Improving Time Series Forecasting Performance by Fuzzy Decision Fusion

Sonia Malvina Djeuda Nzouapet

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Engineering

Eastern Mediterranean University
September 2020
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Ali Hakan Ulusoy
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science in Computer Engineering.

Prof. Dr. H. Işık Aybay
Chair, Department of Computer Engineering

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Science in Computer Engineering.

Assoc. Prof. Dr. Mehmet Bodur
Supervisor

Examining Committee

1. Assoc. Prof. Dr. Adnan Acan

2. Assoc. Prof. Dr. Mehmet Bodur

3. Assoc. Prof. Dr. Mehtap Köse Ulukök

ABSTRACT

Time series data can be collected in many domains including econometric, signal processing, weather forecasting, and earthquake prediction. Accurate prediction of time series prices is essential for investors, meteorologist, or statisticians. Forecasting of the financial time series has intrinsic complexity due to uncertainties of factors that affect it. In this study, better forecasting of the financial stock market time series movements is targeted using the closing prices of the stock market. In this work, our objective is to implement a set of well-known financial time series forecasting models such as Autoregressive-Integrating-Moving-Average (ARIMA), Exponential Smoothing, Support Vector Regression (SVR), Long-Short Time Memory (LSTM), and to merge the forecasted decision by using fuzzy knowledge-based decision system. The difference of this thesis compared to the previous works is mainly the expert-decided membership functions instead of clustering in building the fuzzy rule base. An experimental demonstration has been carried out on the S&P 500 index using the closing prices of this Index. The results shows that the fuzzy decision fusion procedure gives lower cumulative absolute prediction error than cumulative error of forecasts of each individual model.

Keywords: time series, time series forecasting, fuzzy decision fusion, fuzzy logic system, fuzzy rule generation.

ÖZ

Zaman serisi verileri, ekonometri, sinyal işleme, hava durumu tahmini, deprem tahmini dahil olmak üzere birçok alanda üretilir ve kullanılır. Yatırımcılar, meteorologlar ve istatistikçiler için serilerin doğru tahmin edilmesi çok önemlidir. Zaman serilerini etkileyen faktörlerin belirsizliklerinden dolayı, mali zaman serilerinin tahmini içsel karmaşıklığa sahiptir. Bu çalışmada finansal borsa zaman serisi hareketlerinin tahmininin, yalnızca borsa endeksinin kapanış fiyatları ve hacmi kullanılarak iyileştirilmesi hedeflenmiştir. Önerilen yöntem, Otoregressif-Bütünleşik-Hareket-Ortalaması (ARIMA), Üstel yumuşatma (ESM) Destek Vektör Regresyonu (SVR), Uzun Kısa Süreli Bellek (LSTM) gibi mevcut zaman serisi verilerinden bir finansal zaman serisinin gelecekteki fiyatını tahmin etme yeteneğine sahip bir dizi model kullanmakta, ve eğitim veri seti üzerinden öğrendiği bulanık karar birleştirme kuralları bilgi tabanını kullanarak tahmin edilen değerlerin en iyisini seçebilmektedir. Bu çalışmada, önceki tez çalışmalarından farklı olarak, karar birleştirme için bulanık bir kural tabanı oluşturmada öbekleme yöntemleri yerine standart bulanık yöntemler kullanmaya odaklanılmıştır. S&P 500 endeksinden elde edilen veriler üzerinde, bu hisse endeksinin kapanış fiyatları zaman dizisi kullanılarak önerilen yöntemin deneysel bir gösterimi sunulmuştur. Sonuçlar, önerilen bulanık karar füzyon birleştirme yönteminin, bireysel modellerin her bir tahmininin kümülatif hatalarından daha düşük kümülatif mutlak tahmin hatası sağladığını göstermiştir.

Anahtar Kelimeler: zaman serileri, zaman serisi tahmini, bulanık karar füzyonu, bulanık mantık sistemi, bulanık kural üretimi.

ACKNOWLEDGMENT

The completion of this thesis was made possible with the contribution of many whose assistance was a milestone in the realization of this project.

I will first of all like to thank the all mighty God for the gift of life, wisdom, and his constant guidance and motivation throughout my life and more specifically my academic years.

My profound gratitude to my parents Mr. & Mrs. Nzouapet for their unfailing support and continuous encouragement.

I will like to extend my thanks to my supervisor Assoc. Prof. Dr. Mehmet Bodur for his availability, orientation, and advice whenever I needed it.

I extend my thanks to all those who near or far have made this work possible.

Finally, my thanks go to the authorities of Eastern Mediterranean University for providing me with the necessary resources and environment for the completion of this project.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ADF	Augmented Dickey-Fuller
API	Application Programming Interface
AR	Autoregressive
ARIMA	Autoregressive Integrated Moving Average
ARMA	Autoregressive Moving Average
CAE	Cumulative Absolute Error
EMH	Efficient-Market Hypothesis
ETS	Error, Trend, Seasonal
FDF	Fuzzy Decision Fusion
FRBS	Fuzzy Rule-Based System
KB	Knowledge Base
LSTM	Long short-term memory
MA	Moving Average
MSE	Mean Square Error
RBF	Radial Basis Function
RMSE	Root Mean square Error
RNN	Recurrent Neural Network
SVR	Support Vector Regression
TSK	Takagi-Sugeno-Kang

Chapter 1

INTRODUCTION

1.1 Introduction on Forecasting

Forecasting is the process of predicting future values based on available information, knowledge, and data that can impact the prediction. Forecasting is part of our daily life. It is useful in several domains including economic forecast, political forecast, weather forecast, and meteorology. It is used by many businesses to define how to allocate their budgets or plan to anticipate expenses for an upcoming period time therefore it should be reliable [1].

Improving time series accuracies has constantly retained researchers' attention. Several methods have been put in place to forecast time-series data, such as statistical methods, deep learning, and machine learning methods which have made it possible to obtain more or less satisfactory results from one data set to another. Each method has its strengths and weaknesses. The accuracy of each method depends on many factors such as characteristics of the data namely the trend, seasonality, the size of the data, the date range, and the length of the forecast horizon. In general, no method outperforms others in all cases. The stochastic nature of time series data makes it difficult for a single method to capture all intrinsic information of a given time series data. It is therefore very risky to rely on one forecasting method. To avoid this risk and to benefit of the strength of many individual methods, a combination of

the results from different forecasting methods was proposed and their accuracy is generally better than those of the individual model [2].

Stock market forecasting consists of determining the future value of a company. The current and future prices of the investment are two critical prices any investors have to know. Investors generally review past pricing history and use it to influence their future investment decision.

The efficient-market hypothesis (EMH) states that the current share prices already contain sufficient information for predicting future values [3]. This hypothesis is stated in 3 levels: weak, semi-strong, and strong. The weak form claims that technical analysis is insufficient for investors in making a trading decision. The semi-strong form instead believes that if some information is not readily available to the public then technical or fundamental analysis can be used by investors to boost their returns. The strong form believes that all information is present in the current stock prices therefore using proper tools, an investor can boost their chance of beating the market. Further studies in this area show that the stock prices are not efficient and do not follow a random walk [4]. And, several methods such as statistical, deep learning, and machine learning methods have been put in place to forecast time-series data, and have made it possible to obtain more or less satisfactory forecasts depending on time series data sets.

Motivated by making higher return rates, researchers, investors, and investment professionals always attempt to find a stock market model that would make a better forecast for a higher return. For a satisfactory forecast of time series data, the choice of the model is crucial. Each method has its strengths and weaknesses, and its

accuracy depends on many factors such as its characteristics, trend, seasonality, size, range of data, and length of the forecast horizon. No method outperforms the others in all cases, in other to get rid of the question which method to use to aim good or better accuracy, an alternative is to combined results from different forecasting methods.

1.2 The Goal of the Thesis

The goal of this thesis is to improve the cumulative absolute error of the time series prediction by a set of forecasting models using *Fuzzy Decision Fusion* (FDF) that learns the best decision for each fuzzy rule through the training data set. In other words, this study aims to demonstrate the decision fusion ability of a fuzzy rule base for multiple forecasting models namely the *Autoregressive Integral Moving Average*, (ARIMA), *Double Exponential Smoothing*, (EXP), *Support Vector Regression*, (SVR), and *Long Short Term Memory*, (LSTM) on currently available S&P500 time-series dataset.

1.3 Literature Review

There are many models to deal with a forecasting problem. Because it is more and more difficult to improve the performance of a single model, the combination has become one major way to do that and many models combination has been proposed. In decision making, the decision of multiple experts or models is fused to get the final decision. In the past decade, various methodologies were employed to combine decisions. The literature proposes different ways of combining forecasting methods it can be done by using either objective methods which involve objective techniques in its combination or subjective methods involving human judgment techniques [5]. Objective methods combination was introduced by Bates in 1969 [6] as a linear combination of two objective forecasts with k and $(1-k)$ for the first and second

decisions respectively $(D_f = k*D_1 + (1-k)*D_2)$ where k is the factor that minimizes the error variance of the combined forecast.

Subsequently, the combination was extended from 2 to *n,* and combination techniques began to be interpreted as a structural form of regression [7]. After their study, more sophisticated methods were proposed such as *arithmetic mean* methods *neural networks* methods, and *nonlinear combinations* [8]. A combination of decisions is carried also using *Bayesian analysis* in which weights are attributed based on the expected value [9], [10].

The key challenge in model combination forecast is to find the optimal model combination; this can be done at the learning phase for the static method [11] or at the testing for the dynamic method [12].

In 2001 Armstrong developed the so-called *Rule-Based Forecasting* (RBF). It determines the weights for each forecasting method by using IF-THEN rules to provide a rule-based weighted average combination. Kourentzesa et al proposed an approach that consists of selecting the models in the forecast combination [13].

Under Dr. Bodur's supervision, Ahmed Salih [14] implemented a forecasting model. He applied some well-known forecasting methods: *Radial Basis Function, K-Nearest Neighbour* method, *Self-organizing map* methods, and *Autoregressive Fractionally Integrated Moving Average*. Then, he clustered the estimated error space produced by these prediction methods. For each cluster, he obtained the final decision that

provides minimum forecasting error by using majority voting of the training data. For each new entry, the closest estimated error cluster indicates the final decision.

A similar study was supervised by Dr. Bodur, in 2014, that used fuzzy C-means clustering for *decision source selection* among three-time series technical analysis methods; *six-days-moving-average*, *moving-average convergence-divergence*, and *relative strength index* using TSK to select the best forecasting method among those three [15].

1.4 Overview of the Proposed System

In this thesis, *Fuzzy Decision Fusion* is applied to select the finalized forecast among the forecasts of four prediction models by *Fuzzy Decision Source Selection,* which uses the fuzzified estimated errors of the models to select the final decision according to a fuzzy rule base.

In time series forecasting, mostly there are multiple candidates of prediction models available for the purpose. The standard decision making procedure targets to determine the best model by carrying tests on the models using a test data set, and selecting the best model that gives the least prediction error through the test data set.

Decision fusion is a combination of various decisions coming from many sources. Each source, in our case prediction model, makes its own decision, in our case a forecast, with its local information.

This thesis aims to improve time series forecasting (in an accuracy point of view) by using fuzzy logic to select the best among multiple prediction models as proposed in

the internal report by Dr. Mehmet Bodur [16]. The demonstration consists of four prediction models: namely Double Exponential Smoothing EXP (M1), SVM (M2), ARIMA (M3), and, LSTM (M4) to forecast the S&P 500 data series. The first step is to use training data to determine the parameters of models, M1 … M4, and to get the forecasted values. The next stage is to use training data set to train a fuzzy rule base which shall finalize the prediction method to be used depending on the most strongly fired rule of the rule base. The proposed decision merging system is demonstrated on the Standard & Poor's 500 Index (S&P 500) daily time series data which is downloaded from the "*finance.yahoo.com*" data server [17] for 10 years period. 75% of the data points were used for training and 25 were reserved for independent testing.

1.5 Structure of the Thesis

The thesis starts with a short introduction of recent advancements in fuzzy decision fusion in Chapter 1. Chapter 2 explains some properties of time series forecasting on prediction models. Chapter 3 introduces the methods of the forecasting models which serve as the decision sources, and the fuzzy decision fusion algorithm. A demonstration of the proposed method is presented in Chapter 4 on forecasting of S&P 500 data set, including the results and discussions on the improvements of the forecasting by applied fuzzy decision fusion. Finally, we complete the thesis with a conclusion in Chapter 5.

Chapter 2

TIME SERIES FORECASTING

A time-series data is a collection of numerical values at different time points in successive order, usually spaced at regular time steps. It is commonly used in many areas including economics, health care, and so on. They can be univariate, which means collected by observations of a single variable over time, and multivariate where a set of observations of several variables are collected at each time step.

2.1 Stationary and Non-Stationary Time Series

A data set is said to be *stationary* time series when both the mean and variance are constant over time, therefore at large periods its properties are time-independent. A *non-stationary* time series has either a constant mean or a constant variance or both non-constant over large time. Many time series techniques assume that the time series is stationary. For a non-stationary time series, there exist two main methods to transform it into a stationary series. *Differencing* is performed to get rid of the varying mean. $y(t) = x(t) – x(t – 1)$, while *Log-transformation* is a nonlinear transformation to stabilize the non-constant variance of a series.

2.2 Components of a Time Series Data

A time-series can be decomposed into three components. (i) The *trend* means continuation of increasing or decreasing values in a given time series; (ii) the *seasonal* character means a repeating cycle over a specific period such as day, week, month, in a given time series; (iii) the *noise* is the random irregularity of values in a

given time series. Time series forecasting uses information in the sequence of historical values and their associated patterns to predict future activity.

2.3 Time Series Forecasting Models

They are various methods to forecast time-series data, and they can be classified into 2 groups [18] .

Qualitative methods:

For these methods, the forecast is based on opinions judgment, personal experiences emotions, and intuition. Here there is no model or any mathematical computation. They are subjective. We list methods such as Delphi Method, Market Survey, Executive Opinion, and Salesforces Composite in this group.

Quantitative methods:

These methods heavily rely on mathematical quantitative computations and are objective. Quantitative models can be grouped into associative models, often called Causal models in which forecasting is based upon associations between the forecast variable and other variables in the environment and time-series models that look at past patterns of data and attempt to predict the future. We can list here some quantitative methods such as (i) *naive*, (ii) *moving average*, (iii) *exponential smoothing*, (iv) *artificial intelligence* models, and (v) *ensemble* model.

In this thesis, we mainly use quantitative methods that use observed data to fit in a model that provides forecast of future values. Each quantitative model has its structural and fitting parameters and its hypothesis to be considered. The following

part of this chapter describes a non-exhaustive list of some well-known forecasting models that have been used in literature to forecast time series data.

2.3.1 Autoregressive Model (AR)

As its name indicates, the autoregressive model is a regression model to predict the future value by fitting the data series to a linear expression using the past values of the variable. The autoregressive model uses multiple numbers of past linear terms. The number (p) of the lag variable is decided depending on the correlation of variable at time t with respect to time t-p. An autoregressive model of order p (lag) is written in (2.1).

$$y_t = c + A_1 y_{t-1} + A_2 y_{t-2} + \ldots + A_p y_{t-p} + b_t \quad (2.1)$$

where $b_{t.}$ is white noise, y_k (k from t-1 to t-p) is the value of the forecasted variable at different lags. Therefore, AR(p) model is called an autoregressive model of order p.

2.3.2 Moving Average Models (MA)

This model is a linear regression on white noise in other words; it uses the past few forecast errors in a regression-like model,

$$y_t = c + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q.} \quad (2.2)$$

where, ε_t is white noise. MA(q) model is a moving average model of order q.

2.3.3 ARIMA

ARIMA modelling also called Box-Jenkins modelling proposed by Box and Jenkins in 1970 [19] is a mathematical model that forecasts future value by using previous time-series data plus an error. Specifically, it combines the Autoregressive model (AR) and the Moving average (MA). ARIMA model is applied under the assumption of stationary time series which means they have constant variance and mean. In the

ARIMA model, the predictors are lags of the dependent variable and/or lags of the forecast errors

$$\hat{y}_t = c + A_1 y_{t-1} + \ldots + A_p y_{t-p} - B_1 \varepsilon_{t-1} - \ldots - B_q \varepsilon_{t-q.} \quad (2.3)$$

ARIMA(p,d,q), model (2.3) has three structural parameters:

p: the number of autoregressive terms,

d: how many non-seasonal differences are needed to achieve stationarity,

q: the number of lagged forecast errors in the prediction.

In EMU-CMPE department, studies on forecasting stock market time series are carried under the supervision of Dr. M Bodur since 2013 [20]. Shareef demonstrated that filling the missing values of financial time series data improves the forecasting accuracy by extensive experiments, and her experiments indicated the possible high profit of a daily international automatic foreign exchange algorithm that uses a very high order ARMA(p=9,q=10) model for two days ahead of closing price prediction.

2.3.4 Exponential Smoothing Model

Exponential smoothing is a univariate time series forecasting method that models the three components of a time-series: values, trend, and seasonality. It is similar to ARIMA by Box-Jenkins in the sense that it is a weighted average of past observations with the main difference that the weights are decaying exponentially as the observations get older. According to the pattern taken into consideration, we can distinguish three types of exponential smoothing.

Single/Simple Exponential Smoothing (SES):

It models univariate time series data without trend or seasonality. The only parameter here is the smoothing factor coefficient called alpha (α) to control the decaying rate of influence of the previous observations as shown in (2.4).

$$y_{t+1|t} = \alpha y_t + \alpha(1-\alpha)y_{t-1} + \alpha(1-\alpha)2y_{t-2} + \ldots + y_1. \text{ With } 0 \leq \alpha \leq 1 \quad (2.4)$$

Double Exponential Smoothing:

Double Exponential Smoothing is a univariate time series model, and it extends SES by explicitly adding support for trends in the time series. In addition to the parameter α, a smoothing factor called β is added to control the decay of the influence of the change in trend.

Triple Exponential Smoothing:

It models a univariate time series by its trend and seasonality components. It extends the Double Exponential Smoothing by adding support for seasonality. Therefore it introduces alpha, beta, and gamma (2.5), (2.6). Gamma controls the influence on the seasonal component. This method is also called Holts-Winters exponential Smoothing and it is modelled from (2.5) to (2.8).

The forecast equation: $y_{t+h|t} = [\ell_t + (\phi_1 + \phi_2 + \cdots + \phi_h)b_t]s_{t+h} - m_{k+1}.$ (2.5)

Level/values equation: $\ell_t = \alpha(y_t / s_{t-m}) + (1-\alpha)(\ell_{t-1} + \phi b_{t-1}).$ (2.6)

Trend equation: $b_t = K*(\ell_t - \ell_{t-1}) + (1-k*)\phi b_{t-1}$. (2.7)

Seasonality equation $s_t = \gamma y_t(\ell_{t-1} + \phi b_{t-1}) + (1-\gamma)s_{t-m}.$ (2.8)

2.3.5 Support Vector Regression (SVR)

SVR is a regression algorithm that targets to fit the error within a certain threshold instead of minimizing the error rate as it is done in other regression algorithms**.** Before describing SVR operation mode we let us take a look at some important terminologies.

Kernel:

SVR assumes a linear relationship between the input and output variables, even if this assumption is not always verified with the dynamics of time series data. Kernel functions (polynomial, Gaussian Radial Basis Function, Sigmoidal) are employed to capture the nonlinear dynamics of the time series under study. In another word, the kernel function plays the role of moving data from lower-dimensional data into higher dimensional data so as to make it possible to perform linear separation on non-linear data without increasing the computational cost as shown in Figure 1.

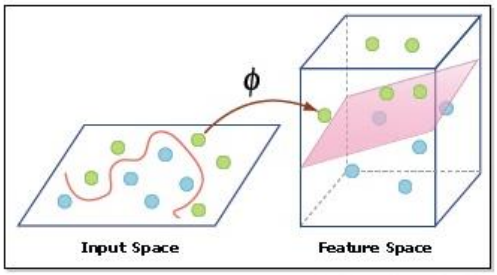

Figure 1: Kernel Mapping by Sethi [21]

Hyper plane:

A hyperplane is a surface that is determined by the kernels to predict the continuous output or target value.

Boundary line:

These are the lines drawn at an error ε (epsilon) distance from the hyperplane. It is used to create demarcation (margin) between the data point.

Support vectors:

These are the data points that are closest to the boundary line with a minimum or least distance.

Mathematical Formulation of SVM Regression

Given some data points, the goal of SVR is to draw a boundary (margin) lines and only consider points between these lines. In other words, SVR attempts to draw a line called hyperplane that fits the data points best and then, put boundaries at -$\boldsymbol{\varepsilon}$, and +$\boldsymbol{\varepsilon}$, distance from this hyperplane.

Linear Support Vector Regression: the Primal Formulation

Assuming we have a multivariate training data set x_n of n observations and a target variable y_n. SVR aims to search for the flat linear function $h(x) = x'k+b$ with minimal norm value $(k' k)$ .This can be formulated as an optimization problem with (2.9) as the goal and (2.10) as constraints:

Minimize

$$s(k) = 1/2\ k'k \tag{2.9}$$

Subject to:

$$\forall n: |\ y_n - (x_n'k+b)\ | \leq \varepsilon. \tag{2.10}$$

These constraints may not be always satisfied in all point of our data set therefore, to prevent against outliers, slack variables such as ξ, and ξ^* can be added in each point leading us to the objective function in (2.11) and (2.12), also called the primal formula [22].

Minimize

$$s\ (k) = 1/2k\ 'k + C \sum_{i=1}^{N} (\xi_i + \xi_i^*) \tag{2.11}$$

Subject to:

$$\forall n: y_n - (x_n'k+b) \leq \varepsilon + \xi_{n,} \tag{2.12}$$

$$\forall n: (x_n'k+b) - y_n \leq \varepsilon + \xi^*_n$$

$$\forall n: \xi^*_n \geq 0,$$

$$\forall n: \xi_n \geq 0,$$

where *C* is a constant that aims to control the penalty on data points lying outside the margin (ε).

This primal formulation, and can be written in a Lagrange dual formulation which makes the problem computationally simpler to solve and enable the primal technique to be extended to nonlinear functions [22]. Figure 2 illustrates a schematic of epsilon deviation bands in SVM process.

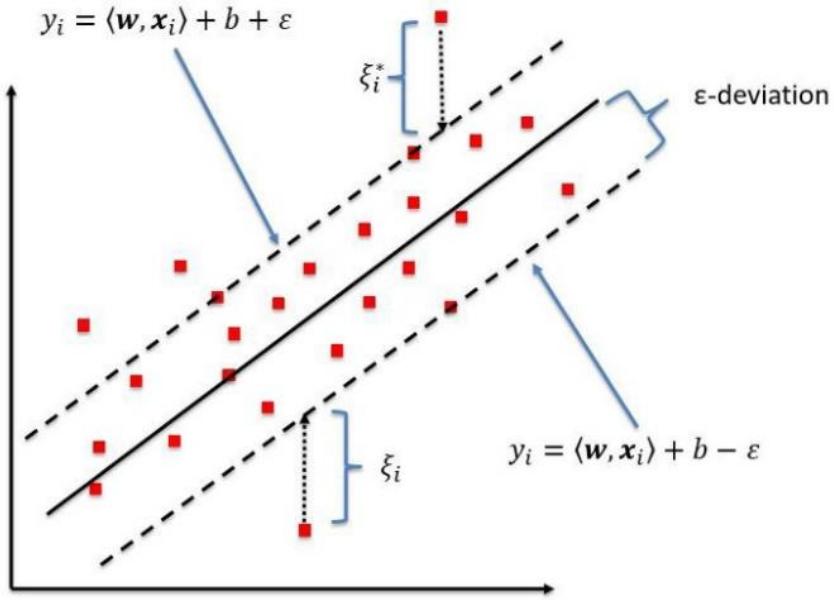

Figure 2: SVR Epsilon Deviation Bands by Bhattacharyya [23]

2.3.6 Long Short Term Memory Networks (LSTM)

In deep learning, LSTM is an artificial *Recurrent Neural Network* (RNN) architecture, which is known as well-suited for processing sequential data. In

traditional neural networks, input and output data are temporally independent which means it cannot memorize the previous outputs. In some cases like text mining or time-series data prediction of the current output depends on the previous input and hence comes the need to remember the previous input. This can then be solved by using Recurrent Neural Network (RNN).

A recurrent neural network is a type of artificial neural network with a memory that is capable of remembering previously computed information. Its main drawback appears if the sequence to process becomes long, where RNN faces the gradient vanishing and exploding problem.

The gradient vanishing and exploding problem

In backpropagation artificial neural network or in neural network with gradient-based learning technique, the error gradient is used to update the weights of the network and to know the right propagation direction. When the sequence to process is very long, the gradient is unable to remember old term dependency and shrinks as it back propagates this then causes its value to become too small and therefore doesn't contribute that much in learning: the network stop learning without been trained sufficiently.

LSTM is an efficient RNN capable of remembering a long time dependencies without being affected by the vanishing gradient problem. Proposed by Hochreiter et al, LSTM is used in the field of deep learning and it has feedback connections [24]. Unlike standard feed-forward neural networks, it is suitable for data sequences such as stock market index, speech or video streams as it is capable to handle lags of unknown duration between important events in a time series.

The LSTM network is composed of the following items.

Cell state: It plays the role of memory. It processes relevant information throughout the processing of the sequence and discards or forgets the irrelevant as shown in Figure 3.

Input Gate: this gate is responsible for adding relevant information to the cell state.

Forget Gate: it is a sigmoid layer has an output between 0 and 1. By this output, it decides which information should stay or be forgotten during the training. An output closer to zero indicates that information is irrelevant; while an output closer to one means relevant information and should stay.

Output Gate: This gate decides which values to be allowed as an output from the current cell.

Activation functions: These are some equations that determine the neural network output we can name: sigmoid function which is a popular function in a neural network. It transforms its input values to values between 0 and 1. Typically, the hyperbolic tangent function (tanh) is a nonlinear function that transforms it input values into values between -1.0 and 1.0.

Given a new sequence value x_t, it will be concatenated to the previous output of the cell h_{t-1}**.** The result is squashed with a tanh layer and then passed to the input gate. The latter will therefore kill off the unrequired element of the input vector using a sigmoid function. The next stage consists of determining the required variable (from the input gate) to be remembered or forgotten using the forget gate. Finally, the variables to be remembered are squashed (tanh) and passed to the output gate. Figure 3 shown an example of LSTM cell and its components.

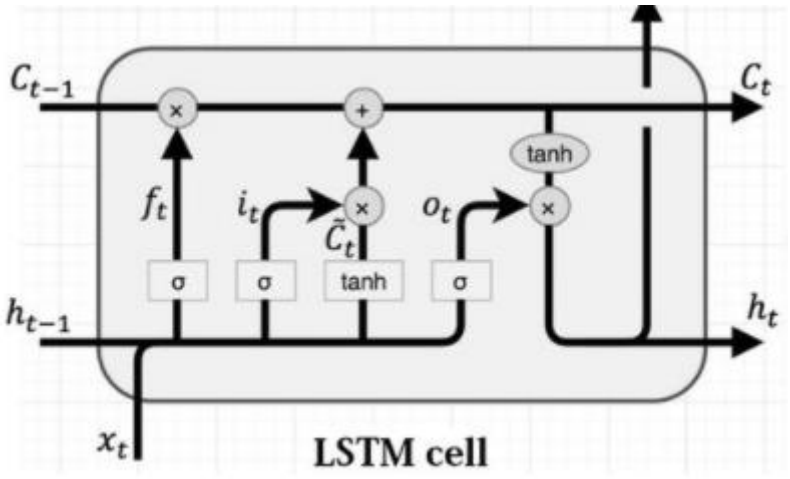

Figure 3: LSTM Cell Diagram by Varsamopoulos [25]

Operations in LSTM cells can be mathematically formulated as:

Input*:* $g = \tanh(b^g + x_t U^g + h_{t-1} V^g)$,

Output of the input gate*:* $i = \sigma\left(b^i + x_t U^i + h_{t-1} V^i\right)$,

where U^g and V^g represent the input and previous cell output weights, respectively, and b^g is the input bias.

The output of the input section is: $g \circ i$,

Forget gate and state loop is: $f = \sigma(b^f + x_t U^f + h_{t-1} V^f)$.

The output from the forget gate is*:* $s_t = s_{t-1} \circ f + g \circ i$.

Output gate: $O = \sigma(b^o + x_t U^o + h_{t-1} V^o)$.

Output of the cell is: $h_t = tanh(s_t) \circ o$.

2.4 Discussion on Quantitative Prediction Models

It is noticeable that the selected quantitative prediction models for the demonstration of the fuzzy decision fusion by source selection have quite different characters. We expect that each one may help in the final decision at a different partition of the expected errors, and take part in the fuzzy rule base as the best model for its niche.

Chapter 3

FUZZY DECISION FUSION

3.1 Fuzzy Representation of Numbers, and Fuzzification

In this work, we used fuzzy sets to represent a vector in the fuzzy domain. In representing numerical variables by fuzzy terms (or labels), a fuzzy set is a function from an interval of numbers to the interval of membership values to describe the uncertainty of the linguistic terms by a continuous membership value between non-member (0) and fully-member (1) [26]. In mathematical terms, a fuzzy set is a set, in which each real number is mapped onto a membership value that lies in the range [0, 1]. Given a collection of elements in the universe of discourse *X*, and each member denoted by *x*, a fuzzy set *A* in *X* is an ordered set of pairs:

$$A= \{(x, UM(x)) \mid x \in X\} \tag{3.1}$$

where *UM (x)* or *M(x)* is the membership function is also called the degree of truth of x in *A*. Linguistic labels such as attributes very low, low, medium, high, very high are represented by a family of membership functions, mostly a parametric mathematical function such as trapezoidal, triangular and Gaussian, for the ease of description and computation [27]. A value of a variable, for example, the closing value of the SP500 shares, c_k=45, is fuzzified in the interval [30, 50] by reading the value of the membership functions corresponding to each of the linguistic labels, resulting in a set of membership degrees u_k = (0.01, 0.1, 0.3, 0.9, 0.4), meaning that it is strongly in high fuzzy-interval, and closer to very high rather than medium.

In fuzzy modelling, the membership degrees of the inputs are processed by t-norm, and co-norm operators similar to the "*and"* and "*or"* operators of binary logic systems. The most widely used t-norm and co-norm pair are *min* and *max* functions, which are used by Zadeh to build decision inference models [26]. Fuzzy logic is a computational approach that uses *"degree of truth"* ($1 \geq u \geq 0$) instead of Boolean logic (*p* is either true or false, represented by either 1 or 0). In other words, it is a logic used to handle the concept of partial truth in which a value can be between completely true and completely false.

On Zadeh's Fuzzy Set Theory, several technics were introduced in the literature for combining the results of models aiming improvement of accuracy. Developments followed one of the two main tracks: (i-) an expert-determined set of membership functions used in modelling a training data set, (ii-) extracting the knowledge to a fuzzy rule base which points an acceptable solution for the modelled system, such as for a given input vector predicting the value of target variable, or determining the correct cluster.

In the past decade, fuzzy modelling techniques applied to decision fusion started in 1994 [28]. Dietrich et al proposed KNN method of classification of Time Series Utilizing temporal and decision fusion [29]. Xie et al built a fuzzy decision support system for demand forecasting based on a decision from market expert, customer, Autoregressive Moving Average (ARMA) model, and time-series analysis based on the decomposition method [30]. Fatemipour et al. proposed a source selection methodology for fuzzy decision fusion systems, where a binary tree-structured fuzzy rule base, decides the best source for a given input vector [31]. Finally, Ronald R. Yager worked on the fusion of multiple multi-criteria aggregation functions with a

focus on the fusion of OWA aggregations [32] and all these works give promising results.

3.2 Fuzzy Rule-Based Decision Fusion System

Fuzzy Rule-Based System (FRBS) is a rule-based system that uses fuzzy logic and fuzzy set to represent different forms of knowledge.

3.2.1 General Architecture of a Fuzzy Rule-Based System

The functioning of FRBSs can be defined as the interaction between knowledge and reasoning, which is the knowledge base (KB) and processing structure. The KB stores the available knowledge about the problem in the form of fuzzy IF-THEN rules. The processing structure uses these rules to puts into effect the inference process on the system inputs. Figure 4 shows a general structure of a FRBSs.

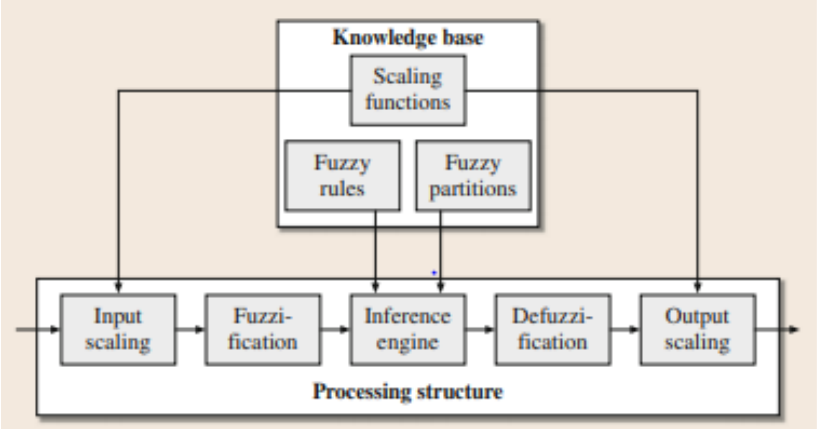

Figure 4: Typical Diagram of a Mamdani FRBS by Magdalena [33]

3.2.2 Knowledge Base (KB)

This component stores all the problem-specific knowledge (the relationship between input and output of the system). It is made of the fuzzy partition (methodology for generating fuzzy sets), rule base (RB), which is the collection of linguistic rules in the form of IF-THEN (i.e. IF X_1 is A_i and X_2 is A_j THEN Y is C) and the scaling

functions that are used to transform between the universe of discourse in which the fuzzy sets are defined from to the domain of the system input and output variables.

3.2.3 The Processing Structure

A fuzzy rule base system contains five main components, namely: (i) *the input scaling* which scales the input from its domain to a normalized domain require by input fuzzy partitions, (ii) *the fuzzification interface* which transforms crisps input values into a fuzzy value that serves as the input to the fuzzy reasoning process, (iii) *the inference engine* which uses the fuzzy input to infer fuzzy output based on the information in the KB, (iv) *the defuzzification interface* that converts the fuzzy inference result to a crisp value and finally (v) *the output scaling* which converts the defuzzified value to output variable domain.

A typical fuzzy rule is: IF x_1 is A_i and x_2 is A_j THEN y is C. The linguistic variables A_i and A_j are called antecedent and C is the consequent. Depending on the form of the consequent we can differentiate two basic FRBS models: (i) linguistic memberships function as the Mamdani model, and, (ii) an arithmetic function of input variables as TSK models.

3.2.4 Fuzzy Decision Fusion

Analysing data with several models provide more insights about this data and hence fusing the decisions made by each model enables one to benefit from multiple views rather than one. Assume we have K prediction models applied on a data set, with X explanatory variables, and $h_i(X)$ the output of each model i ($i = 1 \ldots K$), the goal of decision fusion-based model is to find the function g which suitably combines the prediction models given specific criteria.

A fuzzy decision fusion model works on the principle of space partitioning, which may be obtained either by clustering methods as applied by [14]. Many methods of partitioning are well known including k-means, fuzzy c-means (FCM) method, mountain, and subtractive clustering (SC) method. Partitioning may also be based on expert opinion who decides the fuzzy labels for input variables and specifies the parameters of the membership functions for each of these fuzzy labels. On this track, a supervisory training of membership function parameters is a possibility for further developments as implemented by [27]. Once explanatory variables are partitioned into subspaces, one or more rules are defined for each subspace as follow:

$$\text{If } x \text{ is } C_1 \text{ then } [w_{11}, w_{21}.....w_{k1}]$$

$$\text{If } x \text{ is } C_2 \text{ then } [w_{12}, w_{22}.....w_{k2}]$$

...

$$\text{If } x \text{ is } C_R \text{ then } [w_{1R}, w_{2R}.....w_{kR}]$$

where k is the number of models, C_i is the center of the i^{th} rule with $i = 1....R$ and w_{sk} is the weights vector assigns to model number s and in rule number k. In the *source selection* methodology, the weight $w_{iR} = 1$ just specify the selection of the i^{th} source, while all other weights being zero specifying not to select them.

3.3 The Proposed Data Fusion Method

The proposed data fusion method works in two phases: (i) the learning phase is a supervised algorithm that uses the cumulative absolute errors of the prediction models for strongest fired rule; (ii) the decision phase uses the best-scored prediction model for the strongest fired rule.

Parallel to the standard best model selection strategy, the parameters of prediction models ARIMA (M1), LSTM (M2), SVR (M3), and double exponential smoothing

(M4) were determined using only the training data set. The mean of the predictions of all these models was used as an intermediate estimate of prediction to build the vectors of estimated errors, which were used as the feature set to build the fuzzy rule base that determines the best method to be used for each fuzzy rule, as proposed in the internal report [16].

Once the fuzzy rules were learned, for a new time series inputs, the predictions of methods M1 … M4 were calculated using the trained parameter sets of each method. The predictions were converted to the estimated error features by using the mean of the predictions as the intermediate estimate of the prediction. The estimated error vector (*eemx*) calculated by this estimated prediction is fuzzified, and the firing strengths of the rules are obtained by using max-min t-norm and co-norm. The rule that gets the maximum firing strength specifies which model among the trained M1 … M4 model set performs the best for this input. The predicted value of that model is used as the finalized decision for the new input vector. The performance of the system is compared to each of the methods M1, … , M4 using the vectors in the test data set by the same procedure described for finalized prediction decision, and calculating the cumulative absolute prediction error. Figure 5 describes the process in a visual format, including the three phases: model training, fuzzy decision learning, and inference for new inputs. It also shows the flow of process for the determination of the cumulative prediction errors of M1 … M4 models, in parallel to the cumulative prediction error of proposed fuzzy decision method.

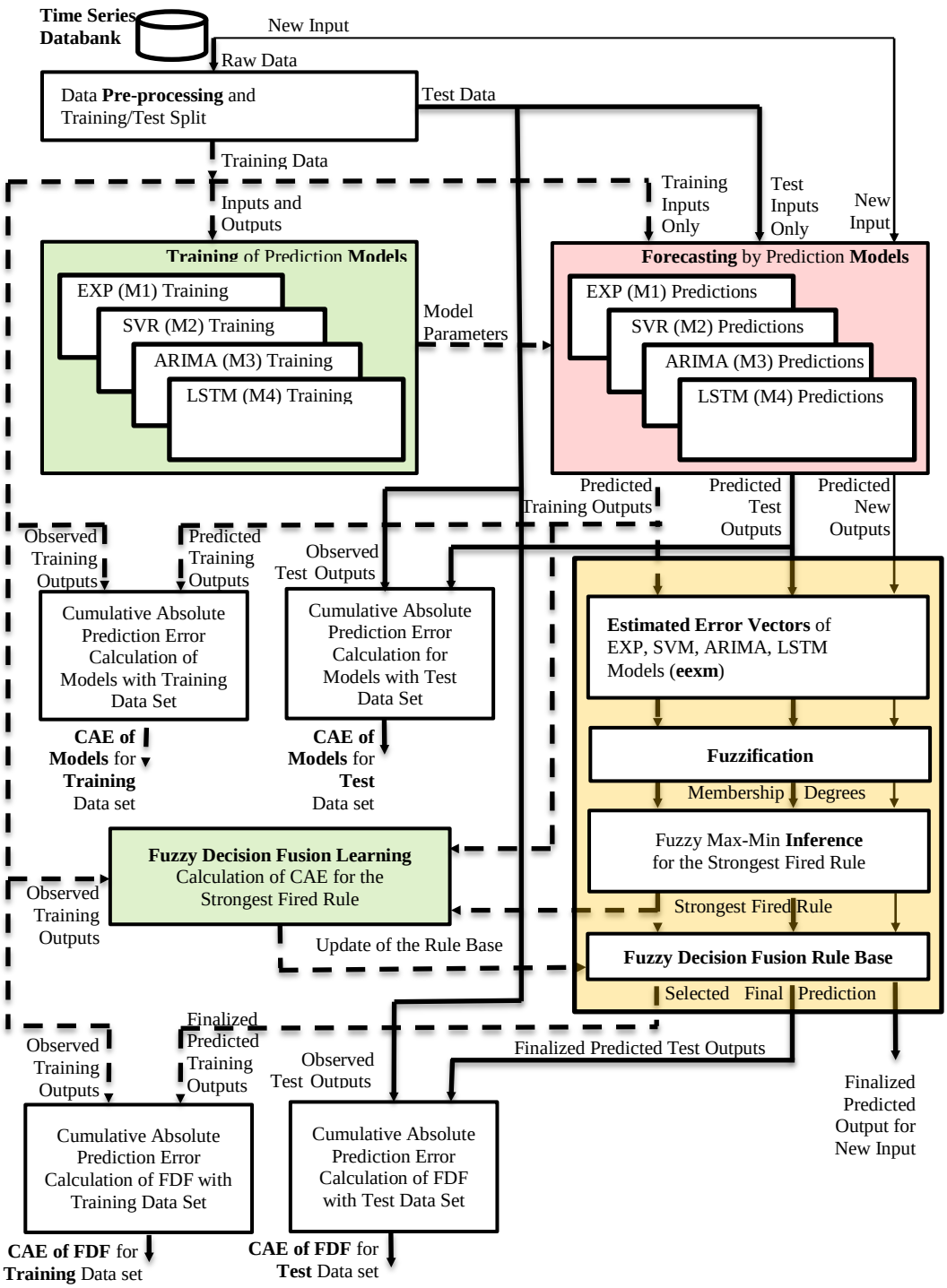

Figure 5: General Structure of the Demonstrated Fuzzy Decision Fusion System

3.3.1 Learning Phase of the Decision Finalization

The general framework for the training phase of the proposed method is described in 5 main phases.

Construction of the Estimated Error Vector

The first task is to construct the error vector from the predicted values of each of the single method. The error is computed as the difference between the predicted and the observed value

Computing Membership Degrees (MD):

The fuzzification process partitions the input variable in a certain number of the linguistic term following a structure called membership function which can be Trapezoid, Triangle, Sigmoid, Gaussian, or Bell-shaped forms [27]. The membership functions are specified by the expert opinion to partition the universe of discourse in a reasonable and convenient form to get rich content of partitions for all linguistic labels.

Normalize Membership Degrees (MD):

The sum of membership degrees obtained this way are not exactly unity and may create unexpected problems in further processing. Each MD of an input variable is normalized by dividing each MD value with the sum of the membership degrees along with the labels of each input variable so that their sum makes unity.

Determine which rule is fired strongest:

The strongest fired rule is determined using the combination of the maximum membership degrees along with the labels of each input variable as seen in Table 1. After all training inputs were evaluated, if some of the rules are non-voted, they are omitted by marking them as obsolete, and the globally best model is assigned as the default for all obsolete rules.

Table 1: Determination of Strongest Fired Rule

Input MD	Strongest Fired Rule
For variable 1, maximum MD is low, q_1=1	Rule = q_1
For variable 2, maximum MD is high, q_2=3	+3 (q_2-1)
…	+ …
For variable *K*, the maximum MD is (med.), q_K=2	$+3^{(K-1)}$ (q_K-1)

Determine the best model by actual errors:

This task consists of finding the model with less cumulative error for each rule. Table 2 explains cumulative calculation of the actual errors $err_{i,j}$ for each training input vector *i* in {1 ... n} and prediction model j in {1… k}.

After all train inputs were evaluated, the minimum error accumulator E_{Rj} of rule-R, for model-*j* is searched to get the best model for that rule. If some of the rules are non-voted, they are marked as obsolete rules, and the global best algorithm is specified as the default for these rules.

Table 2: Selection of the Best Algorithm for each Rule

MD combination	Strongest Fired Rule	Model Error Update (1 … *k*)
For input 1	Rule A	E_{A1}+=$err_{1,1;\ldots}$ E_{Ak}+=$err_{1,k;}$
For input 2	Rule B	E_{B1}+=$err_{2,1;\ldots}$ E_{Bk}+=$err_{2,k;}$
…	…	…
For input n	Rule A	E_{A1}+=$err_{1,1;\ldots}$ E_{Ak}+=$err_{1,k;}$
After all inputs trained:		model-*j* is best for rule-*R* if E_{Rj} =min(E_{R1}, …, E_{Rk}).

3.3.2 Forecasting of a New Input

For the given new input vector, the output may not be given at all. In the training of Fuzzy Decision Fusion, the output value has been used to get $err_{i,j}$ terms. In the forecasting case; (i) the predicted values for each model shall be calculated, (ii) estimated prediction errors *eemx* shall be fuzzified to get membership degrees of each variable of *eemx*, (iii) the maximum MD of each variable shall be combined to get the rule number which points the best model, (iv) among the cumulative model errors the minimum one shall indicate the model to be selected by fuzzy decision fusion as the finalized decision.

3.4 Concluding Remarks

This chapter has completed the description of the components which take part in the proposed method. The next chapter will focus on a step by step case study of our method on S&P 500 data. And the result will be compared to the individual prediction model that took part in the fuzzy fusion.

Chapter 4

COMPUTATIONAL RESULTS

4.1 Data Set and Code Development Environment

For the demonstration of the proposed Fuzzy Decision Fusion method, we used 10 years of historical data set of the Standard & Poor's 500 Index (November 2009 to November 2019). S&P 500 index is a market-capitalization-weighted index of the 500 largest U.S. publicly traded companies. Data contain 6 columns that describe the market behaviour (open price, highest price, lowest price, close price, and the volume). S&P500 index is accessible on the website of finance.yahoo.com. Data contain missing values for weekends and holidays, which can affect the forecast accuracy [20]. Therefore, data pre-processing will consist of filling missing data via interpolation.

All codes in this thesis were developed in the RStudio environment using R language [34], [35]. The code heavily depends on the libraries: keras, tensorflow, ggplot2, quantmod, tseries, timeSeries, forecast, xts, CombMSC, scales, e1071, httr, miceadds, lfl, fdm2id, frbs, Metrics, and DMwR.

4.2 Data Pre-processing

Table 3 shows the first 6 lines of raw data where we notice the absence of values on weekend days since the market is off.

Table 3: Sample of Raw S&P 500 Data Set

Days	Open	High	Low	Close	Volume
2009-11-11	1096.04	1105.37	1093.81	10985.51	4286700000
2009-11-12	1098.31	1101.97	1093.48	1093.48	4160250000
2009-11-13	1087.59	1097.79	1093.48	1093.48	3792610000
2009-11-16	1094.13	1113.69	1094.13	1109.30	4565850000
2009-11-17	1110.52	1110.52	1102.19	1110.32	3824070000

The linear interpolation is one of the most commonly used methods for solving missing data. It proceeds by taking the weighted average of the before missing data and the after missing data. After using linear interpolation, we obtained data including weekend days as it is in Table 4.

Table 4: Data after Filling the Missing Values

Days	Close	Volume
2009-11-11	1098.510	4286700000
2009-11-12	1087.240	4160250000
2009-11-13	1093.480	3792610000
2009-11-14	1098.753	4050356667
2009-11-15	1104.027	4308103333
2009-11-16	1109.300	4565850000
2009-11-17	1110.320	3824070000

For better insight, the training set was plotted in quarter sections. We can notice from the plot in Figure 6, that there is an additive upward trend pattern without outliers nor a sudden shift in the time series data which indicate a non-stationary data set this can also be checked with the Augmented Dickey-Fuller (ADF) test.

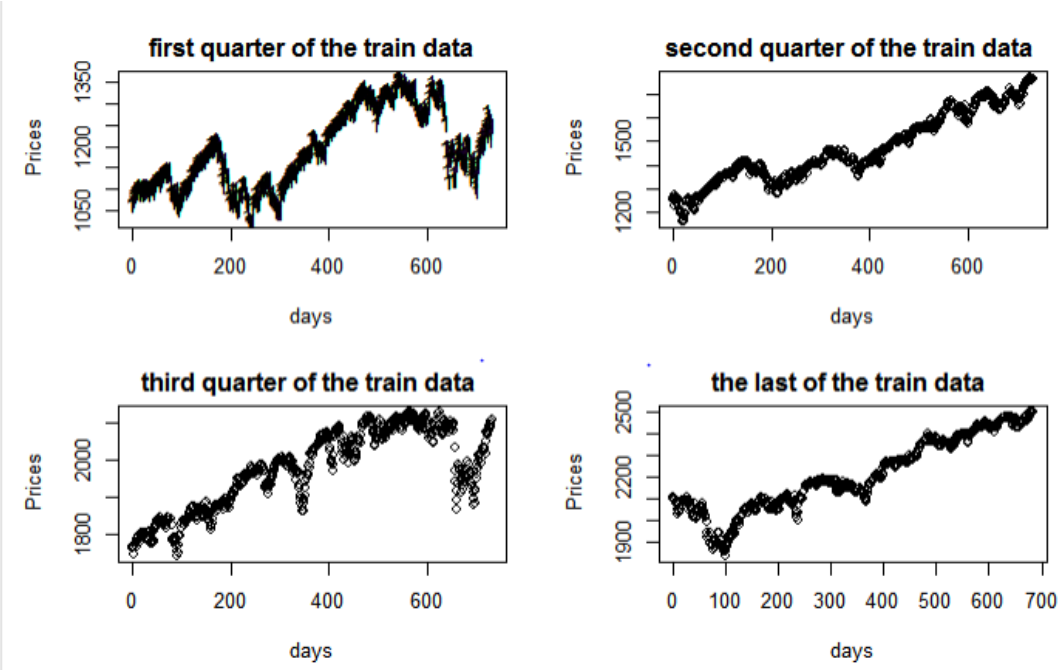

Figure 6: S&P 500 Daily Close Price Plot

4.3 Forecasting Models.

Four well-known prediction models were used, and after prediction by each model, all decisions were fused by a fuzzy rule base that determined the best models for each fuzzy region of the predicted error space. 75% of our data were used as training and 25% for testing.

4.3.1 Test for Stationary Property of Data

The ADF test is a statistic test for testing the null hypothesis that a unit root is present in the time series at some level of confidence such as because of random walk character.

ADF returns the P-Value of the data set. When this value is less than 5% null hypothesis of being stationary can be rejected. As seen in Figure 7 ADF applied on SP500 data returned P-value = 17%, which means that data set is not stationary.

```
adf.test(trainData)
##  Augmented Dickey-Fuller Test
##
## data:  trainData
## Dickey-Fuller = -2.9523, Lag order = 14, p-value = 0.1752
## alternative hypothesis: stationary
```

Figure 7: Code output of the ADF test for S&P 500 data set

4.3.2 ARIMA Model

The auto_arima function requires optimal structural parameters (*p*, *d*, *q*). We tested ARIMA (2, 1, 2) which means AR and MA orders are 2, and stationarity assumption is verified by first differentiation.

Figure 8 shows the training (blue) and fitted ARIMA forecast values (red) while Figure 9 displays the residual from ARIMA forecast.

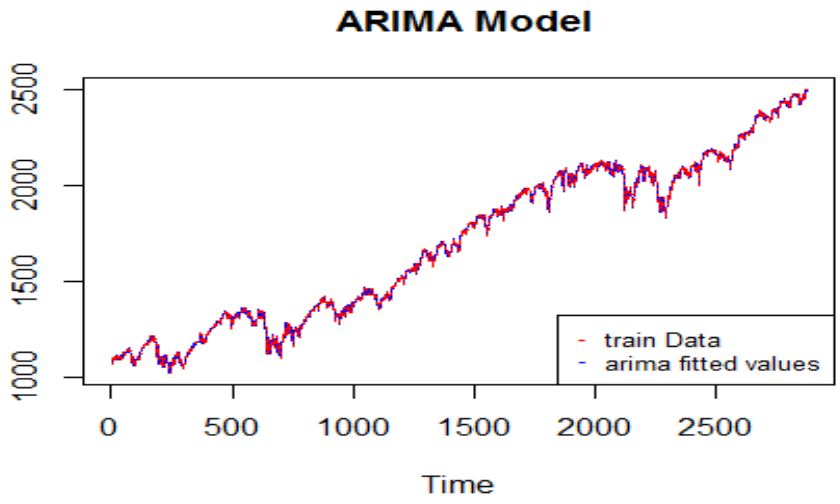

Figure 8: ARIMA Fitted Values Plot and Observed Values Plot

Residual Analysis

Residual is used to evaluate the *appropriateness* of a model which is generally done by observing the residual graph. In statistic there are some assumptions on residual such as: the residual variance should be constant, its variables should be independent, and the residual has to be normality of the distribution. These assumptions hold when the residual is randomly distributed around zero.

ARIMA Residual Plot

From the residual graph in Figure 9, we observe that the ARIMA residual follows a normal distribution, that is, it has a mean of zero and variance is uniform. This shows that, there is no repeating pattern left in the residual.

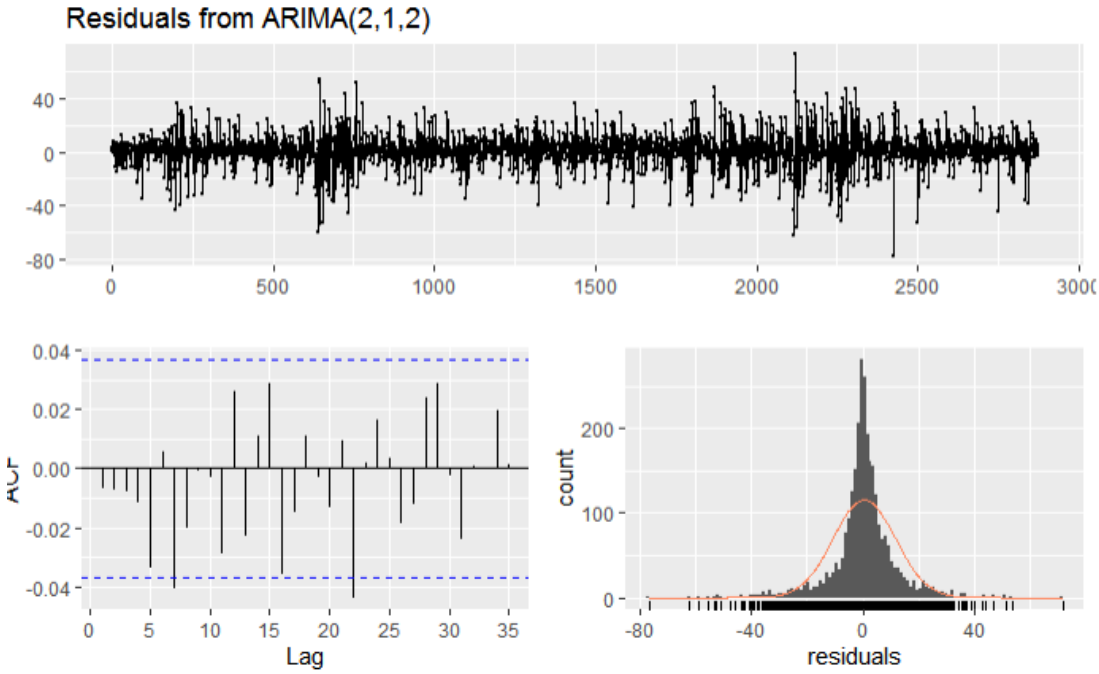

Figure 9: ARIMA Residual Plot

4.3.3 SVR Model

Radial Basis Function (RBF) was used as a kernel. RBF is a real-value function that maps each input from its domain to a real value. This value depends on the distance between the input and some fixed point that can be the origin ($\boldsymbol{\rho}(\boldsymbol{x}) = \boldsymbol{\rho}(||\boldsymbol{x}||)$ or a center *c* ($\boldsymbol{\rho}(\boldsymbol{x}) = \boldsymbol{\rho}(||\boldsymbol{x} - \boldsymbol{c}||)$.

SVR prediction performance can be visualized in Figure 10 and Figure 11 where we can visualize the fitted and residuals plot respectively.

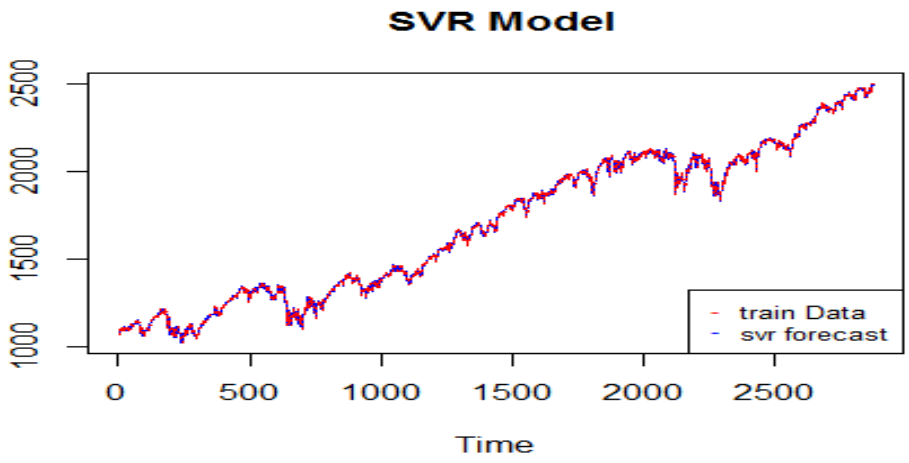

Figure 10: SVR Plot for Fitted and Observed Values

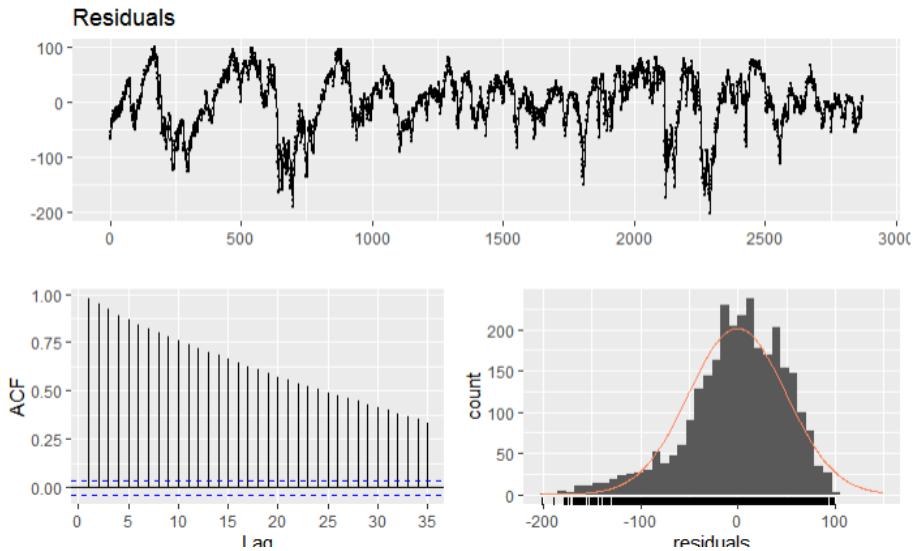

Figure 11: SVR Residual Plot

4.3.4 Double Exponential Smoothing

The error trend seasonality (ETS) model was used to identify the type of exponential smoothing function to use and its appropriate parameters (α, K, γ). The R code for tuning ETS parameters is given in Figure 12.

```
ets_model<-ets(ts(trainData$GSPC.Close),
      model = "AAN",alpha = NULL,gamma = NULL,lambda = NULL,beta = NULL)
etsF<-forecast(ets_model,h=n)
exp_forecast<-(ets_model$fitted) #forescast on train
summary(ets_model)
```

Figure 12: R Code for EXP prediction model

The output of the ETS model summary is given in Figure 13. It indicates the use of a double exponential smoothing with the additive trend with parameters: alpha = 0.998 and beta = 1e-04. The fitted values of the double exponential smoothing model on the training data can be visualized in Figure 14. Additionally, the residual plot in Figure 15 implies that there is no useful information left.

```
ETS(A,A,N)
## Call:
##  ets(y = ts(trainData$GSPC.Close), model = "AAN", alpha = NULL,
##  Call:
##      beta = NULL, gamma = NULL, lambda = NULL)
##   Smoothing parameters:
##     alpha = 0.998
##     beta  = 1e-04
##   Initial states:
##     l = 1081.4624
##     b = 0.5072
##   sigma:  11.2215
##      AIC     AICc      BIC
## 36776.88 36776.90 36806.69
## Training set error measures:
##                       ME     RMSE      MAE          MPE      MAPE      MASE
## Training set -0.003391882 11.21368 7.160736 -0.004591073 0.4475896 0.9986955
##                    ACF1
## Training set 0.03131204
```

Figure 13: ETS Model Summary Output for EXP Model

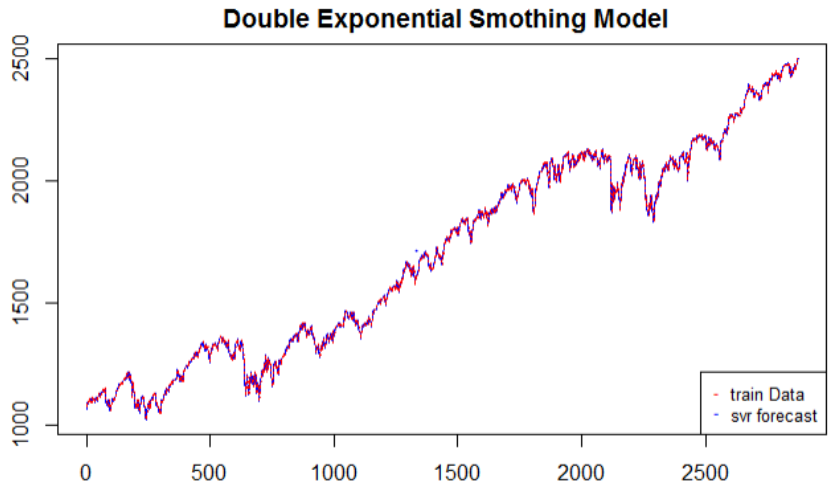

Figure 14: Graph of Exponential Smoothing Forecasted and Observed Values

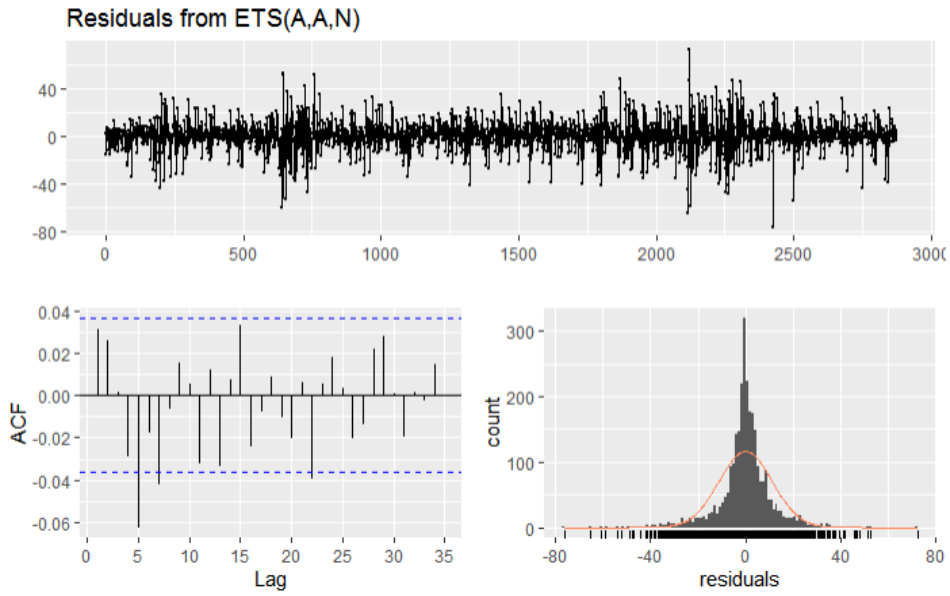

Figure 15: Exponential Smoothing Residual Plot

4.3.5 LSTM Model

LSTM model was build using the python open-source neural-network library called *keras*. It permits rapid experimentation with deep neural networks. It is modular, user-friendly, and extensible.

Data processing for LSTM:

To enable LSTM learning, the data must be divided into multiple input/output patterns. In this case one-time step was used as input (k) and one-time step was used as output. In other words we put our data in 2 dimensions (x-k and x) as shown in Table5.

Table 5: Two Dimensional Representations of the Data for LSTM Learning

Index	x-1	X
1	0.000000	2.670044
2	2.670044	7.926636
3	7.926636	7.926636
4	7.926636	7.926636
5	7.926636	-0.069946
6	-0.069946	5.500000

Data is reshaped to have another dimension as required by *keras* API so that it has dimensions for samples, time steps, and features.

In the demonstration, a vanilla LSTM model is built as a single hidden layer of LSTM unit and an output layer for prediction. In the model, the hidden layer contains 50 LSTM units and produces a single numeric value as output. The model was configured with the Adam stochastic gradient descent as an optimizer, and Mean Square Error (MSE) as the loss function. The predicted and observed values are plotted in Figure 16 and we can notice a close similarity between.

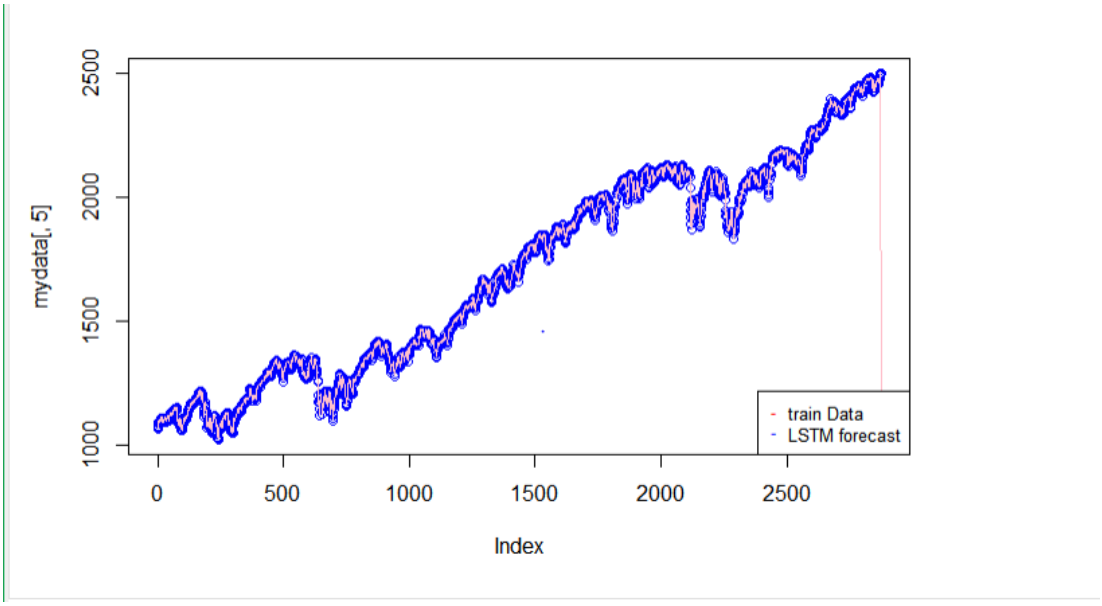

Figure 16: LSTM Predicted and Observed Values

4.4 Fuzzy Decision Fusion

4.4.1 Preparation of the Training Dataset

Once the prediction models M_j, $j = \{1 \ldots k\}$ forecast the next day value $y_{i,j}$ for an input x_i, we used the predicted values $\{ y_{i,j} \mid i=1 \ldots n; j=1 \ldots k\}$ as input for the fuzzy decision fusion system. Instead of using directly the predicted values as input features, we used estimated errors for each model, so that, decisions do not get affected by any interference of predicted variables which carries dominant information of data set together with the differences in the model prediction. The actual output y_i is estimated by the mean of all predictions, $\{y_{i,j} \mid j=1 \ldots k\}$, and the estimated prediction error vector ($eemx_i$) is used as the feature vector of fuzzy decision fusion as shown at Table 6. The plot of all prediction errors for the first 50 inputs against the observed values is shown in Figure 17.

Table 6: The First Six Predicted and Observed Values of each Model

	EXP	SVM	ARIMA	LSTM	Observed
1	1081.97	1132.13	1065.563	1069.393	1066.63
2	1067.166	1131.577	1066.637	1072.063	1069.3
3	1069.802	1131.03	1069.411	1079.989	1077.227
4	1077.719	1130.49	1077.601	1087.916	1085.153
5	1085.646	1129.957	1085.717	1095.843	1093.08
6	1093.574	1129.43	1093.638	1095.773	1093.01

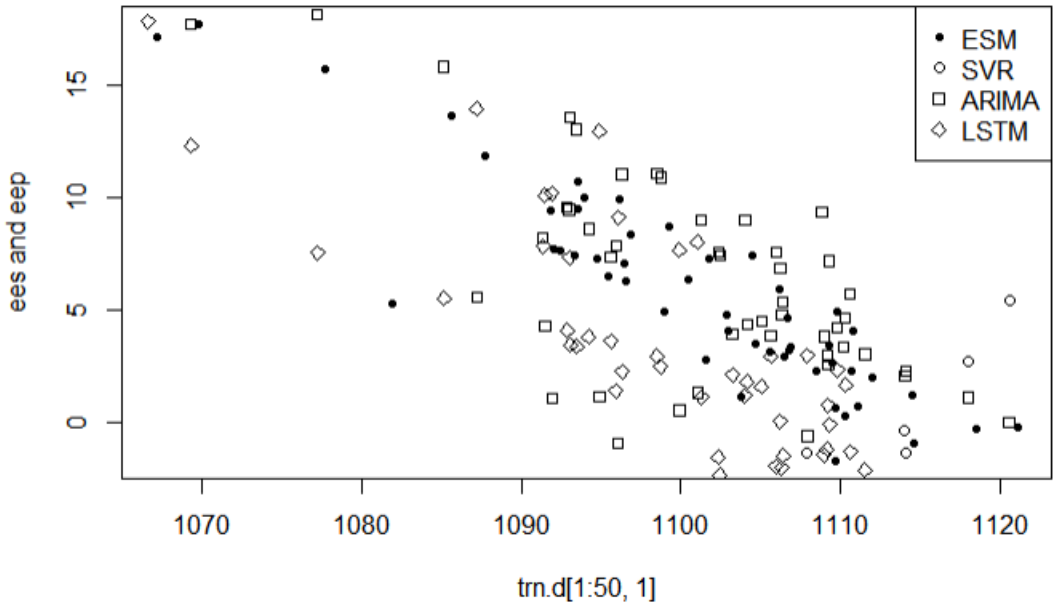

Figure 17: Graph of Prediction Errors vs. Observed Values for all Models

4.4.2 Construct the Estimated Error Vector (*eemx*)

The model prediction errors are obtained by subtracting the predicted values from the observed values. A sample can be seen in Table 7.

Table 7: Samples of Estimated Error Matrix

	eeESM	eeSVR	eeARIMA	eeLSTM
1	5.294295	-44.86607	21.700485	17.871294
2	17.194849	-47.21626	17.72354	12.297866
3	17.75641	-43.47203	18.146717	7.568901
4	15.712699	-37.05878	15.830484	5.515594
5	13.644394	-30.66642	13.574013	3.448014

The prediction performance of each method may be evaluated using the *cumulative absolute error* (CAE), which is computed as the sum of prediction error made for each input data. Table 8 lists the training CAE of the four models. LSTM performs better than the three others by producing the least CAE values.

Table 8: Training Cumulative Absolute Prediction Errors by Models EXP, SVR, ARIMA, and LSTM

Models	EXP	SVR	ARIMA	LSTM
Training CAE	20572.79	111203.4	20549.25	9366.128

4.4.3 Setting Membership Functions for Fuzzification

For the demonstration each input variable is partitioned into 3 Gaussian Membership Functions (MF). In Figure 18Figure 18: Membership Function Plot for Each Input Variable. It is a 5 by *n***k* matrix where *n* is the number of input variables and *k* the number of labels for each variable. The first row of this matrix describes the type of MF where 1: Triangle MF, 2: Trapezoid-left-side MF, 3: Trapezoid-right-side MF, 4: Trapezoid-middle MF, 5: Gaussian MF, 6: Sigmoid MF, and finally 7: for Bell-shaped MF. The other rows indicate the corner points to construct the functions.

Table 9: Parameter Matrix for Membership Function

	eeESM			eeSVR			eeARIMA			eeLSTM		
	Low	Med.	High	Low	Med.	High	Low	Med.	High	Low	Med.	High
1	5	5	5	5	5	5	5	5	5	5	5	5
2	-200	0	50	-200	0	50	-200	0	50	-50	0	400
3	100	20	40	100	20	40	100	20	40	30	10	200
4	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0

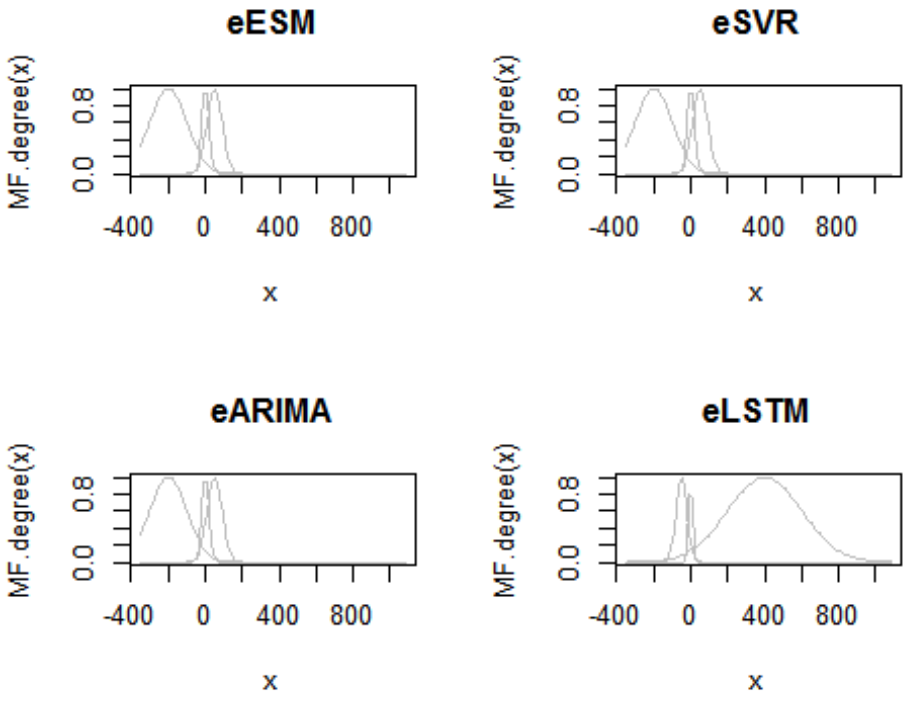

Figure 18: Membership Function Plot for Each Input Variable

Table 10: Membership Degree of the First Six Input Data

	eeESM			eeSVR			eeARIMA			eeLSTM		
	eE1	eE2	eE3	eS1	eS2	eS3	eA1	eA2	eA3	eL1	eL2	eL3
1	0.12	0.97	0.54	0.3	0.08	0.06	0.09	0.56	0.78	0.08	0.2	0.16
2	0.09	0.69	0.71	0.31	0.06	0.05	0.09	0.68	0.72	0.12	0.47	0.15
3	0.09	0.67	0.72	0.29	0.09	0.07	0.09	0.66	0.73	0.16	0.75	0.15
4	0.1	0.73	0.69	0.27	0.18	0.09	0.1	0.73	0.69	0.18	0.86	0.14
5	0.1	0.79	0.66	0.24	0.31	0.13	0.1	0.79	0.66	0.2	0.94	0.14
6	0.11	0.89	0.6	0.22	0.42	0.16	0.11	0.89	0.6	0.16	0.76	0.15

4.4.4 Extract the Membership Degree

The degree of membership to each label is obtained by applying the membership function to a giving input. In other words for each input vector, we compute their degree of belonging to each fuzzy set on each input variable. For the complete set of training inputs, the matrix of Membership Degree is an N*M matrix in which values lied between [0-1] where N is the number of input data and M is equal to the product of the number of input variable by the number of label [36]. For the demonstration M = 4*3=12 as shown on Table 10 which lists the MDs of first six training inputs.

4.4.5 Extract all Fired Rules

The number of rules is proportional to the input variable and the number of fuzzy set for each variable. The demonstration has 4 input variables and 3 membership functions and therefore 3 by 3 by 3 by 3 = 81 rules. From these rules those which were fired were retained, and from the retained rules, we get the best model of each rule searching the model with the lowest error. Table 11 shows the first ten rules, among which we observe that rule number 3 was fired, and, for that rule, model-1 (EXP) worked best.

Table 11: Cumulative Error Accumulators.

Rule No.	1	2	3	4	5	6	7	8	9	10	...
E_E	0	0	0.7	0	0	0	0	0	0	0	...
E_S	0	0	9	0	0	0	0	0	0	0	...
E_A	0	0	1.5	0	0	0	0	0	0	0	...
E_L	0	0	1432.1	0	0	0	0	0	0	0	...

4.4.6 Cumulative Absolute Error of Fuzzy Fusion on Training Data

The performance of the demonstrated method against each of the conventional prediction models was evaluated using the cumulative absolute error (CAE) for trainng inputs. As indicated on

Table 12, the demonstrated decision fusion method outperformed all models for the training phase by 14.31 % improvement compared to LSTM which stands out to be the best method among the four individual models.

Table 12: Cumulative Absolute Training Errors for Models and FDF

Models	**Cumulative Absolute Error**
EXP	20572.8
SVM	111203.4
ARIMA	20549.3
LSTM	9366.1
Fuzzy Decision Fusion	*7934.8*

4.4.7 Training of Fuzzy Decision Fusion Rule Base

During the model-training, the parameters of the models are determined to get the best prediction $y_{i,j}$ for the training input x_i by model-j among the models: EXP, SVR, ARIMA, and LSTM. Once the training of all models is over, the estimated prediction error $eexm_{i,j}$ is computed using the mean of all predictions as an estimate of observed output. Fuzzification of $eexm_i$ vectors partitions the input variables $eexm_{i,j}$ to expert-specified labels through the membership degrees $MD_{i,p}$ in expert-specified Membership-Functions. The combination of the maximum MD values points the strongest fired rule, r. The cumulative absolute error accumulators of rule r is updated by $E_{r,j}$ += $err_{i,j}$; for all models.

After all training inputs are processed, the minimum accumulated error for rule *r* specifies the best model for that rule, i.e., for rule *r* model *j* is best among *k* models if $E_{r,j} = \min(E_{r,1} \ldots E_{r,k})$.

4.4.8 Processing of New Inputs and Test Data Set

Once at the training phase the parameters of prediction models were trained, it also determines the best model for each rule in the fuzzy rules of the fuzzy decision fusion procedure.

The first step of the process for a new input data x_i is to predict target output from the four models: EXP, SVR, ARIMA, and LSTM using the trained parameters. Next, the estimated prediction error $eemx_{i,j}$ by each model-*j*, is computed using the mean of all predictions as an estimate of observed output, even if the observed output values for a new input is missing. The fuzzy decision fusion is carried out by the fuzzification step of *eemx* to obtain the strongest fired rule corresponding to the input data. After normalizing MD vectors for each input value, the normalized MD is used in finding the strongest fired rule. The absolute errors accumulated at the training phase for each model of the fired rule selects the best prediction model for that input as the finalized decision of the fuzzy decision fusion procedure. This process is repeated until all training input data set are processed.

For the test data set, the mean absolute error is computed for each model individually, and for the fuzzy decision fusion selected final predictions. The results of these calculations are listed in Appendix E.

Table 13 contains Membership Degrees of the expected errors for the first three test data. The mean absolute error of all fired rules is available in Appendix E in which a rule with zero cumulative absolute errors for all models represent a non-fired rules. During the learning process of fuzzy decision fusion, each time a rule is fired, it increments the *rulecount* counter of fired rule. Rulecounts of the rules are used to compute the mean absolute error for each model for each rule at the end of the process. Appendix E also shows the rule count of each fired rule during the training process.

Table 13: Membership Degrees of the First Three Test *eexm* Values.

	eeESM			eeSVR			eeARIMA			eeLSTM		
	eE1	eE2	eE3	eS1	eS2	eS3	eA1	eA2	eA3	eL1	eL2	eL3
1	0.1	0.6	0.27	0.105	0.841	0.054	0.107	0.787	0.106	0.11	0.686	0.203
2	0.1	0.6	0.275	0.105	0.84	0.055	0.111	0.789	0.1	0.112	0.685	0.204
3	0.1	0.6	0.273	0.104	0.839	0.056	0.112	0.789	0.099	0.112	0.683	0.205

Table 14: Cumulative Absolute Error Performance of Each Model on Test Data

Models:	EXP	SVM	ARIMA	LSTM	Fuzzy Dec. Fusion
CAE:	14841.033	839040.988	14783.931	15003.846	14764.844

Table 15: Head and Tail of Predictions by all models and FDF

	ESM	SVR	ARIMA	LSTM	Fuzzy Dec. Fusion
1	2502.7	2492.8	2500.2	2504.2	2504.2
2	2503.0	2493.1	2502.7	2505.4	2505.4
3	2504.2	2493.4	2503.9	2506.6	2506.6
4	2507.0	2493.7	2506.8	2509.4	2509.4
5	2508.6	2493.9	2508.4	2511.0	2511.0
	ESM	SVR	ARIMA	LSTM	Fuzzy Dec. Fusion
953	2851.0	1406.6	2854.0	2853.4	2854.0
954	2864.9	1406.6	2865.0	2867.3	2865.0
955	2878.8	1406.6	2876.7	2881.2	2876.7
956	2863.8	1406.7	2860.4	2866.2	2860.4
957	2939.7	1406.7	2940.1	2942.3	2940.1

Cumulative Absolute Error for Fuzzy Decision Fusion on Test Set

In Table 14 we observed a slight improvement of decision fusion of 0.19% compared to LSTM which stands out to be the best method among the four models. A sample of the predicted values is seen in Table 15 which lists the head and tail of the overall output prediction respectively.

4.5 Concluding Remarks

The results of demonstration on S&P 500 time-series data set indicate the successful contribution of fuzzy decision fusion by source selection by decreasing the cumulative absolute test error as well as the cumulative absolute training error. It is noticeable that the success of the method depends on the expert-opinion based selection of proper membership functions for the labels of input variables. Also, the fusion of the predicted decisions requires training and evaluation of prediction models, which requires considerable execution time. For the demonstrated data, an i7 processor with 8 GB main memory running at 2.8 GHz clock takes almost 20 minutes for training, and about 20 seconds for the test data set.

Chapter 5

CONCLUSION

Forecasting stock prices is very challenging and demands a reliable or accurate predictive model. This thesis demonstrated an approach of finalizing the selection of the best performing forecast among multiple well-known predictive models to reduce cumulative forecasting error. The idea is demonstrated on S&P 500 index using the prediction models: ARIMA, Double Exponential Smoothing, SVR, and LSTM. Fuzzy decision fusion combined the decisions of these models.

The demonstration shows that although all four prediction techniques forecasted the target in the best way for their limitations, LSTM outperformed the other three models in mean absolute training errors. Still, each model has its local hyperspaces where they work better than the others. This condition is observed in the Cumulative Error Accumulators of the rules. The Fuzzy Decision Fusion algorithm applied on the demonstration is expected to extract the best model for each fuzzy region. In the run, rule-3 and rule-51 had a local best model, EXP, while all other rules pointed the LSTM as the best. As a result of these local differences, which are successfully detected by FDF, the selected models by FDF outperformed the individual cumulative performances of all models both in training and in testing evaluations.

The fuzzy decision fusion method requires higher computational cost for training and forecasting procedures, but the extra time is not linearly dependent on the number of

models. The training and the forecasting period of EXP, SVR, and ARIMA take almost one-tenth of LSTM, the best performing conventional model. As a result, the 0.2% reduction in test-CAE costs only about 20% extra computational time. For the S&P 500 data set with total 3600 data points, the R code takes almost 20 minutes on an i7 processor with 8GB RAM, even while the source runs on a USB2 connected flash disk. Note that reduction of CAE depends on many factors, and may vary from one data set to another.

The proposed fuzzy decision fusion method has the advantage of combining methods of different natures (statistic, deep learning, or Machine Learning methods) and can be extended to other types of data. However, it presents some deficits. First of all, selecting the set of models to fuse is nontrivial. Next, no optimization method or tool is available to determine expert-proposed MF of input variable labels for an application. The third deficit is the increased computational complexity as discussed in the previous paragraph while explaining the computational complexity of the method.

The most critical part of the proposed method is the expert-opinion based MF settings. Without proper settings, the method cannot distinguish local best regions of the prediction models. In this respect, the proposed model is more art than a straightforward applicable tool for time series forecasting problems.

Future Work

As a perspective and possible future work, the Fuzzy Decision Fusion idea may be applied on a panel or cross-sectional time series analysis to improve the accuracy of

predictions by including other critical data sources on the related financial sectors and even political actions at some extent for a successful stock market forecasting.

REFERENCES

[1] Zemke, S., “Data Mining for Prediction. Financial Series Case.,” Computer and systems science, university's (KTH), 1971.

[2] Andrawis, R. R. , Atiya, A. F. & El-Shishiny, H., “Forecast combinations of computational intelligence and linear models for the NN5 time series forecasting competition.,” *International Journal of Forecasting,,* pp. vol. 27, issue 3, 672-688, 2011.

[3] Ţiţan, G., Alexandra, I., “The Efficient Market Hypothesis,” *Procedia Economics and Finance,* pp. 442-449, 2015.

[4] Lo, A. W. & MacKinlay, A. C. , “Stock Market Prices dot not Follow Random Walks,” *STOR,* pp. 41-66, 1988.

[5] Castello, A.; Mancuso, B.; Werner L., “Review of Combining Forecasts Approaches,” *Independent Journal of Management & Production (IJM&P),* pp. v. 4, n. 1, 2013.

[6] Bates, J. M. & Granger, C. W. J. , “The Combination of Forecasts,” *Journal of the Operational Research Society volume,* p. 451–468, 1969.

[7] Granger, C. & Newbold, P., “Spurious regressions in econometrics,” *Journal of*

Econometrics, pp. vol. 2, issue 2, 111-120, 1974.

[8] Shanming Shi ; Bao Liu, "Nonlinear combination of forecasts with neural networks," in *Proceedings of 1993 International Conference on Neural Networks*, Nagoya, Japan, Japan, 1993.

[9] Clemen, T., "Combining forecasts," *International Journal of Forecasting,* pp. Volume 5, Issue 4, 559-583, 1989.

[10] Werner, L., "A composite model to perform demand forecasting by integrating the combination of forecasts and opinion-based adjustment," Federal University Of Rio Grande, Porto Alegre, 2004.

[11] Burduk, R., Walkowiak, K., "Static Classifier Selection with Interval Weights of Base Classifiers," in *Asian Conference on Intelligent Information and Database Systems*, 2015.

[12] Oliveira, L. E. S., Sabourin, R., & Britto, AS. Jr., "Dynamic selection of classifiers," p. 3665–3680, 2014.

[13] Kourentzesa, N.; Barrowb, D.; Petropoulo, F. , "Another look at forecast selection and combination:evidence from forecast pooling," *International Journal of Production Economics*, pp. 1-40, 2018.

[14] Ibrahim, A. S., “Model Based Multi Criteria Decision Making Methods for Prediction of Time Series Data,” Eastern Mediterranean University, Gazimağusa, North Cyprus, 2014.

[15] Thanoon, M. A. , “Prediction of International Stock Markets Movement Using Technical Analysis Methods and TSK,” Eastern Mediterranean University, Computer Engineering, Gazimağusa, North Cyprus, 2014.

[16] Bodur, M., “Fuzzy Decision Fusion, Internal Report,” Dr. Bodur, EMU, Computer Eng. Dept., G/Magusa, 2020.

[17] Finance.Yahoo, “S&P 500 (^GSPC),” September 2020. [Online]. Available: https://finance.yahoo.com/quote/%5EGSPC?p=^GSPC. [Accessed 21 March 2020].

[18] Karmaker, C. L.; Halder, P. K.; & Sarker, E., “A Study of Time Series Model for Predicting Jute Yarn Demand,” *Hindawi journal of Industrial Engineering,* pp. 1-8, 2017.

[19] Box G., and Jenkins G., “Time series analysis: Forecasting and control,” San Francisco, CA, 1970.

[20] Shareef, J. K., “Prediction of International Stock Market Movements Using a Statistical Time Series Analysis Method,” Eastern Mediterranean University

Computer Engineering, Gazimağusa, North Cyprus, 2013.

[21] Sethi, A., "Support Vector Regression for Machine Learning," *Analytics Vidhya,* pp. 1-7, 2020.

[22] Platt, J. C., "Sequential Minimal Optimization," *Research Gate,* pp. 1-22, 1998.

[23] Bhattacharyya, I., "Support Vector Regression Or SVR," 29 june 2018. [Online]. Available: https://medium.com/coinmonks/support-vector-regression-or-svr-8eb3acf6d0ff. [Accessed 21 05 2020].

[24] Staudemeyer, R. C.; Morris, E. R. , "Understanding LSTM," Schmalkalden University of Applied Sciences, Germany, Schmalkalden, 2019.

[25] Varsamopoulos, S., Bertels, K., & Almudéver, C.G., "Designing neural network based decoders for surface codes," 2018.

[26] Zadeh, L. A., "Outline of a New Approach to the Analysis of Complex Systems and Decision Processes," *IEEE Transactions on Systems, Man, and Cybernetics,* pp. 28 - 44, 1973.

[27] Bodur, M., "Fuzzy System Modeling with the Genetic and Differential Evolutionary Optimization," in *Proc.Int.Conf CIMCA-IAWTIC,* Vienna, 2005.

[28] Buczak, A. L. ; Uhrig, R. E. , “Decision fusion by fuzzy set operations,” in *Proceedings of 1994 IEEE 3rd International Fuzzy Systems Conference*, Orlando, FL, USA, 1994.

[29] Dietrich, C. ; Schwenke, F. ; Palm G., “Classification of Time Series Utilizing Temporal and Decision Fusion,” in *International Workshop on Multiple Classifier Systems*, 2001.

[30] Xie, Y; Burnham, K. , “Fuzzy decision support system for demand forecasting with a learning mechanism,” *Fuzzy Sets and Systems,* pp. 1713-1725, 2006.

[31] Fatemipour M. R; Akbarzadeh T, “Dynamic Fuzzy Rule-based Source Selection in Distributed Decision Fusion Systems,” *Fuzzy Information and Engineering,* pp. 107-127, 2018.

[32] Yager, R., “On the fusion of multiple multi-criteria aggregation functions with focus on the fusion of OWA aggregations,” *Knowledge-Based Systems,* 2020.

[33] Magdalena, L., “Fuzzy Rule-Based Systems,” in *Springer Handbook of Computational Intelligence,* 7 Warsaw, Poland, Janusz Kacprzyk, 2015, pp. 203-218.

[34] RStudio, “RStudio Team,” 2020. [Online]. Available: http://www.rstudio.com/.

[35] R-Core-Team, "(). R: A language and environment for statistical computing. R Foundation for Statistical Computing,," URL https://www.R-project.org/., Vienna,, 2017.

[36] Riza, L. S., Bergmeir, C., & Herrera, F., "Package 'frbs' documentation," december 2019. [Online]. Available: http://dicits.ugr.es/software/FRBS/. [Accessed 18 June 2020].

APPENDICES

Appendix A: Data Retrieval and Split

This code is a step by step r code used to split our S&P 500 data set into train and test data set and later, build our individual model forecast. The fitted values of each model during training were bind as features input to the fuzzy Rule-Based model. The code starts by importing the necessary library files.

```
library(keras)
library(tensorflow)
library(ggplot2)
library(quantmod)
library(tseries)
library(timeSeries)
library(forecast)
library(xts)
library(CombMSC)
library(scales)
library(e1071)
library(httr)
library(miceadds)  # to load .Rdata
library(lfl)
library(fdm2id)
library(frbs)
library(Metrics)
library(DMwR)
rm(list = ls())
start_date <- as.Date("2009-11-05")
end_date <- Sys.Date() - 1
Sys.setenv(TZ = "GMT")
# if the file don't exixt then downloaf and save else just load it
if (!file.exists("GSPC.Data")) {
    re <- getSymbols.yahoo("^GSPC", env = parent.frame(),
    from = start_date, to = Sys.Date(),
    index.class = "Date", periodicity = "daily",
    return.class = "xts",
        auto.assign = "FALSE", frequency = 7)
    save(list = "re", file = "GSPC.Data")
}
load.Rdata("GSPC.Data", "spy")
myData = spy[, 1:6]
# removes tryCatch result
indexClass(myData) <- "POSIXlt"
# indexClass(myData)
tzone(myData) <- Sys.timezone()
allIndex <- index(myData)
print(as.data.frame(myData[1:20, ]))
```

Appendix B: Filling the Missing Values

This part of the code fills the missing data at weekend and holidays by merging missing days using linear interpolation function. It also splits data into train and testing partitions.

```
tt <- seq(from = min(index(myData)), to = max(index(myData)), by =
"day")
newData <- merge(myData, fill = "NA", tt)
## Warning in merge.xts(myData, fill = "NA", tt): NAs introduced by
coercion
newData[, 5] <- na.approx(newData[, 5])
newData[, 4] <- na.approx(newData[, 4])
close_volume <- newData[, 4:5]
# head(index(myData))
print(as.data.frame(close_volume[1:20, ]))
```

Split the data into train and testing data

```
N <- nrow(closeD)
closeD <- ts(close_volume[, 1], frequency = 1)
# Delimit training range
smp_siz = floor(0.75 * nrow(closeD))  # index of 75% of the dtaset
# smp_size= floor(0.75*nrow(volumeD)) # index of 75% of the dtaset

train_data1 <- window(closeD, end = c(smp_siz, 1))

head(train_data1)
## Time Series:
## Start = 1
## End = 6
## Frequency = 1
##      GSPC.Close
## [1,]   1066.630
## [2,]   1069.300
## [3,]   1077.227
## [4,]   1085.153
## [5,]   1093.080
## [6,]   1093.010
# Delimit testing range
test_data1 <- window(closeD, start = c((smp_siz + 1), 1))
head(test_data1)
```

Appendix C: EXP, SVR, ARIMA, LSTM Modeling

The following code builds ARIMA, SVR, and Exponential smoothing models on the training set.

```
n = nrow(ts(train_data))
model_auto <- arima(ts(train_data$GSPC.Close), order = c(2, 1, 2))
arima_forcast <- forecast(model_auto, h = n)$fitted

regress <- svm(train_data$trainSeq, train_data$GSPC.Close, data = train_data,
    kernel = "radial", type = "eps-regression", epsilon = 0.13, cost = 2^4)  #svm(x,y,...)
# prediction on train data
svm_forecast <- regress$fitted
n_test = nrow(test_data)
n <- nrow(train_data)
ets_model <- ets(ts(train_data$GSPC.Close), model = "AAN", alpha = NULL, gamma = NULL,
    lambda = NULL, beta = NULL)
etsF <- forecast(ets_model, h = n)
exp_forecast <- (ets_model$fitted)  #forescast on train
```

The following code builds LSTM Model, starting by data preparation, i.e., generating lagged dataset in two dimensions (x-1 and x)

```
diffedData = diff(closeD, differences = 1)

lagData <- function(x, k = 1) {

    lagged = c(rep(NA, k), x[1:(length(x) - k)])
    DF = as.data.frame(cbind(lagged, x))
    colnames(DF) <- c(paste0("x-", k), "x")
    DF[is.na(DF)] <- 0
    return(DF)
}
superviseData = lagData(diffedData, 1)

Nr = nrow(superviseData)
nr = round(Nr * 0.75, digits = 0)

train_lstm = superviseData [1:n, ]
test_lstm = superviseData [(nr + 1):N, ]
scale_data = function(trainData, test,
  feature_range = c(0, 1)) {
    x =trainData
    fr_min = feature_range[1]
```

Appendix D: Data Preparation for Fuzzy Decision Fusion

The following function (AllModel_test) is called to apply the models on the test set and use the output as input for testing the FRBSs.

```
AllModel_test = function(etsModel, svmRegressor, arimaModel, dataT) {
    n_test <- nrow(dataT)
    # svmF<-predict(svmRegressor, newData=dataT)
    svm_test <- predict(svmRegressor, dataT[, -2])  #prediction on test data

    arima_test <- Arima(dataT[, -1], model = arimaModel)
    arima_test <- fitted(arima_test)
    etsF1 <- ets(dataT[, -1], model = ets_model)  #fit test data values
    ets_test <- etsF1$fitted
    arimaF <- as.data.frame(arima_test)

    # lstm foescast on test set

    L = length(x_test)
    scaler = Scaled$scaler
    Lstm_Test = numeric(L)
    N = nrow(supervised)
    n = round(N * 0.75, digits = 0)
    for (i in 1:L) {
        X = x_test[i]
        dim(X) = c(1, 1, 1)
        yhat = model %>% predict(X, batch_size = batch_size)
        # invert scaling
        yhat = invert_scaling(yhat, scaler, c(-1, 1))
        # invert differencing
        yhat = yhat + closeD[n + i]
        # store
        Lstm_Test[i] <- yhat
    }
    combineF <- data.frame(cbind(volumeTest, ets_test, svm_test, arima_test,
        Lstm_Test, dataT[, -1]))
    colnames(combineF) <- c("Volume", "ETS_pred", "SVM_pred", "ARIMA_pred",
        "LSTM_ped", "Test_Data")
    return(combineF)
}
n_test <- nrow(train_data)
finalTestData1 <- data.frame(cbind(volumeTrain, exp_forecast, svm_forecast,
    arima_forcast, lstm_forecast, train_data[, -1]))

colnames(finalTestData1) <- c("Volume", "ETS_pred", "SVM_pred",
                              "ARIMA_pred", "LSTM_ped", "Train_Data")

finalTrainData <- scale(finalTestData1)
print(as.data.frame(round(finalTrainData[1:10, ], 3)))

finalTestData1 <- AllModel_test(ets_model, regress, model_auto, test_data).
```

```
finalTestData <- scale(finalTestData1)
```

Saving an R data frame as a .csv file as to be used next time instead of running all the models code again.

```
write.csv(FinalTrainData1, "FrbsTrain.csv")
write.csv(finalTestData1, "FrbsTest.csv")
```

Appendix E: Fuzzy Decision Fusion for Time Series

This code follows the frameworks described in Figure 8.

```
## expected value estimate of CLO
trn.epPL= (trn.d[,1]+trn.d[,2]+trn.d[,3]+trn.d[,4])/4
#error matrix
trn.emx=matrix(c(
trn.d[,5] -trn.d[,1],
trn.d[,5] -trn.d[,2],
trn.d[,5] -trn.d[,3],
trn.d[,5] -trn.d[,4]
),ncol=4, byrow=FALSE )
colnames(trn.emx)=c("eESM","eSVR","eARIMA","eLSTM")
#estimated error matrix
trn.eemx=matrix(c(
trn.epPL - trn.d[,1],
trn.epPL - trn.d[,2],
trn.epPL - trn.d[,3],
trn.epPL - trn.d[,4]),ncol=4, byrow=FALSE )
colnames(trn.eemx)=c("eeESM","eeSVR","eeARIMA","eeLSTM")

{
  plot(trn.d$CLO[1:50],trn.eemx[1:50,1],
    pch=20, ylab='ees and eep')
  points(trn.d[1:500,5], trn.eemx[1:500,2], pch=21)
  points(trn.d[1:500,5], trn.eemx[1:500,3], pch=22)
  points(trn.d[1:500,5], trn.eemx[1:500,4], pch=23)
  legend("topright",
    legend=c("ESM","SVR","ARIMA","LSTM"),
    pch=c(20,21,22,23))
}

options(width=120)
trn.eMD = fuzzifier(
trn.eemx,
trn.num.ivar,
trn.num.labels,
 trn.eMF )

colnames(trn.eMD)=c(
  "eE1","eE2","eE3",
  "eS1","eS2","eS3",
  "eA1","eA2","eA3",
  "eL1","eL2","eL3" )
# we need number of input variables, use trn.num.ivar
trn.eMDN=trn.eMD # start with copy of MD
for (i in 1:nrow(trn.eemx)){ # for each training vector
   for (k in 1:trn.num.ivar) { # for each input variable
      # ik is the start index of sum
      ik=sum(trn.num.labels[1:k])-trn.num.labels[k]+1
sumik=sum( trn.eMD[i, ik:(ik+trn.num.labels[k]-1)])
for (m in ik:(ik+trn.num.labels[k]-1) ){
```

```
trn.eMDN[i,m]=trn.eMD[i,m]/sumik}
# cat('i=',i,' k=',k,' eMDN=',eMDN[i,],"\n") # debug
}
}
head(round(trn.eMDN,2))
# a matrix of 6-col x 2-row to count voted algorithms
trn.num.rules = prod(trn.num.labels)
trn.rulemaes= 0*matrix(
   1:(trn.num.rules*(trn.num.ivar)),
   nrow=trn.num.ivar)
trn.rulecount=0*matrix( 1:trn.num.rules, nrow=1)
trn.rulenr = 0*trn.d[,-5][,1]
for (i in 1:nrow(trn.eemx)) { # for each training vector
  # i=3
  trn.rulenr[i] = 0
  # find the most strongly fired rule using shortcut
  # by the largest normalized MD for each variable.
  for (k in 1:trn.num.ivar) { # for each variable
    # k=1; k=2
    # start index of eMDN for that variable
    ik = sum(trn.num.labels[1:k]) - trn.num.labels[k] + 1
    # index of highest in the labels of k th var
    ikmax1 = which.max(
      trn.eMDN[i, ik:(ik + trn.num.labels[k] - 1)])
    # there are evi.label.num[1]*..*evi.label.num[kmax]
    # rules. Positional weight of k th variable is
    pw1 = prod(trn.num.labels[k:trn.num.ivar]
               ) / trn.num.labels[k]
    trn.rulenr[i] = trn.rulenr[i] +
        pw1 * (ikmax1 - 1) # iterative terms
  }
  trn.rulenr[i]=trn.rulenr[i]+1 # last term.
  trn.rulemaes[,trn.rulenr[i]] =
    trn.rulemaes[,trn.rulenr[i]]+ abs(trn.emx[i,])
  trn.rulecount[trn.rulenr[i]] =
    trn.rulecount[trn.rulenr[i]] + 1
}
for (r in 1:length(trn.rulemaes[1, ])) {
  # i=1 i=2
  for (k in 1:trn.num.ivar)
    if (trn.rulemaes[k, r] > 0) {
      trn.rulemaes[k, r] = round(trn.rulemaes[k, r] /
                                 trn.rulecount[r], 1)}
}
options(width = 100)
```

```
          round(tst.rulemaes[k,r]
               /tst.rulecount[r],3)}
}
options(width = 100)
tst.rulemaes
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15] [,16] [,17]
## [1,]    0    0    0    0    0    0    0    0    0     0     0     0     0     0     0     0     0
## [2,]    0    0    0    0    0    0    0    0    0     0     0     0     0     0     0     0     0
## [3,]    0    0    0    0    0    0    0    0    0     0     0     0     0     0     0     0     0
## [4,]    0    0    0    0    0    0    0    0    0     0     0     0     0     0     0     0     0
##      [,18]   [,19] [,20] [,21] [,22] [,23] [,24] [,25] [,26] [,27] [,28] [,29] [,30] [,31] [,32]
## [1,]     0  16.431     0     0     0     0     0     0     0     0     0     0     0     0     0
## [2,]     0 947.531     0     0     0     0     0     0     0     0     0     0     0     0     0
## [3,]     0  16.354     0     0     0     0     0     0     0     0     0     0     0     0     0
## [4,]     0  16.597     0     0     0     0     0     0     0     0     0     0     0     0     0
##      [,33] [,34] [,35] [,36] [,37] [,38] [,39] [,40]  [,41] [,42] [,43] [,44] [,45] [,46] [,47]
## [1,]     0     0     0     0     0     0     0     0  3.360     0     0     0     0     0     0
## [2,]     0     0     0     0     0     0     0     0 32.036     0     0     0     0     0     0
## [3,]     0     0     0     0     0     0     0     0  3.544     0     0     0     0     0     0
## [4,]     0     0     0     0     0     0     0     0  3.158     0     0     0     0     0     0
##      [,48]   [,49]  [,50]  [,51] [,52] [,53] [,54] [,55] [,56] [,57] [,58] [,59] [,60] [,61] [,62]
## [1,]     0  13.691  1.367  5.068     0     0     0     0     0     0     0     0     0     0     0
## [2,]     0 142.259 60.308 84.127     0     0     0     0     0     0     0     0     0     0     0
## [3,]     0  13.910  1.913  5.141     0     0     0     0     0     0     0     0     0     0     0
## [4,]     0  13.255  1.042  5.700     0     0     0     0     0     0     0     0     0     0     0
##      [,63] [,64] [,65] [,66] [,67] [,68] [,69] [,70] [,71] [,72] [,73] [,74] [,75] [,76] [,77]
## [1,]     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
## [2,]     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
## [3,]     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
## [4,]     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
##      [,78] [,79] [,80] [,81]
## [1,]     0     0     0     0
## [2,]     0     0     0     0
## [3,]     0     0     0     0
## [4,]     0     0     0     0

tst.rulecount
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15] [,16] [,17]
## [1,]    0    0    0    0    0    0    0    0    0     0     0     0     0     0     0     0     0
##      [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25] [,26] [,27] [,28] [,29] [,30] [,31] [,32]
## [1,]     0   880     0     0     0     0     0     0     0     0     0     0     0     0     0
##      [,33] [,34] [,35] [,36] [,37] [,38] [,39] [,40] [,41] [,42] [,43] [,44] [,45] [,46] [,47]
## [1,]     0     0     0     0     0     0     0     0    31     0     0     0     0     0     0
##      [,48] [,49] [,50] [,51] [,52] [,53] [,54] [,55] [,56] [,57] [,58] [,59] [,60] [,61] [,62]
## [1,]     0     5     1    41     0     0     0     0     0     0     0     0     0     0     0
##      [,63] [,64] [,65] [,66] [,67] [,68] [,69] [,70] [,71] [,72] [,73] [,74] [,75] [,76] [,77]
## [1,]     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
##      [,78] [,79] [,80] [,81]
## [1,]     0     0     0     0
options(width = 80)
```