**2.2 Time Series Forecasting**

Neglecting to understand and account for future price trends can be detrimental for 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 to the future, by analyzing past time series data (TSD) and predicting future values based on it.

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

**2.2.1 Time Series Components**

TSD is typically composed of the three primary components: trend, seasonality and the 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 (Jose, Page 4). 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 (Jose, Page 5). These patterns indicate the 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 (Jose, Page 5). Examples can be unexpected natural disaster inhibiting crop growth and shortening supply.

TSD can be broken down into its components through decomposition. This process is essential in better understanding and analyzing the underlying properties of a time series.

**2.2.2 Stationarity and Its Importance**

Stationarity is a fundamental concept in time series analysis and presumed by many models, such as the autoregressive ones. 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.

This presumption is necessary to ensure that selected model parameters remain consistent with the given time series. Changing properties would otherwise affect the models 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 modelling.

Differencing is the simplest method of achieving stationarity. It involves stabilizing the mean, by subtracting the previous value of a timestamp from the present value , resulting in the differenced value .

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 TSD with a seasonal component, seasonal differencing can be applied. Here, instead of subtracting neighboring values from another, the differencing is applied on values 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. Conversely, if the root equals 1, this indicates the presence of a unit root, making the series non-stationary, making another round of differencing necessary. When applying the method in code, a p-value of less than 0.05 rejects the null hypothesis and indicates stationarity.

**2.4 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 (Wade 2020, Chapter 5, para. “Design features”). 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.

**2.4.1 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:

Where is the loss function, representing the difference between the true value and the predicted value and is the regularization term to penalize the model complexity. Further, represent the model parameters and the the number of trees in the model.

The regularization term is given as:

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

**2.4.2 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 () and second-order Hessian () of the loss with respect to the model’s prediction at iteration :

where:

* is the first-order gradient, indicating the direction of the loss,
* 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.

**2.4.3 Leaf Weight Optimization**

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

where is the set of instances assigned to leaf .

The resulting quality score for a tree is derived as:

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.

**2.4.4 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 , which provides a way to obtain a good generalization of the model.

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

**2.5 Lasso Regression**

The Least Absolute Shrinkage and Selection Operator, more commonly known as Lasso regression, is a regularization technique used in statistical modelling and machine learning to enhance the prediction accuracy and interpretability of models. It achieves this by imposing a constraint on the sum of the absolute values of the models, effectively driving some of them towards near zero. This characteristic renders it particularly fit for feature selection, as well as creating parsimonious models.

Traditionally, linear regression determines the best-fitted line by minimizing the sum of the squared residuals. Lasso regression alters this approach by adding a penalty term λ that is proportional to the sum of the absolute values of the coefficients. The loss function can be defined as follows:

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 particularly demonstrates its usefulness in high dimensional datasets, where selecting variable automatically is crucial in producing sparse models, stripped of 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 avoid overfitting by balancing the model’s simplicity and its predictive performance on unseen data.

**2.5.1 The Lasso for Time Series**

These functionalities of Lasso regression prove especially useful in time series forecasting, as high dimensionality is commonly found when dealing with time series data. 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:

Where:

* indicate the value at time ,
* are lagged observations,
* are coefficient estimates.

Lasso regression modifies the AR model by introducing the penalty on the absolute values of the coefficients, ensuring some coefficients are driven to 0. The resulting optimization problem of Lasso regression for a time series can be formulated as:

Here:

* is the number of observations,
* is the number of lagged covariates,
* is a hyperparameter controlling the regularization strength.

The penalty term 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.

**2.6 Long Short-Term Memory Networks (LSTM)**

Long Short-Term Memory networks, short LSTMs, are a type of recurrent neural network (RNN) that are designed to 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.

**2.6.1 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 time step lags, so that stable gradients can be preserved over long periods of time.

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 is like a conveyor belt, carrying relevant information throughout the sequencing process without out many transformations. It gets updated through operations that are regulated by the input and forget gates.
* **Input gate:** The input gate determines how much new information enters the cell state. It's defined through the following equation:

where σ denotes the sigmoid function, is the weight matrix for the input gate, ​ is the previous output, is the current input, and is the bias.

* **Forget Gate:** The forget gate controls what information is removed from the cell state. It is formulated as:

The parameter represents the weight matrix for the forget gate.

* **Cell State Update**: Cell state is updated by:

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

* **Output Gate:** The output gate controls the output from the cell state to the next layer:

and the output of the LSTM unit can be defined by:

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

**2.6.2 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. 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.

**2.6.3 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 LSTMs architecture to be tailored to a particular task, which often require faster calculations without a significant trade-off in performance.

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