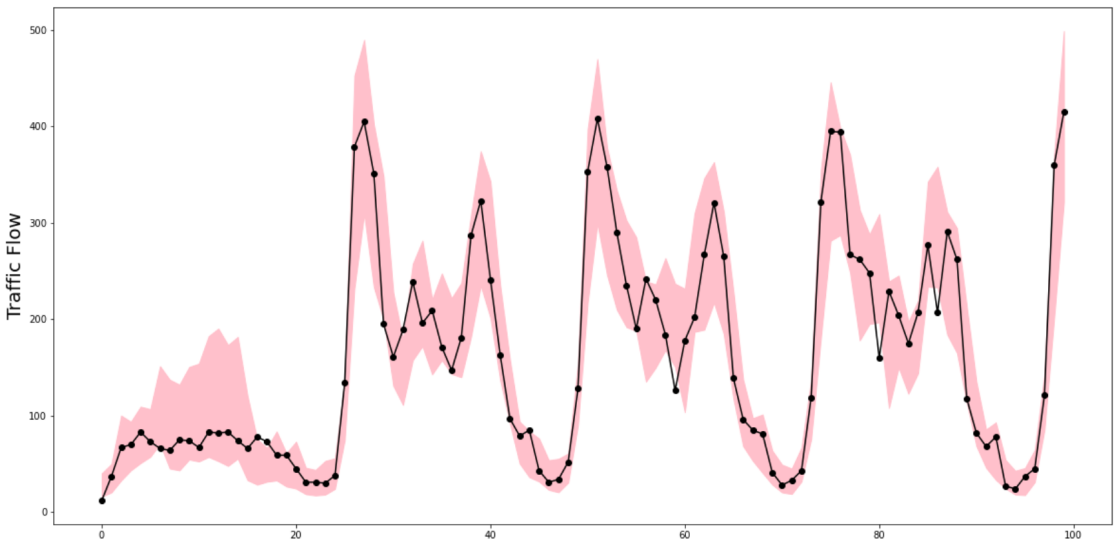

Figure 5.15. The Visualization of QRSLSTM Prediction Intervals.

6. CONCLUSION

In this thesis we studied short-term traffic flow forecasting methodologies by analyzing the existing literature, and introducing new approaches. In addition to point forecasting, interval estimation is also discussed. SXGBoost and QRSLSTM, which have not been previously proposed, have been developed. Both of these two focus on the seasonal trends of the time series. During this study, besides the well-known California dataset, we worked on the traffic flow data of İstanbul which had additional challenges such as the missing data problem. We hope that the approaches in this thesis will be useful in improving the time series methods for traffic flow forecasting.

REFERENCES

1. Goodfellow, I., Y. Bengio and A. Courville, *Deep Learning*, MIT Press, 2016.

2. Dong, H., L. Jia, X. Sun, C. Li and Y. Qin, "Road Traffic Flow Prediction with a Time-Oriented ARIMA Model", *Fifth International Joint Conference on INC, IMS and IDC*, Seoul, Korea, 2009.

3. Kumar, S. V. and L. Vanajakshi, "Short-Term Traffic Flow Prediction Using Seasonal ARIMA Model with Limited Input Data", *European Transport Research Review*, Vol. 7, No. 3, pp. 1–9, 2015.

4. Lin, S.-L., H.-Q. Huang, D.-Q. Zhu and T.-Z. Wang, "The Application of Space-Time ARIMA Model on Traffic Flow Forecasting", *International Conference on Machine Learning and Cybernetics*, Baoding, China, 2009.

5. Lippi, M., M. Bertini and P. Frasconi, "Short-Term Traffic Flow Forecasting: An Experimental Comparison of Time-Series Analysis and Supervised Learning", *IEEE Transactions on Intelligent Transportation Systems*, Vol. 14, No. 2, pp. 871–882, 2013.

6. Shekhar, S. and B. M. Williams, "Adaptive Seasonal Time Series Models for Forecasting Short-Term Traffic Flow", *Transportation Research Record*, Vol. 2024, No. 1, pp. 116–125, 2007.

7. Williams, B. M. and L. A. Hoel, "Modeling and Forecasting Vehicular Traffic Flow as a Seasonal ARIMA Process: Theoretical Basis and Empirical Results", *Journal of Transportation Engineering*, Vol. 129, No. 6, pp. 664–672, 2003.

8. Hong, W.-C., "Traffic Flow Forecasting by Seasonal SVR with Chaotic Simulated Annealing Algorithm", *Neurocomputing*, Vol. 74, No. 12-13, pp. 2096–2107, 2011.

9. Smith, B. L. and M. J. Demetsky, "Traffic Flow Forecasting: Comparison of Modeling Approaches", *Journal of Transportation Engineering*, Vol. 123, No. 4, pp. 261–266, 1997.

10. Smith, B. L., B. M. Williams and R. K. Oswald, "Comparison of Parametric and Nonparametric Models for Traffic Flow Forecasting", *Transportation Research Part C: Emerging Technologies*, Vol. 10, No. 4, pp. 303–321, 2002.

11. Cao, J., G. Cen, Y. Cen and W. Ma, "Short-Term Highway Traffic Flow Forecasting Based on XGBoost", *15th International Conference on Computer Science & Education*, Delft, Netherlands, 2020.

12. Dong, X., T. Lei, S. Jin and Z. Hou, "Short-Term Traffic Flow Prediction Based on XGBoost", *IEEE 7th Data Driven Control and Learning Systems Conference*, Enshi, China, 2018.

13. Lartey, B., A. Homaifar, A. Girma, A. Karimoddini and D. Opoku, "XGBoost: A Tree-Based Approach for Traffic Volume Prediction", *IEEE International Conference on Systems Man and Cybernetics*, Melbourne, Australia, 2021.

14. Lu, W., Y. Rui, Z. Yi, B. Ran and Y. Gu, "A Hybrid Model for Lane-Level Traffic Flow Forecasting Based on Complete Ensemble Empirical Mode Decomposition and Extreme Gradient Boosting", *IEEE Access*, Vol. 8, pp. 42042–42054, 2020.

15. Chen, H., H. Ai, Z. Yang, W. Yang, Z. Ye and D. Dong, "An Improved XGBoost Model Based on Spark for Credit Card Fraud Prediction", *IEEE 5th International Symposium on Smart and Wireless Systems within the Conferences on Intelligent Data Acquisition and Advanced Computing Systems*, Dortmund, Germany, 2020.

16. Sun, F., F. Fang, R. Wang, B. Wan, Q. Guo, H. Li and X. Wu, "An Impartial Semi-Supervised Learning Strategy for Imbalanced Classification on VHR Images", *Sensors*, Vol. 20, No. 22, pp. 6699–6719, 2020.

17. Fernandes, B., F. Silva, H. Alaiz-Moretón, P. Novais, C. Analide and J. Neves, "Traffic Flow Forecasting on Data-Scarce Environments Using ARIMA and LSTM Networks", *World Conference on Information Systems and Technologies*, Galicia, Spain, 2019.

18. Fu, R., Z. Zhang and L. Li, "Using LSTM and GRU Neural Network Methods for Traffic Flow Prediction", *31st Youth Academic Annual Conference of Chinese Association of Automation*, Wuhan, China, 2016.

19. Nourani, V. and N. Behfar, "Multi-Station Runoff-Sediment Modeling Using Seasonal LSTM Models", *Journal of Hydrology*, Vol. 601, pp. 126672–126686, 2021.

20. Vinayakumar, R., K. Soman and P. Poornachandran, "Applying Deep Learning Approaches for Network Traffic Prediction", *International Conference on Advances in Computing, Communications and Informatics*, Manipal, India, 2017.

21. Zhao, Z., W. Chen, X. Wu, P. C. Chen and J. Liu, "LSTM Network: A Deep Learning Approach for Short-Term Traffic Forecast", *IET Intelligent Transport Systems*, Vol. 11, No. 2, pp. 68–75, 2017.

22. Liao, C.-W., I.-C. Wang, K.-P. Lin and Y.-J. Lin, "A Fuzzy Seasonal Long Short-Term Memory Network for Wind Power Forecasting", *Mathematics*, Vol. 9, No. 11, pp. 1178–1193, 2021.

23. Jurafsky, D. and J. H. Martin, *Speech and Language Processing*, NJ.: Prentice Hall, 2020.

24. Borovikov, I. and M. Sadovsky, "A Relative Information Approach to Financial Time Series Analysis Using Binary N-Grams Dictionaries", ArXiv:1308.2732, 2013.

25. Du, Q., F. Yin and Z. Li, "Base Station Traffic Prediction Using XGBoost-LSTM with Feature Enhancement", *IET Networks*, Vol. 9, No. 1, pp. 29–37, 2020.

26. Luo, X., D. Li, Y. Yang and S. Zhang, "Spatiotemporal Traffic Flow Prediction with KNN and LSTM", *Journal of Advanced Transportation*, 2019.

27. Che, J. and J. Wang, "Short-Term Electricity Prices Forecasting Based on Support Vector Regression and Autoregressive Integrated Moving Average Modeling", *Energy Conversion and Management*, Vol. 51, No. 10, pp. 1911–1917, 2010.

28. Che, J. and J. Wang, "Short-Term Load Forecasting Using a Kernel-Based Support Vector Regression Combination Model", *Applied Energy*, Vol. 132, pp. 602–609, 2014.

29. Chi, Z. and L. Shi, "Short-Term Traffic Flow Forecasting Using ARIMA-SVM Algorithm and R", *5th International Conference on Information Science and Control Engineering*, Zhengzhou, China, 2018.

30. Li, K.-L., C.-J. Zhai and J.-M. Xu, "Short-Term Traffic Flow Prediction Using a Methodology Based on ARIMA and RBF-ANN", *Chinese Automation Congress*, Jinan, China, 2017.

31. Du, S., T. Li, X. Gong and S.-J. Horng, "A Hybrid Method for Traffic Flow Forecasting Using Multimodal Deep Learning", *International Journal of Computational Intelligence Systems*, Vol. 13, No. 1, pp. 85–97, 2018.

32. Brockwell, P. J. and R. A. Davis, *Introduction to Time Series and Forecasting*, Springer, 2002.

33. Fuller, W. A., *Introduction to Statistical Time Series*, John Wiley & Sons, 2009.

34. Hyndman, R. J. and G. Athanasopoulos, *Forecasting: Principles and Practice*, OTexts, 2018.

35. Williams, B. M., "Multivariate Vehicular Traffic Flow Prediction: Evaluation of ARIMAX Modeling", *Transportation Research Record*, Vol. 1776, No. 1, pp. 194–

200, 2001.

36. Haykin, S., *Neural Networks and Learning Machines*, Pearson Education India, 2009.

37. Vapnik, V. N., *The Nature of Statistical Learning*, Springer, 1995.

38. Welling, M., "Support Vector Regression", *Department of Computer Science, University of Toronto, Toronto*, 2004.

39. Pai, P.-F., K.-P. Lin, C.-S. Lin and P.-T. Chang, "Time Series Forecasting by a Seasonal Support Vector Regression Model", *Expert Systems with Applications*, Vol. 37, No. 6, pp. 4261–4265, 2010.

40. Chen, T. and C. Guestrin, "Xgboost: A Scalable Tree Boosting System", *Proceedings of the 22nd ACM International Conference on Knowledge Discovery and Data Mining*, San Francisco, CA, USA, 2016.

41. Friedman, J., T. Hastie and R. Tibshirani, "Additive Logistic Regression: A Statistical View of Boosting", *Annals of Statistics*, pp. 337–374, 2000.

42. Friedman, J. H., "Greedy Function Approximation: A Gradient Boosting Machine", *Annals of Statistics*, pp. 1189–1232, 2001.

43. Myles, A. J., R. N. Feudale, Y. Liu, N. A. Woody and S. D. Brown, "An Introduction to Decision Tree Modeling", *Journal of Chemometrics: A Journal of the Chemometrics Society*, Vol. 18, No. 6, pp. 275–285, 2004.

44. Natekin, A. and A. Knoll, "Gradient Boosting Machines: A Tutorial", *Frontiers in Neurorobotics*, Vol. 7, pp. 21–42, 2013.

45. Rosenblatt, F., "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain", *Psychological Review*, Vol. 65, No. 6, pp. 386–408,

1958.

46. Shumway, R. H., D. S. Stoffer and D. S. Stoffer, *Time Series Analysis and Its Applications*, Vol. 3, Springer, 2000.

47. Bottou, L., "Stochastic Gradient Descent Tricks", *Neural networks: Tricks of the Trade*, pp. 421–436, Springer, 2012.

48. Bottou, L., F. E. Curtis and J. Nocedal, "Optimization Methods for Large-Scale Machine Learning", *Siam Review*, Vol. 60, No. 2, pp. 223–311, 2018.

49. Liu, K., L. Ziyin and M. Ueda, "Noise and Fluctuation of Finite Fearning Rate Stochastic Gradient Descent", *International Conference on Machine Learning*, 2021.

50. Duchi, J., E. Hazan and Y. Singer, "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization", *Journal of Machine Learning Research*, Vol. 12, pp. 2121–2159, 2011.

51. Zeiler, M. D., "Adadelta: An Adaptive Learning Rate Method", ArXiv:1212.5701, 2012.

52. Kingma, D. P. and J. Ba, "Adam: A Method for Stochastic Optimization", ArXiv:1412.6980, 2014.

53. Alibrahim, H. and S. A. Ludwig, "Hyperparameter Optimization: Comparing Genetic Algorithm against Grid Search and Bayesian Optimization", *IEEE Congress on Evolutionary Computation*, Kraków, Poland, 2021.

54. Liashchynskyi, P. and P. Liashchynskyi, "Grid Search, Random Search, Genetic Algorithm: A Big Comparison for NAS", ArXiv:1912.06059, 2019.

55. Bergstra, J. and Y. Bengio, "Random Search for Hyperparameter Optimization",

Journal of Machine Learning Research, Vol. 13, pp. 281–305, 2012.

56. Feurer, M. and F. Hutter, "Hyperparameter Optimization", *Automated Machine Learning*, pp. 3–33, Springer, Cham, 2019.

57. Hyndman, R. J., "The Difference Between Prediction Intervals and Confidence Intervals", *Hyndsight Blog*, 2013.

58. Koenker, R., *Quantile Regression*, Cambridge University Press, 2005.

59. Nowotarski, J. and R. Weron, "Computing Electricity Spot Price Prediction Intervals Using Quantile Regression and Forecast Averaging", *Computational Statistics*, Vol. 30, No. 3, pp. 791–803, 2015.

60. Cannon, A. J., "Quantile Regression Neural Networks: Implementation in R and Application to Precipitation Downscaling", *Computers & Geosciences*, Vol. 37, No. 9, pp. 1277–1284, 2011.

61. Cao, J., L. Yang, C. Qian and J. Yu, "Short Term Electricity Price Probability Density Prediction Based on QRLSTM Algorithm in Spot Market", *IEEE Sustainable Power and Energy Conference*, Perth, Australia, 2021.

62. Alexander, C., *Value-at-Risk Models*, John Wiley & Sons, 2008.

63. Misiorek, A., S. Trueck and R. Weron, "Point and Interval Forecasting of Spot Electricity Prices: Linear vs. Nonlinear Time Series Models", *Studies in Nonlinear Dynamics & Econometrics*, Vol. 10, No. 3, pp. 1–36, 2006.

64. Panagiotelis, A. and M. Smith, "Bayesian Density Forecasting of Intraday Electricity Prices Using Multivariate Skew t Distributions", *International Journal of Forecasting*, Vol. 24, No. 4, pp. 710–727, 2008.

65. Clements, M. P. and J. H. Kim, "Bootstrap Prediction Intervals for Autoregressive

Time Series", *Computational Statistics & Data Analysis*, Vol. 51, No. 7, pp. 3580–3594, 2007.

66. Efron, B. and R. J. Tibshirani, *An Introduction to the Bootstrap*, CRC Press, 1994.

67. Nowotarski, J. and R. Weron, "Recent Advances in Electricity Price Forecasting: A Review of Probabilistic Forecasting", *Renewable and Sustainable Energy Reviews*, Vol. 81, pp. 1548–1568, 2018.

68. Maciejowska, K., J. Nowotarski and R. Weron, "Probabilistic Forecasting of Electricity Spot Prices Using Factor Quantile Regression Averaging", *International Journal of Forecasting*, Vol. 32, No. 3, pp. 957–965, 2016.

69. Chen, C., "A Finite Smoothing Algorithm for Quantile Regression", *Journal of Computational and Graphical Statistics*, Vol. 16, No. 1, pp. 136–164, 2007.

70. Huber, P. J., "Robust Statistics", *New York: John Wiley*, 1981.

71. Zhou, Y. and S. Liang, "LSTM Based Quantile Regression Method for Holiday Load Forecasting", *IEEE 4th Conference on Energy Internet and Energy System Integration*, Wuhan, China, 2020.

72. Gneiting, T. and A. E. Raftery, "Strictly Proper Scoring Rules, Prediction and Estimation", *Journal of the American Statistical Association*, Vol. 102, No. 477, pp. 359–378, 2007.

73. Winkler, R. L., "A Decision-Theoretic Approach to Interval Estimation", *Journal of the American Statistical Association*, Vol. 67, No. 337, pp. 187–191, 1972.

74. Huang, W., G. Song, H. Hong and K. Xie, "Deep Architecture for Traffic Flow Prediction: Deep Belief Networks with Multitask Learning", *IEEE Transactions on Intelligent Transportation Systems*, Vol. 15, No. 5, pp. 2191–2201, 2014.

75. Zhang, K., L. Wu, Z. Zhu and J. Deng, "A Multitask Learning Model for Traffic Flow and Speed Forecasting", *IEEE Access*, Vol. 8, pp. 80707–80715, 2020.

76. Rankin, E. D. and J. C. Marsh, "Effects of Missing Data on the Statistical Analysis of Clinical Time Series", *Social Work Research and Abstracts*, 1985.

77. Tawn, R., J. Browell and I. Dinwoodie, "Missing Data in Wind Farm Time Series: Properties and Effect on Forecasts", *Electric Power Systems Research*, Vol. 189, pp. 106640–106647, 2020.

78. Anava, O., E. Hazan and A. Zeevi, "Online Time Series Prediction with Missing Data", *International Conference on Machine Learning*, Lille, France, 2015.

79. Bashir, F. and H.-L. Wei, "Handling Missing Data in Multivariate Time Series Using a Vector Autoregressive Model-Imputation (VAR-IM) Algorithm", *Neurocomputing*, Vol. 276, pp. 23–30, 2018.

80. Velicer, W. F. and S. M. Colby, "A Comparison of Missing Data Procedures for ARIMA Time Series Analysis", *Educational and Psychological Measurement*, Vol. 65, No. 4, pp. 596–615, 2005.