Grenze International Journal of Engineering and Technology, Jan Issue

Treasury Bond Price Prediction using Time Series and Sentiment Analysis

Jugal Kothari[1], Archishman VB[2] and Dr. Jyothi R[3]
[1-3]PES University, CSE Department, Bengaluru, India
Email: jugalprakashk19@gmail.com, archishmanvb27@gmail.com

***Abstract*— This research paper distinguishes between stocks and bonds as distinct financial instruments and addresses the relative lack of computational finance research in bond price forecasting compared to stock market prediction. To bridge this gap, we propose and evaluate three forecasting models using historical bond price data: Support Vector Regression (SVR), Facebook Prophet, and RNN-LSTM. We also look into the possibility of increasing forecast accuracy by adding sentiment analysis as a new parameter. Mean Absolute Error (MAE), Mean Square Error (MSE), Root Mean Square Error (RMSE), Mean Absolute Percentage Error (MAPE), and Mean Square Percentage Error (MSPE) are some of the evaluation metrics used for model comparison. Our findings show that when sentiment encoding is included as a contributing element, Facebook Prophet beats the other models and exhibits improved accuracy. This study has useful implications for improving bond price forecasting methodologies and makes significant contributions to the field of computational finance.**

***Index Terms*— Support Vector Regression (SVR), Facebook Prophet, RNN-LSTM.**

I. INTRODUCTION

One of the areas of the financial markets where players trade debt instruments is fixed income. Bonds are the most under explored debt instrument and the subject of this study, although the market is much larger and includes a wide range of assets, including treasury bills, commercial paper, certificates of deposit, and government securities. Because they are provided to investors and have an obligation to repay the initial investment while paying an agreed- upon rate of interest, bonds are known as fixed-income securities. Bonds are a sort of financial contract or fixed- income instrument that governments, financial institutions, and businesses issue to raise loans and money from the general public to support operations and projects. The bond's nature makes it a legal document that serves as proof of debt. Bondholders are the issuer's debtors or creditors.

Treasury Bonds, being a debt instrument, have witnessed relatively lower popularity as an investment vehicle due to several factors, such as:

A. *Potential Returns:* Historically, stocks have provided longer-term investors with larger returns compared to bonds. Stocks have the potential for significant rewards despite having more volatility and risk. On the other hand, bonds often offer higher stability but lower yields.

B. *Risk Perception*: Compared to stocks, bonds are typically seen as less hazardous. Governments and corporations issue bonds as a kind of debt, and the issuer's capacity to repay the principle and interest determines the bond's value. On the other hand, the market and a company's success have an impact on the value of stocks, which

Grenze ID: 01.GIJET.10.1.193

represent ownership in a company. Bonds are viewed as being safer by some investors, while stocks may offer better potential returns for those who are willing to take on more risk.
C. *Interest Rates*: The desirability of bonds can be affected by the level of interest rates. Bond yields are low when interest rates are low, which makes bonds less desirable to investors looking for better returns. In contrast, when interest rates increase, bond prices could decline, further lowering their appeal.
D. *Market Sentiment:* The popularity of different asset classes is significantly influenced by investor sentiment. Investors may be more likely to invest in equities during optimistic market periods when stocks are performing well in order to benefit from the upward momentum. On the other hand, investors may prefer the relative stability of bonds as a safe-haven investment during periods of uncertainty or economic downturns.
Through the use of various machine learning approaches, this research article aims to highlight the importance of treasury bonds as a crucial investing tool while exploring the underlying relationship between important characteristics. This study seeks to get important understandings of the behaviour of treasury bonds and their relationships with pertinent variables by conducting a detailed examination. We aim to shed light on the variables impacting treasury bond performance, risk, and yield through the application of modern machine learning approaches, thereby contributing to a greater knowledge of the dynamics inside the fixed-income market. In the end, the research's conclusions aim to offer insightful advice to help investors and financial institutions improve their risk management and bond-investment strategies.

II. Related Work

The topic of stock price prediction has seen a tremendous amount of study and development throughout time. Developing complex algorithms and models to estimate stock prices effectively has taken a lot of time and effort from researchers, data scientists, and financial specialists. These initiatives seek to harness the complexity of financial markets, find underlying patterns, and generate knowledgeable forecasts about future price movements.
On historical stock data, models like ARIMA, RNN-LSTM, and MARS (Multivariate Adaptive Regression Splines) have been trained, and it has been found that MARS performs the best at forecasting [1]. In order to anticipate bond prices based on historical data, SVR is presented as a potent machine learning technique that can handle high-dimensional data and grasp nonlinear correlations [2]. The difficulties and restrictions of using machine learning to anticipate bond prices are discussed. They talk on the requirement for constant model updating in dynamic financial markets, as well as issues with data quality, over fitting, model interpretability, and model interpretability [3]. Traditional financial models for time series and models of the yield curve for fixed income securities have been thoroughly investigated [4]. Research has been done on the merits of sentiment analysis for stock market indicators in order to forecast stock prices. Sentiment analysis, also known as opinion mining, analyses user opinions, evaluations, sentiments, attitudes, and emotions in order to find and extract subjective information using text mining and natural language processing (NLP) [5]. For the purpose of event-driven stock market prediction, deep learning has been proposed. Events from news text are extracted and then represented as dense vectors with the help of a novel neural tensor network [6]. A model for forecasting stock price is also looked at using emotional data from news headlines and previous pricing. The method is able to narrow the gap between expected values and actual values in addition to drawing superior conclusions. The absence of news and sentiment lexicon can be compensated for by the combined model of time series and sentiment analysis [7]. The usage of LSTM networks on stock market prediction has been studied. Additionally, an effort has been made to research the connection between news and stock movement, presuming that news pieces have an influence on the stock market [8].
Index Terms—ARIMA (Auto Regressive Integrated Moving Average), RNN-LSTM (Recurrent Neural Network, Long Short Term Memory), MARS (Multivariate Adaptive Regression Splines).

III. Data Description

For our dataset, we used historical data from the Yahoo! Finance website for the 10-year US Treasury bond for the period from 10 November 2016 through 19 July 2023. We obtained the historical bond information and the news headlines for the same time period, from the Financial Times website.
There are five columns in the historical data for the 10-year US Treasury bond that list the bond's pricing, and one column lists the relevant date. The five columns are labelled as " Open", "High", "Low", "Close" and "Adj Close". We have selected "Close" values as the input feature for the various prediction models in our study. The dates and the corresponding "Close" values were then stored in a CSV file for data preprocessing.

The news headlines for the 10-year US Treasury bond for the same time period was obtained from the Financial Times website, through web-scraping. The headlines were then stored in a CSV file for data preprocessing.

IV. Data Preprocessing

As seen in Fig. 1, the historical data acquired lacks ”Close” values for a number of days. We used a technique called ”Data Imputation” to obtain continuous Time series data. Imputation is a technique for keeping most of the data and

Historical Data	Period		No. of Days	No. of days without Close values	Number of days with Close values
	From	To			
US Treasury 10 Year Yield	10-11-2016	19-07-2023	2442	413	2029

Figure 1: Time Period for Data collection (Historical Data)

News Headlines	Period		No. of Days	No. of days without news	Number of days with news
	From	To			
US Treasury 10 Year Yield	10-11-2016	19-07-2023	2442	1392	1050

Figure 2: Time Period for Data collection (News headlines Data)

information in the dataset by replacing missing data with a new value. The many imputation methods include:

A. Forward Fill (Last Observation Carried Forward, or LOCF): This approach fills in the missing values up until the next observed data point by propagating the last known value forward. It makes the assumption that the missing numbers won’t change until more data is available.

B. Backward Fill (Next Observation Carried Backward, or NOCB): propagates the last known value backward to fill in the missing values up to the previous observed data point, much like Forward Fill does.

C. Mean/Median Imputation: With this method, the mean or median value of the series’ currently accessible data is used to replace the missing values. If the time series includes outliers, this technique might not work.

For our historical dataset, we have applied Mean Imputation to replace the missing values. The dataset obtained after imputation is then stored in a CSV file. For the news headlines dataset, we first dropped the duplicate dates (dates which had multiple news articles) and then applied the Forward Fill (Last Observation Carried Forward) data imputation method.

After imputation, Sentiment Analysis was performed on the acquired dataset using VADER (Valence Aware Dictionary and sEntiment Reasoner) sentiment analysis tool. Based on a predetermined lexicon of terms, VADER analyses the text and assigns sentiment scores. The dictionary includes words and the sentiment intensity scores that correspond to them, indicating whether a word is good or negative. Additionally, VADER employs standards for capitalization, punctuation, and handling modifiers to deliver results for sentiment analysis that are more precise. The VADER sentiment analysis produces three primary results:

A.Positive Score: The percentage of the text that is deemed positive or exhibits a positive sentiment is represented by the positive score. It has a scale of 0 to 1, with 0 denoting no positive sentiment and 1 denoting a strong positive sentiment.

B.Negative Score: The percentage of the text that is deemed negative or expresses a negative sentiment is represented by the negative score. It also runs from 0 to 1, with 0 denoting no negative sentiment and 1 denoting a strongly unfavourable sentiment.

C. Compound Score: The compound score is a single number that captures the sentiment of the text as a whole. Its values vary from -1 to +1, with -1 denoting the most negative, +1 the most positive, and 0 denoting a neutral feeling. The compound score weighs the strength and percentage of each feeling in the text and is calculated as the weighted sum of the positive, negative, and neutral values.

In our study, we’ll employ the Compound Score as a sentiment analysis metric. The ”merge()” function of the pandas library is then used to combine the two datasets mentioned above, namely the historical bond price dataset and the news headlines dataset. With the help of the pd.merge() function in pandas, we may join Data-Frames based on shared columns or indexes. It supports a variety of joins, including inner, left, right, and exterior joins. We have merged the dataset based on the ”Dates” column.

To ascertain if the combined data are steady, a variety of statistical tests are run on them. Time series analysis employs the statistical tests KPSS (Kwiatkowski-Phillips-Schmidt Shin) and ADF (Augmented Dickey-Fuller) to

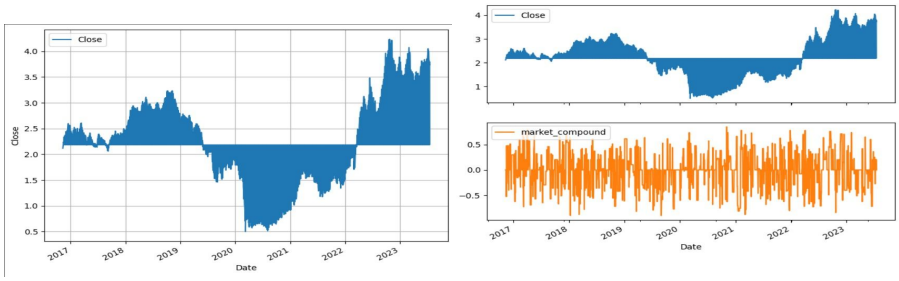
determine if a time series is stationary. Stationary is a crucial concept in time series analysis. The Augmented Dickey-Fuller (ADF) test is a unit root test that determines if a time series is stationary. It examines the null hypothesis that the time series has a unit root in order to establish if it is stationary. The null hypothesis is rejected and it is determined that the time series is stationary if the p value of the ADF test is less than a selected significance level, which is often 0.05. If the p-value is higher than the threshold for significance, the time series is not stationary, and the null hypothesis cannot be ruled out.

In econometrics and finance, the ADF test is frequently used to determine whether trends exist and whether time series need to be transformed to become stationary before being used for further research or modeling.

The Kwiatkowski-Phillips-Schmidt-Shin (KPSS) Test, often known as the unit root test, evaluates a time series' stationarity under the null hypothesis. The KPSS test investigates the alternative hypothesis as opposed to the ADF test. The time series being stationary around a deterministic trend serves as the null hypothesis for the KPSS test. We reject the null hypothesis and arrive to the conclusion that the time series is non-stationary around a deterministic trend if the p- value is smaller than the significance level. The time series is deemed stationary or trend-stationary if the p-value above the significance level, which indicates that we are unable to reject the null hypothesis.

Historical Data + Market Sentiment	Period		No. of Days with both Close values and News
	From	To	
US Treasury 10 Year Yield	10-11-2016	19-07-2023	2029

Figure 3: Time Period for Data collection (combined)



(a) Plot of Close price vs Dates (b) Plot of Close Price and Compound Sentiment scores vs Dates

Figure 4: Combined Figures

The KPSS test and the ADF test are frequently combined to provide a more thorough knowledge of a time series' stationarity characteristics. The KPSS test determines whether a deterministic trend exists in the time series, whereas the ADF test looks for the presence of a unit root and determines whether differencing is required to establish stationarity. Applying the statistical tests indicated above leads us to the conclusion that the collected historical bond data is non stationary in nature.

The merged data is then subjected to Time series decomposition method. A time series can be dissected into it's under-lying components using the decomposition technique, which often identifies trend, seasonality, and residual (or error). By breaking down the time series data into its constituent parts, decomposition aims to better comprehend the patterns and structures that are present in the data and to separate the various elements for further analysis, forecasting, or anomaly identification. The different types of decomposition are:

A. Additive Decomposition: The time series is assumed to be the sum of its individual components, i.e., trend, seasonality, and residual. When the amount of seasonality is largely constant across several levels of the time series, additive decomposition is appropriate. When the time series' various components, such as trend and seasonality, can be understood and exploited to shed light on underlying patterns and behaviours, additive decomposition is widely used. The formula for additive decomposition is given by (1)

B. Multiplicative Decomposition: The time series is thought to be the total of its trend, seasonality, and residual components in multiplicative decomposition. Multiplicative decomposition requires multiplying the constituent parts to separate them. This method is particularly useful when the seasonal variations are not constant and change in relation to the trend. When the seasonality's magnitude varies with the time series' level and the seasonal

variations rise or fall in response to the trend, multiplying decomposition is the right strategy. The multiplicative decomposition formula is given by (2). The decomposition of the merged data is shown in the Fig. 5.

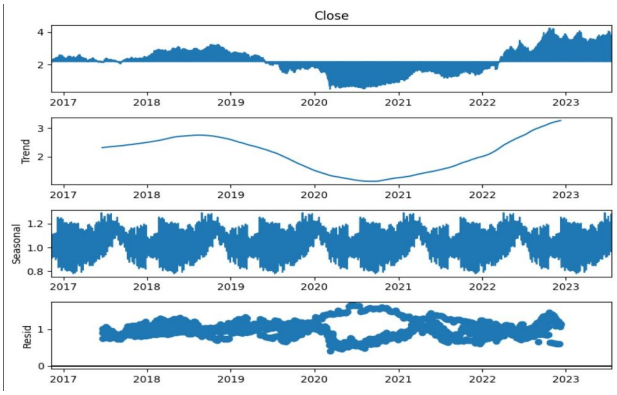


Figure 5: Plot of Decomposition of the final dataset into Trend, Seasonal and Residual components

Y(t)=Trend(t)+" Seasonality " (t)+" Residual " (t) (1)

Y(t)=Trend(t)*" Seasonality " (t)*" Residual " (t) (2)

V. FACEBOOK PROPHET

Facebook Prophet is an open-source forecasting tool developed by the Facebook Core Data Science team. with the goal of making time series forecasting simple and efficient. Prophet, which is based on the Stan probabilistic programming language, uses an additive model to take into account seasonal trends as well as the effects of holidays, making it the perfect tool for handling datasets with numerous seasonality's. Its capacity to automatically manage missing data and outliers with little to no data pretreatment is one of its main features. Prophet also offers user-friendly diagnostic tools and visualizations that make evaluating and validating forecasts easier. Facebook Prophet has become a well-liked option for time series forecasting by fusing adaptability, performance, and accessibility, powering a variety of applications across sectors. Through the use of an additive, decomposable model, Facebook Prophet forecasts future values using past time series data from the dataset X. The trend, seasonality, and holidays make up the model's three primary elements.

A. Trend Component: The time series data's fundamental direction is modelled by the trend component. It captures the long-term trends, whether they are rising or falling. Usually, the trend is represented by a logistic growth curve or a piecewise linear model.

B. Seasonality Component: The seasonality component takes into account periodic patterns like daily, weekly, or yearly patterns that repeat over set intervals. Prophet can capture both short-term and long-term seasonality because it is modelled as a Fourier series.

C. Holiday Component: This component addresses how holidays and other special occasions may affect the time series in a major way. Prophet gives users the option to identify unique holiday events and the impact they will have on the forecast.

Facebook Prophet formulates the forecasting problem as a probabilistic model and then uses the Bayesian technique to derive the parameters for the trend, seasonality, and holiday components. All model parameters are taken into account in a Bayesian framework as random variables with prior distributions that reflect our pre-data opinions about their values. The posterior distributions, which reflect our assumptions about the parameters after taking into account the observed data, are updated as fresh data is observed, replacing the prior distributions. The following procedures are a part of the Bayesian approach:

A. Model Formulation: Prophet outlines a probabilistic model that breaks down the time series into trends, seasonality, holiday impacts, and error terms. Hyper parameters in the model regulate the degree of regularization and the sensitivity of the model to changes.

B. Prior Distributions: All model parameters are assigned prior distributions. These prior distributions indicate the assumptions made on the parameters prior to any data being observed. For instance, it might be assumed that the trend component has a Gaussian (normal) distribution.

C. Parameter Estimation: The model employs Bayesian inference to convert the prior distributions to posterior distributions given the prior distributions and the observed data. The Stan probabilistic programming language is used for this operation.

$$y(t)=g(t)+s(t)+h(t)+\epsilon(t) \quad (3)$$

where:
y(t) = observed value at time t
g(t) = trend component at time t
s(t) = seasonality component at time t
h(t) = holiday component at time t
ϵ(t) = error term or noise or irregularity at time t
D. Markov Chain Monte Carlo (MCMC): To select samples according to posterior distributions, Prophet uses MCMC sampling. The simulation-based method known as MCMC creates a series of samples that represent the posterior distribution. The resulting means and uncertainties of the parameters are calculated using these samples.
E. Forecasting: Prophet can produce forecasts for future time periods by projecting the trend, seasonality, and holiday effects based on the calculated parameters once the model parameters have been computed using the Bayesian technique.
Markov Chain Monte Carlo (MCMC) sampling is a powerful technique for estimating the posterior distribution of model parameters in Bayesian statistics and machine learning. The posterior distribution is used to generate a series of samples

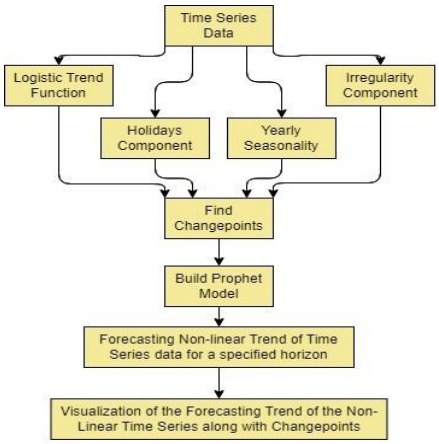


Figure 6: Flow diagram of Facebook Prophet

using this simulation-based method. Building a Markov chain with each state representing a sample from the posterior distribution is the main concept. The Markov chain is built in a way that finally causes the sample distribution to converge to the desired posterior distribution. A factor that determines whether to accept or reject the proposed state is the acceptance probability. The acceptance probability formula used by the Metropolis-Hastings algorithm, an MCMC technique, is as follows:

$$\text{acceptance probability} = \min(1, \alpha) \quad (4)$$

where:

$$\alpha = P(newstate)/P(currentstate)$$

The Metropolis-Hastings method proposes moves in the parameter space and accepts or rejects them depending on the acceptance probability to produce samples from the target distribution iteratively. The gathered samples create a Markov chain over a significant number of iterations that eventually converges to the desired distribution.

VI. Support Vector Regression

The well-known Support Vector Machine (SVM) method has a variant called Support Vector Regression (SVR) that was developed expressly to address regression problems. Instead of discrete class labels, SVR seeks to model and predict continuous numeric data. The fundamental objective of SVR is to maximize the gap between the

predicted values and a predefined error tolerance (epsilon) while finding a hyper plane that best matches the data points. The final regression model is influenced by the support vectors, which are the data points that fall inside this margin. Using kernel functions like the radial basis function (RBF) or polynomial kernels, SVR may handle correlations between features and the target variable that are both linear and non-linear. SVR prevents over fitting and enhances the model's generalization capabilities by applying regularization. In time series analysis, Support Vector Regression (SVR) is important, especially for forecasting and prediction tasks. SVR is a potent technique for capturing intricate temporal patterns in time series and producing precise forecasts based on past data. SVR, in contrast to conventional regression techniques, is capable of handling non-linear correlations between time dependent variables, making it appropriate for modeling and forecasting a variety of time series phenomena, including stock prices, weather patterns, and energy consumption. SVR makes use of the idea of support vectors to pinpoint crucial data points that have a major impact on the regression model, allowing it to concentrate on the time series' most important patterns.

Seasonality, trends, and other common temporal features in time series data can also be handled by SVR thanks to its ability to use kernel functions. By optimizing the hyper parameters and fine-tuning the model, SVR can produce reliable forecasts and predictions that support decision-making processes across a wide range of industries, including finance, economics, healthcare, and many others where time-dependent patterns are essential for comprehending and planning future outcomes.

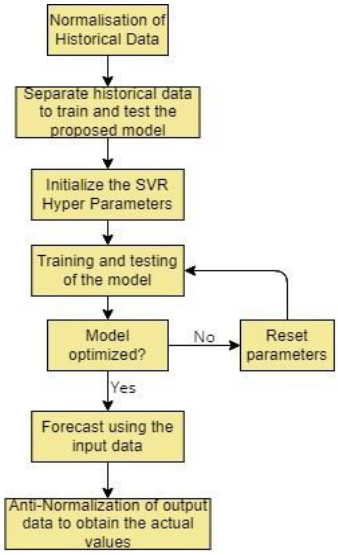


Figure 7: Flow diagram of SVR

A number of mathematical ideas and formulas are crucial to comprehend while performing time series analysis with Support Vector Regression (SVR). They are:

A. Regression Model For SVR: In SVR, the objective is to learn a regression function that forecasts a continuous output (target variable) from a collection of input features (time series data). The SVR regression model's generic form is given by (5)

$$f(x) = \sum_{i=1}^{n_{SV}} \alpha_i K(x, x_i) + b \quad (5)$$

where:

f(x) : Represents the predicted output for a given input feature vector x.

∑_(i=1)^(n_SV) : Denotes the summation over all support vectors, where nSV is the number of support vectors.

αi : Represents the Lagrange multipliers (coefficients) associated with each support vector xi.

K (x,xi) : Denotes the kernel function that measures the similarity between the input feature vector x and the support vector xi. It allows SVR to handle non-linear relationships between variables.

b : Represents the bias term or the y-intercept, which adjusts the position of the regression line.

B. Loss Function: The goal of SVR's loss function is to penalize departures from the target values' actual values. The (ε)-insensitive loss function, which permits a tolerance (ε) for errors within a specific range, is commonly used for time series analysis. A single data point's loss function is as follows:

$$L(y, f(x)) = \max(0, |y - f(x)| - \varepsilon) \quad (6)$$

where:
L(y,f(x)) : Represents the loss function, denoting the error between the true target value y and the predicted value f(x).
max : Denotes the maximum function, which selects the larger value between the two arguments.
0 : Represents the value zero.
|y − f(x)| : Denotes the absolute difference between the true target value y and the predicted value f(x).
ε : Represents the epsilon symbol, which is the specified error tolerance or margin of tolerance around the true target values. Predictions within this tolerance incur zero penalty, while errors outside the tolerance are penalized based on their distance from the threshold.
C. Objective Function: For the purpose of limiting model complexity and avoiding over fitting, the objective function in SVR combines the loss function and a regularization term. The formulation of the objective function is given by Eq. (7):

$$\min_{w,b,\xi,\xi^*} \frac{1}{2}||\omega||^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*) \quad (7)$$

where:
min : Represents the minimization objective, aiming to find the optimal values of the variables.
w : Denotes the weight vector in the SVR model.
b : Represents the bias term or the y-intercept in the SVR model.
ξ : Denotes the slack variable for data points above the margin.
D. Kernel Trick: In time series analysis, the data may not be linearly separable, and a linear kernel may not be adequate to detect intricate patterns. Using the kernel method, SVR can find a non-linear decision boundary by implicitly transforming the input into a higher-dimensional space. In time series analysis, common kernel functions include: Kernels:

- Linear Kernel:

$$K(x_i, x_j) = x_i \cdot x_j \quad (8)$$

- Polynomial Kernel:

$$K(x_i, x_j) = (x_i \cdot x_j + r)^d \quad (9)$$

- Radial Basis Function (RBF) Kernel:

$$K(x_i, x_j) = \exp\left(-\gamma \left||x_i - x_j|\right|^2\right) \quad (10)$$

- Sigmoid Kernel:

$$K(x_i, x_j) = \tanh(\alpha x_i \cdot x_j + c) \quad (11)$$

The Radial Basis Function (RBF) kernel's adaptability makes it useful for a variety of applications since it can handle data that cannot be efficiently segregated by a linear decision boundary. Additionally, the gamma () parameter of the RBF kernel regulates the decision boundary's smoothness, offering finetuning possibilities to accommodate various datasets, Hence RBF kernel is utilized in our SVR model.
E. Hyperparameter Optimisation: SVR requires adjusting kernel-specific parameters as well as hyper parameters like C and . In our SVR model, a hyperparameter grid is defined, including the 'C' (regularization parameter) and '' (kernel coefficient) for the RBF kernel. The SVR model is created with an initial epsilon value of 0.1, and hyperparameter tuning is performed using GridSearchCV with 5-fold cross-validation and parallel processing.
F. Support Vectors: In SVR, the data points that fall within the margin and have an impact on the final regression model are referred to as support vectors. When creating predictions and setting the decision boundaries, they are extremely important.

VII. RNN-LSTM.

An artificial neural network model known as an RNN-LSTM (Recurrent Neural Network - Long Short-Term Memory) is a particular kind of artificial neural network made for sequential data processing. It is especially well suited for tasks requiring time-series data, as well as for tasks involving NLP, speech recognition, and other procedures where the context and order of the input data are essential.

Time series analysis now has access to a potent and adaptable tool in RNN-LSTM models. For jobs like stock market forecasting, energy demand prediction, weather forecasting, and others, they are well suited because of their innate capacity to recognize sequential relationships and long-term patterns. Sequentially processing historical data allows LSTM units to acquire complicated temporal correlations and preserve valuable temporal information, resulting in more accurate predictions and a better comprehension of the underlying trends in time series data. Additionally, the model can adjust to various time scales and irregular data patterns thanks to its ability to handle variable-length sequences. The parts of an RNN-LSTM model are as follows:

A. Recurrent Neural Network (RNN): An RNN is a kind of neural network that incorporates loops into its architecture, allowing for the persistence of information over time. It analyses data sequences one element at a time while keeping hidden states that preserve the sequence's historical context. Standard RNNs, however, are constrained by the vanishing gradient problem in capturing long-term dependencies.

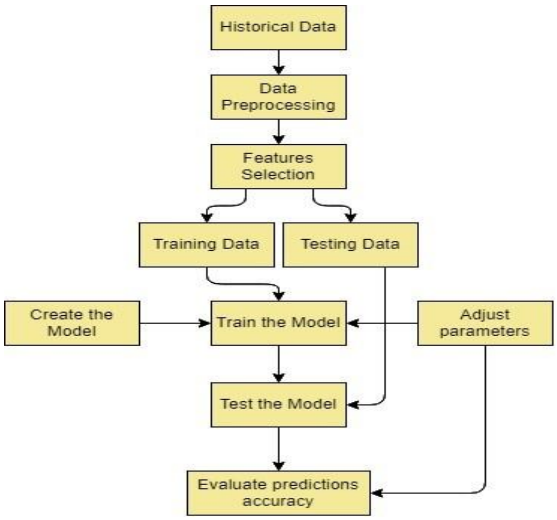


Figure 8: Flow diagram of RNN-LSTM

B. Long Short-Term Memory (LSTM): The basic RNN is expanded by the LSTM to address the vanishing gradient issue by adding specialized memory cells. The three primary parts of these memory cells are an input gate, a forget gate, and an output gate. These gates regulate the information flow into, out of, and within the memory cells, helping LSTMs learn and retain knowledge for longer periods of time and improving their ability to recognize long-term dependencies in the data.

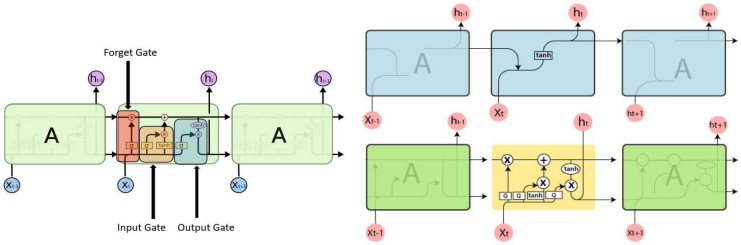
The architecture of an RNN-LSTM model for time series analysis consists of a number of crucial elements that work in concert to analyse time series input and produce predictions. It is as follows:

A. Input Sequence: The time series data is represented as a series of data points, each of which has one or more features. A simple univariate time series, for instance, might simply examine one feature for each time step, while a multivariate time series might take into account several aspects.

B. LSTM Layers: The core elements of the RNN-LSTM model are the LSTM layers. The input gate, forget gate, and output gate are the three main gates of the LSTM cells that make up these layers. These gates control the information flow within the LSTM cell, allowing it to gradually gather and store important facts. The model can keep memory and capture dependencies over time because the LSTM cells process the sequential data and send the hidden state from one time step as input to the following time step.

C. Output Layer: To create the final prediction, the output of the final LSTM layer is often routed via one or more fully linked (dense) layers. The output layer may include a single node for regression tasks, where the model predicts a continuous value, or numerous nodes for classification activities, where there are various classifications to choose from, depending on the particular task.

D. Training: To reduce the variation between the anticipated output and the actual target values, the model's parameters including its weights and biases—are iteratively changed during training. A suitable loss function that measures the prediction error is used to evaluate the performance of the model.
E. Forecasting: Once trained, the model can be utilized for time series forecasting. When forecasting, the model consumes a series of previous data and incrementally predicts future values. The model can produce a forecast for the full future time horizon since each phase's anticipated value is given back into it as input for the subsequent stage.
An input gate, an output gate, a forget gate, and a cell make up a standard LSTM unit. The cell may store values for any amount of time, and these gates regulate the flow of information into and out of the cell.



(a) General LSTM Unit (b) Complete RNN Cell

Figure 9: LSTM Unit and RNN Cell

Input Gate (i_t)

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (12)$$

where:
i_t : Activation value of the input gate at time step t.
σ : Sigmoid activation function, which squashes the value between 0 and 1.
W_i: Weight matrix for the input gate.
Forget Gate (f_t)

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (13)$$

where:
f_t : Activation value of the forget gate at time step t.
σ : Sigmoid activation function.
W_f: Weight matrix for the forget gate.
h_((t-1)): Previous hidden state at time step t − 1.
x_t : Current input at time step t.
b_f: Bias term for the forget gate.
Output Gate (i_t)

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (14)$$

where:
o_t: Activation value of the output gate at time step t.
σ : Sigmoid activation function.
W_o : Weight matrix for the output gate.
h_((t-1)) : Previous hidden state at time step t − 1.
x_t: Current input at time step t.
b_o: Bias term for the output gate.

VIII. Experimental Result Analysis.

A. Historical Data: In this experiment, historical bond data of the US Treasury 10 Year Yield (TNX) from 10 November 2016 to 19 July 2023 have been taken into account. The total number of samples gathered for this particular bond has been separated into training and testing data sets after pre-processing. The bond's Close price and the associated date make up the training sample.

i. Scaling the dataset: Scaling is important in machine learning because it ensures that all feature ranges and magnitudes are comparable, preventing the dominance of any particular feature in the learning process. Numerous machine learning algorithms depend on mathematical calculations and optimization strategies that work best when feature values are on a similar scale. Without scaling, our models could have trouble locating the best solutions, which would result in longer training times and less accurate prediction. Since it guarantees that all features are scaled equally, the MinMaxScaler is a well-liked option for scaling data. This can help the SVR and RNN-LSTM models perform better. By providing a fair comparison of feature importance, it makes sure that no one feature dominates the learning process. This scaler transforms our data by scaling each feature to range between 0 and 1.

$$\text{scaled\_value} = \frac{x-min}{max-min} \qquad (15)$$

where:
x is the original value of a feature,
min is the minimum value of that feature,
max is the maximum value of that feature.
The MinMaxScaler has two functions fit transform and transform. Both fitting the scaler to the training data and converting the training data into the scaled version are done using the MinMaxScaler fit transform method. Using the scaling parameters discovered from the training data, the transform method is used to scale the test data. The Min-Max scaling technique converts all features in the training and test sets to a common range of [0, 1], which enhances the performance of the SVR model, speeds up training convergence, and prevents potential numerical instabilities.
ii. Hyperparameter Optimization: Grid search was used to identify the SVR model's ideal hyper parameters. The grid search criteria were as follows:
C (Regularization parameter) =[1, 10, 100]
(Kernel coefficient for the rbf kernel) = ['auto', 'scale']
(Epsilon tube parameter) = [0.1,0.01]
For historical data, C=1 turned out to be the ideal regularization parameter
(C) value. We have taken into account that the RNN-LSTM model will have 1 feature and 48 lag hours as we are only utilizing historical data to train our model. With 100 epochs and a batch size of 30, the dataset was fitted to the model. In order to estimate the model parameters, Facebook Prophet uses a technique known as Markov Chain Monte Carlo (MCMC) sampling. Prophet takes care of all of this in a discreet manner. There is no need to provide any criteria.
iii. Evaluation Metrics: The models have been evaluated on the basis of three parameters - Mean Squared Error (MSE), Root Mean Squared Error (RMSE) and Mean Absolute Percentage Error.
Mean Squared Error (MSE): A well-liked statistic for evaluating the effectiveness of regression models is mean squared error. A lower MSE signifies a better fit to the data, with the predictions of the model being nearer to the true values. Because of the squaring operation, it is highly susceptible to outliers because larger errors are penalized more severely than smaller ones. The formula for calculating MSE is as follows:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2 \qquad (16)$$

where:
n is the number of data points,
y_iis the actual (observed) value,
ŷ_i is the predicted value,
Σ denotes the summation over all data points.
Root Mean Squared Error (RMSE): The average squared difference between the actual (observed) values and the model-predicted values is what is evaluated. Since it provides a measurement of the typical error in the model's predictions in the same units as the target variable, RMSE is frequently preferred to MSE. A lower RMSE reflects a better fit to the data because it shows that the model's predictions are closer to the true values. The formula for calculating RMSE is as given below:

$$RMSE = \sqrt{\left\{\left(\frac{1}{n}\right)\cdot\Sigma(y_i - \hat{y_i})^2\right\}} \qquad (17)$$

where:
n is the number of data points,

y_i is the actual (observed) value,
(y_i)ˆis the predicted value,
Σ denotes the summation over all data points.

Mean Absolute Percentage Error (MAPE): Mean Absolute Percentage Error (MAPE) is used to assess how accurate forecasts or predictions are. MAPE, which is expressed as a percentage, offers a relative measurement of the prediction error. With a value of 0% denoting a perfect match between predictions and actual observations, a lower MAPE indicates greater accuracy. The formula for MAPE is:

$$\text{MAPE} = \frac{100}{n}\sum_{i=1}^{n}\left|\frac{y_i-\widehat{y_i}}{y_i}\right| \qquad (18)$$

where:
n is the number of data points,
y_i is the actual (observed) value,
(y_i)ˆis the predicted value,
Σ denotes the summation over all data points.

Following the training phase, a training model is produced, and the remaining samples are tested using this model. The close price of the bond (historical data) was used as the sole parameter for training and testing the three models. The findings are summarized in the following three tables. The best-performing model is indicated by the highlighted row, and the provided charts display its performance.

B. Historical Data and Market Sentiment: Historical bond information for the US Treasury 10 Year Yield (TNX) has been taken into account, along with news headlines about this bond from 10 November 2016 to 19 July 2023. The exact same procedures that were described before for historical data have been used for training and testing. The sentiment score of each day's news headlines was obtained using VADER. VADER is pre-trained on a labeled dataset and relies on a lexicon containing words and their corresponding sentiment scores. All compound sentiment scores have been encoded as 0, 1, and -1 to make training the model easier. In order to express a positive sentiment in the market, all headlines having a compound sentiment score greater than 0.1 have been encoded as 1. Similar to this, every headline with a compound score below -0.1 has been encoded with the number -1 to signify negative market sentiment. To indicate neutral sentiment, all compound scores in the range (-0.1,0.1) have been encoded as 0.

The hyper parameters for fitting the dataset to the SVR model alter because we are now using two factors, namely the bond's close price and the sentiment score of the news headlines. Following Grid Search technique to discover the best hyperparameter, 10 and 0.1 are found to be the optimal regularization parameter (C) and kernel coefficient for the rbf kernel (gamma) respectively.

With two parameters to train our model, we have presumed that the number of features for the RNN-LSTM model would be 2, and the number of lag hours would be 48 hours. The number of epochs and batch size used to fit the dataset to the model remain the same as before (epochs=100, batch size=30).

The best parameters are estimated and the dataset is fitted to the model once again by Facebook Prophet using Markov Chain Monte Carlo (MCMC) sampling, a type of Bayesian inference. The close price of the bond (Historical data) and the sentiment score of the news headlines (Market Sentiment) were the two parameters used for training and testing the three models. The results are summarized in the following tables.

80-20 split	MSE	RMSE	MAPE
SVR	1.3246	1.1509	30.873
RNN-LSTM	0.280	0.530	14.611
Facebook Prophet	0.0127	0.1130	3.756

(a) 80-20 train-test split

80-20 split	MSE	RMSE	MAPE
SVR	0.0155	0.1245	2.997
RNN-LSTM	0.193	0.440	10.06
Facebook Prophet	0.0131	0.1146	3.813

(b) 80-20 train-test split

70-30 split	MSE	RMSE	MAPE
SVR	2.2057	1.4851	41.895
RNN-LSTM	0.356	0.597	16.601
Facebook Prophet	0.0102	0.1012	3.223

(c) 70-30 train-test split

70-30 split	MSE	RMSE	MAPE
SVR	0.0104	0.1022	2.363
RNN-LSTM	0.275	0.524	15.33
Facebook Prophet	0.0101	0.1023	3.235

(d) 70-30 train-test split

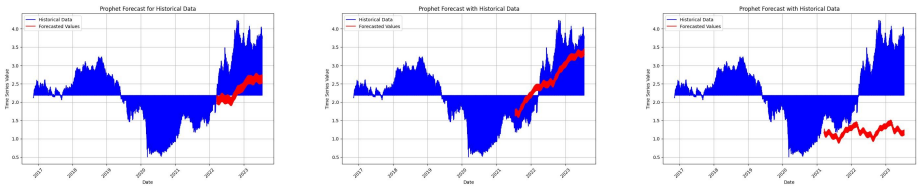
60-40 split	MSE	RMSE	MAPE
SVR	0.8917	0.9443	33.056
RNN-LSTM	0.230	0.479	14.490
Facebook Prophet	0.0141	0.1189	8.477

(e) 60-40 train-test split

60-40 split	MSE	RMSE	MAPE
SVR	0.0081	0.0903	2.352
RNN-LSTM	0.239	0.489	15.86
Facebook Prophet	0.0139	0.1181	16.343

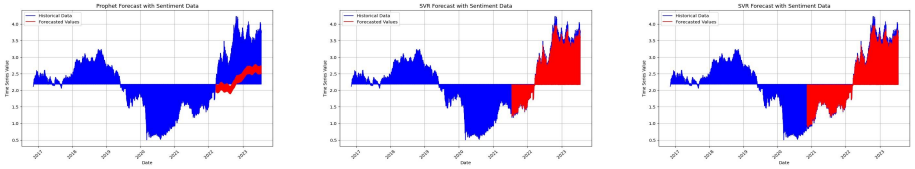
(f) 60-40 train-test split

Figure 10: Time Series Prediction and Time Series Prediction + Sentiment Analysis Results Column-wise



(a) 80-20 train-test split (b) 70-30 train-test split (c) 60-40 train-test split

Figure 11: Row I: Time Series Prediction



(a) Prophet - 80-20 split (b) SVR - 70-30 split (c) SVR - 60-40 split

Figure 12: Row II: Time Series Prediction + Sentiment Analysis Results

IX. Conclusion

Two analogies are made in this paper. The first comparison is based on the three models' accuracy: Support Vector Regression, RNN-LSTM, and Facebook Prophet. In the second comparison, a model that just used historical data to train is put up against a model that also considered market sentiment. The performance of the models is examined using Mean Square Error (MSE), Root Mean Square Error (RMSE), and Mean Absolute Percentage Error (MAPE). Any discrepancies between the three values to determine the best model have been resolved by examining the RMSE value. According to our research, Facebook Prophet has been able to identify patterns on historical data the most precisely and is the closest to actual values. Both SVR and Facebook Prophet perform well when both, Historical Data and Market Sentiment are incorporated into training the models, but SVR appears to have a slight advantage over Prophet in this case. Additionally, we see that the predictions improve when market sentiment is taken into account during model training, demonstrating its influence on future bond prices.

Acknowledgements

This work was supported by Centre for Data Science and Applied Machine Learning (CDSAML), PES University, Bangalore, India, Pin: 560085.

References

[1] Ananda Chatterjee, Hrisav Bhowmick and Jaydip Sen, "Stock Price Prediction Using Time Series, Econometric, Machine Learning, and Deep Learning Models", IEEE, December 2021, doi: 10.1109/MysuruCon52639.2021.9641610.

[2] Smita Mohanty and Rajashree Dash, "A Support Vector Regression Framework For Indian Bond Price Prediction", IEEE, May 2019, doi: 10.1109/ICAML48257.2019.00021 .

[3] Swetava Ganguli and Jared Dunnmon, "Machine Learning for Better Models for Predicting Bond Prices", Cornell University, March 2017, doi: 10.48550/arXiv.1705.01142 .

[4] Manuel Clemente Mendonca Nunes, "Machine Learning in Fixed Income Markets: Forecasting and Portfolio Management - Thesis", University of Southampton, January 2022, ORCiD: 0000-0002-7116-5502

[5] Aditya Bharadwaj and Yogendra Narayan, "Sentiment Analysis for Indian Stock Market Prediction Using Sensex and Nifty",Procedia Computer Science, Volume 70, 2015, doi: 10.1016/j.procs.2015.10.043.

[6] Xiao Ding, Yue Zhang, Ting Liu, Junwen Duan, "Deep Learning for Event-Driven Stock Prediction", IJCAI 2015 - Proceedings of the 24th International Conference on Artificial Intelligence, doi: https://dl.acm.org/doi/10.5555/2832415.2832572.

[7] Hsiao-Chuan Chou1 and Kandethody M. Ramachandran, “Combining Time Series and Sentiment Analysis for Stock Market Forecasting“, ICSTA July 2021, Proceedings of the 3rd International Conference on Statistics: Theory and Applications, doi: 10.11159/icsta21.132 .
[8] Ashish Shrestha and Mr. Tej Bahadur Shahi, “Stock Market Forecasting with LSTM and Sentiment Analysis“, January 2020, Central Department of Computer Science and Information Technology, Tribhuvan University, Nepal.
[9] Faruk Ahmad and Md-Mizanur Rahoman, ”Named Entity Classification using Dependency Grammar”, 2017 20th International Conference of Computer and Information Technology (ICCIT), 22-24 December, 2017.
[10] Lee, K.L. Billings, S.A.. (2002). Time Series Prediction Using Support Vector Machines, the Orthogonal and the Regularised Orthogonal Least Squares Algorithms. International Journal of Systems Science. 33. 811-821.10.1080/0020772021000017317.
[11] Yahoo Finance: https://finance.yahoo.com
[12] Financial Times: https://www.ft.com/us-treasury-bonds
[13] LSTM Diagrams: https://www.javatpoint.com/long-short-term-memoryrnn-in-tensorflow