Duale Hochschule Baden-Württemberg Mannheim

Project Thesis

Machine Learning-Based Price Forecasting: A Case Study of Commodity Prices in Nepal

Business Information Systems

Data Science

Author:	Lukas Stamm
Matriculation No.:	8402366
Semester:	4th
Company:	Accenture
Department:	Data & AI
Course:	WWI22DSA
Degree Program Director:	Prof. habil. Dr.-Ing. Dennis Pfisterer
Scientific Advisor:	Jan O. Bauer
Company Advisor:	Yerzhan Mussatayev
Time Frame:	14.11.2024 – 07.01.2025

Disclaimer usage of Generative AI

All ideas, research, and conclusions presented in this paper are the original work of the author. No section of the text was generated by Generative AI tools of any form. AI was only used in a way to improve clarity and formulation of text already written by the author. No AI-enhanced text was directly included in the paper, but solely served as inspiration to reword the text manually.

Statement of Originality

The author of the present work, titled *“Machine Learning-Based Price Forecasting: A Case Study of Commodity Prices in Nepal”* hereby confirms that no other source or tools were used to create this paper other than those indicated. The submitted electronic version is in accordance with the printed version.

Mannheim, 06.01.2025

Place, Date

Lukas Stamm

Abstract

Accurate forecasting of agricultural commodity prices is essential for infirmed decision-making by farmers, traders and policymakers. This study evaluates the performance of three different machine learning models, LASSO Regression, XGBoost and LSTMs, in predicting Nepalese commodity prices with focus on the red potato. Utilizing a time series dataset with price data between 2013-2021, the study investigates the effect of trends, seasonality and anomalies on the performances of the models. Other external factors, such as weather and economic data, were also included in an effort to enhance the predictions, though limitations on data resolution and accuracy ultimately shaped the final approach.

The conducted simulation study concluded with a LSTM as the best performing model throughout various scenarios including amplified trends, noise, and anomalies. XGBoost achieved lower performance scores, due to its inability to handle abrupt changes and anomalies within the data, but still provided decent performance. LASSO regression, while interpretable, struggled to account for non-linear patterns or dynamic changes.

Overall, the results emphasized the need for sufficient external data and contextual information to account for real-world complexities and factors, such as sudden market changes brought upon events like COVID-19. It underlines the need for domain specific knowledge, data collection and advanced modeling techniques to account for complex patterns within the data and offers analytical insight into commodity prices within Nepal.

Table of Contents

List of Equations

List of Figures

List of Tables

1 Introduction

The accurate prediction of agricultural commodity prices offers vital knowledge towards farmers, traders, as well as policymakers, enabling them to make informed decisions to minimize risks and maximize opportunities. Time Series Forecasting emerges as the essential approach for enabling these predictions, by analyzing historical data and predicting future trends through analyzed patterns. By capturing complex patterns, seasonality and external factors that drive prices, time series forecasting provides a powerful tool to make accurate predictions.

This study investigates the performance and limitations of various machine learning models in the context of Nepalese agricultural commodity prices with a focus on red potatoes. It will compare how various models adapt to unique challenges within time series data, such as trends, seasonality and sudden changes due to external factors. Particular attention will be given to the ability of these models to balance accuracy and adaptability to the dynamic changes in prices.

The methodology involves feature collection and generation, model development and a simulation study to evaluate the performance of the developed models. The study will focus on comparing LASSO regression, XGBoost and LSTMs, with the goal of understanding their strengths and weaknesses to capture complex temporal dependencies and external factors. Additionally, it will focus on the importance of feature collection and the presence of contextual data, such as weather data, economic and agricultural data.

This study adds to the extensive body of literature on commodity price forecasting by analyzing the trade-offs between model complexity, data availability and forecasting accuracy. These results not only provide insights into model selection in specific scenarios but also emphasize the potential for hybrid approaches and enriched data collection to overcome challenges in real-world forecasting scenarios. They seek to provide actionable recommendations for more robust decision making in agricultural markets and beyond.

2 Theoretical Background

2.1 Time Series Forecasting

Neglecting to understand and account for future price trends can be detrimental to any farmer, trader, or policymaker who is trying to plan and shape decisions and policies for the future. Time series forecasting offers a methodology which can help anticipate these price trends and offer crucial insights into the future, by analyzing past time series data and predicting future values based on it.

Performance of these models is highly dependent on the understanding and correct analysis of the time series data and its key components, as well as its connected concept of stationarity. This section will serve as an introduction to these concepts.

2.1.1 Time Series Components

Time series data is typically composed of three primary components: trend, seasonality and residual.

1. **Trend:** The trend component, also referred to as the level depicts the slow-moving, general direction of a time series over time, either increasing or decreasing steadily [1]. Trends are critical for long-term price forecasting as they offer insights into the general direction of the market, an example being inflation, steadily moving prices upwards in the long term.
2. **Seasonality:** The seasonal component shows the regular and predictable patterns of a time series that reoccur at fixed intervals of time, such as daily, monthly, or yearly [1]. These patterns indicate positive and negative deviations from the trend and can be caused by natural or artificial factors. For example, prices of certain fruits can vary over the year based on their seasonal availability.
3. **Residual:** After accounting for the trend and seasonality, all remaining variations are represented within the residual. It captures the random noise or unexplained fluctuations, that cannot be attributed to the main components of the series and cannot be foreseen [1]. Examples can be unexpected natural disasters inhibiting crop growth and shortening supply.

Time series data can be broken down into its components through decomposition. This is an essential step for capturing and understanding the underlying properties of the data.

2.1.2 Stationarity and Its Importance

Required by many models, like the Autoregressive models, stationarity poses as a fundamental concept in time series analysis. A time series is presumed stationary, when its properties, such as the mean, variance and autocorrelation, do not change over time and are therefore independent of it [2].

This presumption is necessary to ensure that selected model parameters remain consistent with the given time series. Changing properties would otherwise affect the model's fit to the data and render possible predictions inaccurate. In most real-world cases, stationary time series are not present from the get-go. This is often due to trend and seasonality being naturally present, requiring the time series to be transformed to become stationary before modeling.

Differencing is the simplest method of achieving stationarity. It involves stabilizing the mean, by subtracting the previous value of a timestamp y_{t-1} from the present value y_t, resulting in the differenced value y'_t [2].

$$y'_t = y_t - y_{t-1} \tag{2.1}$$

Additionally, in order to stabilize the variance a log function can be applied to the series.

Applying differencing once is referred to as first-order differencing. If necessary, multiple rounds of differencing (second-order, third-order differencing etc.) can be applied until the series is ultimately stationary.

For time series data with a seasonal component, seasonal differencing can be applied. Here, instead of subtracting neighboring values from another, the differencing is applied on values m data points apart to account for the seasonality.

It is important to remember to inverse difference the model results afterwards, to return the datapoints to their original unit measurements and make them interpretable.

After each round of differencing the augmented Dickey-Fuller (ADF) test can be applied to test whether the transformation successfully made the series stationary. This method determines stationarity by testing for the null hypothesis that a unit root is present. If the root of a time series lies between -1 and 1, the series is stationary, as there is no unit root present [2]. Conversely, if the root equals 1, this indicates the presence of a unit root, making the series non-stationary, and making another round of differencing necessary. A common way to determine whether the hypothesis can be rejected or not is calculating the p-value. A p-value of less than 0.05 rejects the null hypothesis and indicates stationarity.

2.2 LASSO Regression

Regularization methods such as the Least Absolute Shrinkage and Selection Operator (LASSO regression) are used in statistical modeling and machine learning to improve prediction accuracy and interpretability. LASSO achieves this by placing a constraint on the sum of the absolute values of the models, thus shrinking some of them close to zero. This makes it especially suitable for feature selection and creating parsimonious models.

In the traditional approach, linear regression calculates the best-fitted line by minimizing the sum of the squared residuals. LASSO regression approaches this by appending a penalty term λ proportional to the sum of absolute values of the coefficients. The loss function can be defined as follows [3]:

$$Loss\ Function\ =\ Sum\ of\ Squared\ Residuals\ +\ \lambda \sum |\beta_j| \qquad (2.2)$$

Here λ is a non-negative hyperparameter that controls the strength of the penalty. When λ is set to 0, LASSO regression operates the same as ordinary least squares regression, where no penalty is applied at all. Increasing λ and therefore increasing the penalty, leads to a greater shrinkage of coefficient estimates towards zero.

This enables LASSO regression to shrink some coefficients to exactly zero whenever λ is large enough. Through this, feature selection among predictors can be facilitated, by excluding less important predictors, leading to simpler and more

interpretable models. This capability demonstrates its usefulness in high dimensional datasets, where selecting variables automatically is crucial in producing sparse models, free from redundant attributes.

It is important to note that choosing an appropriate λ is important, as it controls the amount of shrinkage. A common method for determining λ is cross-validation, which helps prevent overfitting by balancing the model's simplicity and its predictive performance on unseen data [4].

2.2.1 The LASSO for Time Series

Therefore, because of the commonly found high dimensionality in time series data, LASSO regression proves especially useful in making accurate predictions. Effective handling of overfitting and determination of the most relevant predictors is achieved through focusing on the important lagged variables (autoregressive terms), while excluding weak ones, thus leading to more streamlined models.

A univariate autoregressive (AR) model can be defined as [2]:

$$y_t = \beta_1 y_{t-1} + \beta_2 y_{t-2} + \cdots + \beta_p y_{t-p} \tag{2.3}$$

Where:

- y_t indicate the value at time t,
- $y_{t-1}, y_{t-2}, \ldots, y_{t-p}$ are lagged observations,
- $\beta_1, \beta_2, \ldots, \beta_p$ are coefficient estimates.

LASSO regression modifies the AR model by introducing a penalty on the absolute values of the coefficients, ensuring some coefficients are driven to 0. The resulting optimization problem of LASSO regression can be formulated as [3]:

$$Minimize \sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \tag{2.4}$$

Here:

- n is the number of observations,
- p is the number of lagged covariates,
- λ is a hyperparameter controlling the regularization strength.

The penalty term $\lambda \sum_{j=1}^{p} |\beta_j|$ enforces the desired sparsity, driving the coefficients of irrelevant variables to zero.

This makes LASSO regression a valuable tool for simplifying time series models by focusing on the most relevant lagged variables.

2.3 XGBoost for Regression

Extreme Gradient Boosting, or short XGBoost is an optimized implementation of the gradient boosting algorithm designed for achieving superior performance and accuracy by leveraging advanced optimization techniques and pushing computational limits [5]. It is especially popular as it allows for efficient scaling on high-dimensional, large-scale datasets, as is often the case in time series data.

2.3.1 Boosting in XGBoost

At its core, XGBoost is a supervised learning algorithm based on the boosting principle, where an ensemble of weak learners, typically decision trees, are combined sequentially to produce a strong final model. In this iterative approach, each subsequent tree tries to minimize the residual error (loss) of the previous trees. This stands in contrast to bagging techniques, as used by Random Forest, where each tree is trained independently and grown randomly. This targeted learning mechanism is what allows XGBoost to efficiently model complex, non-linear relationships.

Mathematically, the goal of boosting is to minimize the following objective function [6]:

$$obj(\theta) = \sum_{i}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \omega(f_k) \tag{2.5}$$

Where $l(y_i, \hat{y}_i)$ is the loss function, representing the difference between the true value y_i and the predicted value $\hat{y}_i$ and $\omega(f_k)$ is the regularization term to penalize the model complexity. Further, θ represent the model parameters and the T the number of trees in the model.

The regularization term $\omega(f)$ is given as [6]:

$$\omega(f) = \gamma T + \frac{\lambda}{2}\sum_{j=1}^{T} w_j^2 \qquad (2.6)$$

where γ penalizes the number of leaves T in the tree and λ puts an L2 penalty on the leaf weights w_j. This regularization mechanism prevents XGBoost from fitting overly complex models that are susceptible to overfitting, while still capturing essential patterns in the data.

2.3.2 Gradient Boosting with Second-Order Approximation

To further improve both accuracy and computational efficiency, XGBoost optimizes the objective using a second-order Taylor approximation of the loss function. This leverages both the first-order gradient (g_i) and second-order Hessian (h_i) of the loss with respect to the model's prediction at iteration t [7]:

$$\bar{L}^{(t)} \approx \sum_{i=1}^{n}\left[g_i f_t(x_i) + \frac{1}{2}h_i f_t^2(x_i) + \Omega(f_t)\right] \qquad (2.7)$$

where:

- $g_i = \frac{\partial l(yi,\hat{y}i)}{\partial yi}$ is the first order gradient, indicating the direction of the loss,
- $h_i = \frac{\partial^2 l(yi,\hat{y}i)}{\partial \hat{y}_i^2}$ is the second order Hessian, which captures the curvature of the loss function.

The inclusion of the Hessian into XGBoost provides information about the curvature of the loss, which enables faster and more stable optimization than other methods where only the first-order information is used.

2.3.3 Leaf Weight Optimization

Given a tree structure, XGBoost optimizes the weights w_j of every leaf j to minimize the regularized loss. The optimal weight for a leaf is computed as [7]:

$$w_j^* = -\frac{\sum_{i\in \mathrm{I}_j} g_i}{\sum_{i\in \mathrm{I}_j} h_i + \lambda} \tag{2.8}$$

where I_j is the set of instances assigned to leaf j.

The resulting quality score for a tree is derived as [7]:

$$L_{opt} = -\frac{1}{2}\sum_{j=1}^{T} \frac{\left(\sum_{i\in \mathrm{I}_j} g_i\right)^2}{\sum_{i\in \mathrm{I}_j} h_i + \lambda} + \gamma \mathrm{T} \tag{2.9}$$

This score is used to determine how well the split fits and is used as a criterion for selecting the best split when constructing the tree.

2.3.4 Sparsity-aware Learning and Regularization Techniques

XGBoost uses advanced techniques to improve efficiency and generalization. For datasets with missing or sparse features, a sparsity-aware algorithm learns the best default split directions for handling missing values. Furthermore, shrinkage (also known as learning rate scaling) and column subsampling are used to prevent overfitting and improve computation. Shrinkage multiplies the contribution of each new tree by a constant $\eta \in (0,1]$, which provides a way to obtain a good generalization of the model [7].

Together, these features enable the efficient handling of large datasets while providing a robust model for time series data and prediction tasks.

2.4 Long Short-Term Memory Networks (LSTM)

Long Short-Term Memory networks, short LSTMs, are a type of recurrent neural network (RNN) that address the challenges associated with learning long-term dependencies in sequential data while avoiding issues such as vanishing and

exploding gradients. This capability is particularly helpful in time series analysis, where dependencies may extend over long temporal intervals.

2.4.1 Fundamental Architecture

LSTMs incorporate a mechanism known as Constant Error Carousel (CEC) which is responsible for preserving a constant error flow during training. This is critical for enabling the network to accommodate to very long sequences, exceeding a thousand timestep lags so that stable gradients can be preserved over long periods of time [8]:

Each LSTM unit contains a set of gates to manage the information flow: the input gate, the forget gate, the output gate as well as the central component, the cell state. These elements are responsible for the adaptability of the LSTM to different data sequences.

- **Cell State:** The cell state $c(t)$ is like a conveyor belt, carrying relevant information throughout the sequencing process without many transformations. It gets updated through operations that are regulated by the input and forget gates [9].

- **Input gate:** The input gate $i(t)$ determines how much new information enters the cell state. It's defined through the following equation [9]:

$$i(t) = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.10}$$

  where σ denotes the sigmoid function, W_i is the weight matrix for the input gate, h_{t-1} is the previous output, x_t is the current input, and b_i is the bias.

- **Forget Gate:** The forget gate $f(t)$ controls what information is removed from the cell state. It is formulated as [9]:

$$f(t) = \sigma\big(W_f \cdot [h_{t-1}, x_t] + b_f\big) \tag{2.11}$$

  The parameter W_f represents the weight matrix for the forget gate.

- **Cell State Update**: Cell state is updated by [9]:

$$c(t) = f(t) * c(t-1) + i(t) * tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \quad (2.12)$$

W_c here is the weight matrix associated with the cell state, and $tanh$ is the hyperbolic tangent function that adds non-linearity to the model.

- **Output Gate:** The output gate $o(t)$ controls the output from the cell state to the next layer [9]:

$$o(t) = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (2.13)$$

and the output of the LSTM unit can be defined by [9]:

$$h(t) = o(t) * tanh(c(t)) \quad (2.14)$$

where W_o is the weight matrix associated with the output gate.

2.4.2 Application to Time Series

The ability of LSTMs to remember and forget is critical in time series forecasting. This selective memory allows the network to keep only essential historical information that is useful for the model, such as trends or seasonality [9]. This feature enables LSTMs to forecast future data points based on complex patterns and dependencies in past data, which is critical for accurate predictions in areas like financial markets, energy load forecasting, and more.

2.4.3 Enhanced Variants and Practical Use

Several variants of the basic LSTM architecture, including peephole connections and gated recurrent units (GRUs), have been proposed to improve computational efficiency and enhance model performance. Such improvements enable the LSTM architecture to be tailored to a particular task, which often requires faster calculations without a significant trade-off in performance [8].

Such flexibility and efficiency position Long Short-Term Memory networks as a key component for modeling a range of complex time series modeling applications across multiple domains.

3 Related Work

3.1 Advancements in Commodity Price Forecasting

There is an increasing body of literature highlighting several machine learning techniques, which are able to model the complex, non-linear dynamics of commodity price movements. For instance, Mishra et al. [10] demonstrate how machine learning models can process high-dimensional data and make reliable predictions in the case of cotton prices. Their study highlights the benefits of feature selection and integration of external variables such as weather patterns and market indices into the models. Those insights are consistent with broader trends in forecasting, such as greater use of external factors like global demand swings and trade policies, affecting the price of commodities.

Similarly, Gu et al. [11]. adopted neural networks customized for agricultural price forecasting, namely LSTMs. The study concluded that LSTMs were able to outperform classical regression models through capturing temporal dependencies more effectively and avoiding overfitting with noisy datasets. These findings are supported by other studies which show that LSTMs are well-suited for the typical seasonality and sudden price fluctuations in agricultural markets. However, they caution that these models can be computationally intensive and data-hungry, therefore not always being suitbale.

Furthermore, studies have examined the robustness of simpler learning methods such as LASSO regression and advanced ensemble algorithms such as XGBoost. Research by Huang et al. [12] on predictive modeling of blood pressure finds that these models also provide high performance in environments with limited data, offering a compromise between model interpretability and performance. LASSO regression is recognized for its ability to address multicollinearity, which is a common challenge in economic data, by automatically selecting the most relevant features and avoiding overfitting the model. On the other hand, XGBoost is known for its

ability to handle large datasets and its versatility, which allows it to be used for large-scale prediction tasks.

3.2 Comparative Insights Across Studies

A recurring theme across literature is the trade-off between model complexity and real-world applicability. While more advanced models such as LSTMs can provide better accuracy in learning complex non-linear trends and seasonal components, simpler models can be suitable for cases where compute power is limited, or only short-term forecasting is required. For example, Huang et al. [12] show that given the right conditions, linear regression models with regularization such as the LASSO can perform as well or better than more complex algorithms. This emphasizes the necessity of customizing model selection for the specified data and environment conditions.

Notably, a study by Luo et al. [13] discusses how domain-specific knowledge feeds into model selection and hyperparameter tuning. Their work illustrates how augmenting domain fundamentals with data-driven approaches can improve the interpretability of predictions. In the case of this study, by leveraging multiple data sources, such as aggregating price trends with insights into supply chain disruptions, one can not only greatly improve forecast accuracy but yield actionable, informed insights for decision-making.

3.3 Summary and Conclusion of Literature

The literature on commodity price forecasting shows a wide variety of methodologies, each with distinct strengths and weaknesses. Machine Learning models like LSTMs and XGBoost offer state-of-the-art performance for capturing complex temporal relations. However, less complex models like LASSO regression still provide good performance and high interpretability, while requiring less computational power.

Therefore, in the case of price predictions for Nepalese commodities, data availability will play a crucial role in evaluating the performance of the models, as well as effectively leveraging their trade-offs.

4 Methodology

4.1 Data Collection

4.1.1 Original Dataset Description

The studies dataset was obtained from Kaggle's Agriculture Vegetables and Fruits Time Series Prices dataset [14]. It consists of daily price records for different fruits and vegetables collected across Nepalese markets from 1. January 2013 to 31. December 2021. It is comprised of seven columns including SN (serial number), Commodity, Unit (e.g. kg), Date, Minimum, Maximum and Average.

For this implementation, the analysis and predictions were limited to the commodity “Potato Red”, an important vegetable in Nepal’s agricultural sector [15]. The data was filtered based on the commodity column to only include records for “Potato Red”.

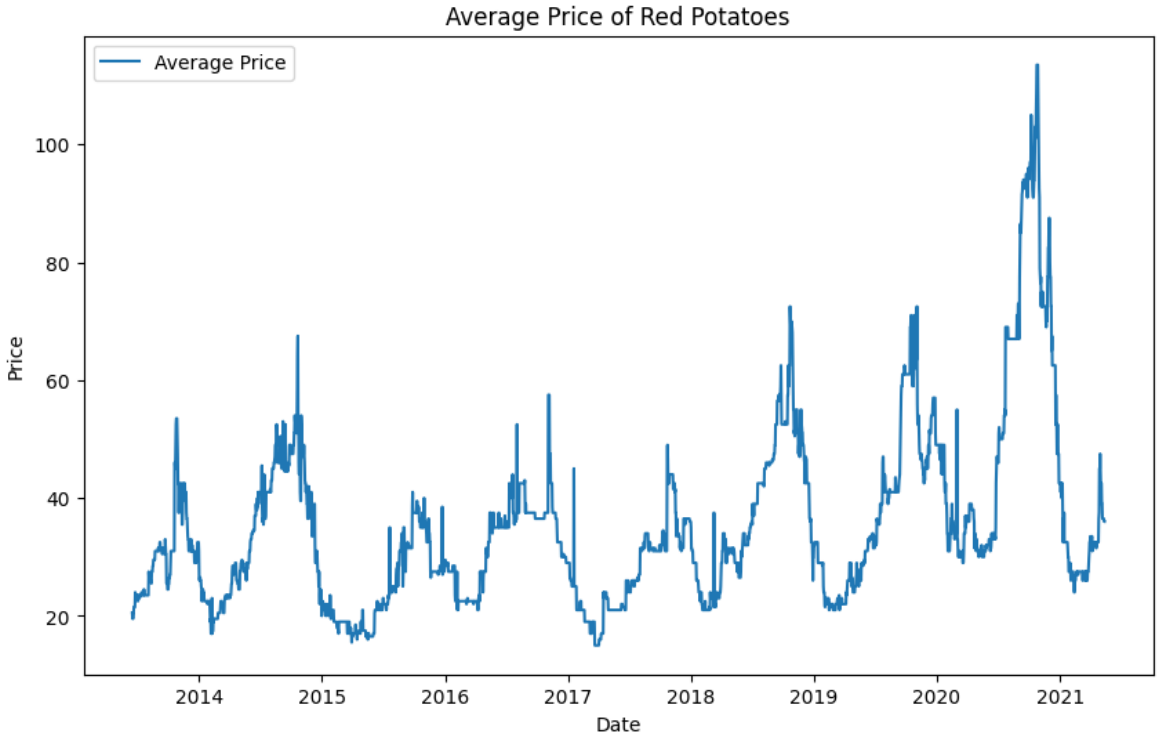

Figure 4.1: Average price of red potatoes from mid-2013 to mid-2021.

From figure 1, one can observe the price changes of red potatoes over the years. A clear seasonal pattern can be observed with spikes around the middle of the year.

A significant spike can be seen between years 2020 to 2021. This can be explained through the beginning of the COVID-19 pandemic, likely driving prices upwards.

After filtering, two key columns were retained:

- Date: The date column provides the temporal dimension of the data, with daily records that enable detailed analysis for price variations over time. In the case of the red potatoes, the time frame ranged from 16.06.2013 to 13.05.2021.
- Average: This column represents the average daily price of red potatoes on a given date and is the target variable for the prediction. The choice to focus on the average price, instead of the minimum or maximum, was made to provide a generalized measure of market conditions, avoiding unnecessary complexity from modeling price ranges.

Other columns like SN, Unit, Minimum and Maximum were dropped as they provide no additional information relevant to the time series.

In order to capture information about the seasonal and periodic trends available to the model, features such as Year, Quarter, Month, Week and Day were derived from the Date feature. For instance:

- **Year** can reflect long-term trends, like gradual price rises or falls brought on by macroeconomic shifts.
- **Quarter** and **Month** may denote seasonal cycles associated with harvest periods or demand variation.
- **Week** and **Day** can help detect short-term trends and periodic patterns like weekly market activities.

This data set formed the basis for the predictive modeling task with its daily granularity providing insights into short-term fluctuations as well as seasonal trends. By focusing on the Date and Average columns, the dataset provides the necessary structure to analyze the relationship between the daily price of "Potato Red" and external factors such as economic indicators, weather conditions, and holidays.

4.1.2 External Feature Collection

The feature collection process integrated a variety of external data to construct a solid foundation of information about external factors influencing vegetable prices in Nepal. It focused on economic and agricultural indicators, weather data and information about national holidays, with the aim to capture key variables critical to price dynamics, while maintaining consistent temporal resolution.

Economic data was obtained via the World Bank API, covering indicators such as Consumer Price Index (CPI), exchange rates, GDP growth and agricultural productivity metrics. They were selected based on their relevance to economic and agricultural conditions that affect pricing. Due to the data only being available on a yearly basis, interpolation was used to reduce the data to monthly intervals. While this approach introduces the risk of distorting the data it was deemed necessary to align with the temporal resolution of the other data.

Additionally, FAOSTAT data on agricultural metrics regarding red potatoes specifically, was sourced to enrich the dataset with information on the commodity itself. This data contained annual statistics on import and export volumes, production and yield quantities and harvested area. Once again, as with the economic data these metrics also had to be resampled to a monthly resolution using interpolation.

Using the Open-Meteo API and due to its impact on agricultural yield and possible availability of red potatoes weather data was collected for the city of Kathmandu, which reflects the agricultural climate of central Nepal. The data, which is available on a daily basis was aggregated to monthly values, ensuring consistency.

Finally, public holiday data was joined into month-wise counts to account for possible market disruptions or demand shifts during festival seasons.

The monthly resolution was chosen as a compromise between the coarser intervals of economic and agricultural data and the finer daily granularity of the target price variable, as well as the weather data. Even though interpolation is likely to introduce some risk of distortion or loss of temporal detail, the monthly resolution ensures consistency across all data and provides a practical foundation for integrating diverse sources of information, while also retaining a reasonable level of temporal detail.

The resulting dataset combines macroeconomic, agricultural, environmental, and social factors. It offers a foundation for predictive modeling while acknowledging trade-offs in aligning data from varied sources.

4.1.3 Data Leakage Prevention

In time-series forecasting and many other ML tasks, avoiding data leakage is crucial to ensure the validity and reliability of the model's performance. Data leakage occurs when information that shouldn't be available at the time of prediction gets leaked into the training data, resulting in overly optimistic predictions that cannot be achieved under realistic conditions [16]. This becomes particularly significant for external features as they get integrated into the modeling pipeline, since they also have to be constraint to the respectable timeframe.

In this study, the introduction of external features, such as economic and weather data, introduced unique challenges in that regard. Since the goal is to forecast into the future, subsequent test features, couldn't be sourced from the available dataset, as that would introduce information about the data, which otherwise wouldn't be available yet in a real scenario. As a result, the prediction pipeline would have to include predicting the external features themselves, before using them for the final model training. This approach is essential in ensuring that data leakage can't occur in the modeling pipeline.

4.1.4 Feature Prediction Framework

The ARIMA (AutoRegressive Integrated Moving Average) model was used for the feature predictions. ARIMA is a common, statistical model that combines autoregressive (AR) terms, differencing to achieve stationarity (I), and moving average (MA) terms [17]. Each feature was treated as an independent time series, and stationarity tests were performed prior to modeling. Where required, differencing was applied to make the series stationary.

However, this method encountered substantial constraints due to the distortion introduced by interpolation and aggregation beforehand. ARIMA-predicted values for many of these features deviated substantially from the true values, leading to high error margins and unreliable predictions.

4.1.5 Feature Exclusion

Due to the poor predictive accuracy of the economic and agricultural features, generally deviating by upwards of 600%, dropping them for the final modeling pipeline was necessary. While these variables are theoretically important for explaining price dynamics, their poor prediction accuracy would have introduced a lot of noise into the model. Additionally, their original temporal resolution rendered them impractical for real-world forecasting at a finer frequency level.

With the exclusion of the economic and agricultural data, the modeling shifted from a monthly to a daily approach. The decision was driven by the daily availability of all the remaining features, e.g. price, weather and holidays. By working with the original frequency of the data the pipeline avoided unnecessary resampling and aggregation, ensuring consistency across all features.

Furthermore, due to general weather prediction being very accurate to real-world events, it can be assumed that the data present in the test set, would be available at the time of performing predictions. Therefore, although the ARIMA-predicted values were fairly accurate with a deviation of around 15%, they were deemed unnecessary and thus the original weather data was kept for the modeling process. This approach ensured simplicity and reliability without compromising the model's real-world applicability.

4.2 Initial Modeling

Building on the refined dataset and the decision to exclude economic features, the modeling phase focused on implementing LASSO regression, XGBoost, and an LSTM neural network. These models were chosen to explore both traditional machine learning techniques and advanced deep learning methods for predicting the daily price of red potatoes. This section outlines the additional preprocessing steps and the implementation of each model, the results of hyperparameter tuning and the insights gathered from the feature importance analyses.

4.2.1 Preprocessing

For the preprocessing a MinMaxScaler was used to scale both the target variable as well as all the features to a value range between 0 and 1. Both LASSO regression

and LSTMs are highly dependent on scaled data [18]. The L1 regularization in LASSO penalizes the coefficients relative to their absolute values, not scaling them would therefore result in features with larger value ranges disproportionately influencing the regularization process, potentially skewing feature selection. Regarding LSTMS, scaling ensures stable gradient descent optimization during training. XGBoost natively handles scaling due to its decision tree-based architecture, so scaling it manually wouldn't be required specifically, but was still performed for consistency.

For LSTMS, the data was also reshaped into 7-day sequences, enabling the model to learn from weekly dependencies over the past weeks. This sequencing step is critical for LSTMs, as it allows the model to process time-series data in chunks, leveraging its memory mechanism to capture short- and long-term dependencies. Categorical data, such as holidays, was encoded as numerical values to ensure seamless integration with the machine learning pipeline.

4.2.2 LASSO Regression

LASSO regression was chosen as a baseline model because of its simplicity and its ability to perform feature selection through regularization.

1. **Initial Implementation:** The model was trained on the MinMax scaled features and achieved a mean absolute error (MAE) of 17.2 on the test set.

2. **Feature Importance:** Among the features, Quarter (17.19) and Year (10.20) emerged as the most influential, suggesting that long-term temporal trends were key drivers of price dynamics. Short-term temporal features (Day, Month, Week) and weather variables were excluded completely, with coefficients at 0. This indicates that LASSO struggled to capture the impact of non-linear or shorter-term patterns.

3. **Hyperparameter Tuning:** Hyperparameter tuning was conducted using a grid search with cross-validation, optimizing parameters such as *alpha*, *max_iter*, and *tol*. The best configuration included *alpha* = 0.24, *max_iter* = 1000, and *tol* = 0.0001. However, the MAE increased slightly to 17.77 after tuning, suggesting that the initial model had overfitted to the training data,

and tuning improved generalization at the expense of marginally worse test performance.

4.2.3 XGBoost

XGBoost was used to capture non-linear relationships between the features. Its capacity to model complex interactions and capture non-linear dependencies made it a strong candidate for capturing the dependencies between weather, time and price features.

1. **Initial Implementation:** The initial XGBoost model, trained with default parameters, achieved an MAE of 17.17, comparable to LASSO regression.

2. **Feature Importance:** Computed feature importance scores proposed both temporal and weather-related features as key contributors. The most important features were Year (score of 2266), Week (2103) followed by other variables such as temperature_2m_min (1061) and Month (1019). Where LASSO failed to capture the effects of weather features XGBoost was able to effectively capture these relationships, highlighting its ability to model non-linear relationships.

3. **Hyperparameter Tuning:** Performed by RandomizedSearchCV to optimize parameters such as *learning_rate*, *max_depth*, *min_child_weight*, *gamma*, *subsample*, *colsample_bytree*, and n_estimators. The best configuration included a *learning_rate* of 0.0034, *max_depth* of 5, and *n_estimators* of 400. Post-tuning, the MAE increased to 18.46, reflecting that the initial model overfitted to the training data, and tuning improved generalization at the expense of test performance.

4.2.4 LSTM

Recognizing the sequential nature of the data, an LSTM neural network was implemented to leverage its ability to capture temporal dependencies.

1. **Initial Implementation:** The LSTM model consisted of two bidirectional LSTM layers with dropout regularization to prevent overfitting.

2. **Learning Rate Optimization:** To determine the optimal learning rate, a learning rate scheduler was employed. The scheduler adjusted the learning rate dynamically during training using the formula $10^{-8} * 10^{\frac{epoch}{20}}$. This approach progressively increased the learning rate over 100 epochs, and the learning rate corresponding to the lowest loss was identified as optimal.

3. **Training**: The Adam optimizer, a gradient-based optimization that effectively adjusts the learning rates [19], configured with the dynamically adjusted learning rate, was then used to train the model with a final optimal learning rate of 0.0005.

4. **Results:** The model was compiled using the mean_squared_error loss function and trained for 300 epochs. After training, the LSTM achieved a test set MAE of 16.62. While the LSTM model effectively captured sequential patterns, its performance was comparable to LASSO regression and XGBoost, highlighting the challenge of improving predictions beyond simpler models for this dataset.

4.2.5 Comparative Analysis and Insights

The initial modeling phase demonstrated the varying strengths and limitations of the tested approaches.

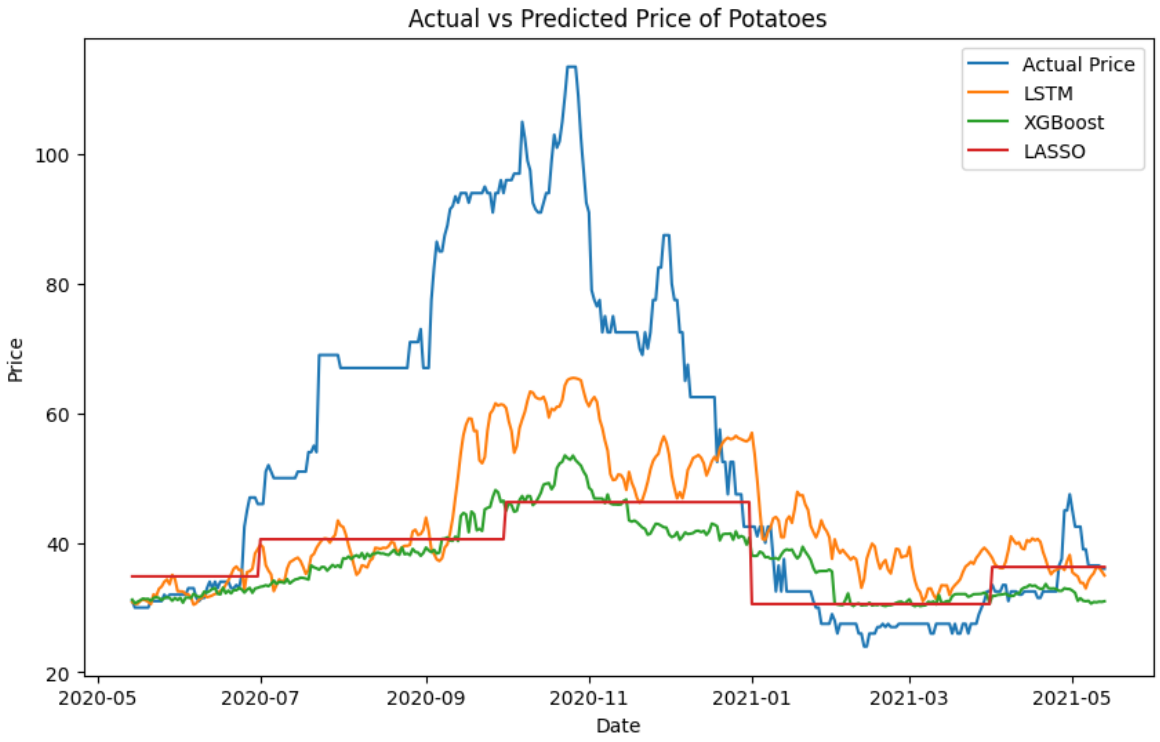

Figure 4.2: Price of potato red from the original data compared to predicted values from initial modeling.

LASSO regression, while interpretable, was noticeably limited by its linearity and therefore couldn't capture non-linear relationships, nor short-term dynamics. XGBoost was more flexible, capturing non-linear interactions and incorporating the importance of weather features. However, its performance was limited by overfitting in the initial implementation, barring the question of how well the tuned model will perform on varied data. The LSTM, while theoretically well suited for sequential data, still performed very similarly to less complex models such as LASSO and XGBoost.

The feature importance analyses emphasize the significance of temporal features, and in particular Year, Quarter, and Week contributions for capturing price trends. The weather variables, while not being considered by LASSO, demonstrated their importance for non-linear interactions through XGBoost. These results highlight the importance of thoughtful feature engineering and model selection in accordance with the nature of the data.

4.3 Simulation Study

To evaluate the robustness of the models developed in the initial modeling phase, a simulation study will be performed using synthetic datasets based on the original data. The study aims to evaluate the various model's performance under different hypothetical scenarios, representative of real-world complexions, such as structural breaks, outlier, or changes in seasonality. The simulation pipeline was developed to systematically manipulate key elements of the original data while maintaining its general temporal structure.

4.3.1 Data Preparation

The basis for the data simulation was the decomposed components of the original dataset: Trend, seasonality and residuals. They were derived through decomposition. Each decomposed component was processed to handle missing values via forward and backward filling, ensuring complete datasets for generating synthetic variations.

Nine synthetic datasets were created by systematically altering the decomposed components to simulate various real-world phenomena:

1. **Amplified Trend:** Increased the magnitude of the trend component to simulate stronger long-term growth or decline.
2. **Increased Noise:** Added random Gaussian noise to emulate more volatile data.
3. **Reduced Seasonality:** Decreased the amplitude of seasonal components to mimic weakening periodic effects.
4. **Nonlinear Trend:** Introduced a quadratic trend to simulate non-linear growth patterns.
5. **Added Anomalies:** Inserted random spikes into the data to represent outliers or abrupt changes.
6. **Structural Break:** Imposed an abrupt shift in the trend mid-series to simulate a sudden regime change.

7. **Shortened Seasonality:** Adjusted the seasonal frequency to simulate shorter cycles, such as those caused by external disruptions.

8. **Non-Stationary Variance:** Gradually increased the residual variance over time to reflect changing levels of unpredictability.

9. **Regime Switching:** Modeled two distinct regimes with contrasting trends to test adaptability under varying conditions.

Each synthetic dataset was constructed by combining the modified components with the remaining unaltered elements, ensuring a realistic representation of hypothetical scenarios.

4.3.2 Model Evaluation Framework

For each synthetic dataset, the models saved during the initial development phase i.e., the LASSO regression, XGBoost, and LSTM were used to predict. The evaluation pipeline was as follows:

- **Data Scaling:** Target and Features were scaled with MinMaxScaler to maintain consistency with the requirements of the models.
- **Train-Test Split:** The final year (365 days) of data in each synthetic dataset was reserved as the test set, with the preceding data used for training.
- **Prediction and Metrics:** Models were evaluated using the Mean Absolute Error as the metric to determine how well the predicted values aligned with the true values in the synthetic test set.

4.4 Evaluation of Results

The models' performance was assessed using the Mean Absolute Error Across the original and synthetic datasets. The following table provides a summary of the results.

Dataset	Lasso (MAE)	XGBoost (MAE)	LSTM (MAE)
Original	17.77	18.46	16.62
Original Synthetic	23.46	24.08	17.78
Amplified Trend	28.23	29.42	19.58
Increased Noise	23.33	23.04	17.54
Reduced Seasonality	23.01	23.73	18.82
Added Anomalies	23.57	23.80	16.72
Structural Break	27.24	28.58	21.03
Shortened Seasonality	20.10	19.62	20.07
Regime Switching	13.30	11.50	22.69

Table 1: Simulation Study Modeling Results

As evaluated in the initial modeling step, the LSTM emerged as the most performant model with the lowest MAE of 16.62, compared to XGBoost with 17.17 and LASSO with 18.46. In most of the synthetic scenarios, LSTM remained the best performer, being able to capture anomalies, noise and amplified trends with generally better results. Nonetheless, XGBoost showed relative strengths in being able to produce reliable predictions in scenarios that involved sudden changes in the data as well. LASSO regression, in comparison, failed to adapt to relationships within the data and produced consistently higher error margins and mediocre predictions, across complex scenarios.

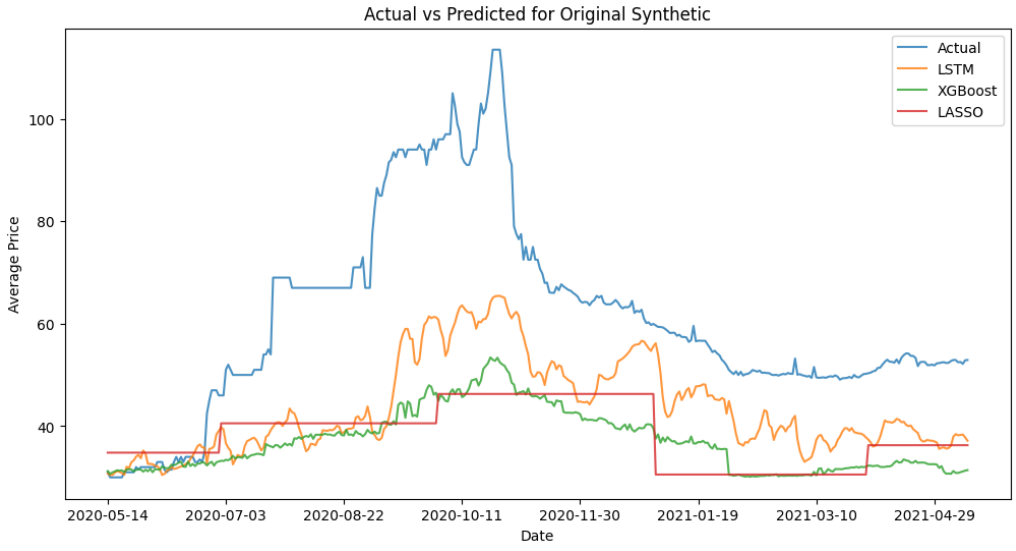

Figure 4.3: Price of red potatoes in the original synthetic dataset compared to predictions.

Amplified Trend: The ability to adapt to a stronger long-term trend was best again demonstrated by the LSTM, outperforming the other models with an MAE of 19.58. Observing the graphs reveals that the LSTM's predictions closely follow the trend, while XGBoost and LASSO consistently underestimated peaks and did not sufficiently adjust to the trend's magnitude. This reflects the LSTM's advantage in modeling cumulative dependencies over time.

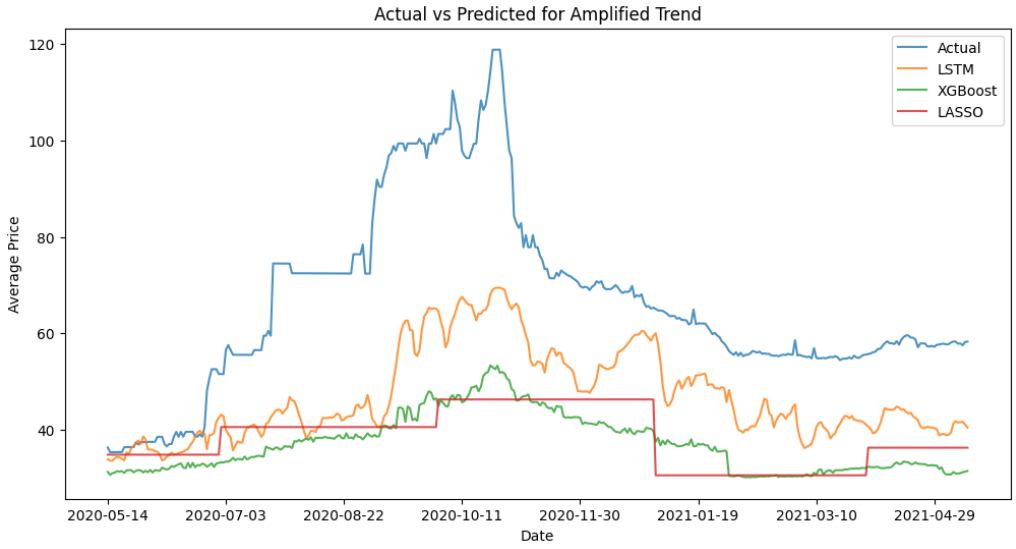

Figure 4.4: Price of red potatoes in the amplified trend dataset compared to predictions.

Increased Noise: Increased noise added random variability to the dataset. With a MAE of 17.43, the LSTM predicted the most accurate and demonstrated its capability to capture the increased variability introduced into the data while preserving the underlying trend. Compared to that, predictions from the other two models smoothed out the predictions too much, disregarding the introduced noise, resulting in an increased MAE of 23.33 (LASSO) and 23.04 (XGBoost).

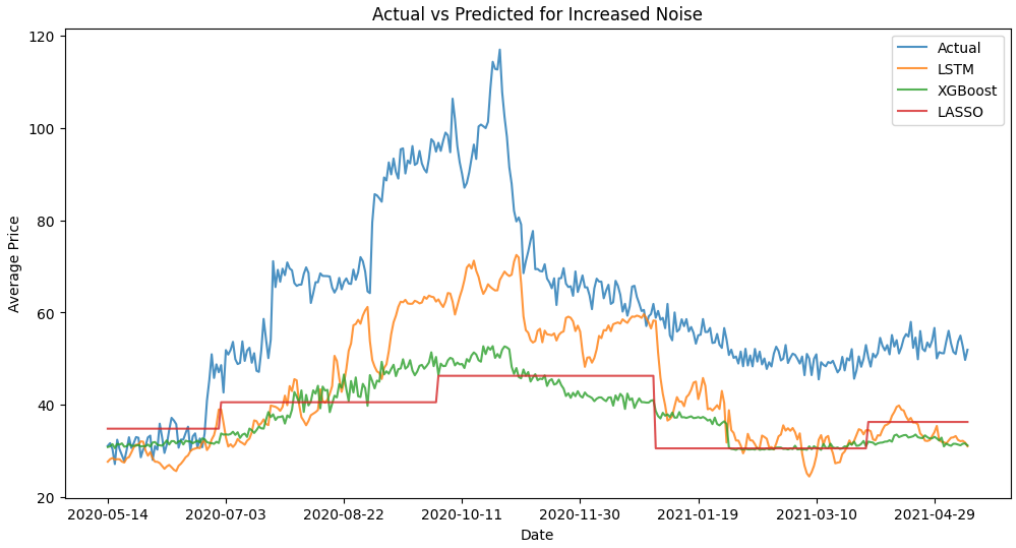

Figure 4.5: Price of red potatoes in the increased noise dataset compared to predictions.

Reduced Seasonality: With reduced seasonality the LSTM emerged as the best-performing model with a MAE of 18.82. XGBoost's and LASSO's worse performance can be derived from their limited ability to capture seasonal impacts within the data, leading to consistently underestimated predicted values. In contrast, the LSTM is able to adjust to the higher impacts, although still introducing variations that are not present in the original data.

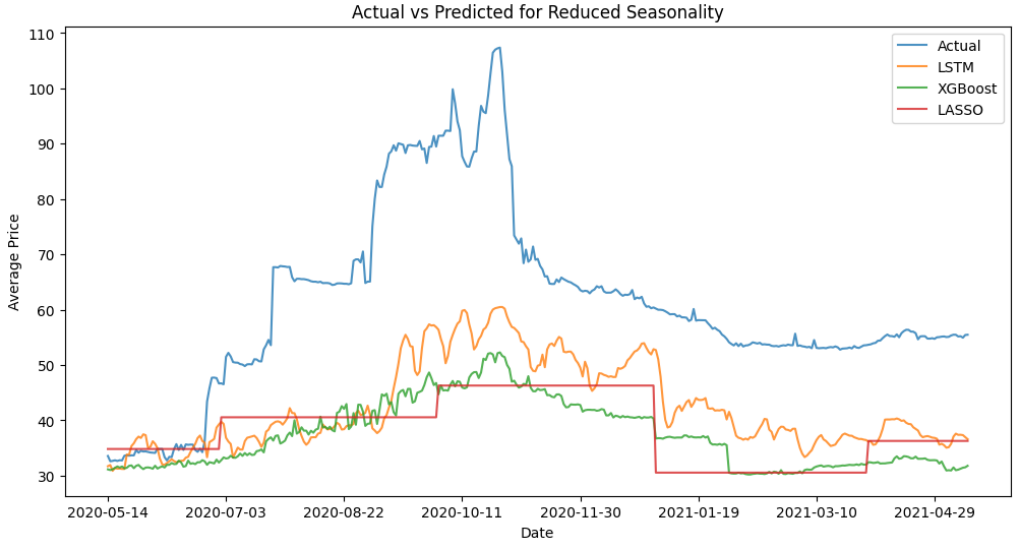

Figure 4.6: Price of red potatoes in the reduced seasonality dataset compared to predictions.

Added Anomalies: The anomalies dataset introduced sudden increases in the data to mimic shocks within the market, representing one-off events that alter the price.

Here the LSTM produced the lowest MAE with 16.72, highlighting its ability to factor in anomalies into its predictions. Visual analysis also confirms its ability to capture spikes more effectively than the other models. XGBoost and LASSO produced smoother predictions, without capturing anomalies and treating them as noise instead. This implies that the LSTM's sequential architecture allowed it to better recognize and integrate anomalies into its forecast.

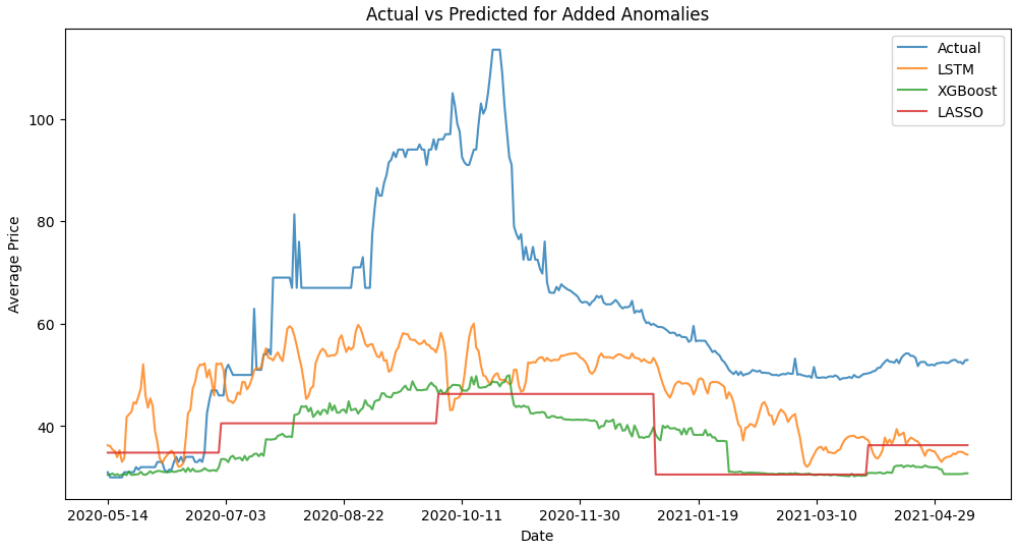

Figure 4.7: Price of red potatoes in the added anomalies dataset compared to predictions.

Structural Break: The introduction of sudden shifts in trend mid-series introduced unique challenges for the models. Once again, the LSTM performed the best with the lowest MAE of 21.03 compared to XGBoost (28.58) and LASSO (27.24), adapting to the trend more effectively than the other two. Visual inspection revealed that LSTM predictions adjusted gradually to the post-break trend, whereas XGBoost and LASSO failed to realign, resulting in persistent residuals. Nonetheless, the LSTM's delayed adaptation, suggests difficulties in predicting abrupt changes.

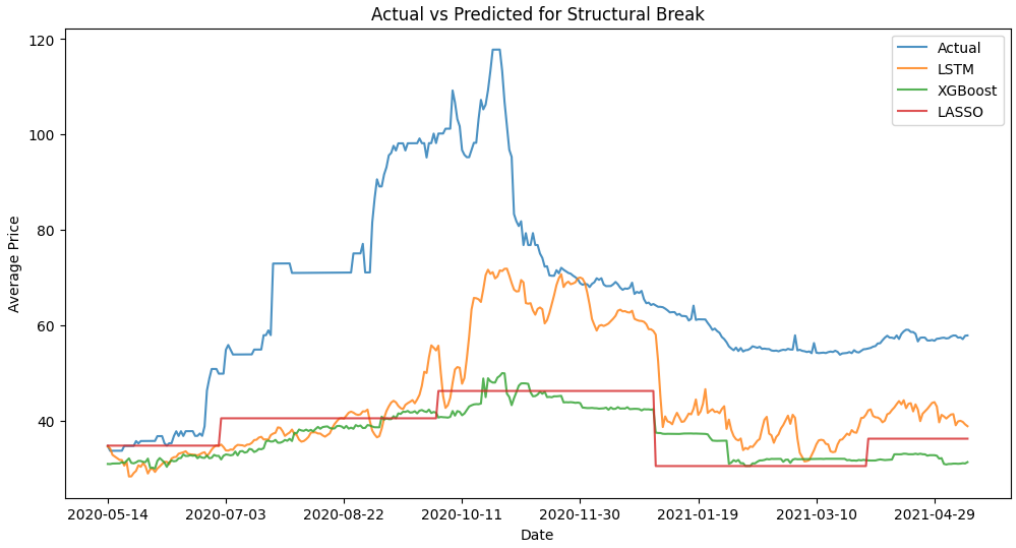

Figure 4.8: Price of red potatoes in the structural break dataset compared to predictions.

Shortened Seasonality: Shortened seasonality posed higher challenges for all models, with XGBoost achieving the lowest MAE of 19.62 and LASSO the highest with 20.07. While this could indicate better performance for XGBoost a visual inspection reveals that the LSTM generally fitted better to the compressed seasonal cycles, whereas XGBoost failed to do so. The higher MAE for the LSTM in this scenario reflects some misalignment with the shorter cycles, though its prediction overall reflected the actual data more closely.

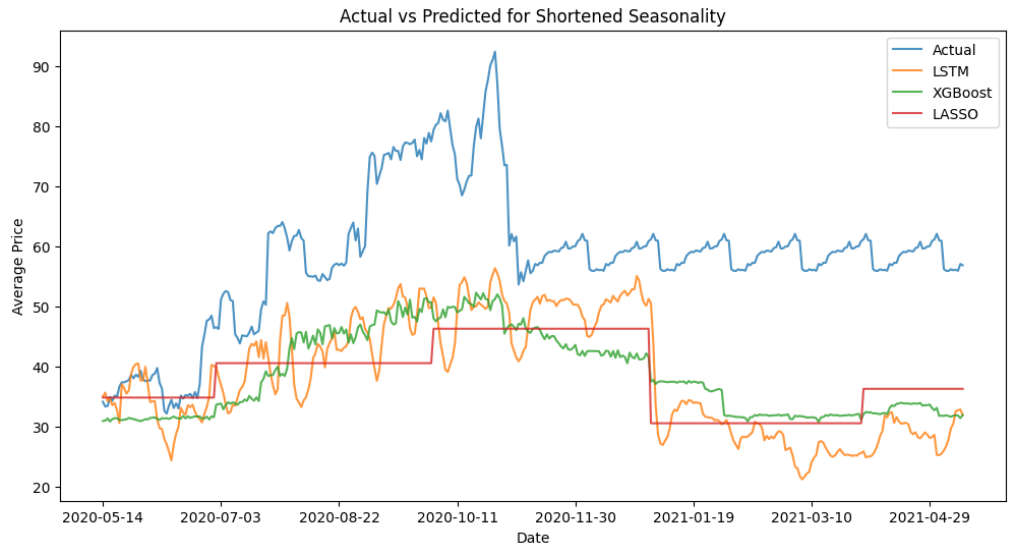

Figure 4.9: Price of red potatoes in the shortened seasonality dataset compared to predictions.

Regime Switching: Finally, for the regime switch XGBoost performed the best with an MAE of 11.50, followed by LASSO with 13.3 and the LSTM with 22.69. The visual analysis suggests that XGBoost was more effective in adapting to the transition than the LSTM. It is to be called into question whether these predictions were based on a better analysis and understanding of the data's structure or simply by chance. In this case, the LSTM predictions smoothed over the transition, failing to differentiate between the regimes. This highlights the limitation of this implementation to handle discontinuous patterns.

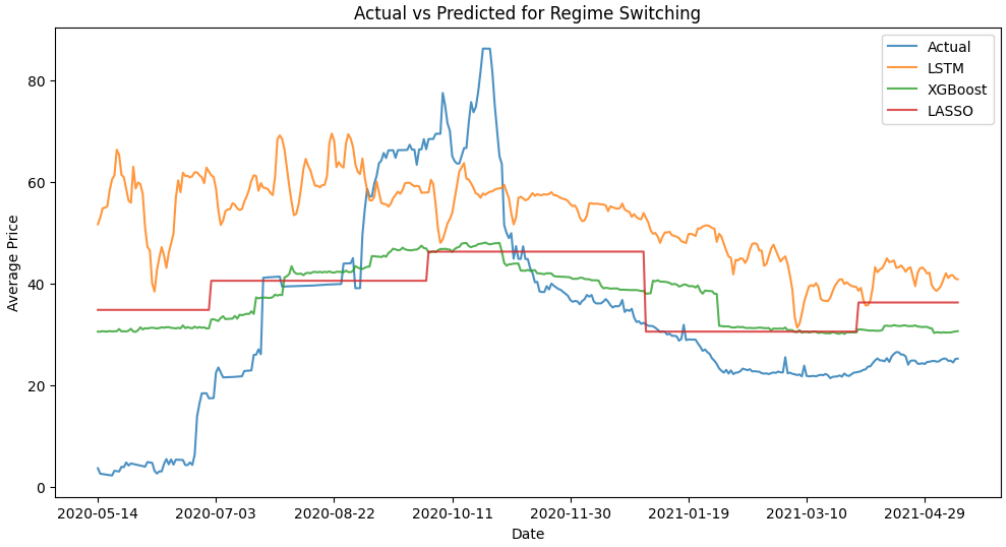

Figure 4.10: Price of red potatoes in the regime switching dataset compared to predictions.

4.5 Interpretation of Results

The simulation study demonstrates the strengths and limitations of LASSO regression, XGBoost and LSTMs in time series forecasting under diverse synthetic scenarios. The results highlight the reliance of the models on the data characteristics, the importance of contextual information and the challenges of accurately modeling complex dynamics with limited data.

4.5.1 Key Insights and Model Behavior

In scenarios requiring sequential learning, such as amplified trends, noise and anomalies, the LSTM demonstrated consistent strengths in delivering reliable

predictions. Its ability to leverage temporal dependencies allowed it to capture patterns much better than any of the other models. Despite these advantages, the reliance on sequential data introduced challenges in scenarios with abrupt shifts, including regime switches and structural breaks, where it struggled to adapt and occasionally smoothed over important transitions.

XGBoost's performance was less consistent. Although it demonstrated lower MAE than the LSTM in some scenarios, it appeared to be more due to chance than its capacity to capture the structure of the data itself. The visual analysis also revealed its tendency to produce smoothened predictions, unable to capture important shifts and variations in the data. Its ability to adapt to abrupt changes was limited, further indicating that the apparent success of predictions in certain scenarios resulted out of chance and not a reflection of robust learning.

LASSO regression, as a simpler linear model, fell short in comparison to the other approaches in more complex scenarios. Its limitations in capturing non-linearity or responding to sudden changes positioned it as a simple baseline model rather than a solution of dynamic forecasting tasks.

4.5.2 Broader Challenges and Context

A major limitation across all models represented their failure to reproduce the higher price peaks observed in the original dataset, likely driven by the global COVID-19 crisis during that time frame. This peak, caused by extraordinary external factors such as supply chain disruptions or market instability, demonstrates the dependence of machine learning models on sufficient available data. Without external, contextual features to capture these conditions within the models, none of the models were able to account for and predict the resulting price peaks properly. This limitation highlights the importance of enriching the predictive features with all necessary contextual information, to improve real-world forecasting accuracy.

5 Conclusion and Outlook

This study focused on evaluating the performance of various machine learning models in time series forecasting for Nepalese commodity prices with a focus on the red potato. The research detailed the strengths and limitations of these models in both real-world and simulated scenarios.

The LSTM emerged as the best-performing model across various scenarios, due to its ability to analyze sequential patterns, predicting well under varying trends, noise and anomalies. However, it struggled to respond to abrupt changes, such as regime shifts, as it tended to smooth over significant transitions. XGBoost showed robust predictions but failed to capture complex patterns in many cases, not being able to capture noise and structural changes effectively. LASSO regression, while simple and interpretable, was unable to handle non-linear relationships and adapt to dynamic changes.

All models faced challenges in adapting to the high demand peak caused by external factors such as COVID-19. This emphasized the need to incorporate sufficient external data about economic and market disruptions, which can improve the forecasting accuracy.

Overall, while advanced models such as LSTMs and in part, XGBoost result in better performance within complex scenarios, simpler models like LASSO regression can still serve as an effective baseline in straightforward scenarios. Choosing the right model depends on the nature of the data, as well as the forecasting goals and the associated, available data.

The future of commodity price prediction will depend on a combination of extensive data sourcing based on domain-specific insights and advanced integration of different modeling techniques.

- **Incorporation of Further Contextual Data:** Future studies should explore ways to integrate further external data, such as global demand fluctuations, trade policies, and weather anomalies. Advanced feature engineering and data fusion techniques can help address issues of data resolution mismatches and temporal alignment, ensuring that models are enriched with critical contextual information.

- **Hybrid Modeling Approaches:** Combining the strengths of different models could lead to better outcomes. For instance, hybrid models that integrate LSTM's sequential learning capabilities with XGBoost's non-linear modeling or even traditional statistical models may address the limitations observed in this study, such as handling abrupt shifts or regime changes.

The potential of time series forecasting in agricultural and economic contexts remains huge. Integrating state-of-the-art analytics and modeling techniques with domain knowledge can not only lead to enhanced prediction accuracy but also offer actionable insights to address challenges such as food security and market volatility. This study serves as a foundation, demonstrating the challenges and opportunities of forecasting Nepalese commodity prices.

Bibliography

[1] Jonath Jose, *INTRODUCTION TO TIME SERIES ANALYSIS AND ITS APPLICATIONS*, 2022. [Online]. Available: https://www.researchgate.net/publication/362389180_INTRODUCTION_TO_TIME_SERIES_ANALYSIS_AND_ITS_APPLICATIONS

[2] M. Peixeiro, *Time series forecasting in Python*. Shelter Island NY: Manning Publications Co, 2022. [Online]. Available: https://learning.oreilly.com/library/view/-/9781617299889/?ar

[3] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning: With applications in R* (Springer texts in statistics 103). New York: Springer, 2013. Accessed: Jan. 7, 2025. [Online]. Available: https://static1.squarespace.com/static/5ff2adbe3fe4fe33db902812/t/6009dd9fa7bc363aa822d2c7/1611259312432/ISLR+Seventh+Printing.pdf

[4] E. Iturbide, J. Cerda, and M. Graff, "A Comparison between LARS and LASSO for Initialising the Time-Series Forecasting Auto-Regressive Equations," *Procedia Technology*, vol. 7, pp. 282–288, 2013. doi: 10.1016/j.protcy.2013.04.035. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2212017313000364

[5] C. Wade, *Hands-On Gradient Boosting with XGBoost and scikit-learn*. Birmingham: Packt Publications Co, 2020. [Online]. Available: https://learning.oreilly.com/library/view/hands-on-gradient-boosting/9781839218354/

[6] "Introduction to Boosted Trees — xgboost 2.1.3 documentation." Accessed: Jan. 7, 2025. [Online]. Available: https://xgboost.readthedocs.io/en/stable/tutorials/model.html

[7] T. Chen and C. Guestrin, "XGBoost," vol. 11, no. 34, pp. 785–794, 2016. doi: 10.1145/2939672.2939785. [Online]. Available: https://arxiv.org/pdf/1603.02754

[8] G. van Houdt, C. Mosquera, and G. Nápoles, "A review on the long short-term memory model," *Artif Intell Rev*, vol. 53, no. 8, pp. 5929–5955, 2020. doi: 10.1007/s10462-020-09838-1. [Online]. Available: https://www.researchgate.net/publication/340493274_A_Review_on_the_Long_Short-Term_Memory_Model

[9] R. C. Staudemeyer and E. R. Morris, "Understanding LSTM -- a tutorial into Long Short-Term Memory Recurrent Neural Networks," Sep. 2019. [Online]. Available: http://arxiv.org/pdf/1909.09586

[10] P. Tanwar, R. Shah, J. Shah, and U. Lokhande, "Cotton Price Prediction and Cotton Disease Detection Using Machine Learning," *Intelligent Data Communication Technologies and Internet of Things*, vol. 101, pp. 115–128, 2022. doi: 10.1007/978-981-16-7610-9_9. [Online]. Available: https://link.springer.com/chapter/10.1007/978-981-16-7610-9_9?utm_source=chatgpt.com

[11] Y. H. Gu, D. Jin, H. Yin, R. Zheng, X. Piao, and S. J. Yoo, "Forecasting Agricultural Commodity Prices Using Dual Input Attention LSTM," *Agriculture*, vol. 12, no. 2, p. 256, 2022, doi: 10.3390/agriculture12020256.

[12] J.-C. Huang *et al.,* "Predictive modeling of blood pressure during hemodialysis: a comparison of linear model, random forest, support vector regression, XGBoost, LASSO regression and ensemble method," *Computer Methods and Programs in Biomedicine*, early access. doi: 10.1016/j.cmpb.2020.105536.

[13] X. Luo, D. Zhang, and X. Zhu, "Deep learning based forecasting of photovoltaic power generation by incorporating domain knowledge," Energy, vol. 225, p. 120240, 2021. doi: 10.1016/j.energy.2021.120240. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360544221004898

[14] R. K. Rijal. "Time Series Price Vegetables and Fruits." Accessed: Jan. 7, 2025. [Online]. Available: https://www.kaggle.com/datasets/ramkrijal/agriculture-vegetables-fruits-time-series-prices/data

[15] "1631268941_Latest Corrected_12th proceeding-57-71," [Online]. Available: https://www.horticulturenepal.org/uploads/main_attachment/1631268941_Latest%20Corrected_12th%20proceeding-57-71.pdf

[16] M. A. Bouke, S. A. Zaid, and A. Abdullah, "Implications of Data Leakage in Machine Learning Preprocessing: A Multi-Domain Investigation," 2024. doi: 10.21203/rs.3.rs-4579465/v1. [Online]. Available: https://www.researchsquare.com/article/rs-4579465/v1

[17] R. Adhikari, "An Introductory Study on Time Series Modeling and Forecasting [Online]. Available: https://arxiv.org/pdf/1302.6613

[18] J. Brownlee, "How to Scale Data for Long Short-Term Memory Networks in Python," *Machine Learning Mastery*, 06 Jul., 2017. Accessed: Jan. 7, 2025. [Online]. Available: https://machinelearningmastery.com/how-to-scale-data-for-long-short-term-memory-networks-in-python/

[19] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," Dec. 2014. [Online]. Available: http://arxiv.org/pdf/1412.6980