What is the impact of fuel index on other indexes in inflation forecasting in the UK?

Team Andromeda

Farah Yesilkaya, Moronfolu R Durodola, Priyanshu Jha, Vijay Jawali

MSc Data Science group project
06-32252

May 2023

School of Computer Science
College of Engineering and Physical Sciences
University of Birmingham

Acknowledgements

We would like to thank our Supervisor Hayat Adeyemo for his continual weekly support with our project. His support enabled us to answer the question and come up with an appropriate solution to answer the question and was very supportive.
We would also like to thank Professors Qamar Natsheh and Leonardo Stella for providing us with very insightful lectures which helped pave the way in producing a suitable question for our project. And lastly, we would like to thank Qamar by providing us with useful resources which made this project possible.

Table of Contents

1. Introduction

Inflation is defined as the rate of increase in prices and is reflected in a decrease of purchasing power [1]. Hence there has been a lot of apprehension and concern by the government, business owners, consumers, and financial institutions of the inflation level, due to the large impact it has on these different stakeholders. There are many factors known to influence inflation rates, and fuel prices in particular, is considered a significant factor; so, having a thorough understanding of how the prices of fuel impact inflation is extremely important, to accurately forecast inflation. As a result, it enables various stakeholders to make more informed decisions regarding the economy.

The relationship between fuel prices and inflation have been the subject of several research papers in different contexts. According to one study by Daniel F. Meyer [2], the volatility of fuel prices can impact the stability of the global economy due to importing and exporting fuel. It also states that an alarming rate of increase in fuel prices lead to inflationary pressures within the economy of South Africa. It was discovered in another study that 'the effects of these oil price shocks have been less damaging to economic growth and less inflationary' than it was decades ago [3]; this realisation is backed up by other research as well [4, 5]. These findings emphasise the significance of understanding the association between fuel prices and inflation, which also varies between countries. Moreover, due to the recent cost of living crisis, there has been increased speculation on the how much increased fuel prices affects the inflation rates, along with other inflation indexes [6]. Specifically, we look at Consumer Price Index, Food Price Index, Energy Price Index, Headline Consumer Price Index and Producer Price Index.

The objective of this report is to investigate the impact of fuel prices on inflation forecasting. Our focus is primarily on the UK as there is an advanced financial sector and the economy is vulnerable to economic shocks. Additionally, the ongoing crises has caused the inflation level to rise, reaching 9.2% as of February 2023 [7] (measured by CPIH). Specifically, we intend to develop a model that can accurately predict future fuel prices and use this to gauge the forthcoming inflation level. Throughout our investigation we use statistical analysis in addition to machine learning techniques such as time series analysis to create suitable models and evaluate these to determine the most appropriate.

The report is structured as follows: we begin by providing an overview of the relevant studies already accomplished about our chosen topic. We then proceed with some exploratory data analysis after sourcing and preparing the data to better understand the configuration of our data, this is followed by the rationale behind the forecasting models – including hyperparameter tuning and evaluation metrics – created in addition to their implementation. Next, we discuss the results of the models, including a model evaluation. Finally, we address the implications of our findings as well as limitations, which is followed by potential areas for future study.

1.1 Question development

Any research project requires a question development process which entails identifying a topic of interest and ensuring that the chosen question is precise and pertinent to the goals of the project. We commenced this process by researching potential topics individually and finding suitable datasets in this area where possible. We then gathered and ranked our options in three categories: opportunity to investigate, readily available and substantial dataset, and interest in topic. Whichever idea belonged in all three groups were ranked higher, so the question could then be refined around the topic. This process was not as linear as described; there were multiple revisions and discussions had. We settled on the question, "What is the impact of fuel index on other indexes in inflation forecasting in the UK?", because:

1. We were positive that the research outcomes would be meaningful and remain relevant.
2. We had considered existing literature surrounding the topic of fuel prices and inflation and felt we could contribute ore knowledge where there were gaps.
3. We could see that our solution had the potential to influence decision making at government level.

1.2 Related Work

Latest figures from the Office of national statistics demonstrated that gas prices in the UK rose by 129.4% in 12 months to February 2023 and was one of the main drivers of the increasing inflation rate, contributing to the current cost of living crisis [8]. A multitude of papers have previously investigated the ability to forecast

inflation and forecast fuel prices using machine learning [9,10]. Thus, there have been multiple studies investigating the correlation between rising fuel price and inflation, but no studies have investigated the effect of variations in fuel prices affecting the ability to forecast inflation. However, this paper is the first to investigate the impact of fuel prices on other inflation indexes during inflation forecasting. Since the mid-80s inflation volatility has decreased, but since, it has been harder to outperform random walk type-forests. For instance, it was demonstrated that averaging over the previous 12 months provides a more accurate forecast of the inflation for the next 12 months than a Phillips curve that looks backwards [9]. We use LSTMs in our research due to the benefits displayed in the literature for exploring inflation data, namely being flexible and able to capture the complexities that inflation data carries, namely its non-linear nature. For example, LSTMs outperformed autoregressive models (AR), random walk models (RW), Markov switching models and a fully connected neural network in forecasting monthly US CPI inflation [10]

2. Data

2.1 Retrieving data

2.1.1 Inflation Dataset

In this section, we describe the method used to retrieve data from the Global Inflation Database (GID) [11], the primary source of inflation data used in the current study. Finding the relevant dataset for the study under consideration was the first stage in the data retrieval procedure. The GID was chosen because of its thorough coverage of inflation data across nations and historical periods. The database's web interface allowed users to access the standardised inflation statistics that had been acquired from several sources such as national statistical agencies, central banks, and international organisations. Subsequently, we selected the specific variables and time periods of interest within the dataset. In the current research project, monthly inflation rates from 2003 to 2022 were retrieved. The variables of concern were the inflation rate as mentioned in [Table 1].

Table 1 Inflation index definitions

Index	Definition
Official Core Consumer Price Index	A measure of inflation that excludes the prices of energy and food, which are typically more volatile than other consumer goods, from the calculation[12]
Energy Price Index	A measure of inflation that tracks the changes in the prices of energy goods, including gasoline, electricity, and natural gas. [13]
Food Price Index	A measure of inflation that tracks the changes in the prices of food items, including meat, dairy, and grains. [14]
Headline Consumer Price Index	A measure of inflation that includes all goods and services purchased by consumers, including food and energy. [15]
Producer Price Index	A measure of inflation that tracks the changes in the prices of goods and services at the producer level before they reach the consumer market. [16]

2.1.2 Fuel Dataset

In our study, we used the Weekly Road Fuel Prices dataset [17] as supplementary information to our primary dataset to examine the influence of energy prices on the UK economy. The inclusion of this dataset was deemed essential as it provides average retail prices of road fuels on a weekly basis, which is critical for monitoring variations in UK road fuel prices over time. The Energy Prices Statistics Team is responsible for maintaining and publishing this dataset. To obtain this dataset, we accessed the gov.uk website, which is the primary source of the Weekly Road Fuel Prices dataset. The data was updated regularly, with the most recent update occurring for the week starting 13 February 2023. The data was downloaded in CSV format and subsequently imported into our software for further analysis.

2.2 Data Pre-Processing

Inflation data pre-processing

The inflation data is structured in the form of an Excel workbook containing multiple worksheets, each corresponding to a distinct inflation index. To facilitate ease of data retrieval, each worksheet is saved as a separate CSV file (Appendix 1.1) containing economic data, specifically Consumer Price Index (CPI),

Producer Price Index (PPI), Household Consumption Price Index (HCPI), Fuel Price Index (FCPI), and Electricity Price Index (ECPI) for various countries and filter the data to only include information for Great Britain (GBR) and are concatenated. The first four columns of the data containing metadata are then removed, and only the data starting from the fifth column is kept. The data frames are then concatenated into one, transposed, and the column names are updated to the first row of the data frame that reflects the month. The first row is then removed, and a new datetime index is created by converting the existing monthly index values. This new datetime index is set as the index for the data frame. The 'Energy Price Index' column is converted to a numeric format, and any missing values in the data frame are filled in using forward fill and backward fill methods (Appendix 1.2).

Data Normalisation

We employ various normalization methods to rescale and standardize data to enable comparison and interpretation. The four methods discussed are the Min-Max Approach, z-score method, Maximum Absolute Scaling, and Robust Scaling. The Min-Max Approach adjusts the range of the feature to [0,1] by subtracting the minimum value of the feature and dividing by the range. The z-score method standardizes the data to have a mean of 0 and a standard deviation of 1 by calculating the difference between each value and the mean and then dividing it by the standard deviation. In contrast, the Maximum Absolute Scaling method scales each observation by dividing it by the absolute maximum value of the feature. The Robust Scaling method is particularly useful when dealing with data containing outliers, as it scales each feature by subtracting the median and dividing by the interquartile range, which is a robust measure of dispersion.

To compare the effectiveness of each method, line plots [Figure 1] of the normalized data frames are generated (Appendix 1.3) using the Seaborn library, and they are plotted on four subplots in one figure. After inspecting the plots, the Robust Scaling method is selected since it provides the most consistent and stable results regardless of the data distribution.

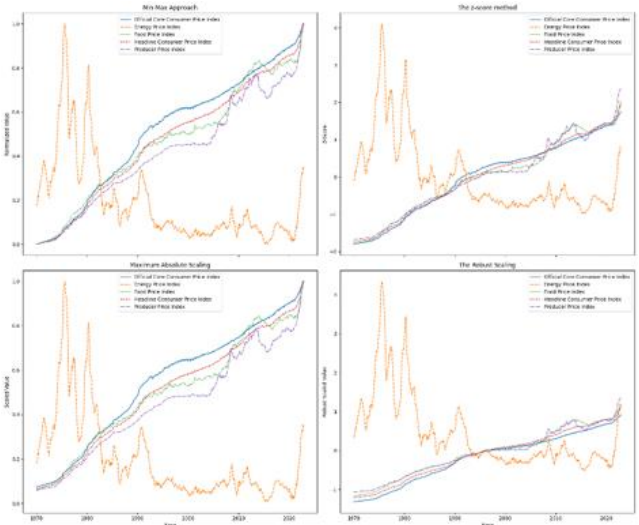


Figure 1 Scaling mechanisms applied to raw data: Min Max (top-left), z-score (top-right), Maximum Absolute (bottom-left) and Robust (bottom-right)

Auxiliary data pre-processing

To make fuel data suitable for analysis, it undergoes pre-processing steps such as cleaning, normalization, and transformation similar to inflation data (Appendix 1.4). Additionally, the fuel data is resampled from a weekly to a monthly coarse level, the original weekly data is resampled to a monthly level using the "month start" frequency ('MS') and then averaged within each monthly period. This is a common technique used to reduce the temporal resolution of the data and facilitate trend analysis over longer time periods.

3 Exploratory Data Analysis

The following section will briefly talk about the Exploratory data analysis (EDA), which is used to analyse and investigate data sets and summarize their main characteristics, often employing data visualization methods. It helps determine how best to manipulate data sources to get the answers, making it easier to discover patterns, spot anomalies, test a hypothesis, or check assumptions.

3.1 General EDA

Histograms - A useful visualisation technique for examining the distribution of data in a dataset is the histogram plot. When examining the distribution of continuous data like inflation indices, they are very helpful. A histogram graphic can show the frequency and distribution of inflation rates over time in the context of inflation indices.

Box Plots - Box plots are a useful tool for representing and contrasting data distributions visually. It helps in understanding the central tendency and variability of each inflation index as well as spotting outliers or extreme values in the case of inflation indices.

Pair Plot - Pair plots are used to determine the most distinct clusters or the best combination of features to describe a connection between two variables. It is essential when forecasting inflation because it provides a broad understanding of how indices are related to one another.

3.2 Timeseries Data Visualisations

Autocorrelation analysis

It is a crucial component of time series forecasting's exploratory data analysis. The process is used to find hidden patterns and prove that the data are random. When using autoregressive-moving-average (ARMA) models for forecasting, the importance of autocorrelation analysis is particularly notable since it enables the selection of the best model parameters. Plots for the autocorrelation function (ACF) and partial autocorrelation function (PACF) are examined as part of the investigation.

Autocorrelation Plot - A graphical tool called an autocorrelation plot is used to investigate the relationship between a time series and its prior values at various time delays. An autocorrelation graphic illustrates the degree of connection between the current month's inflation rate and the inflation rates in preceding months up to a given number of lags in the context of inflation indices for monthly time series data.

The correlation between two variables y1, y2 is defined as:

$$\rho = \frac{\frac{1}{n}\{\sum_{\{i=1\}}^{n}[(y_1 - \mu_1)(y_2 - \mu_2)]\}}{\sigma_1 \sigma_2} \quad (1)$$

Partial Autocorrelation Plot - It is a statistical method used to examine how present and historical values of a variable are related while accounting for other factors that could be correlated. It involves calculating the partial correlation coefficients, corrected for various lags in between, between the inflation index's present value and its prior lags. The graphic illustrates the degree of correlation between the current inflation index and historical lags while taking other intervening factors into account.

The formula for the Partial Autocorrelation Function (PACF) for k lags can be defined as:

$$\rho_{(y_i,k)} = \frac{Covariance\left([y_i|y_{(i-1)}, y_{(i-2)}, \ldots, y_{(i-x+1)}], [y_{(i-k)}|y_{(i-1)}, y_{(i-2)}, \ldots, y_{(i-k+1)}]\right)}{Variance\left([y_i|y_{(i-1)}, y_{(i-2)}, \ldots, y_{(i-x+1)}]\right) * Variance\left([y_{(i-k)}|y_{(i-1)}, y_{(i-2)}, \ldots, y_{(i-k+1)}]\right)} \quad (2)$$

4 Stationarity

Stationarity tests

When analysing time series data, it is crucial to evaluate whether the data exhibits stationarity. Stationarity refers to the constancy of the statistical properties over time including the mean, variance, and covariance [18]. In contrast, non-stationary time series demonstrate trends or seasonality, making it challenging to apply models that require stationarity such as autoregressive (AR) and moving average (MA) models.

Therefore, prior to utilizing AR or MA models for forecasting, it is essential to conduct stationary tests on the time series data. The commonly used test for stationarity is the Augmented Dickey-Fuller (ADF) test [19], which evaluates the null hypothesis that the time series includes a unit root signifying non-stationarity compared to the alternative hypothesis that the time series is stationary. This is crucial to ensure accurate forecasting, as non-stationarity can lead to erroneous predictions when using models that require stationarity. Before conducting a parametric Augmented Dickey-Fuller (ADF) test on a time series dataset, we conduct preliminary checks for stationarity using non-parametric tests, such as the Hodrick-Prescott filter (Appendix 2.1). This particular test is designed to separate a time series into two distinct components: a trend component and a cyclical component. If the trend component is found to be non-zero, then it indicates that the time series is non-stationary. The Hodrick-Prescott filter is widely used in the fields of economics and finance to decompose time series data into its underlying trend and cyclical components. This is useful for understanding the patterns and dynamics present in the data, as well as for identifying trends and anomalies that may have important implications for forecasting and decision-making. The [Figure 2] below depicts the application of a filter to the Diesel price.

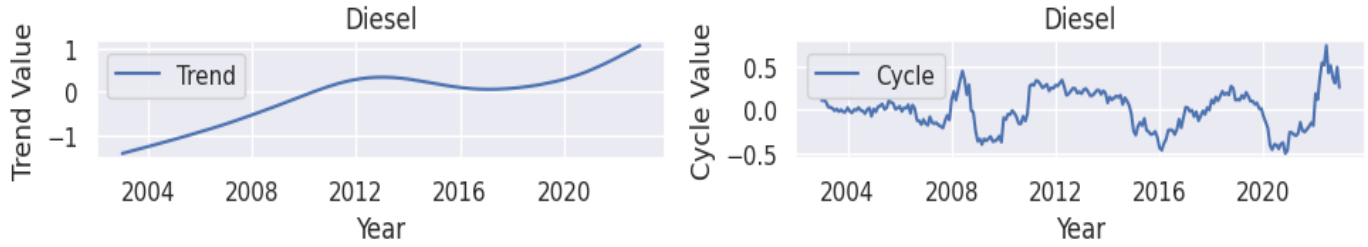


Figure 2 Hodrick-Prescott applied to timeseries data, trend component (left) and cyclical component (right)

After conducting a non-parametric test, it was determined that the data was non-stationary. To further test for stationarity, an Augmented Dickey-Fuller (ADF) test was conducted. If the p-value from the ADF test is less than the chosen significance level, typically 0.05, the null hypothesis is rejected, and the time series is considered stationary. Conversely, if the p-value is greater than the significance level, the null hypothesis cannot be rejected, and the time series is considered non-stationary.
Apart from the p-value, the ADF test also produces a test statistic (ADF Stat) which can be compared to the critical value at the selected significance level. If the ADF Stat is less than the critical value, the null hypothesis is rejected, indicating that the time series is stationary. However, if the ADF Stat is greater than the critical value, the null hypothesis cannot be rejected, and the time series is considered non-stationary (Appendix 2.2). Applying ADF test confirmed the data to be non-stationary and validated the preliminary findings done on non-parametric test.

Converting Non-Stationary data to Stationary

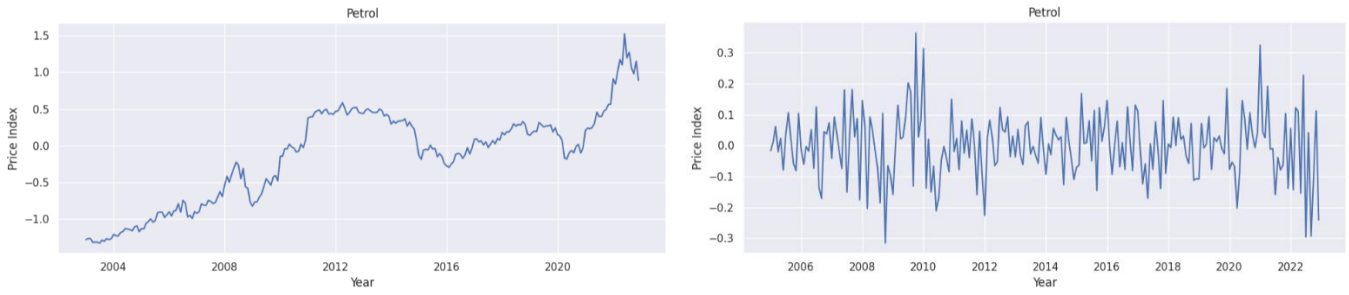


Figure 3 Comparison plot of non-stationary (left) and stationary data (right)

This is achieved by applying several transformations successively. First, a rolling mean with a window size of 12 is computed and then subtracted from the original data to eliminate any underlying trend. Next, seasonality is removed by taking the difference between the data at the same time point in the previous season, using the diff () function with a period of 12. Then, the code examines autocorrelation by computing the

difference between the data and its lagged value and removes it using the diff () function (Appendix 2.3). The resulting data is considered stationary that can be used for further analysis and modelling that requires stationary data. To validate stationarity, Augmented Dickey-Fuller is re-run and we see that the null hypothesis is rejected, concluding the data has indeed been converted to stationary (Appendix 2.4). The [Figure 3] illustrates a comparison between the raw data and the converted stationary data for Petrol price index.

5 Modelling theory and implementation

5.1 Linear models

5.1.1 Principal Component Analysis

Principal component analysis, or PCA, is a technique for reducing the number of dimensions in large data sets by condensing a large collection of variables into a smaller set that retains most of the large set's information. Because PCA is an easy, non-parametric way of removing pertinent information from complex data sets, it is widely utilized in various types of study, from neuroscience to computer graphics. A simple roadmap is provided by PCA for how to simplify a complex data set to a lower dimension to show the sometimes concealed, underlying structure [20].

Theory

PCA involves transforming interdependent variables (called "correlated" in statistics) into new variables that are uncorrelated with each other. The information provided by the data will be described by these variables, which are referred to as the principal components. We need to look at the covariance matrix because covariance is a measure of the joint variability of two random variables, to maintain the PI and discard the rest in the PCA, we must compute the covariance of our data matrix and search for the directions or vectors that gather the most information [21].

1. Standardization of the data

For each continuous initial variable to contribute equally to the analysis, this phase standardizes the range of the variables. Standardization must be done before PCA because the latter is very sensitive to the variances of the starting variables. That is, if there are significant disparities in the initial variable ranges, the bigger range variables will predominate over the smaller range variables, resulting in biased findings. If "A" is the matrix which describes the data, thus each row is an example, and each column is a feature. Let's set n equal to the number of features et m the number of examples. Thus, the matrix A is a matrix mxn.

$$A^i_{Standard} = \frac{A - mean}{standard\ deviation} \tag{3}$$

where A^i *is one of the feature of* A *and* $A_{standard}$ *is the new data matrix*

2. Covariance matrix and diagonalization

The purpose of this stage is to determine the relationship, if any between the variables in the input data set and how they differ from the mean in relation to one another. Because variables can occasionally be highly connected to the point where they include redundant data. We compute the covariance matrix to find these associations [22].The covariance of a matrix is given by following formula:

$$cov(A_{standard}) = \sum = \frac{1}{m}\sum_{i=1}^{n}\left(A^i_{standard}\right)\left(A^i_{standard}\right)^T = A^T A \tag{4}$$

3. Dimensionality reduction

In this part, we want to determine how many eigen vectors are required to adequately describe our data. Therefore, we must first determine the importance of each eigenvalue before calculating the cumulative variance. The significance of an eigen value is represented as:

$$Significance\ \sigma_i = \frac{abs(\sigma_i)}{\sum_{i=1}^{n} abs(\sigma_i)} \tag{5}$$

4. Choose the k eigen vectors

In this part we choose k eigen vectors based on the threshold we defined.

$$cummulative\ variance\ explained = \frac{\sum_{i=1}^{k} \sigma_i}{\sum_{i=1}^{n} \sigma_i} \geq threshold \quad (6)$$

Implementation

We first generated the covariance matrix, and then we retrieved the eigen values and eigen vector from that covariance matrix. We have set the threshold value for dimensionality reduction to 99.5%, which will essentially encompass all the significant characteristics. Eigen values of characteristics that fall below the threshold are not taken into consideration.

5.2 Statistical Models

5.2.1 Exponential Smoothing

Theory

We use Exponential Smoothing as a base model to understand how the changes in inflation fluctuations from past affect the future predictions. The Exponential Smoothing Model [23] functions by computing a weighted average of past observations, where recent observations are given more weight than older ones. The weight given to past observations gradually decreases over time based on a smoothing parameter that can be adjusted to suit the time series under analysis. This Model is advantageous due to its ability to capture trends and seasonality present in the data, resulting in more precise forecasts of future inflation rates. The model can be customized to accommodate numerous seasonal patterns including weekly and monthly fluctuations.

Alpha is the smoothing factor for the level of the time series, which controls the weight given to past observations. A higher alpha value places greater emphasis on recent observations, while a lower alpha value places more weight on older observations.

Beta is the smoothing factor for the trend of the time series, which controls the rate of change over time. A higher beta value results in a more responsive trend to recent changes, while a lower beta value results in a more stable trend that changes more slowly.

Seasonal periods are the number of time steps in a seasonal cycle, such as the number of months in a year for monthly data or the number of quarters in a year for quarterly data. This parameter is used to capture seasonal patterns in the data and adjust for them in the forecast.

$$y_t = l_{t-1} + b_{t-1} + s_{t-m} + \epsilon_t \quad (7.1)$$

$$\ell_t = \ell_{t-1} + b_{t-1} + \alpha\varepsilon_t \quad (7.2)$$

$$b_t = b_{t-1} + \beta\varepsilon_t \quad (7.3)$$

$$s_t = s_{(t-m)} + \gamma\varepsilon_t \quad (7.4)$$

Where m *is the number of seasons in a year*, $0 \leq \alpha \leq 1, 0 \leq \beta \leq \alpha,$ and $0 \leq \gamma \leq 1 - \alpha$

y_t *is the basic additive model equation at time t and is the sum of the previous level* l_{-1},
the previous trend b_{t-1}, *the seasonal component* s_{t-m}

l_t *is the level smoothing equation which calculates the new level value* l_t,
the sum of the previous level l_{t-1}, *the previous trend* b_{t-1}, *and a smoothed error term* $\alpha\varepsilon_t$

Implementation

We choose the parameter values of alpha and beta to be 0.7 to place greater emphasis on the more recent trends in the time series data. This is because the inflation rates have been highly variable in recent years, and it is important to capture these fluctuations in the forecasting model. The seasonal periods parameter was set to 12 to correspond with the monthly frequency of the time series data being analysed [24]. The testing dataset is selected from the most recent twelve months, while the remaining data is employed as the training dataset (Appendix 3.1). For each index, the model is executed and assessed based on metrics such as mean square error, mean absolute error, Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC) (Appendix 3.2).

5.2.2 Auto-Regressive Model

Theory

While Exponential Smoothing models are particularly well-suited for data exhibiting seasonal or cyclical patterns. On the other hand, AR models are more appropriate for data that exhibit strong autocorrelation and trends. The Autocorrelation Function (ACF) plot reveals that the inflation index data displays a high degree of correlation and presents additive trends (Appendix 4.1). Additionally, AR models have the ability to integrate numerous lagged values of the time series variable, thus enabling them to capture more intricate associations between past and future values.

The Auto-Regressive model assumes that the current value y_t is dependent on previous values $y_{\{(t-1)\}}, y_{\{(t-2)\}}, \ldots$. Because of this assumption, we can build a linear regression model. AR(p) model:

$$Y_t = c + \phi_1 Y_{\{t-1\}} + \phi_2 Y_{\{t-2\}} + \ldots + \phi_p Y_{\{t-p\}} + \varepsilon_t \quad (8)$$

where:

Y_t is the value of the time series at time t.

c is a constant (also known as the intercept).

$\phi_1, \phi_2, \ldots, \phi_p$ are the parameters of the model that represent the coefficients on the past values of the time series.

$Y_{\{t-1\}}, Y_{\{t-2\}}, \ldots, Y_{\{t-p\}}$ are the past values of the time series.

ε_t is the error term (also known as the residual) at time t, which represents the part of the time series that is n ot explained by the past values.

The parameter p is the order of the AR model, which specifies how many past values of the time series are included in the model.

Implementation

AR models assume that the underlying process generating the time series data is stationary [25], which means that the statistical characteristics of the process do not alter over time. The next step in the AR modelling process is to specify the model order denoted by the value "p," which indicates the number of prior values of the time series variable used to forecast its value at time "t". For example, an AR (1) model only considers the preceding observation to estimate the current value, while an AR (2) model uses the two previous observations. We take the data converted to stationarity as input for AR Model. We can then establish the parameter value of "p" by analysing the autocorrelation function (ACF) plot, which indicate a strong correlation to the preceding year's data and as a result a value of 15 is assigned (Appendix 4.2). The accuracy of the AR model's predictions is evaluated using statistical measures such as mean absolute error, mean squared error, Akaike Information Criterion (AIC) [26] and Bayesian Information Criterion (BIC) (Appendix 4.3).

5.2.3 Moving Average Model

The moving average (MA) model is a mathematical framework used to analyse and forecast time series data, such as inflation, by using past error terms to predict future values. We decided to choose a moving average model because it is able to help smooth the data and provide a clear visualisation of the trend of the data,

Theory

The MA model assumes that the current value of a time series variable can be predicted by a linear combination of the past error terms, where the order of the model (represented by "q") determines the number of lagged error terms used in the prediction. For instance, a MA (1) model would utilize the previous error term and a coefficient to forecast the current value of the time series variable. The formula for a MA (1) model includes a constant term, the current error term, and the product of the coefficient and the lagged error term. By analysing the coefficients and residual errors in the MA model, we can gain insights into the dynamics of the time series data and make forecasts for future values. It is important to note that the effectiveness of the MA model relies on the specific characteristics of the data being analysed and that it is one of several time series models used to analyse and forecast inflation data.

The first-order moving average model, which is represented by MA (1), is expressed as:

$$x_t = \mu + \omega_t + \theta_1 \omega_{t-1} \tag{9.1}$$

The second-order moving average model, which is represented by MA (2), is expressed as:

$$x_t = \mu + \omega_t + \theta_1 \omega_{t-1} + \theta_2 \omega_{t-2} \tag{9.2}$$

The qth order moving average model, which is represented by MA(q), is expressed as:

$$x_t = \mu + \omega_t + \theta_1 \omega_{t-1} + \theta_2 \omega_{t-2} + \cdots + \theta_q \omega_{t-q} \tag{9.3}$$

To figure the order of an MA model, you need to look at the ACF. The order of the ACF plot allows an estimation of the order q.

Implementation

The moving average model has a precondition of only using stationary data. After completing our stationary tests, and finding the data was in fact non-stationary, we converted the data to stationary.

The testing data that we used for the MA model was the last 15 data point of stationary data, and our training dataset was all the rows, excluding the last 12.

5.2.4 ARIMA Model

Theory

The AR in ARIMA stands for Auto Regressive and it models the relationship between an observation and a lagged version of itself. The AR component captures the trend in the data. The MA in ARIMA represents the Moving Average and models the relationship between an observation and a lagged error term. It captures the noise of the data [27].

The second component of the ARIMA model is Integration, also known as Differencing. This element models the degree of differencing needed to make the time series stationary, i.e., to remove any trend or seasonality present in the data [28] to obtain meaningful results. It can be expressed as:

$$Y_t' = (1 - L)^d \cdot Y_t \tag{10}$$

Where Y_t' is the differenced series, L is the Lag operator and d is the order of differencing.

The equation for the ARIMA model, with notation p (AR order), d (degree of differencing, q (MA order), applied to inflation data looks like:

$$Y_t = c + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \ \ldots \ + \phi_p Y_{\{t-p\}} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t \tag{11}$$

Implementation

The ARIMA model is best implemented with a stable time series data set. So, it is important to check if the data has a constant variance and mean over time [27]. Using multiple tests, we found our time-series data to be non-stationary, so it was converted to stationary by applying some transformations, allowing us to execute the ARIMA model.

On the data is stationary, the values of p and q can be determined using the autocorrelation function (ACF) and partial autocorrelation function (PACF) plots. The ACF plot shows the correlation between each observation and its lagged values, while the PACF plot shows the correlation between an observation and its lagged values after removing the effect of correlations due to intermediate lags.

The number of AR terms (p) can be determined by looking at the significant lags in the PACF plot, while the number of MA terms (q) can be determined by looking at the significant lags in the ACF plot. A lag is considered significant if its corresponding correlation coefficient is outside the bounds of the confidence interval.

Using a test size of 12 (the last 12 points of our data), we define a model that loops through the data frame of the train set, highlighting the differentiating aspect of the ARIMA model. We then fit a model that calculates AIC, BIC, MSE and MAE, all of which will assist in accurately assessing the results of the model prediction. The 'order' parameter is a tuple that specifies the model order as (1, 0, 15) meaning the model has a single autoregressive term (p=1), no differencing (d=0), and 15 moving average terms (q=15).

5.2.5 SARIMA Model

SARIMA (Seasonal Autoregressive Integrated Moving Average) is a statistical model used for time series forecasting. It is an extension of the popular ARIMA (Autoregressive Integrated Moving Average) model, which is used for non-seasonal time series data. Seasonal impacts are frequently seen in many time series data. the typical temperature recorded in a place with four seasons, for instance, every year there will be a seasonal effect, and this season's temperature will unquestionably be strongly correlated with the temperature recorded last year at the same season [29].

Methodology

The SARIMA (Seasonal Autoregressive Integrated Moving Average) methodology is a time series forecasting technique that extends the ARIMA model to incorporate seasonality. The general form of a SARIMA model is denoted as SARIMA (p, d, q)(P, D, Q, s), where:

- p: the number of autoregressive terms
- d: the number of differences needed to make the time series stationary
- q: the number of moving average terms
- P: the number of seasonal autoregressive terms
- D: the number of seasonal differences needed to make the time series stationary
- Q: the number of seasonal moving average terms
- s: the period of the seasonality (e.g., 12 for monthly data with annual seasonality)

The formula for a SARIMA model can be expressed as:

$$(1 - \varphi^1 B - \varphi^2 B^2 - \cdots - \varphi_p B^{\mathrm{p}})(1 - \Phi^1 B^{\mathrm{s}} - \Phi^2 B^{2\mathrm{s}} - \cdots - \Phi_p B^{\mathrm{Ps}})(1 - B)^{\mathrm{d}}(1 - B^{\mathrm{s}})^{\mathrm{D}} y_t = (1 + \theta^1 B + \theta^2 B^2 + \cdots + \theta_p B^{\varphi})(1 + \Theta^1 B^{\mathrm{s}} + \Theta^2 B^{2\mathrm{s}} + \cdots + \Theta_p B^{\mathrm{Ps}})\varepsilon_t \quad (12)$$

$where$:
$y_t\ is\ the\ time\ series\ at\ time\ t$
$B\ is\ the\ backshift\ operator, such\ that\ By_t = y_t - 1$
$\varepsilon_t\ is\ the\ error\ term\ at\ time\ t$
$\varphi^1, \ldots, \varphi_p\ are\ the\ autoregressive\ coefficients$
$\Phi^1, \ldots, \Phi_p\ are\ the\ seasonal\ autoregressive\ coefficients$
$\theta^1, \ldots, \theta_p\ are\ the\ moving\ average\ coefficients$
$\Theta^1, \ldots, \Theta_p\ are\ the\ seasonal\ moving\ average\ coefficients$
$d\ and\ D\ are\ the\ non - seasonal\ and\ seasonal\ differences, respectively$

This formula represents a linear equation that models the relationship between the time series at time t and its lagged values, as well as the errors in the model. The coefficients in the model can be estimated using statistical methods such as maximum likelihood estimation. Once the model is fitted to the data, it can be used to forecast future values of the time series.

Implementation

We have used the function "tsa.SARIMAX(sarimaTrainDF[col], order= (0, 1, 1), seasonal_order=(0, 1, 1, 12)" of the "statsmodels.tsa" module. This function is used to fit a seasonal ARIMA (SARIMA) model to a time series. The sarimaTrainDF[col] argument specifies the column of the sarimaTrainDF Data Frame that contains the time series data. The order argument specifies the order of the SARIMA model, which is a tuple of three integers. The first integer specifies the number of autoregressive (AR) terms, the second integer specifies the number of moving average (MA) terms, and the third integer specifies the number of seasonal autoregressive (SAR) terms. In this case, the order is (0, 1, 1) with seasonal order of 12, which means that the model has two AR terms, two MA terms, and no SAR term, the SAR term is 0 because we found that there was minimal seasonality during time series decomposition. The SARIMAX () function returns a statsmodels.tsa.statespace.SARIMAXResults object. This object contains information about the fitted model, such as the estimated parameters, the AIC and BIC scores, and the forecast errors.

5.3 Deep-learning models

5.3.1 Recurrent Neural Network Model

Theory

Recurrent neural network (RNN) modelling is a machine learning algorithm used for sequential datasets. Thus, it is applied to timeseries which is the exact structure of our fuel data rendering this model suitable to use. The main idea behind RNNs is that it considers the current input, as well as previous inputs [30]. RNNs use an internal memory which assists in the preservation of information retrieved from previous data points. They utilise a feedback loop that helps to determine the order in which the data values were input into the recurrent neural network – this distinguishes RNNs from traditional neural networks [31]. Consequently, previous values are considered in the recurrent model.

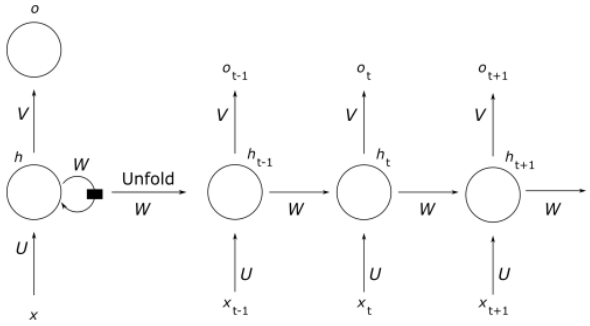


Figure 4 Recurrent neural network with the shared weights of U, V, and W [31]

[Figure 4] demonstrates the notation of the RNN and shows its transformation into the full network. The process of carrying the memory forward is expressed as:

$$a_t = \mathrm{b} + Wh_{t-1} + Ux_t \quad (13.1)$$

$$h_t = \tanh(a_t) \quad (13.2)$$

$$o_t = \mathrm{c} + Vh_t \quad (13.3)$$

$$\hat{y}_t = \mathrm{softmax}(o_t) \quad (13.4)$$

Where h_t is the current hidden state ('memory') at time t, x_t represents the input at time t, o_t represents the output at time t, and U, V and W are variables that represents the weight matrices.

Implementation

Univariate Forecasting: We implement the RNN model for both univariate and multivariate forecasting. Firstly, the dataset is split into training and testing using a test size of 12. We then define a model function using TensorFlow module containing the RNN initialiser, output, model, and loss function. We use a batch size of 64, our epoch is defined as [50, 100, 150] and we have a learning rate of [0.001, 0.01, 0.1]. There are multiple values for each because a learning rate close to 1 can increase the pace of the training process of the RNN model however this may cause overshooting which increases the margin of error. Additionally, the number of epochs governs a trade-off between the length of time it takes to train the RNN and its accuracy. Having a range of values allows us to decide which is optimal for our results. Next, we compile the model using the Adam optimiser algorithm as it is well-suited for large datasets and MSE as our loss.

We then implement the model, calculating the AIC, BIC MSE and MAE metrics as well. On giving our results we merge the train, test, and prediction values into one data frame and produce a plot for each index.

To obtain prediction values for the future, we begin by collecting the last three columns of our original data, 'datagreaterthan2003', and operate the model defined to generate predicted values for a given future date range, with a line plot of the results.

Multivariate forecasting: To proceed with multivariate forecasting, we begin by filtering the principal components of our data set for each index: Producer Price Index, Energy Price Index, Official Core Consumer Price Index, Food Price Index, Petrol and Diesel. Then we split the data as before, reshape and follow the same process as for the univariate set.

In this instance, we also produce a heat map displaying the correlation between each of the variables.

5.3.2 Long Short-Term Memory networks

Long Short-Term Memory (LSTM) models are frequently used as forecasting models within a business application, such as predicting inflation. As a result, we decided that a LSTM neural network would be an excellent tool to allow for forecasting future consumer, energy, food, headline, and producer price index, as well as petrol and diesel prices. We decided to use LSTMs due to their superiority in handling long-term dependencies, which was important due dealing with time-series data [32]. RNNs suffer an issue known as the vanishing gradient problem. The vanishing gradient problem relates to deep learning gradient descent algorithms of the RNN [33]. During the time series of the network, gradient descent is combined with a backpropagation algorithm to update synapse weights of the neural network and calculate cost function at each point. Each calculation of cost function at deeper neurons of the layer is used to change weights of neurons in shallower layers, this process is multiplicative, meaning the gradients will be decreasing as the gradient calculation moves throughout the network-leading to the vanishing gradient problem. Conversely, LSTMs can fix this issue since the architecture of an LSTM is able to leave more dimensions untouched. Thus, since our dataset is multivariate, the LSTM would be able to portray the more complex relationships which a multivariate dataset would display as opposed to a univariate dataset.

Theory

LSTMs contain four interacting layers within a single repeating module. In an LSTM, the "cell state" acts as system to allow information through, with some minor interactions occurring [34]. The cell state may gain information or lose information, which is regulated by gates, aiding the transmission or stoppage of information. Gates are a sigmoid neural network layer, either outputting 0 - no transmission or 1 for allow transmission.

The input stage: First step of the LSTM, is to decide what information is disregarded from the cell state. The sigmoid layer inspects h_{t-1} and x_t then outputs zero or one for each number c_{t-1} [35].

$$f_t = \sigma\big(W_f \cdot [h_{t-1}, x_t] + b_f\big) \tag{14.1}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{14.2}$$

$$C_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_C) \tag{14.3}$$

The forget stage: In the second step of the LSTM, the now outdated c_{t-1} is updated to c_t as the new cell state. Consequently f_t is multiplied by the previous c_{t-1}, and then i_t. C_t is added, this is essentially the new values of state scaled by the update parameter.

$$C_{t1} = f_t \cdot C_{t-1} + i_t \cdot C_t \tag{15}$$

The output stage: In the last stage of the LSTM, is the output, which will be based on a filter version of the cell state. It includes two sub-steps; running a sigmoid layer to decide output and putting the cell state through tanh to push the values between –1 and 1 and multiplying by output of the sigmoid gate. Demonstrated by the following two equations:

$$\sigma_t = \sigma(W_o[\, h_{t-1}, x_t\,] + b_o) \tag{26.1}$$

$$h_t = o_t \cdot \tanh(C_t) \tag{16.2}$$

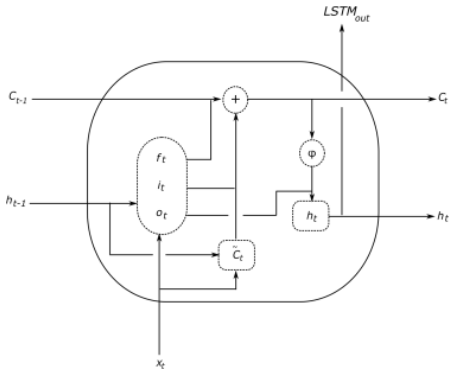


Figure 5 Basic architecture of an LSTM memory block [31]

Implementation and reasonings for parameters

The dataset 'dataGreaterthan2003' was first split into training and testing data sets using a test size of 12. The batch size is utilised for training of the LSTM, essentially the training samples are provided as batches, hence the batch size can affect performance accuracy. We chose a batch size of 64, as we believed this could present the most optimal results for prediction.

The optimisation algorithm that we chose was the Adam optimiser, the Adam optimiser can update the learning rate for each network weight during training- unlike a single learning rate. We decided to choose the Adam optimiser because it was straightforward and easy to implement, thus, it is also computationally efficient and suited for large multivariate datasets such as ours.

An epoch refers to the single time an entire training dataset is passed through the LSTM. Generally, the greater the number of epochs, the more the LSTM network can learn from the data, leading to increased accuracy on testing data. However, overtime, increasing the number of epochs can lead to overfitting. For this reason, we started with 50 epochs, and increased in multiples of 50 until 150. By doing this, we continued to train, until the performance stopped improving.

The learning rate hyperparameter dictates the step size for updating weights of the network during the training stage. The learning rate is a critical component of the LSTM because it determines the convergence rate to the optimal weights which can minimise the loss function. For example, if the learning rate is too large, sub-optimal range of weights may be learned too fast, or if the learning rate is too small, the training process would take a considerable amount of time. For this reason, we chose learning rates 0.001.0.01 and 0.1. We started with a small learning rare and increased the value to find the optimal learning rate.

Table 2 Parameters used for LSTM model.

Batch size	64
Test size	12
Optimiser	Adam
Loss	Mean squared Error
Window	3
Epochs	50,100,150
Learning rate	0.001,0.01,0.1

5.3.3 Transformer Model

Introduction

Transformers and a system known as a sequence-to-sequence architecture are both discussed in the paper [36]. A neural network called series-to-Sequence (also known as Seq2Seq) translates one series of elements, such as the words in a phrase, into another sequence. Transformer is an architecture that uses two components (Encoder and Decoder) to transform one sequence into another, however it varies from previous sequence-to-sequence models in that it does not imply any recurrent networks.

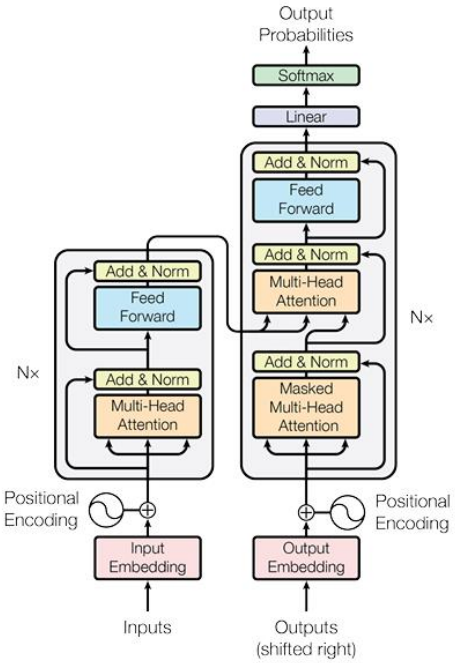


Figure 6 The Transformer-model architecture [36].

Methodology

The basic idea behind transformers is to use self-attention to learn long-range dependencies between different parts of a sequence. Self-attention is a mechanism that allows a neural network to attend to different parts of an input sequence, and to learn how these parts are related to each other. The encoder maps an input sequence of symbol representations $(x_1, \dots, x_n)$to a sequence of continuous representations $z = (z_1, \dots, z_n)$. Given z, the decoder then generates an output sequence $(y_1, \dots, y_m)$of symbols one element at a time. At each step the model is auto regressive, consuming the previously generated symbols as additional input when generating the next [36].

- **Input Embedding**
  The input text must first be transformed into a numerical format that the neural network can use. This is typically done using word embeddings, which map each word to a high-dimensional vector. A matrix is created by concatenating the input embeddings together.
- **Positional Encoding**
  Since the Transformer model does not use recurrent layers to process the input, it needs a way to capture the position of each word in the sequence. This is achieved using positional encoding, which adds a fixed vector to each input embedding based on its position in the sequence. The positional encoding vectors are defined [36]:

$$PE_{(pos,2i)} = \sin\left(\frac{\text{pos}}{10000^{\frac{2\text{i}}{\text{d}_{\text{model}}}}}\right) \tag{17.1}$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{\text{pos}}{10000^{\frac{2\text{i}}{\text{d}_{\text{model}}}}}\right) \tag{17.2}$$

where pos is the position and i is the dimension, and d_{model} is the dimensionality of the embedding vector.
That is, each dimension of the positional encoding corresponds to a sinusoid.
The wavelengths form a geometric progression from 2π to $10000 \cdot 2\pi$.

- **Encoder Self-Attention**
  Each encoder layer has a multi-head self-attention mechanism as its initial sub-layer, which enables every position in the input sequence to pay attention to every other position. As shown below, the self-attention mechanism is calculated.

$$Attention(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (18)$$

$where\ Q\ ,K,and\ V\ are\ the\ query,key,and\ value\ matrices,respectively\ ,6.$

$Q\ ,K,and\ V$ are computed from the input embeddings and positional encodings. The value matrix's values are weighted using the distribution that results from the SoftMax function's computation of a distribution over the key matrix's values. A weighted total of the values from the self-attention mechanism is the output, and it is added to the input embeddings after being put through a normalization layer.

- **Multi-Head Attention**

The Attention module of the Transformer performs its calculations repeatedly and concurrently. These are referred to as Attention Heads each. The Attention module divides its Query, Key, and Value parameters N-ways and independently routes each split through a different Head. A final Attention score is subsequently generated by combining all these related Attention calculations. With the use of a technique known as "multi-head attention," the Transformer is better able to encode various relationships and nuanced aspects for each word [37].

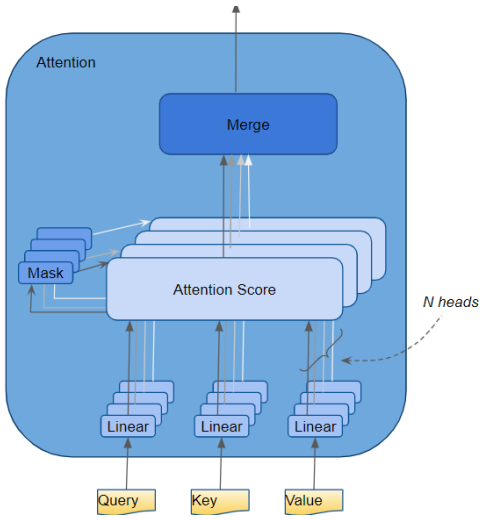


Figure 7 Multi-head attention [37].

$$Attention(Q,K,V) = softmax\left(\frac{QK^T + Mask}{\sqrt{d_k}}\right)V \quad (19)$$

$where\ Q\ ,K,and\ V\ are\ the\ query,key,and\ value\ matrices,respectively\ ,6.$

- **Feed-Forward Network**

  The second sub-layer of each encoder layer is a position-wise fully connected feed-forward network. The feed-forward network is defined as:

$$FFN(x) = \max(0, xW_1 + b_1)\,W_2 + b_2 \quad (20)$$

$where\ x\ is\ the\ input\ vector, W_1, b_1, W_2, and\ b_2 are\ the\ weight\ matrices\ and\ biases\ of\ the\ two\ linear\ layers$

- **Decoder Self-Attention**

  The decoder also uses a multi-head self-attention mechanism, but it is slightly different from the encoder self-attention. Each location in the decoder can only focus on positions in the input sequence that the encoder has previously processed and positions in the decoder sequence up to the present position. This is how the decoder's self-attention mechanism is calculated [38, 39]:

$$Attention(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (21)$$

$where\ Q\ ,K\ ,and\ V\ are\ the\ query,key,and\ value\ matrices$

Q , K , $and\ V$ are computed from the decoder input embeddings and positional encodings. The resulting distribution is used to weight the values in the value matrix, and the output of the self-attention mechanism is added to the decoder input embeddings and passed through a normalization layer.

- **Encoder-Decoder Attention**
  The encoder-decoder attention mechanism is computed as follows:

$$Attention(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (22)$$

$$where\ Q\ ,K\ ,and\ V\ are\ the\ query, key, and\ value\ matrices$$

Q , K , $and\ V$ are computed from the decoder self-attention output and the encoder output, respectively [38].

- **Feed-Forward Network**
  Like the feed-forward network used in the encoder, the second sub-layer of each decoder layer is similarly a position-wise completely connected feed-forward network.
- **Output Layer**
  A linear layer and a SoftMax function are applied after the output of the last decoder layer to produce the final output distribution over the vocabulary. The definition of the output layer is:

$$P(y_t|y_{<t}, x) = softmax(y_t W_o) \quad (23)$$

$$where\ y_t\ is\ the\ t-th\ output\ symbol, y_{\{<t\}}\ are\ the\ previously\ generated\ symbols,$$
$$x\ is\ the\ input\ sequence, W_o\ is\ the\ weight\ matrix\ of\ the\ linear\ layer$$
$$and\ P(y_t|y_{<t}, x)\ is\ the\ probability\ of\ generating\ the\ symbol\ y_t\ given\ the\ previously$$
$$generated\ symbols\ and\ the\ input\ sequence.$$

Implementation

The transformer model consists of an encoder and a decoder. The encoder processes the input sequence while the decoder generates the output sequence. In this implementation, the encoder consists of a stack of eight transformer encoder layers each with ten attention heads. The encoder also includes a positional encoding layer, which adds information about the position of each token in the sequence. The encoder is followed by a linear layer, normalization, and ReLU activation functions. The decoder has five linear layers with ReLU activation functions and dropout regularization. It concatenates the output of the encoder with the input sequence and the resulting tensor is passed through the linear layers to generate the final output. The model's forward function takes a source sequence as input, generates a mask to prevent the model from attending to future tokens then applies positional encoding to the input and passes it through the transformer encoder. The output of the encoder is then normalized and passed through the decoder layers and the final output is returned. This implementation is flexible with several hyperparameters that can be adjusted such as the model's dimensionality, number of layers, dropout rate and the number of attention heads.

Encoder Layer - nn.Transformer Encoder Layer is a building block of the transformer model that performs multi-head self-attention on the input sequence and applies a feedforward neural network (FFN) to the resulting representations. The layer takes an input tensor of shape (sequence_length, batch_size, d_model) and returns an output tensor of the same shape.

During the self-attention step, the input tensor is split into num_heads smaller tensors, and each of these tensors is passed through an attention mechanism that computes the importance of each token in the sequence relative to the others. The outputs of these attention heads are then concatenated and passed through a linear layer to generate the attention output.

The attention output is then passed through a two-layer feedforward neural network (FFN) that applies a ReLU activation function and dropout regularization to the intermediate representation. The output of the FFN is added to the input tensor using residual connections, and the resulting tensor is normalized using layer normalization.

nn.TransformerEncoderLayer takes several hyperparameters, such as d_model, which sets the size of the input and output tensors, nhead, which sets the number of attention heads, and dim_feedforward, which sets the size of the intermediate representation in the FFN. The layer also includes dropout regularization to prevent overfitting during training.

5.4 Performance Metrics

All models were implemented with four common performance metrics to evaluate them uniformly. The metrics used are mean square error, mean absolute error, akaike information criterion and bayesian information criterion.
The main reason MSE is used as an evaluation metric in our prediction models is because it penalizes larger errors more heavily than smaller errors. Since the errors are squared before taking the average, larger errors have a greater impact on the final MSE value. This makes MSE particularly useful in regression problems, where the goal is to predict a continuous target variable. MAE is used because it is easy to interpret and gives a good sense of the magnitude of the errors in the model's predictions. It is also less sensitive to outliers.
By using both metrics together, we can get a more comprehensive understanding of the performance of an ML model. Together, they can provide a more nuanced evaluation of the model's predictive power. While MSE and MAE provide the measure of model's performance, they do not directly account for the complexity of the model.
AIC and BIC, on the other hand consider both the goodness of fit and the complexity of the model. They penalize models with more parameters, thus avoiding overfitting and providing a trade-off between model complexity and goodness of fit.

$$loglik = -0.5 * len(residuals) * log(mean(residuals)^2) - 0.5 * modelparams \quad (24.1)$$

$$AIC = -2 * loglik + 2 * modeparams \quad (24.2)$$

$$BIC = log\big(len(y_{test})\big) * len(modeltrainableweights) - 2 * log(mse) \quad (24.3)$$

Where,
loglik: is the log − likelihood of the model.
residuals: differences between the observed values and the predicted values.
model*params*: *the parameters of the model.*
y_{test}: *is the test set.*
mse: is the mean squared error

6 Experimental Results

6.1 Exploratory Data Analysis

6.1.1 General Exploratory Data Analysis

First steps in understanding data using EDA were to use Histogram [Figure 5]. Histograms when applied to inflation indices provide insights into the frequency distribution of each index. Analysis of the histograms reveals that the Headline Consumer Price Index exhibits a left-skewed distribution, while the other indices do not show any skewness. Conversely, the symmetrical distribution in the other indices implies that their data is evenly distributed around the centre of the histogram. This suggests that these indices may be less sensitive to changes in prices and have a more stable trend over time. The normal distribution observed in the Headline Consumer Price Index implies that there is a consistent trend in the rate of inflation during the analysed period, which may be attributed to factors such as economic growth, changes in government policies, and market fluctuations.

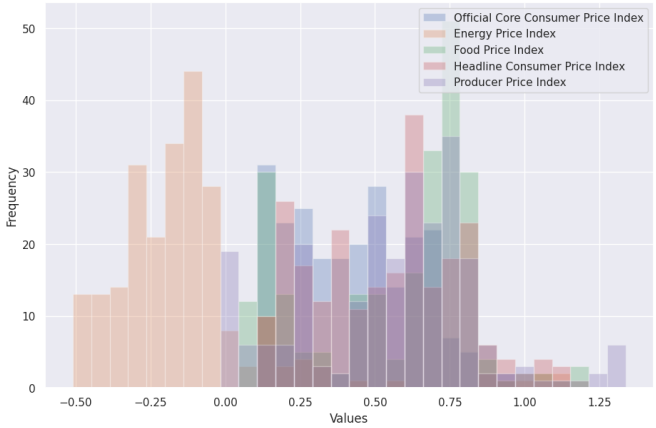


Figure 5 Histogram plot of scaled inflation indices

Next step in EDA involves using Box plot [Figure 6]. Box plots are a useful tool particularly in the case of inflation indices, as they provide insight into central tendency, variability, and outliers. The Official Core Consumer Price Index appears stable, with minimal outliers and a median close to the 50th percentile. The Energy Price Index is more volatile, with more outliers and a median closer to the lower quartile. The Food Price Index has a similar range to the Official Core Consumer Price Index, but with a slightly higher median indicating a more rapid rate of escalation. The Headline Consumer Price Index has a wider range and more outliers, but a median close to the 50th percentile, making it a good representation of overall inflation trends. The Producer Price Index is the most volatile, with a wider range and more outliers than other indices, and a median closer to the 50th percentile, indicating higher producer prices overall.

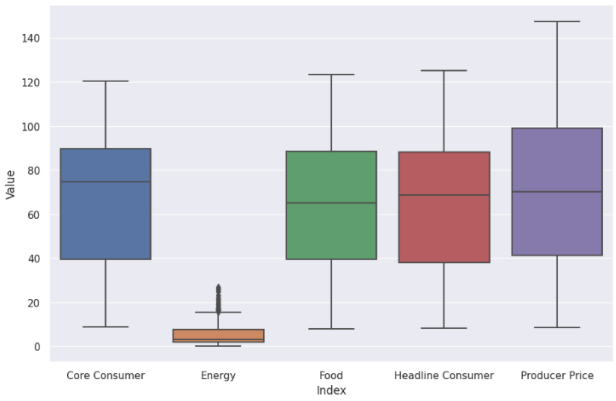


Figure 6 Box plot comparison of inflation indices

Next, we use pair plot [Figure 7(a)] to understand the relationships between variables and identifying patterns or trends. It creates scatter plots for every pair of variables in the dataset, with each variable plotted against every other variable. Finally, we apply correlation analysis [Figure 7(b)] on all the inflation indices along with fuel data, the analysis show fuel data are more closely correlated to Producer price index.

6.1.2 Timeseries Exploratory Data Analysis

First step in Timeseries EDA is Autocorrelation Analysis that include autocorrelation plot [Figure 8(a)] (Appendix 4.1) and partial autocorrelation plot [Figure 8(b)] (Appendix 4.4). The ACF plot shows large number of lagged values with significant correlation, suggesting that the model is highly dependent on past values. The PACF plot shows a significant lag at lag 1, this suggests that there is a strong correlation between the time series and its first lagged value after accounting for the contributions of shorter lags. This may indicate that the time series exhibits some kind of trend or cyclical pattern that occurs on a regular basis, such as a daily or weekly seasonality.

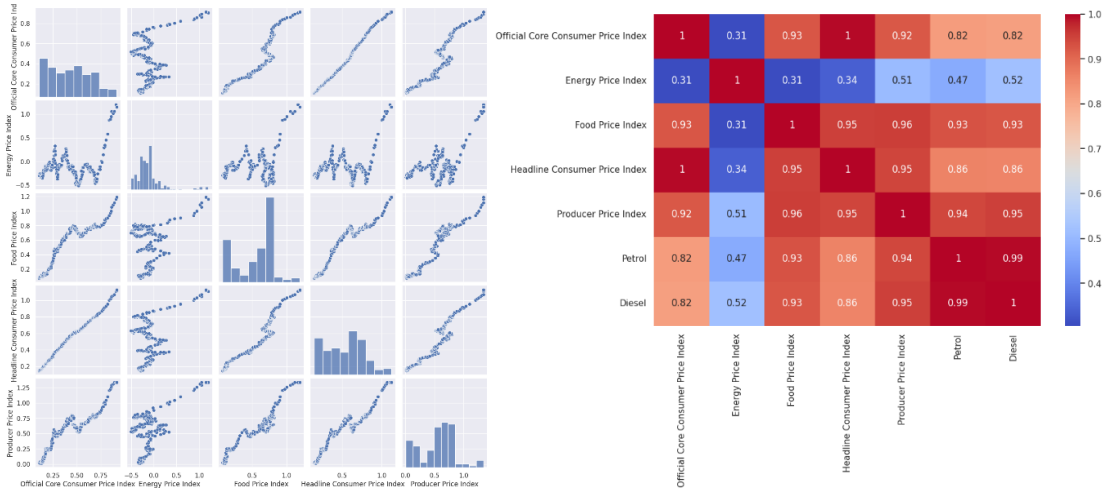


Figure 7 a) Pair plot (left) and b) Tree map (right) for correlation Analysis on Inflation indices along with Fuel data.

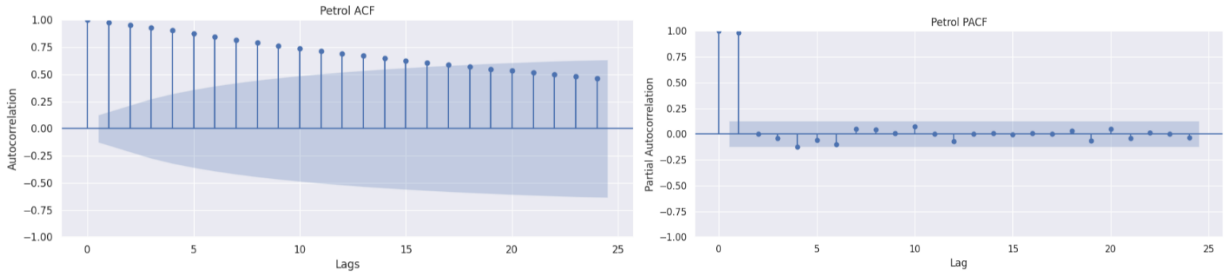


Figure 8 a) Autocorrelation plot(left) b) Partial Autocorrelation plot (right)

Next, we go ahead and plot monthly and quarterly plot to explore the repeating patterns. However, the monthly [Figure 9(a)] (Appendix 4.5) and quarterly plots [Figure 9(b)] (Appendix 4.6) show the same graph for all months and quarters, this suggests that the time series does not exhibit any such patterns at these frequencies.

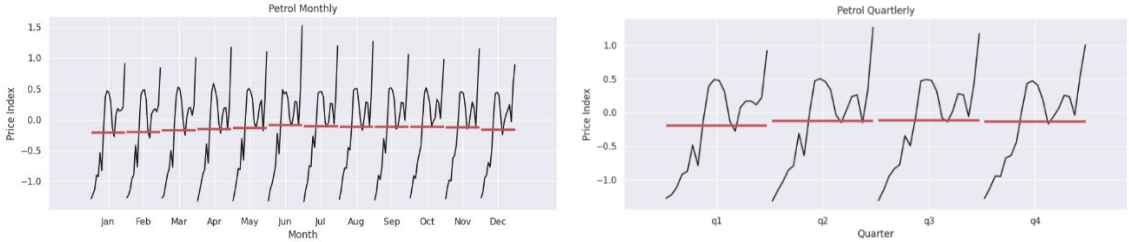


Figure 9 a) Monthly(left) b) Quarterly plot(right)

Finally, we end this section by performing timeseries decomposition [40] (Appendix 4.7) and discover that model is additive, has seasonality with yearly frequency, because the trend line exhibits non-linear behaviour and seasonal component remains constant over a period of year. Illustration of an index is shown in [Figure 10].

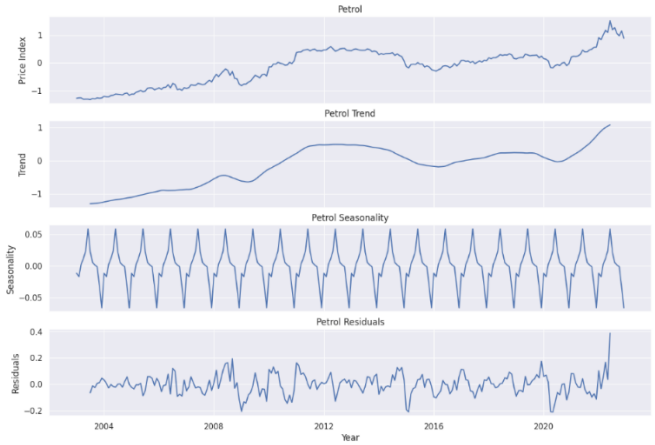


Figure 10 Timeseries decomposition of Energy price index showing Trend, seasonality, and residuals.

6.1.3 Principal Component Analysis

PCA was implemented on five inflation indexes chosen along with petrol and diesel fuel data, these principal components shown in [Figure 11] would later be used in multivariate prediction using Deep Learning models, our analysis show that four principal components can preserve all (99.5%) of the variances in inflation. The four principal index components are Producer Price, Energy, Core Consumer and Food. The principal components in newer orthogonal spaces that explain the original subspace are described in [Table 3].

Table 3 Principal Components explaining the variance of inflation and fuel indices.

	Official Core Consumer Price Index	Energy Price Index	Food Price Index	Headline Consumer Price Index	Producer Price Index	Petrol	Diesel
PC1	0.3878	0.2032	0.40214	0.39804	0.4113	0.39866	0.40009
PC2	-0.23537	0.92123	-0.20529	-0.20317	0.02436	0.04387	0.09998
PC3	0.52682	0.27312	-0.10579	0.39867	0.03328	-0.50637	-0.46929
PC4	-0.36138	0.06842	0.72859	-0.1491	0.32195	-0.41757	-0.18337

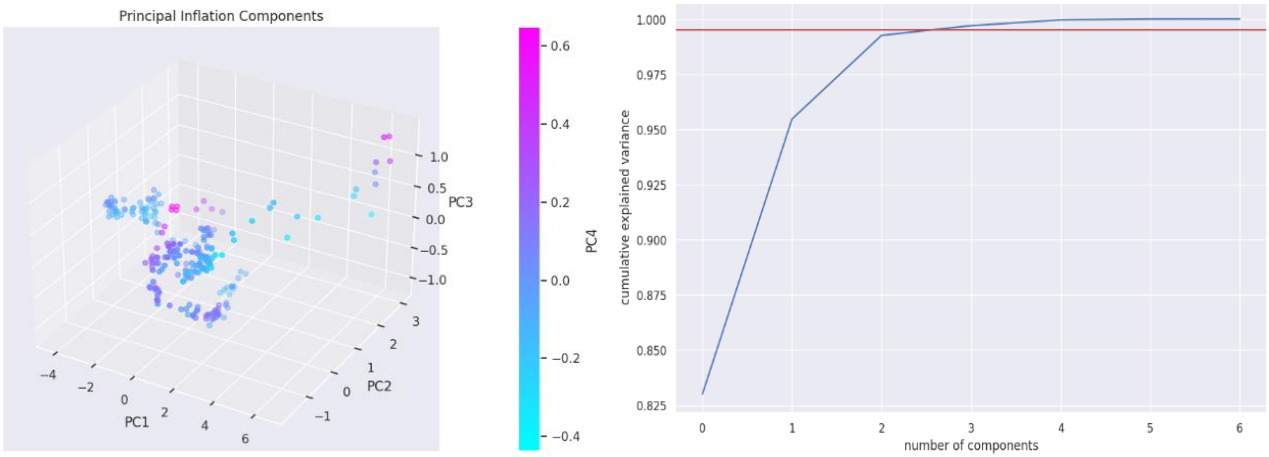


Figure 11 Three-dimensional view of first four components (left) and variance preserved graph (right)

6.2 Prediction Models

In our research project, we aim to forecast future index movements. To achieve this, we begin by selecting a basic model and gradually choose more complex models while examining its behaviour in response to inflation data to each model. We then employ defined metrics to evaluate each model's performance over a year-long period and ultimately selecting three models that demonstrate robust data responsiveness and capacity for accurate predictions over the next five years. These models are subjected to rigorous evaluation, allowing us to compare their performance against one another. [Figure 12] provides a visual representation of the models that we have identified as suitable for use in our current research project.

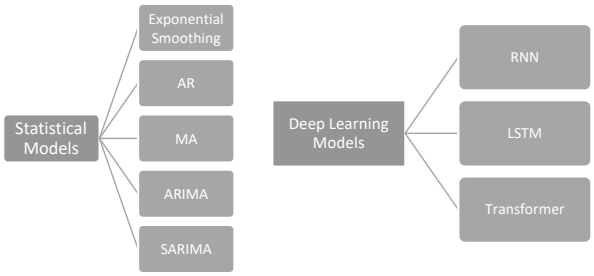


Figure 12 Statistical and Deep Learning models used in Inflation Forecasting

In the following sections, petrol price is used in the report to explain the performance of model, while other indices are added to appendix with relevant references.

6.2.1 Statistical Forecasting Models

Exponential smoothing is a simple yet effective method for time series forecasting that places more emphasis on recent data points while gradually reducing the impact of older ones. This approach is particularly useful for predicting inflation rates, which can be highly volatile and sensitive to economic events and policy decisions. However, the results [Figure 13] (Appendix 5.1) show that the underlying data generating process is stable over time and produces inaccurate forecasts in the presence of shocks.

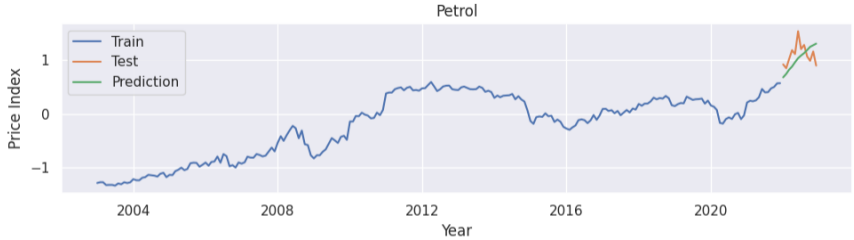


Figure 13 Exponential Smoothing Predictions

The next approach employed was the AR model, which is well-suited for datasets with strong patterns and commonly used for short-term forecasting. Autocorrelation analysis identified a significant trend, serving as the basis for short-term prediction. The dataset used for model fitting and prediction was stationary, which is a prerequisite for the AR model. The evaluation of the model indicated minimal mean square errors concerning the selected indices (Appendix 4.3), and it exhibited satisfactory predictive accuracy in the initial stages. However, as the prediction horizon extended, the model's accuracy decreased, as shown in [Figure 14(a)] (Appendix 5.2).MA model was the next method implemented with "q", the order of seasonality as 15. The results [Figure 14(b)] (Appendix 5.3) for this this method also begins to diverge after initial steps, as the average required to calculate the next steps depend on previous steps and they begin to lose accuracy for the newly predicted steps.

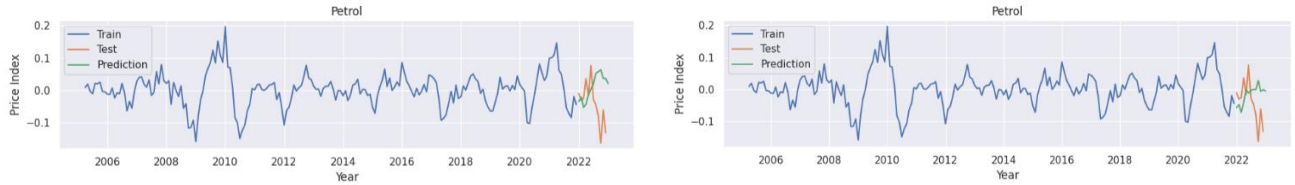


Figure 14 a) AR model prediction (left) b) MA model prediction (right)

ARIMA and SARIMA are the final models implemented as part of the statistical forecasting models. They can forecast on non-stationary data and is often used to make medium to long-term predictions. ARIMA was trained and predicted on stationary data while SARIMA on non-stationary data. The results [Figure 15] (Appendix 5.4, 5.5) for these methods show they yield good results when the data is linear but fail to capture sudden shocks which has been often the case in near past.

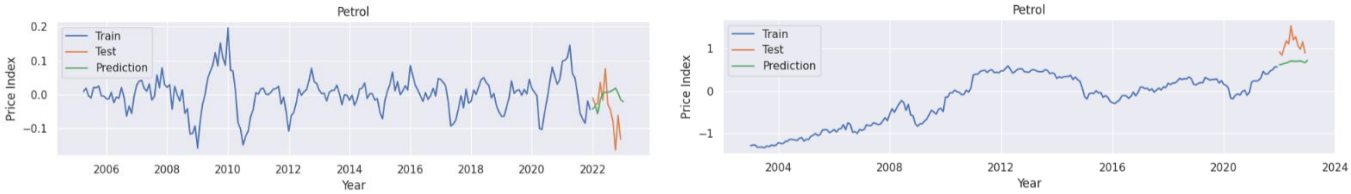


Figure 15 (a) ARIMA (left) and (b) SARIMA (right) Model Predictions

6.2.2 Deep Learning Forecasting Models

Deep learning models are advantageous in capturing intricate patterns in time series data, which may be challenging to identify using statistical models. This is attributed to their vast number of parameters, which can be fine-tuned to capture the nonlinear relationships existing between past and future data points.

The comparison of Recurrent Neural Network (RNN), Long-Short Term Memory (LSTM), and Transformer models in time series forecasting [Figure 16] (Appendix 5.6) has shown promising results compared to traditional statistical models. RNN models are effective in handling sequential data by passing information from one time step to the next. LSTM models, on the other hand, utilize a memory cell to maintain information for extended periods, thus enhancing their ability to capture long-term dependencies in the data. Transformer models, which were originally developed for natural language processing tasks, have also demonstrated their effectiveness in time series forecasting by leveraging attention mechanisms to identify important patterns in the data.

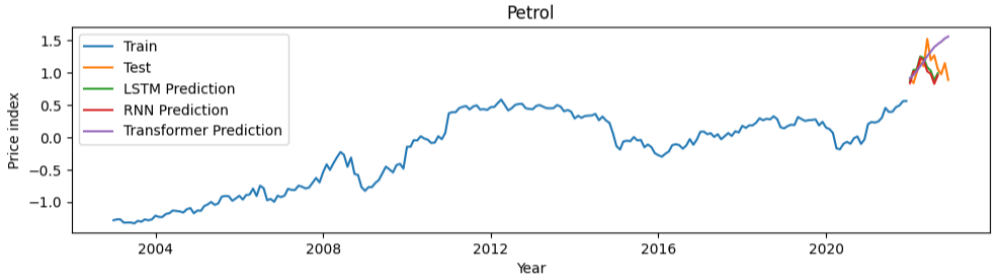


Figure 16 Deep Learning Model Predictions

A comparative investigation of loss functions [Figure 17] in deep learning models revealed a general tendency for transformer models to outperform RNN and LSTM models in predicting inflation. Interestingly, the LSTM

model exhibited an abnormally high loss for the Energy Price Index. This outcome can be attributed to the sudden fluctuations in energy prices that occurred after a decade of stability, particularly after 2020, which posed a challenge for the LSTM model in capturing such abrupt changes within small duration.

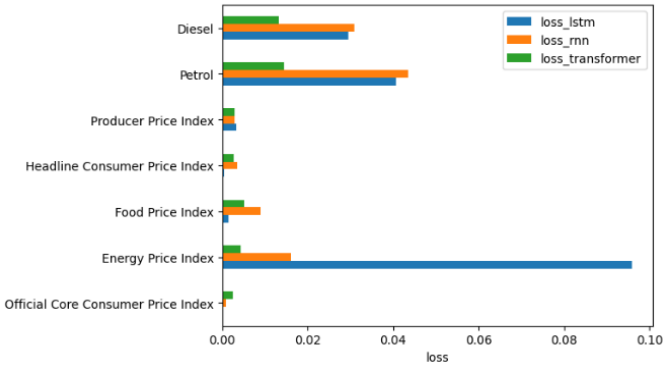


Figure 17 Bar plot to compare deep learning model losses.

The challenging and nonlinear nature of forecasting inflation indices can be difficult to quantify and predict accurately by traditional statistical methods. This complexity poses a challenge for traditional statistical models that may struggle to capture the intricate relationships and patterns in the data, leading to inaccurate predictions. A promising alternative lies in deep learning models, which can automatically learn intricate patterns and relationships in the data, making them better suited to forecast inflation. After comparing the outcomes with three deep learning models, it revealed that there will be a decrease in the prices of petrol [Figure 18] and diesel (Appendix 5.7) in the upcoming future. The recurrent neural network (RNN) and long short-term memory (LSTM) models anticipate a continuous decrease. However, the transformer model predicts an upward trend after three years due to the historical pattern of price increase in the past.

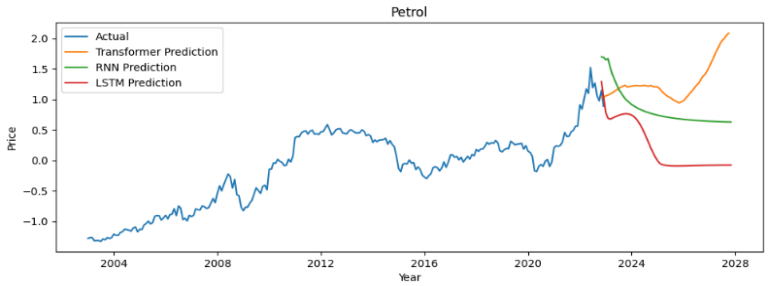


Figure 18 Five-year inflation forecasts by deep learning model

6.2.3 Performance Metrics

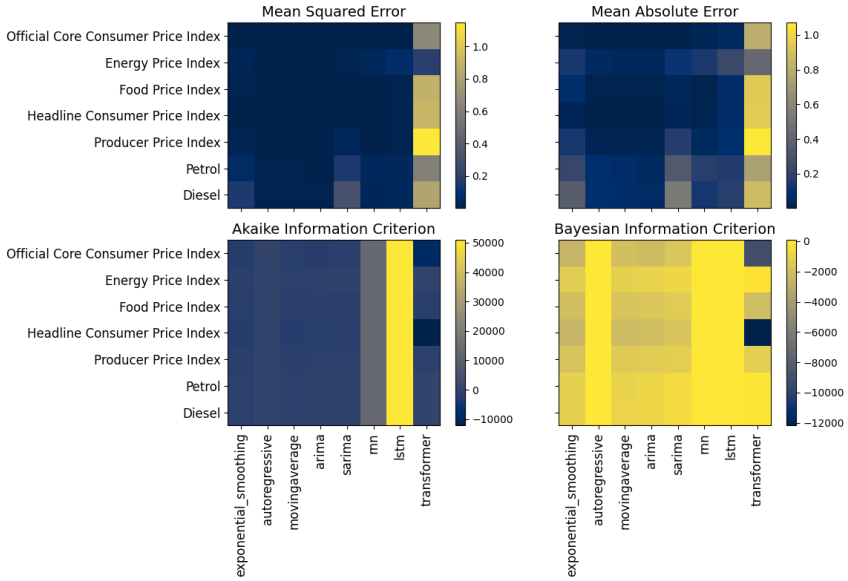


Figure 19 Performance metrics

In this study, four different metrics were employed, namely MSE, MAE, AIC, and BIC. The outcomes for each index are presented in Appendix 5.8, while the [Figure 19] provides an overview of the comparison across all the indices and metrics. The MSE and MAE metrics demonstrate that the values are comparatively lower for statistical models than for deep learning models. Although MSE and MAE are commonly used for assessing the goodness of a model, the AIC and BIC metrics are more appropriate for evaluating deep learning models. This is because they provide a more comprehensive measure of model performance that considers both accuracy and model complexity. In particular, the AIC and BIC values for RNN and LSTM models were relatively high indicating that these models are more prone to overfitting than the Transformer model.

7 Discussion

The first insight into our question; "What is the impact of the fuel index on other indexes in inflation forecasting in the UK" was through our correlation analysis results. The autocorrelation plots demonstrated a strong correlation between fuel indexes (diesel and petrol), and other indexes. After seeing these results, we wanted to understand how the autocorrelation plots' correlation values could change after utilising our predictive inflation data from each of our deep learning models.

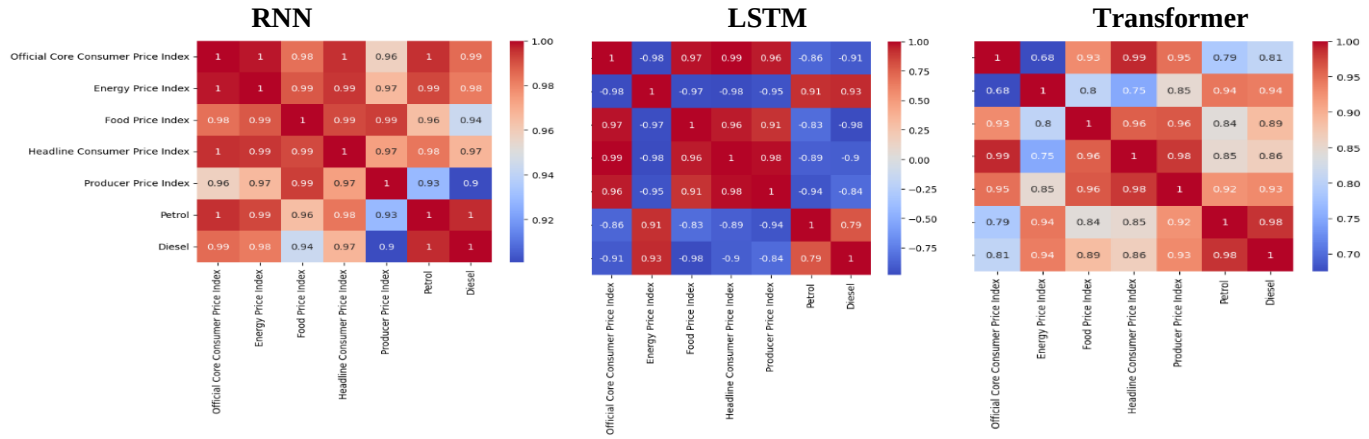


Figure 20 Autocorrelation results using prediction data of each Deep learning model; RNN (Left), LSTM (Middle), and Transformer model (Right)

An interesting observation made, was that the RNN data in the correlation plot (Figure 23) produced extremely high positive correlation results, the transformer model also only produced positive correlation results. Contrastingly, the LSTM model produced a greater proportion of negatively correlated results. A possible reason for this is that the LSTM prediction produced a smaller decrease in diesel and petrol prices for the timeseries prediction, whereas the comparison model for RNN, and Transformer model had a more significant decrease in future fuel prices, to roughly the levels seen in 2020. In addition to this, energy price index seems to have a negative correlation between producer price index, headline consumer price, food price and consumer price. Whereas energy price index is positively correlated with petrol and diesel indicating the fact that energy price index could be overlaying with these fuel indexes.

Interestingly, the transformer model has the worst MSE and MAE metrics of all the models. At first, these results are counterintuitive, as typically higher error suggests poorer performance. However, this could be due to the transformer model being overly complex, leading to greater difficulty in optimising the model. Alternatively, a possible reason for such high errors in these metrics of the transformer model could be due to overfitting. On the topic of overfitting, the LSTM produced higher errors in MSE and MAE than the RNN. This is not what we expected since LSTMs are able to handle long term dependencies better (explained in the LSTM section). We put this down to the LSTM requiring more data than the data we supplied from 2003-2022. LSTMs require larger quantities of data; therefore, it is plausible that the LSTM overfitted the data we supplied.

When trying to answer our question and gain the most valuable insight into the impact of fuel prices on other indexes in inflation forecasting, it was important we chose the best neural network for analysis. We decided that although the transformer model produced the worst metrics for MAE and MSE, it produced by far the

best AIC and BIC metrics (Appendix 5.8). As a result, we went ahead with the transformer model to visualise the impact of fuel index on other indexes. The AIC and BIC values for the transformer model are negative values, whereas the RNN and LSTM produced positive AIC and BIC values. The lower value for AIC and BIC does not equate to a better model, but the fact that the difference between the transformer model and other models AIC and BIC values is greater suggests the transformer model is a better fit.
Moreover, in the first paper to introduce the transformer model [41], the transformer outperformed the RNN model in language translation tasks. Notably, our transformer model utilised a seq2seq architecture allowing it to process more complex data, such as our inflation data which tends to hold seasonality and can be affected by drastic changes to the economic climate. Whilst the architecture of the transformer model differs to the LSTM, it can handle long term dependencies through its attention mechanism. The self-attentive mechanism within a transformer disregards analysis in a chronological order, allowing the model to dispel chances of suffering the vanishing gradient problem. This allows the transformer to perform better than the RNN model, in addition, it is impossible to allow parallelization within an RNN but is possible in transformer models.

Although Transformer models are a new and exciting architecture, the fact that Transformer models still have not been explored in great depth, creates a black box of information to be uncovered. Transformer models typically have a larger black box than LSTM and RNN models, but of course this depends on the size of neural network, such as the number of layers. Therefore, although the size of the black box may differ from task to task, the Transformer model was certainly the most elusive model in our project. The extremely complex architecture and large number of hyperparameters, including the number of attention heads, and the dropout rate demonstrates the difficulty to tune the model to increase performance. Therefore, if we were trying to improve our Transformer model we would increase the number of attention heads, or experiment using different hyperparameters, such as changing the batch size. The reason we were unable to do these alterations was due to the time constraints of our project, as well as the high computational requirement, and large memory requirements.
Focussing our attention onto the LSTM correlation values that were gathered, the highest correlation values were found between the producer price index (PPI) and the rest of the indexes (core consumer price (CCP), energy price, food price, consumer price, petrol, and diesel). PPI measures wholesale inflation and is calculated based on the changes in prices paid to producers of goods and services. Fuel prices directly impact transportation, energy, and manufacturing expenses, as well as the overall supply chain costs. Therefore, since rising fuel prices increases the transportation expenses, it results in a surge of manufacturing costs of transferring raw materials, components, and finished goods from one location to another, hence directly affecting PPI. The two indexes least correlated with petrol and diesel appears to be CCP index, a 0.79 correlation value with petrol and 0.81 correlation value with diesel. Therefore, suggesting that fuel index has least impact on the ability to forecast CCP than other indexes.

8 Dashboard

Our solution is presented in the form of an interactive dashboard (link) implemented in plotly dash that allows users to visualise inflation data for over 70 countries from 2003-2023, as well as view forecasts for future inflation rates in the users' desired country (Appendix 6.1). Users are also granted the ability to decide on a specific period that they may be interested in.
Upon entering the site, users can select a year range using a sliding tool to view the different inflation indices in the form of line plots, histograms, violin plots, and pair plots (Appendices 6.3, 6.4, 6.5). They can then select a specific index and again select year range to view the time series decomposition plot (Appendix.6.6), which displays the trend, seasonal plot, and residual plot. Furthermore, users are presented with a 3D map of the world (Appendix.6.7) where they can select a country and assess the inflation indices for a chosen month up to 2022 and index price (Appendix.6.8). Lastly, users can use the inflation prediction tool to see the forecast within a specified date range and can also select the degree of precision from low, medium, and high that is preferable.
We achieved this firstly by filtering all the countries within the original data set that had all the information required to produce all the illustrations. HTML is used to create the drop-down list for the available countries, the month/year selection slider and general interface design of the dashboard. A 'call back' function from the

dash module is used so that plots (e.g., line, pair etc) only need to be implemented once and it applies the same plot for each chosen country and date range and updated in real-time when the user changes options. This also applies for the time series decomposition and the world map. Additionally with the world map an if else is used to decide what to display when a certain index is chosen. For the prediction plots, LSTM model is implemented.
Additionally, the dashboard has been deployed on Google Cloud Platform that can be accessed here. All the plots available on deepnote can also be visualised in GCP except the prediction model due to memory constraints in the free tier of GCP service.
Overall, the site provides interactive visualisations, making it an ideal resource for students, academics, and other people that show interest in this field.

9 Conclusion

Inflation is a multifaceted phenomenon that is affected by a variety of factors including interest rates, consumer spending, exchange rates and government policies. These factors often have intricate relationships with each other making it difficult to accurately model their impact on inflation. The present study aimed to forecast inflation indices and fuel prices for the next five years to understand how fuel prices affect each individual inflation parameter. Multiple indices were considered and predicted to comprehend the effects of fuel prices on these indices. After extensive data preparation, we applied eight statistical and machine learning models to the dataset. The models predicted a decline in fuel prices in the near future after a historical rise in the past couple of years, while other indices were expected to follow similar trends due to their strong correlation with fuel prices. These analyses and predictions hold true only if the underlying dynamics remain stable over time. Based on this study, we suggest two recommendations. First, fuel prices must be controlled because prices and inflationary indices are directly dependent on fuel prices, and volatility in fuel prices could translate to volatility of the entire economy. Second, governments should promote the use of alternative energy sources such as electric vehicles, biofuels, and renewable energy to reduce demand for fossil fuels and help stabilize prices. Currently, the energy index has the highest correlation, ranging from 0.9 to 0.99, showing direct dependence on fossil fuels. Decreasing dependence on fossil fuels could be helpful in cushioning the volatility of fuel markets.

9.1 Scope for further research

It is evident that Long Short-Term Memory (LSTM) models excel in capturing temporal dependencies in time series data while transformer models are known for their ability to parallel process information. Therefore, to achieve more accurate predictions of inflation data an ensemble model can be developed that balances these two factors. By combining the strengths of both LSTM and transformer models an enhanced ensemble model can be created that can more accurately predict inflation data
While in our study, we incorporated wide range of indices that get factored in while inflation prediction, additional factors such as social trends, and weather patterns can be included into time series models that can help improve their accuracy and make them more useful for decision-making.
For the deep learning models, more studies could be done on hyperparameter tuning. In the current research project hyperparameters have been tuned on learning rate and number of epochs, these could be expanded to include multiple types of optimisers, the number of connected layers, increasing the number of stages. For transformer model, hyperparameters such as the number of layers, the number of attention heads, the dropout rate, the learning rate, and the batch size can be tuned to improve accuracy.

10 References

[1] J. Fernando, “Inflation,” *Investopedia*, Mar. 14, 2023. Available: https://www.investopedia.com/terms/i/inflation.asp%20

[2] D. Meyer, “The Impact of Changes in Fuel Prices on Inflation and Economic Growth in South Africa,” Nov. 2018, doi: https://doi.org/10.5281/zenodo.1569053%20.

[3] C. Bermingham, “Quantifying the Impact of Oil Prices on Inflation Quantifying the Impact of Oil Prices on Inflation,” Jan. 2009. Available: https://www.centralbank.ie/docs/default-source/publications/quarterly-bulletins/quarterly-bulletin-signed-articles/quantifying-the-impact-of-oil-prices-on-inflation.pdf%20

[4] K. Kpodar and B. Liu, “The distributional implications of the impact of fuel price increases on inflation,” *Energy Economics*, vol. 108, no. 1–2, p. 105909, Apr. 2022, doi: https://doi.org/10.1016/j.eneco.2022.105909.

[5] G. Przekota, “Do High Fuel Prices Pose an Obstacle to Economic Growth? A Study for Poland,” *Energies*, vol. 15, no. 18, p. 6606, Sep. 2022, doi: https://doi.org/10.3390/en15186606.

[6] P. Bolton, “Petrol and diesel prices,” *House of Commons Library Briefing Paper no. 4712*, Mar. 31, 2023.

[7] ONS, “CPIH ANNUAL RATE 00: ALL ITEMS 2015=100 - Office for National Statistics,” *Ons.gov.uk*, Feb. 15, 2023. Available: https://www.ons.gov.uk/economy/inflationandpriceindices/timeseries/l55o/mm23

[8] Office for National Statistics, “Cost of Living Insights - Office for National Statistics.” Www.ons.gov.uk, 2023. Available: https://www.ons.gov.uk/economy/inflationandpriceindices/articles/costoflivinginsights/energy.

[9] A. Atkeson and L. E. Ohanian, “Are Phillips Curves Useful for Forecasting Inflation?” *Quarterly Review*, vol. 25, no. 1, Dec. 2001, doi: https://doi.org/10.21034/qr.2511.

[10] A. Almosova and N. Andresen, *Nonlinear Inflation Forecasting with Recurrent Neural Networks,* 2019. Available: https://www.ecb.europa.eu/pub/conferences/shared/pdf/20190923_inflation_conference/L2_Almosova.pdf.

[11] H. Jongrim; K. M. Ayhan; O. Franziska, "One-Stop Source: A Global Database of Inflation." Policy Research Working Paper; No. 9737. World Bank, Washington, DC.", 2021.

[12] F.S. Mishkin, “The economics of money, banking, and financial markets”. Pearson Education, 2007.

[13] B. Fattouh, L. Kilian, and L. Mahadeva, “The Role of Speculation in Oil Markets: What Have We Learned so Far?”, *The Energy Journal*, vol. 34, no. 3, pp. 7–33, Jul. 2013, doi: https://doi.org/10.5547/01956574.34.3.2

[14] FAO, “Food price indices”, Food and Agriculture Organization of the United Nations, 2022.

[15] O. Blanchard and D. R. Johnson, *Macroeconomics*, 6th ed. Boston: Pearson, 2013.

[16] U.S. Bureau of Labor Statistics. Producer Price Indexes - October 2021, Oct. 2021. Available: https://www.bls.gov/ppi/

[17] Department for Energy Security and Net and Department for Business, Energy & Industrial Strategy, “Road Fuel Price Statistics Providing Average UK Retail ‘pump’ Prices on a Weekly Basis,” 2022. Available: https://www.gov.uk/government/statistics/weekly-road-fuel-prices (accessed Feb. 13, 2023).

[18] M. Loretan, & P. C. Phillips, “Testing the covariance stationarity of heavy-tailed time series: An overview of the theory with applications to several financial datasets”, *Journal of empirical finance*, 1(2), 211-248, 1994.

[19] D. A. Dickey and W. A. Fuller, “Distribution of the Estimators for Autoregressive Time Series with a Unit Root,” *Journal of the American Statistical Association*, vol. 74, no. 366a, pp. 427–431, Jun. 1979, doi: https://doi.org/10.2307/2286348

[20] J. Shlens, “A Tutorial on Principal Component Analysis.”

[21] J. Leban, “Theory of Principal Component Analysis (PCA) and Implementation on Python,” May 18, 2020. Available: https://towardsdatascience.com/theory-of-principal-component-analysis-pca-and-implementation-on-python-5d4839f9ae89%20 (accessed Apr. 10, 2023).

[22] Z. Jaadi and B. Whitfield, “A Step-by-Step Explanation of Principal Component Analysis (PCA),” Aug. 08, 2022.

[23] E. S. Gardner, “Exponential smoothing: the State of the Art,” *Journal of Forecasting*, vol. 4, no. 1, pp. 1–28, 1985, doi: https://doi.org/10.1002/for.3980040103.

[24] B. Billah, M. L. King, R. D. Snyder, and A. B. Koehler, “Exponential smoothing model selection for forecasting,” *International Journal of Forecasting*, vol. 22, no. 2, pp. 239–247, Apr. 2006, doi: https://doi.org/10.1016/j.ijforecast.2005.08.002

[25] R. S. Tsay, “Analysis of Financial Time Series”, *Wiley*. pp. 24-97, 2010.

[26] H. Akaike, “A Bayesian analysis of the minimum AIC procedure,” *Annals of the Institute of Statistical Mathematics*, vol. 30a, no. 1, pp. 9–14, Dec. 1978, doi: https://doi.org/10.1007/bf02480194

[27] C. B. A. Satrio, W. Darmawan, B. U. Nadia, and N. Hanafiah, “Time series analysis and forecasting of coronavirus disease in Indonesia using ARIMA model and PROPHET,” *Procedia Computer Science*, vol. 179, p. 529, Jan. 2021, doi: https://doi.org/10.1016/j.procs.2021.01.036

[28] R. J. Hyndman and G. Athanasopoulos, *Forecasting: Principles and Practice*, 2nd ed. Heathmont, Vic.: Otexts, 2018, pp. 221–229. Available: https://otexts.com/fpp2/stationarity.html

[29] K. Foo, “Seasonal lags: SARIMA modelling and forecasting,” *Medium*, Jan. 03, 2018. Available: https://medium.com/@kfoofw/seasonal-lags-sarima-model-fa671a858729

[30] G. Petneházi, “Recurrent Neural Networks for Time SeriesForecasting,” Doctorate, University of Debrecen, 2019. Accessed: Apr. 26, 2023. [Online]. Available: https://www.researchgate.net/publication/330102696_Recurrent_Neural_Networks_for_Time_Series_Forecasting

[31] K. Bandara, C. Bergmeir, and S. Smyl, “Forecasting Across Time Series Databases using Recurrent Neural Networks on Groups of Similar Series: A Clustering Approach,” *arXiv:1710.03222 [cs, econ, stat]*, vol. 140, Sep. 2018, Accessed: Apr. 24, 2023. [Online]. Available: https://arxiv.org/pdf/1710.03222.pdf

[32] J. Zhang, Y. Zeng, and B. Starly, “Recurrent neural networks with long term temporal dependencies in machine tool wear diagnosis and prognosis,” *SN Applied Sciences*, vol. 3, no. 442, Mar. 2021, doi: https://doi.org/10.1007/s42452-021-04427-5

[33] S. Hochreiter, “The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions,” *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 06, no. 02, pp. 107–116, Apr. 1998, doi: https://doi.org/10.1142/s0218488598000094

[34] F. A. Gers, J. Schmidhuber, and F. Cummins, “Learning to Forget: Continual Prediction with LSTM,” *Neural Computation*, vol. 12, no. 10, pp. 2451–2471, Oct. 2000, doi: https://doi.org/10.1162/089976600300015015

[35] A. Sherstinsky, “Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network,” *Physica D: Nonlinear Phenomena*, vol. 404, p. 132306, Mar. 2020, doi: https://doi.org/10.1016/j.physd.2019.132306

[36] A. Vaswani *et al.*, “Attention Is All You Need,” Jan. 2017, Available: http://arxiv.org/abs/1706.03762

[37] K. Doshi, “Transformers Explained Visually (Part 3): Multi-head Attention, deep dive,” *Towards Data Science*, Jan. 17, 2021. Available: https://towardsdatascience.com/transformers-explained-visually-part-3-multi-head-attention-deep-dive-1c1ff1024853%20 (accessed Apr. 26, 2023). P7

[38] K. Doshi, “Transformers Explained Visually (Part 2): How it works, step-by-step,” *Towards Data Science*, Jan. 02, 2021. https://towardsdatascience.com/transformers-explained-visually-part-2-how-it-works-step-by-step-b49fa4a64f34 (accessed Apr. 26, 2023). P6

[39] K. Doshi, “Transformers Explained Visually (Part 1): Overview of Functionality,” *Towards Data Science*, Dec. 13, 2020. https://towardsdatascience.com/transformers-explained-visually-part-1-overview-of-functionality-95a6dd460452 (accessed Apr. 26, 2023). P8

[40] M. West, “Miscellanea. Time Series Decomposition,” *Biometrika*, vol. 84, no. 2, pp. 489–494, Jun. 1997, doi: https://doi.org/10.1093/biomet/84.2.489%20

[41] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

11 Group work

The team worked very well together. The team had no issues or any major disagreements during the project. Each member of the group showed initiative and effort to help when we were faced with a problem, and each of us would offer our opinion on alternative solutions. Additionally, every member demonstrated resourcefulness in finding solutions to our problems using the literature plus internet resources. For example, a difficulty in coding the transformer model was faced, but we were able to fix this. To conclude, the team showcased a massive amount of resilience and determination, which helped us to succeed as a team.

Communication was frequent between us all on “Teams”, as well as on WhatsApp. We would hold weekly meetings, if not multiple meetings a week. In each meeting, we would help each other out on each other’s respective sections, if help was needed.

The group kept organised throughout the project using a Gantt chart. This enabled us to finish our solution to the project prior to the end of semester, allowing us to have a break at the end of semester. This level of organisation and motivation helped the team to complete work on time.

We are delighted with the calibre of work that has been produced and sincerely believe that it is a testament to the excellent standards upheld by our group.

12 Individual contributions

12.1 – Farah Yesilkaya

My contributions for the project are as follows:

1. General Exploratory Data Analysis (EDA): I helped to produce data visualisations with the rest of the group, such as the histograms, and line plots. This enabled us to understand the dataset better.

2. Principal Component Analysis: Coded the PCA section in Deepnote, I found the most important indexes in our data. To do this, it involved computing the covariance matrix, and also finding the corresponding eigenvalues and eigenvectors.

3. LSTM: I worked with Vijay Jawali to help code the LSTM deep learning model to do deep learning time series forecasting of the inflation data. The LSTM was able to deal with our vanishing gradient problem that the RNN faced, allowing us to have a greater insight into the future predictions of inflation.

4. Organising the Team: Helped with organising meetings, contacting TA's with issues, and submitting work. As well as this I made notes to help our groups understanding of tasks in each meeting.

The following sections were my contributions for the report:

Section 1.2 – Relevant works
Section 5.2.3 – Moving Average Model
Section 5.3.2 – LSTM section
Section 7 – Discussion

My assessment of the group is as follows:

I think that everyone was determined to do well, and this was demonstrated by the high level of determination and contributions that everyone made,
I am very thankful for my team, our group dynamic was excellent, there was never any friction, and communication was never an issue.
I am grateful for everyone on my team for their hard work, and continued dedication right from the start of the project to the very end.

12.2 – Priyanshu Jha

My contributions for the project are as follows:

Contributions were made in data preparation, general exploratory data analysis (EDA), and the application of the Transformers model during the duration of this project. The contributions listed below are:

1. Data Preparation: Oversaw gathering, processing, and cleansing the raw data to get it ready for analysis. In addition, found and fixed data quality problems to guarantee the data's accuracy and completeness. Overall, the effort put forth to prepare the data ensured that it was fit for analysis.

2. General Exploratory Data Analysis (EDA): Conducted the exploratory data analysis to gain insights into the data and identify patterns or trends. Have used a range of data visualisation techniques, such as scatterplots, boxplots, and histograms, to visualise the data. EDA's work aided in improving data comprehension and provided information for further modelling choices.

3. Transformers Model Implementation: Worked with Vijay and Moronfolu to implement the Transformers model using Python and PyTorch. Developed scripts to train, validate, and test the model, it also included the tuning of hyperparameters of the model, such as the number of layers and the learning rate, to optimize model performance. The work on the Transformers model has made it possible to classify data with a great deal of accuracy.

The following sections were my contributions for the report:

Section 5.1.1 – Principal Component Analysis
Section 5.2.5 – SARIMA model
Section 5.3.3 – Transformer Model

My assessment of the group is as follows:

Overall, team showed a high level of cooperation throughout the project. Everyone worked well together and were willing to help each other out when needed. Overall, team showed great communication as we had regular meetings about the project and completed the project with good coordination.

12.3 – Moronfolu Durodola

My contributions for the project are as follows:

1. Organisation: At the start of the project, I was responsible for creating the Gantt chart. I then utilised this to ensure that we remained on track throughout the process and updated it where necessary. Alongside this I also ensured we were on track for the progress check and mid-term demonstration. Doing this made certain that we completed all aspects in a timely fashion.

2. SARIMA: I contributed at the initial stage of implementing the SARIMA model which mainly consisted of getting the train and test sets and defining the model.

3. Transformer model: The transformer model was implemented by Priyanshu and me as part of the deep learning models to forecast inflation as accurately as possible. I defined the encoder-decoder function, and also trained, validated, and tested the model using our data. I also worked on the visualisations involved with this model.

4. Model evaluation: At this stage, I was mainly involved with creating the plots to compare the different results of the models as well as compare the different indexes. This was useful for us to analyse which model produced the best outcome.

The following sections were my contributions for the report:

Section 1 – Introduction
Section 1.1 – Question development
Section 3.2 – Timeseries Data Visualisations
Section 5.2.4 – ARIMA Model
Section 5.3.1 – Recurrent Neural Network
Section 8 – Dashboard

My assessment of the group is as follows:

Overall, I enjoyed working within this group as each member demonstrated a high level of work ethic throughout this process which also encouraged me to make every effort in completing this project. Our frequent interactions made it easier to track our progress as well as support each other where needed.

12.4 – Vijay Jawali

My contributions for the project are as follows:

1. At data retrieval stage, my contributions included discovering and collecting source data for inflation and fuel data from world bank and U.K Govt websites.
2. My contributions in EDA stage consisted of timeseries data visualisation such as plotting Autocorrelation and Partial Autocorrelation plots, Monthly and Quarterly plots and Timeseries decomposition plots.
3. In the data preparatory stage, I was responsible for data validation, conducting parametric and non-parametric stationarity tests and converting non-stationary data to stationary.
4. In the modelling stage, I implemented Exponential smoothing model, AR model, MA model, ARIMA model and SARIMA model in statistical modelling. For Deep Learning model, I was responsible for Implementing RNN model. Additionally, I was responsible for implementing performance metrics for all models implemented.
5. For the dashboard, I was responsible for implementing country selection, year range selection, line plot, histogram, violin plot, pair plot, time series decomposition plot, Inflation Forecasting using LSTM on plotly and deployment of dashboard on Google Cloud Platform using docker.
6. Assisted Priyanshu and Rhoda in Implementation of Inflation Forecasting using Transformer model along with Hyperparameter tuning.
7. I was responsible for implementing Model Evaluation notebook and metrics visualisations.

The following sections were my contributions for the report:

Section 2 – Data
Section 3 – Stationarity
Section 5.2.1 – Exponential Smoothing
Section 5.2.2 – Statistical Models (AR)
Section 5.4 – Performance Metrics
Section 6 – Experimental Results
Section 9 – Conclusion

My assessment of the group is as follows:

We had regular meetings on team apart from TA meetings and everyone was an active participant.
We had open and frank discussions on every topic in the meeting, additionally we also had a group chat as means to communicate.
Team was fluid and open to change course whenever it was needed, contrary points were encouraged and discussed with utmost seriousness, and I appreciate the team for having professional courtesy during all meeting.
We prepared a Gantt chart at the beginning of project and every member of the team had their work completed with dedication.
I am thankful to T.A and team for their guidance during the entire time, they helped me improve my skills, especially during the project report preparation.

Appendices

Appendix 0

Link to Deepnote Project: https://deepnote.com/workspace/andromeda-data-science-group-project-c0ac1109-a8e0-4436-b4de-5b1043d1159a/project/Data-Science-Group-Project-638889fe-2546-41ac-8d2f-f9d8b29d307c

GitHub Link: https://github.com/vijayjawali/Andromeda.git

Link to Dashboard: https://638889fe-2546-41ac-8d2f-f9d8b29d307c.deepnoteproject.com/

Link to Dashboard on GCP: https://andromeda-dashboard-2ms2zpkv2q-ew.a.run.app/
source code: https://github.com/vijayjawali/andromeda-dashboard.git

Steps to Initiate Dashboard on Deepnote:

a) Navigate to Andromeda project on Deepnote.
b) Navigate to Notebook named Dashboard.
c) Clear the existing variables.
d) Run the whole notebook.
e) Click on dashboard link.

Appendix 1

Appendix 1.1

```
ccpiData = pd.read_csv('ccpi_m.csv')
ccpiData = ccpiData[ccpiData['Country Code'] =='GBR'].iloc[:,4:]
ecpiData = pd.read_csv('ecpi_m.csv')
ecpiData = ecpiData[ecpiData['Country Code'] =='GBR'].iloc[:,4:]
fcpiData = pd.read_csv('fcpi_m.csv')
fcpiData = fcpiData[fcpiData['Country Code'] =='GBR'].iloc[:,4:]
hcpiData = pd.read_csv('hcpi_m.csv')
hcpiData = hcpiData[hcpiData['Country Code'] =='GBR'].iloc[:,4:]
ppiData = pd.read_csv('ppi_m.csv')
ppiData = ppiData[ppiData['Country Code'] =='GBR'].iloc[:,4:]
```

Appendix 1.2

```
rawDF = pd.concat([ccpiData, ecpiData, fcpiData, hcpiData, ppiData])
rawDF = rawDF.T
rawDF.rename(columns=rawDF.iloc[0,:], inplace = True)
rawDF = rawDF.tail(-1)
rawDF.drop(rawDF.tail(1).index,inplace=True)
time = pd.DatetimeIndex([i[:-2]+'-'+i[-2:] for i in rawDF.index])
rawDF = rawDF.set_index(time)
rawDF['Energy Price Index'] = pd.to_numeric(rawDF['Energy Price Index'])
df = rawDF.fillna(method='ffill').fillna(method='bfill')
```

Appendix 1.3

```
fig, ax = plt.subplots(nrows=2, ncols=2, sharex=True, sharey=False)
fig.set_size_inches(18, 15)
sns.lineplot(normalized_df, ax=ax[0][0])
```

```
sns.lineplot(df_zscore, ax=ax[0][1])
sns.lineplot(df_scaled, ax=ax[1][0])
sns.lineplot(df_robust, ax=ax[1][1])
ax[0, 0].set_title("Min Max Approach")
ax[0, 1].set_title("The z-score method")
ax[1, 0].set_title("Maximum Absolute Scaling")
ax[1, 1].set_title("The Robust Scaling")
fig.tight_layout()
```

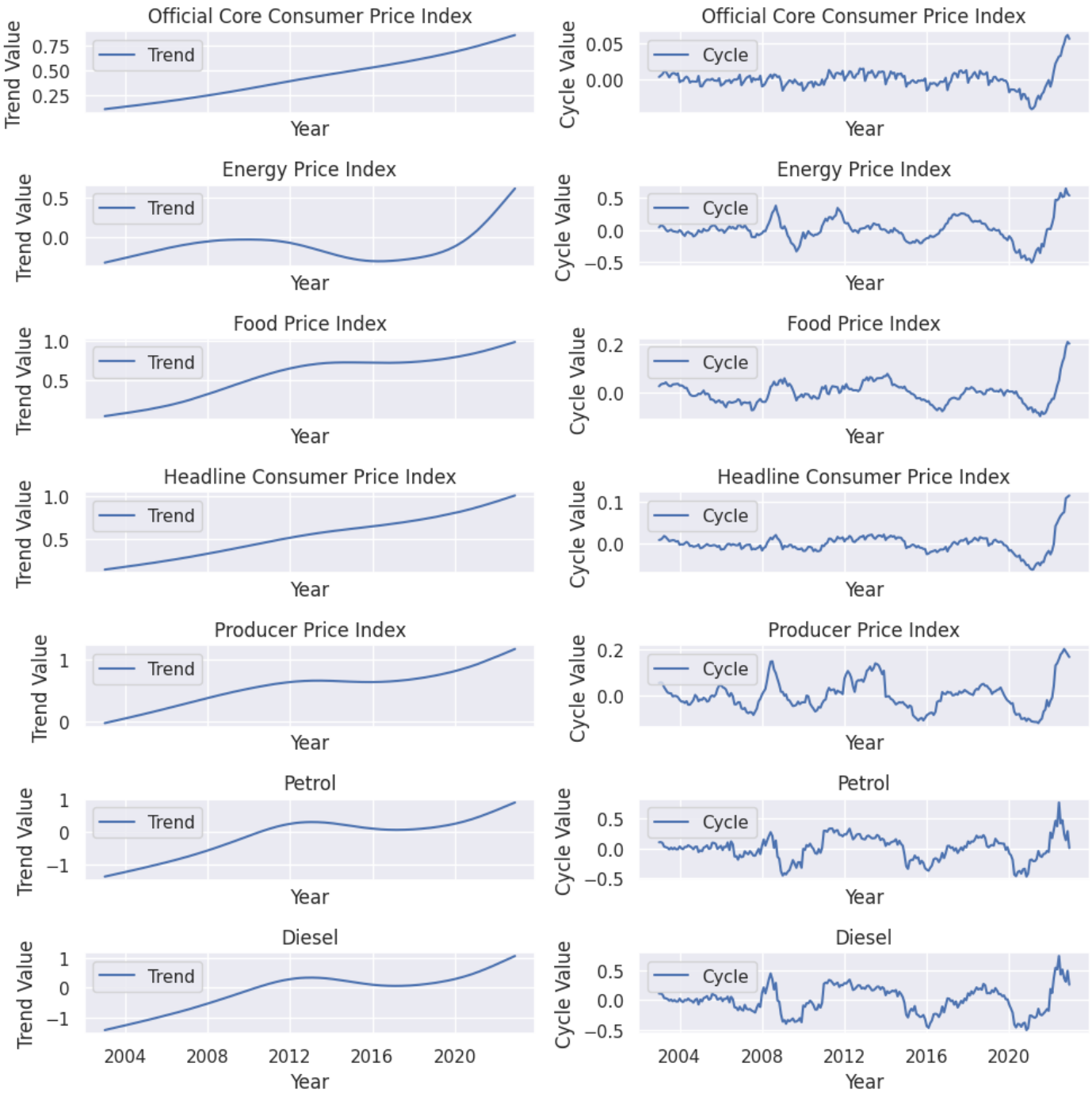
Appendix 1.4

```
fuelData = pd.read_csv('fuel.csv').iloc[:,:7]
petrolData = fuelData.iloc[:,1::2]
dieselData = fuelData.iloc[:,2::2]
petrolData.index = fuelData['Weekly Prices time series'].iloc[0::]
dieselData.index = fuelData['Weekly Prices time series'].iloc[0::]
petrolData.rename( columns={i:j for i,j in zip(petrolData.columns, ['Pump Price','Duty Rate', 'Vat']) }, inplace=True)
dieselData.rename( columns={i:j for i,j in zip(dieselData.columns, ['Pump Price','Duty Rate', 'Vat']) }, inplace=True)

petrolData = (petrolData.tail(-2)).apply(pd.to_numeric)
dieselData = dieselData.tail(-2).apply(pd.to_numeric)
dieselData['Total'] = dieselData['Pump Price']+dieselData['Duty Rate']+dieselData['Vat']
petrolData['Total'] = petrolData['Pump Price']+petrolData['Duty Rate']+petrolData['Vat']
dieselData = dieselData.set_index(fuelData['Weekly Prices time series'].iloc[2::])
petrolData = petrolData.set_index(fuelData['Weekly Prices time series'].iloc[2::])

finalFuelData = pd.DataFrame({'Petrol':petrolData['Total'], 'Diesel':dieselData['Total']})
finalFuelData = finalFuelData.set_index(pd.DatetimeIndex(finalFuelData.index))

for column in finalFuelData.columns:
    finalFuelData[column] = (finalFuelData[column] - finalFuelData[column].median())  / (finalFuelData[column].quantile(0.75) -
finalFuelData[column].quantile(0.25))
finalFuelData.head()
```

Appendix 2

Appendix 2.1

```
for col in dataGreaterthan2003.columns:
    cycle, trend = sm.tsa.filters.hpfilter(dataGreaterthan2003[col], lamb=129600)
    results_hp[f"{col}_trend"] = trend
    results_hp[f"{col}_cycle"] = cycle
```



Appendix 2.2

```
for col in dataGreaterthan2003.columns:
    results = adfuller(dataGreaterthan2003[col])
    results_df = results_df.append({'Column': col,
                                    'ADF_Stat': results[0],
                                    'p-value': results[1],
                                    'Lags': results[2],
                                    'Observations': results[3],
                                    'Crit-1%': results[4]['1%'],
                                    'Crit-5%': results[4]['5%'],
                                    'Crit-10%': results[4]['10%']},
                                    ignore_index=True)
```

Column	ADF_Stat	p-value	Lags	Obs	Crit-1%	Crit-5%	Crit-10%

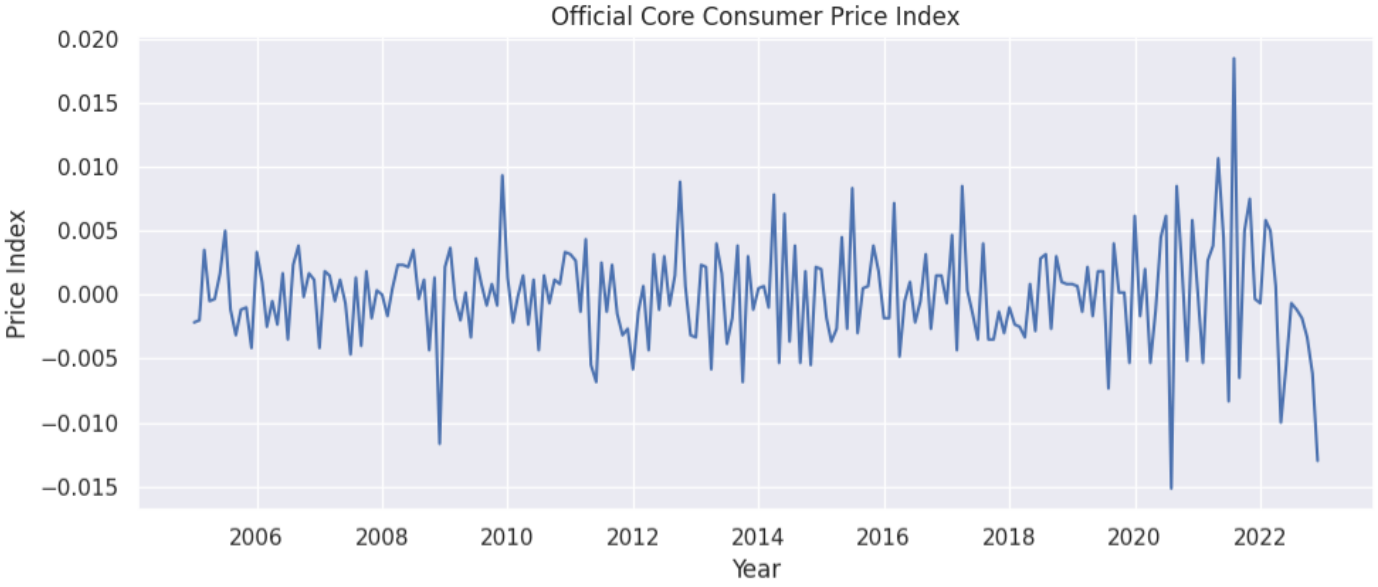
Official Core Consumer Price Index	1.42194	0.99721	15	224	-3.45988	-2.87453	-2.57369
Energy Price Index	-1.24488	0.65399	12	227	-3.45949	-2.87435	-2.57360
Food Price Index	-0.13536	0.94580	15	224	-3.45988	-2.87453	-2.57369
Headline Consumer Price Index	1.15247	0.99564	15	224	-3.45988	-2.87453	-2.57369
Producer Price Index	-0.02545	0.95637	3	236	-3.45836	-2.87386	-2.57333
Petrol	-1.15950	0.69077	7	232	-3.45885	-2.87408	-2.57345
Diesel	-1.03561	0.74005	7	232	-3.45885	-2.87408	-2.57345

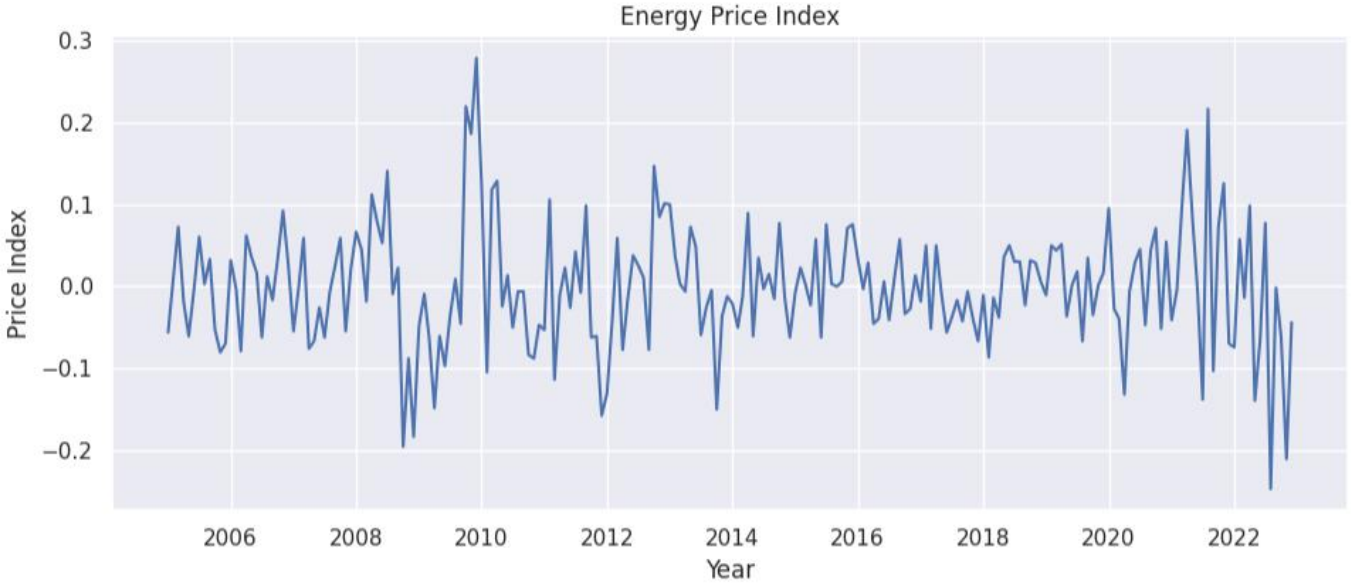
```
hypothesisclassificationconditions = [
    ((results_df['p-value'] <= 0.05) & (results_df['ADF_Stat'] <= results_df['Crit-5%'])),
    ((results_df['p-value'] > 0.05) & (results_df['ADF_Stat'] > results_df['Crit-5%']))
    ]

hypothesisclassificationvalues = ['Rejected', 'Accepted']
```

Index	null_hypothesis	Conclusion
Official Core Consumer Price Index	Accepted	Non-Stationary
Energy Price Index	Accepted	Non-Stationary
Food Price Index	Accepted	Non-Stationary
Headline Consumer Price Index	Accepted	Non-Stationary
Producer Price Index	Accepted	Non-Stationary
Petrol	Accepted	Non-Stationary
Diesel	Accepted	Non-Stationary

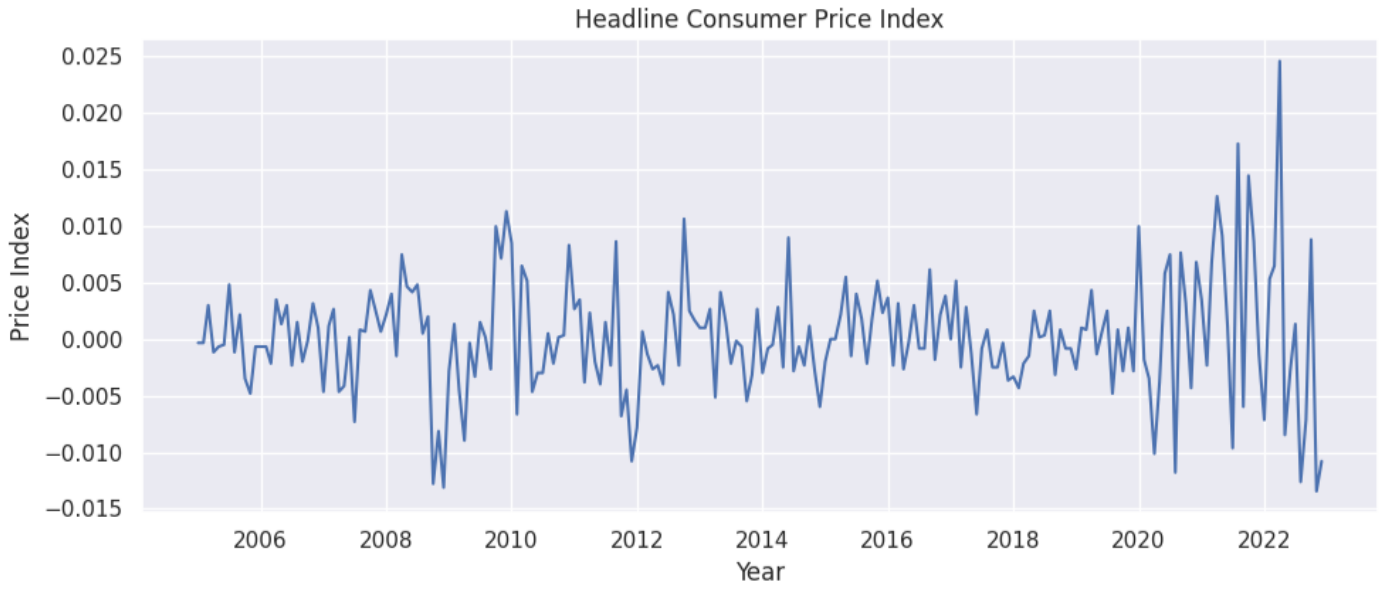
Appendix 2.3

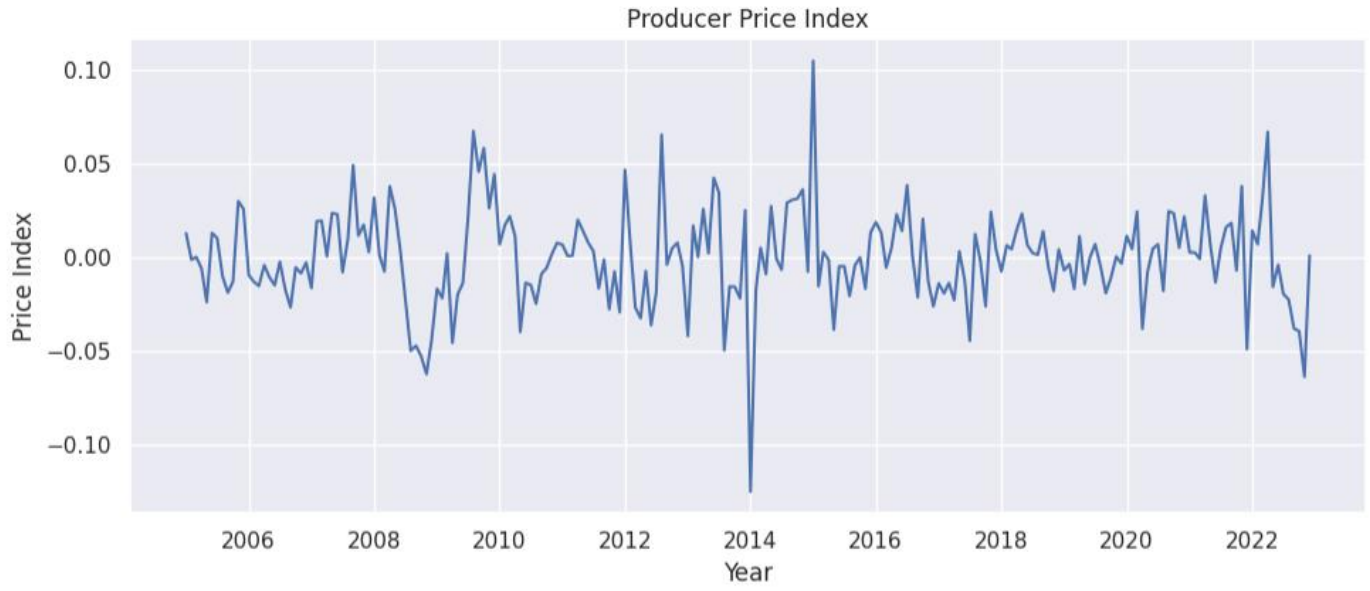
Column	ADF_Stat	p-value	Lags	Obs	Crit-1%	Crit-5%	Crit-10%
Official Core Consumer Price Index	-4.72400	0.00007	15	197	-3.46398	-2.876325	-2.57465
Energy Price Index	-4.40293	0.00029	15	197	-3.46398	-2.87632	-2.57465
Food Price Index	-5.43411	0.000002	14	198	-3.46381	-2.87625	-2.57461
Headline Consumer Price Index	-4.14056	0.00082	15	197	-3.46398	-2.87632	-2.57465
Producer Price Index	-4.51371	0.00018	13	199	-3.46364	-2.87617	-2.57457
Petrol	-4.93489	0.00002	15	197	-3.46398	-2.87632	-2.57465
Diesel	-5.11297	0.00001	15	197	-3.46398	-2.87632	-2.57465

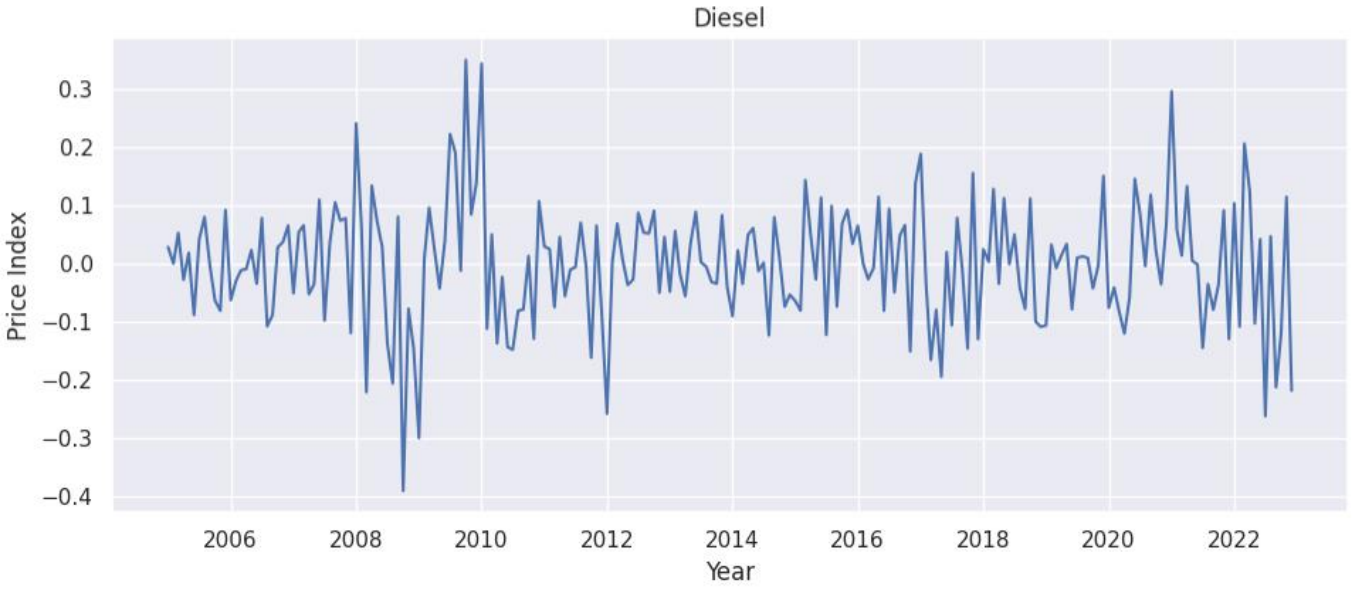

Official Core Consumer Price Index
Price Index
0.020
0.015
0.010
0.005
0.000
−0.005
−0.010
−0.015
2006
2008
2010
2012
2014
2016
2018
2020
2022
Year


Energy Price Index
Price Index
0.3
0.2
0.1
0.0
−0.1
−0.2
2006
2008
2010
2012
2014
2016
2018
2020
2022
Year


Food Price Index
Price Index
0.04
0.02
0.00
−0.02
−0.04
2006
2008
2010
2012
2014
2016
2018
2020
2022
Year







Appendix 2.4

Column	null_hypothesis	Conclusion

Official Core Consumer Price Index	Rejected	Stationary
Energy Price Index	Rejected	Stationary
Food Price Index	Rejected	Stationary
Headline Consumer Price Index	Rejected	Stationary
Producer Price Index	Rejected	Stationary
Petrol	Rejected	Stationary
Diesel	Rejected	Stationary

Appendix 3

Appendix 3.1

```
exponential_aic_dict = {}
exponential_bic_dict = {}
alpha = 0.7
beta = 0.7
seasonal_periods = 12
exponentialSmoothingResultDF = pd.DataFrame()
for col in exponentialSmoothingTrainDF.columns:
    model = ExponentialSmoothing(exponentialSmoothingTrainDF[col],
                                 seasonal_periods=seasonal_periods,
                                 trend='add',
                                 seasonal='add')
    model_fit = model.fit(smoothing_level=alpha, smoothing_slope=beta)
    exponential_aic_dict[col] = model_fit.aic
    exponential_bic_dict[col] = model_fit.bic
    predictions = model_fit.forecast(test_size)
    exponentialSmoothingResultDF[col] = predictions
```

Appendix 3.2

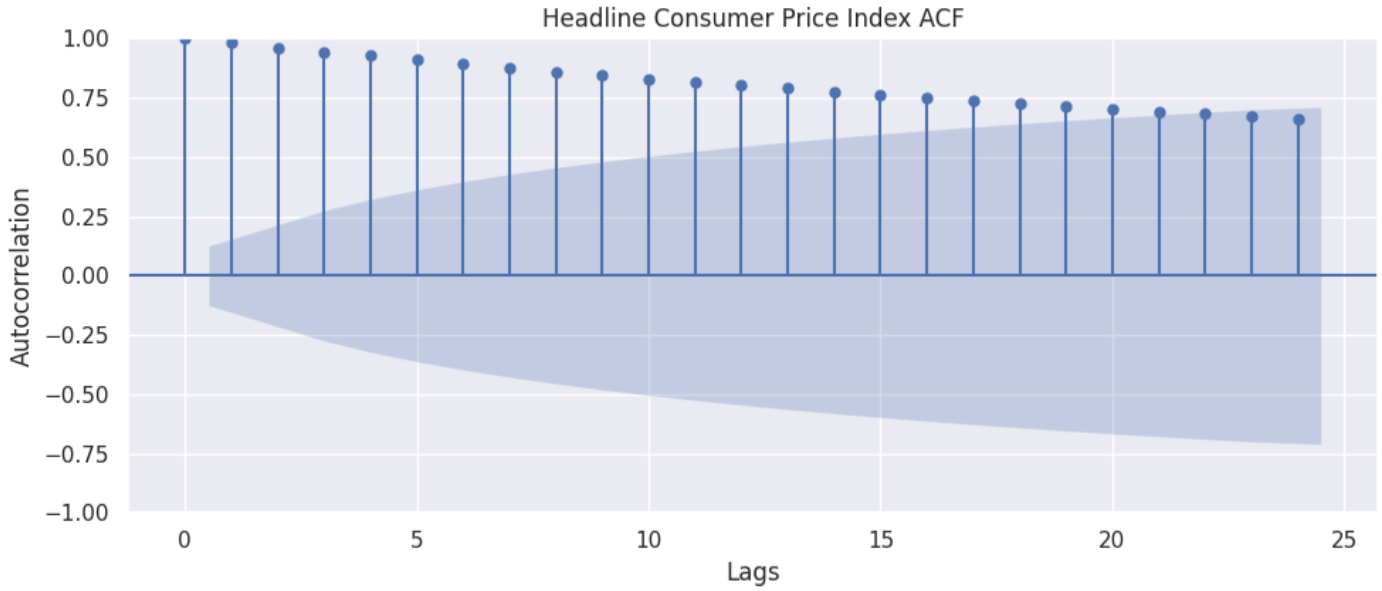
```
exponential_mse_dict = {}
exponential_mae_dict = {}
for col in exponentialSmoothingTrainDF.columns:
    mse = np.mean(np.square(exponentialSmoothingTestDF[col] - exponentialSmoothingResultDF[col]))
    mae = mean_absolute_error(exponentialSmoothingTestDF[col], exponentialSmoothingResultDF[col])
    exponential_mse_dict[col] = mse
    exponential_mae_dict[col] = mae
exponential_metrics_dict_list = [exponential_mse_dict, exponential_mae_dict, exponential_aic_dict, exponential_bic_dict]
metricsDf = pd.DataFrame(exponential_metrics_dict_list, index=[exponential_mse_dict, exponential_mae_dict, exponential_aic_
dict, exponential_bic_dict])
metricsDf = metricsDf.T
metricsDf.columns = ['mse', 'mae', 'aic', 'bic']
```

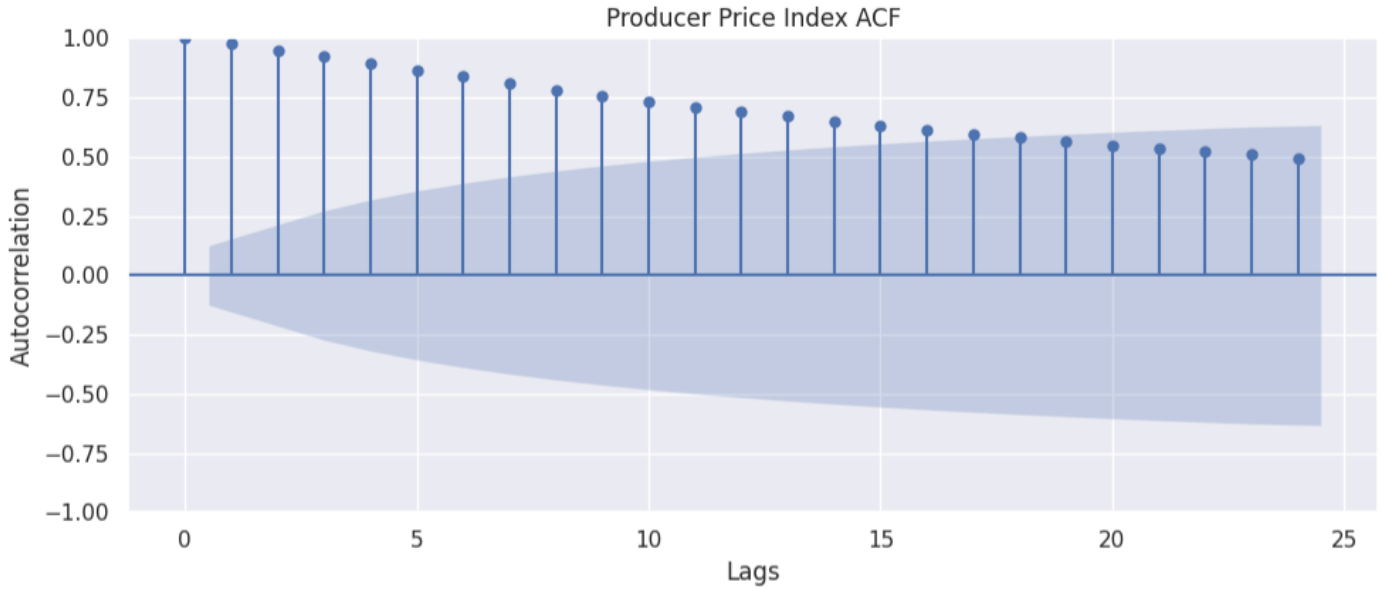
Inflation index	mse	mae	aic	bic
Official Core Consumer Price	0.0001	0.00837	-2555.22619	-2500.35666
Energy Price	0.03002	0.13359	-1350.24938	-1295.37985
Food Price	0.00933	0.07553	-2045.1979	-1990.32837
Headline Consumer Price	0.00116	0.0298	-2511.41097	-2456.54144

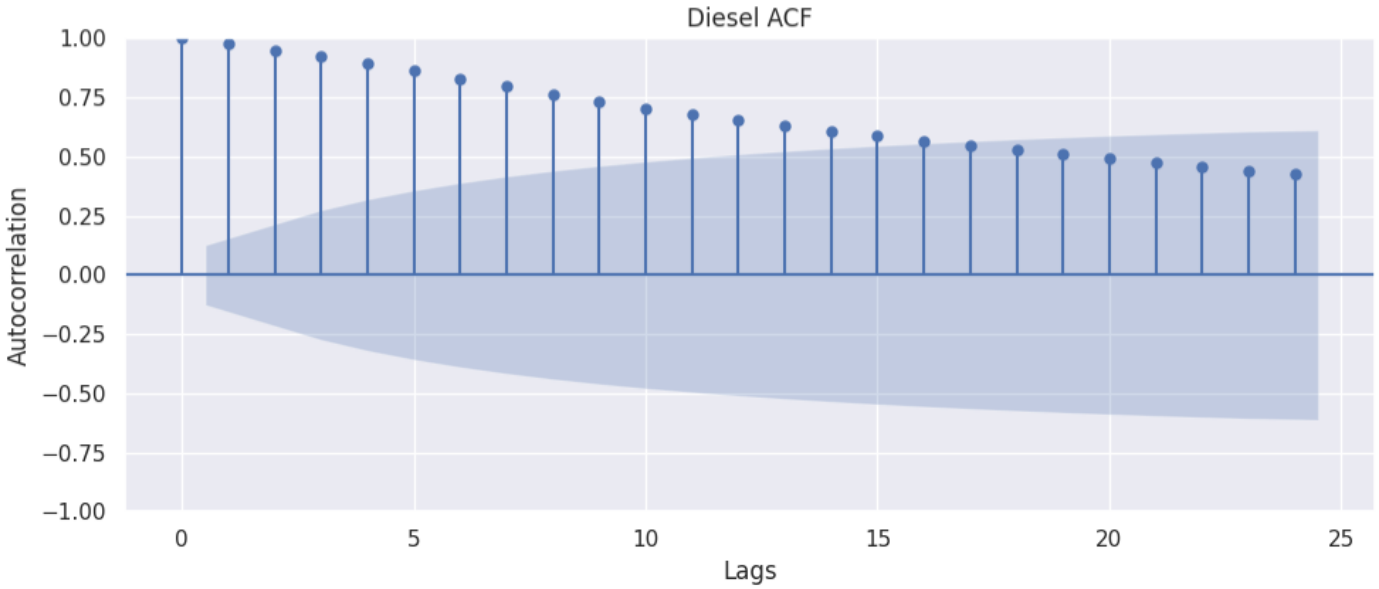
Producer Price	0.0224	0.13462	-1785.95788	-1731.08835
Petrol	0.06368	0.22149	-1118.11915	-1063.24962
Diesel	0.15553	0.36141	-1127.73178	-1072.86225

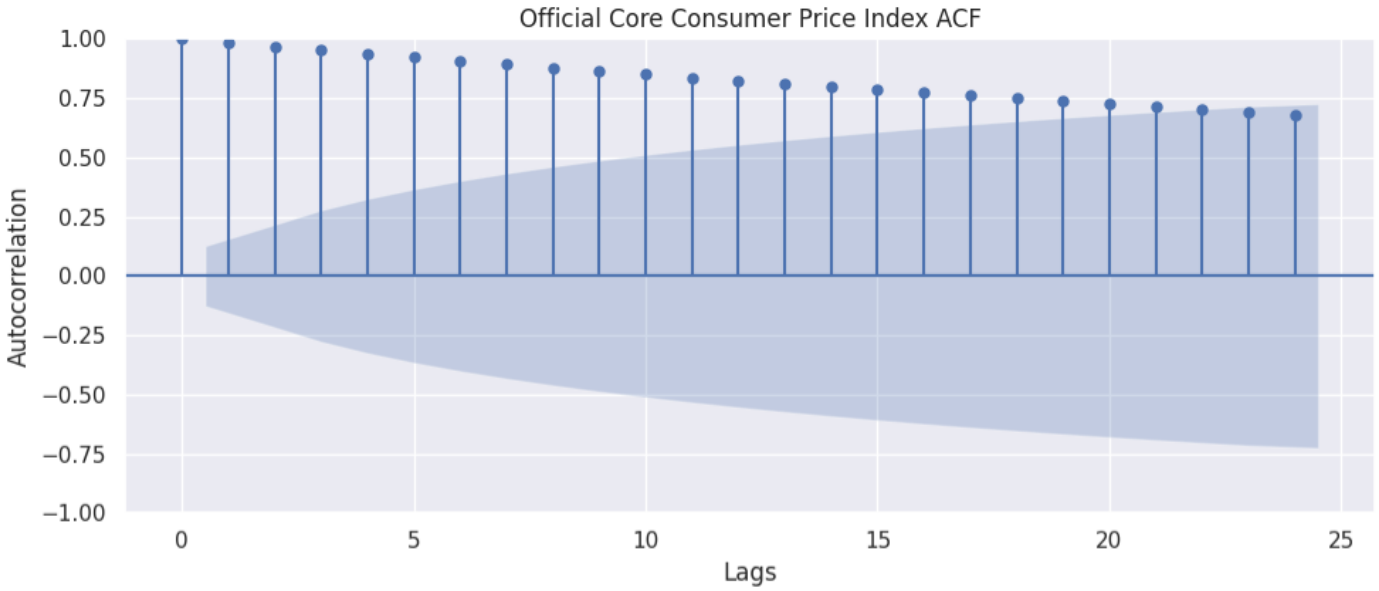
Appendix 4

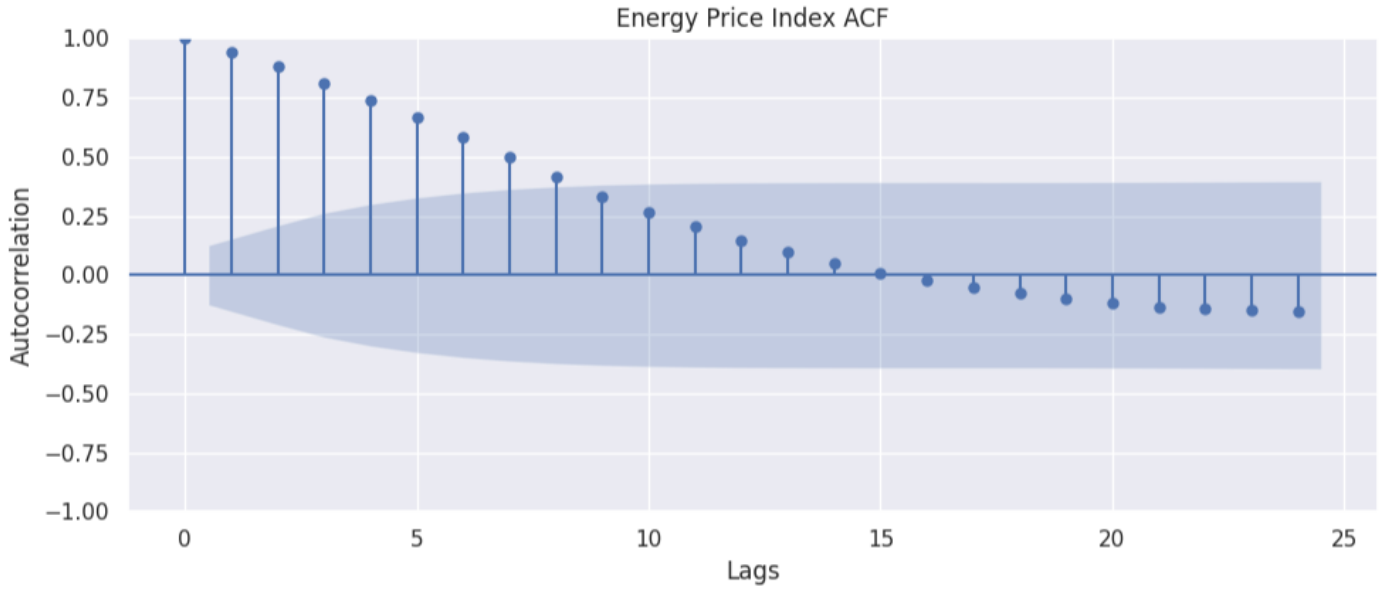
Appendix 4.1






Diesel ACF
Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25
Lags


Official Core Consumer Price Index ACF
Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25
Lags



Appendix 4.2

```
autoRegressiveResultDF = pd.DataFrame()
autoregressive_aic_dict = {}
autoregressive_bic_dict = {}

# loop through each column in `df`
for col in autoRegressiveTrainDF.columns:
    # Create the Auto-Regressive model for the current column
    model = AutoReg(autoRegressiveTrainDF[col], lags=15)

    # fit the model and make predictions
    model_fit = model.fit()

    # Calculate AIC and BIC
    n = len(autoRegressiveTrainDF[col])
    k = len(model_fit.params)
    autoregressive_aic_dict[col] = 2*k - 2*np.log(model_fit.llf)
    autoregressive_bic_dict[col] = np.log(n)*k - 2*np.log(model_fit.llf)
```

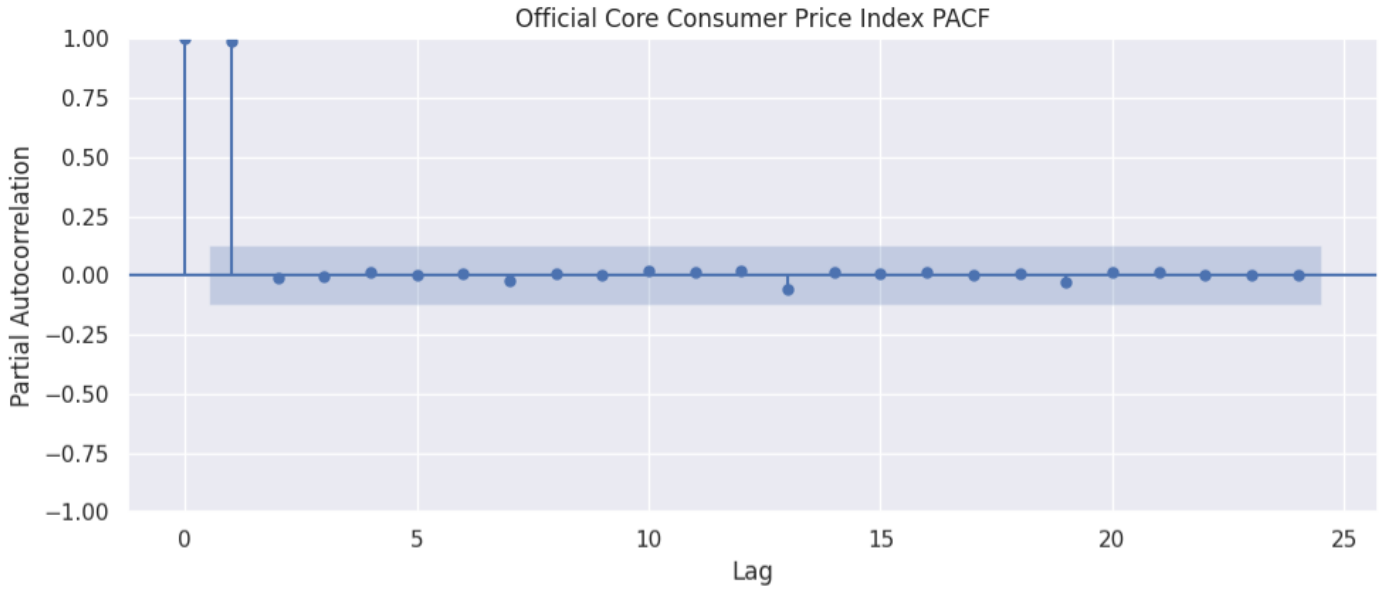
```
    predictions = model_fit.predict(start=len(autoRegressiveTrainDF), end=len(autoRegressiveTrainDF)+12)

    # add the predictions to the result dataframe
    autoRegressiveResultDF[col] = predictions
```

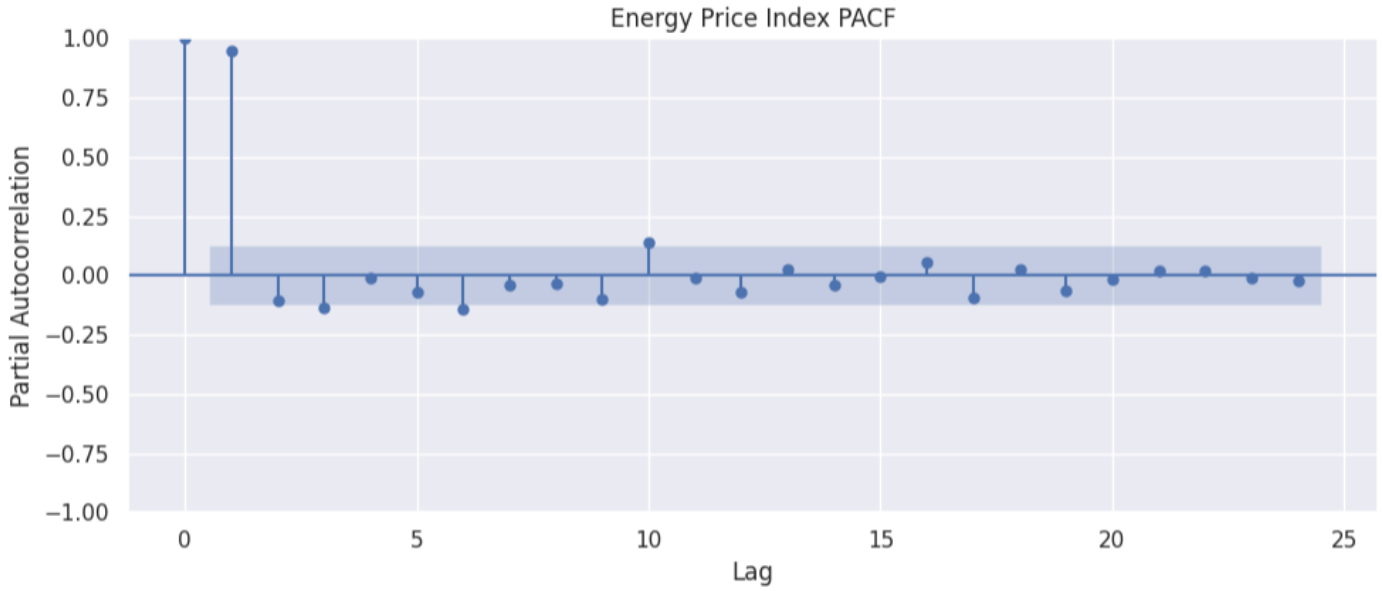
Appendix 4.3

```
autoregressive_mse_dict = {}
autoregressive_mae_dict = {}

for col in autoRegressiveTrainDF.columns:
    mse = np.mean(np.square(autoRegressiveTestDF[col] - autoRegressiveResultDF[col].head(12)))
    mae = mean_absolute_error(autoRegressiveTestDF[col], autoRegressiveResultDF[col].head(12))
    autoregressive_mse_dict[col] = mse
    autoregressive_mae_dict[col] = mae

autoregressive_metrics_dict_list = [autoregressive_mse_dict, autoregressive_mae_dict, autoregressive_aic_dict, autoregressive_bi
c_dict]
metricsDf = pd.DataFrame(autoregressive_metrics_dict_list, index=[autoregressive_mse_dict, autoregressive_mae_dict, autoregre
ssive_aic_dict, autoregressive_bic_dict])
metricsDf = metricsDf.T
metricsDf.columns = ['mse', 'mae', 'aic', 'bic']
```

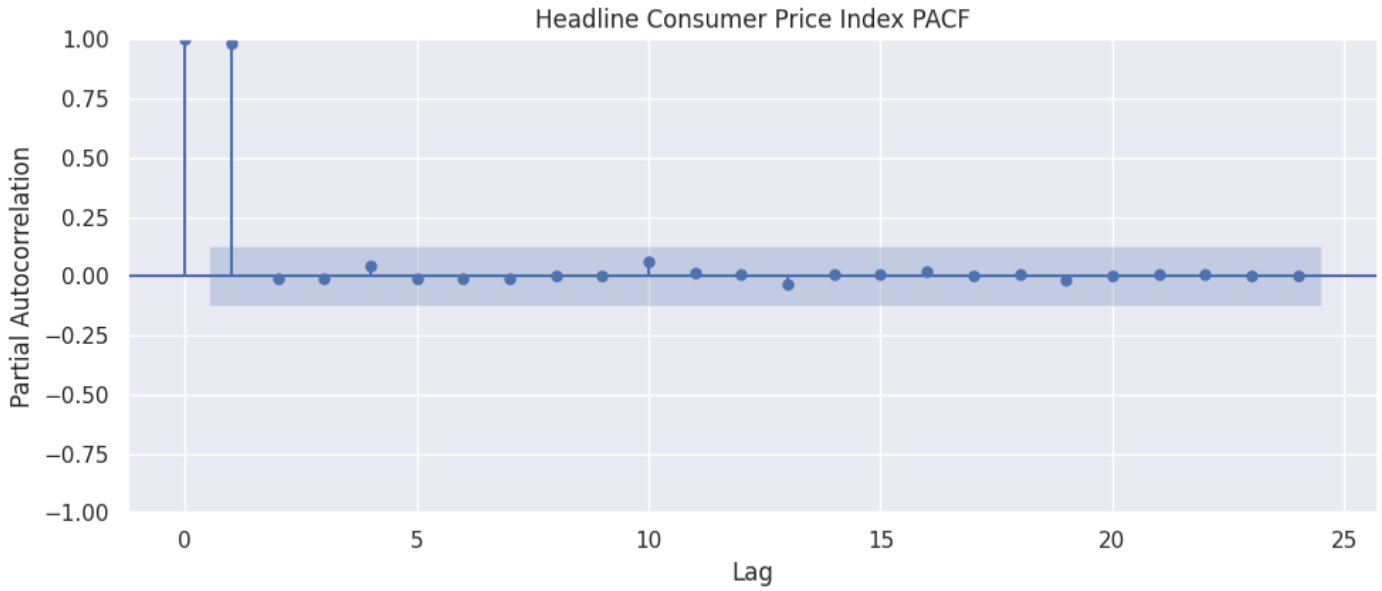
Inflation index	mse	mae	aic	bic
Official Core Consumer Price	1e-05	0.00219	18.13619	70.98907
Energy Price	0.00336	0.04876	19.55395	72.40683
Food Price	0.00017	0.0101	18.57138	71.42426
Headline Consumer Price	3e-05	0.00445	18.2161	71.06898
Producer Price	0.0006	0.02147	18.91722	71.7701
Petrol	0.01109	0.08257	19.90056	72.75344
Diesel	0.00929	0.08454	19.90165	72.75453

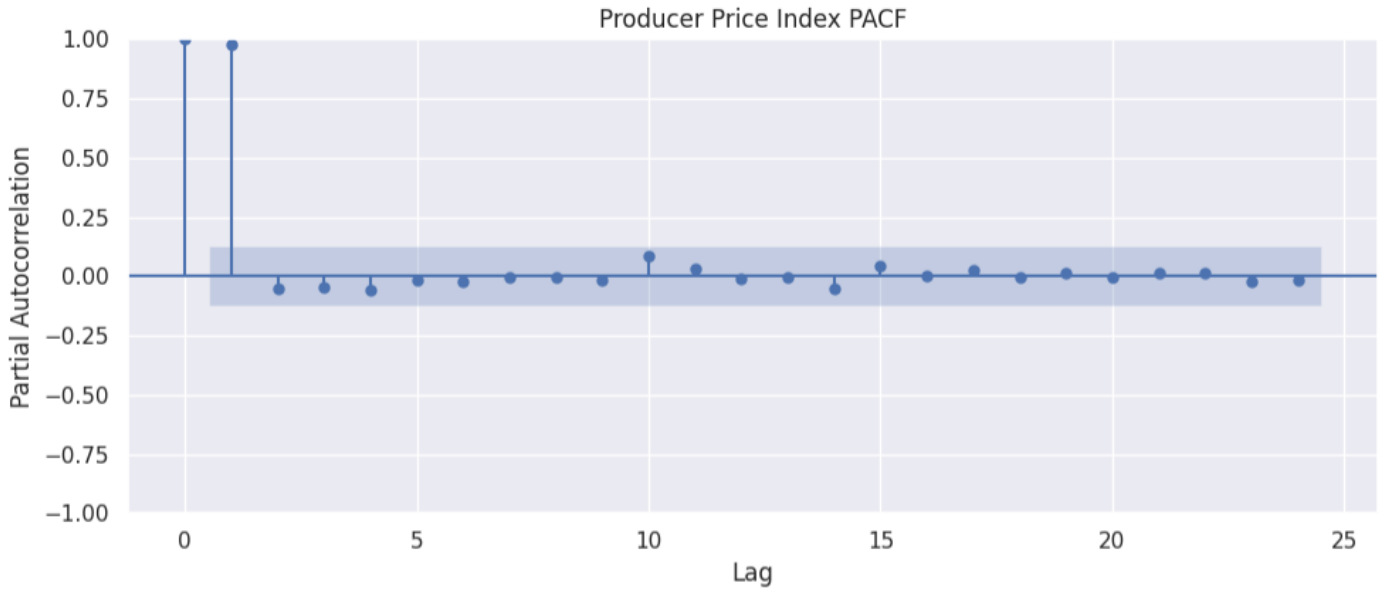
Appendix 4.4

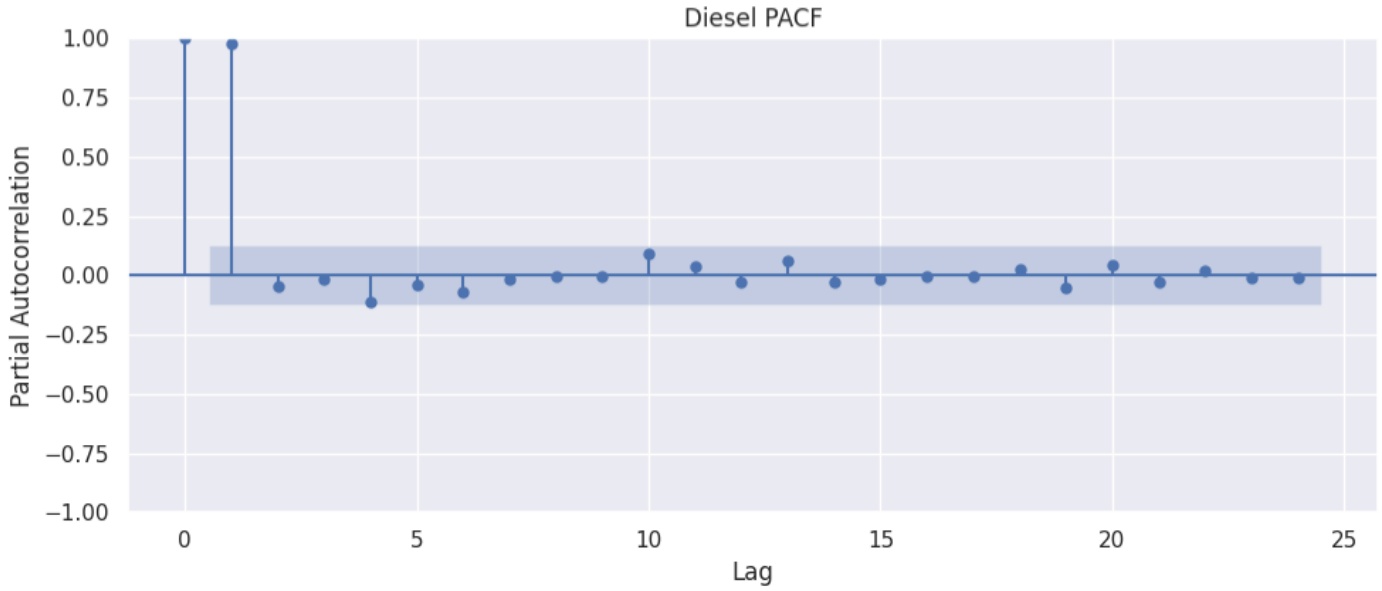

Official Core Consumer Price Index PACF
Partial Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25
Lag


Energy Price Index PACF
Partial Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25
Lag

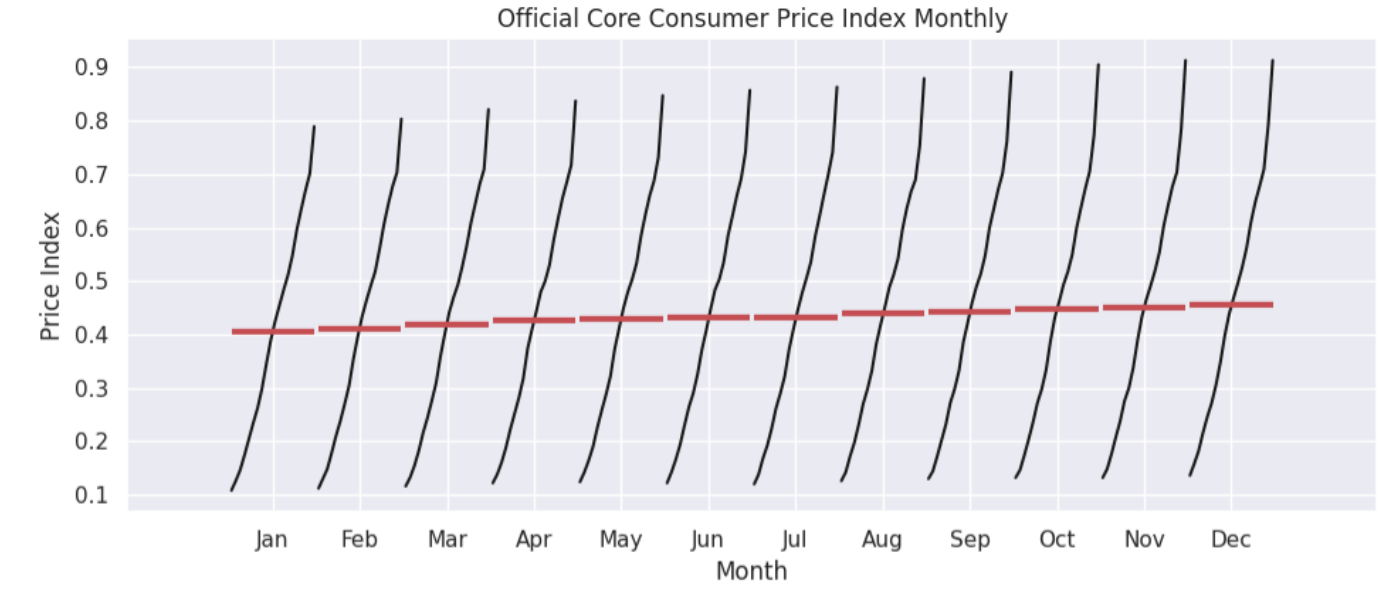

Food Price Index PACF
Partial Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25
Lag

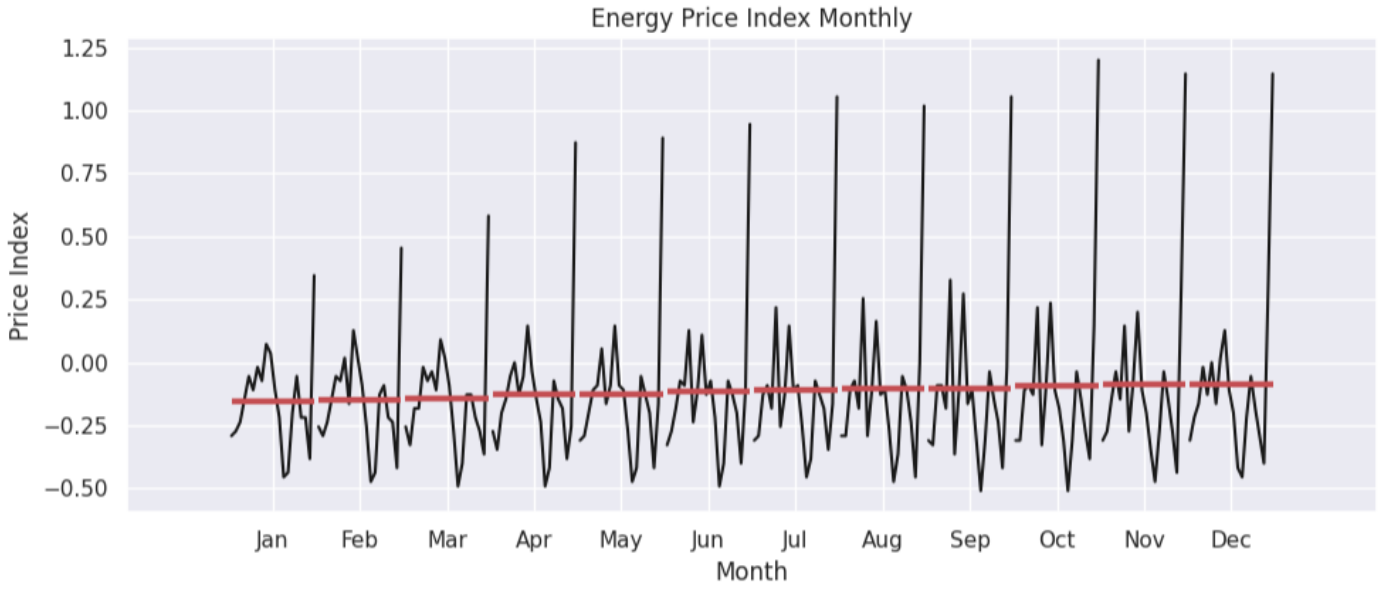

Headline Consumer Price Index PACF
Partial Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25
Lag

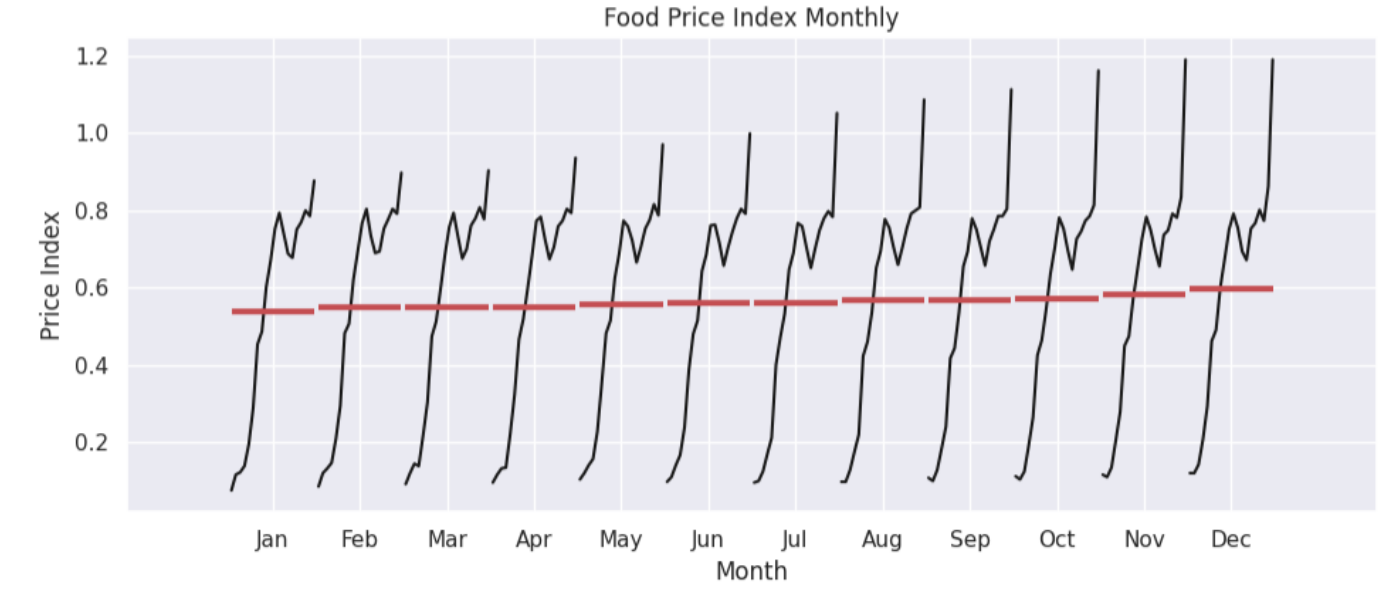

Producer Price Index PACF
Partial Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25
Lag

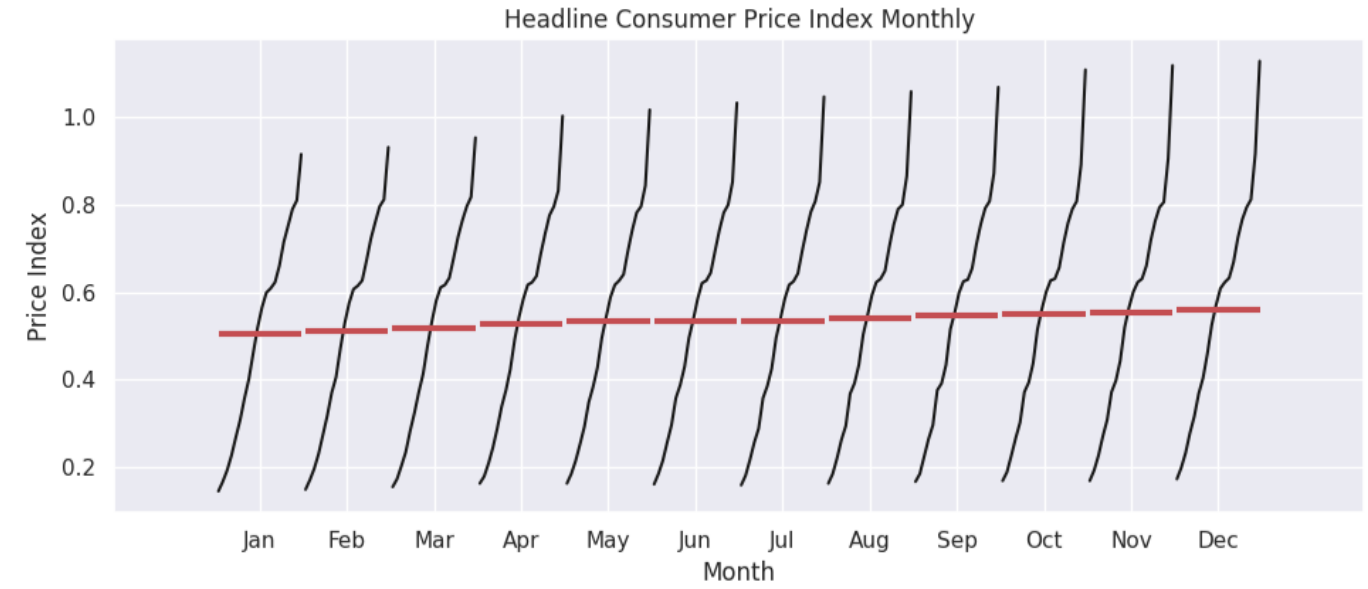

Diesel PACF
Partial Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25
Lag

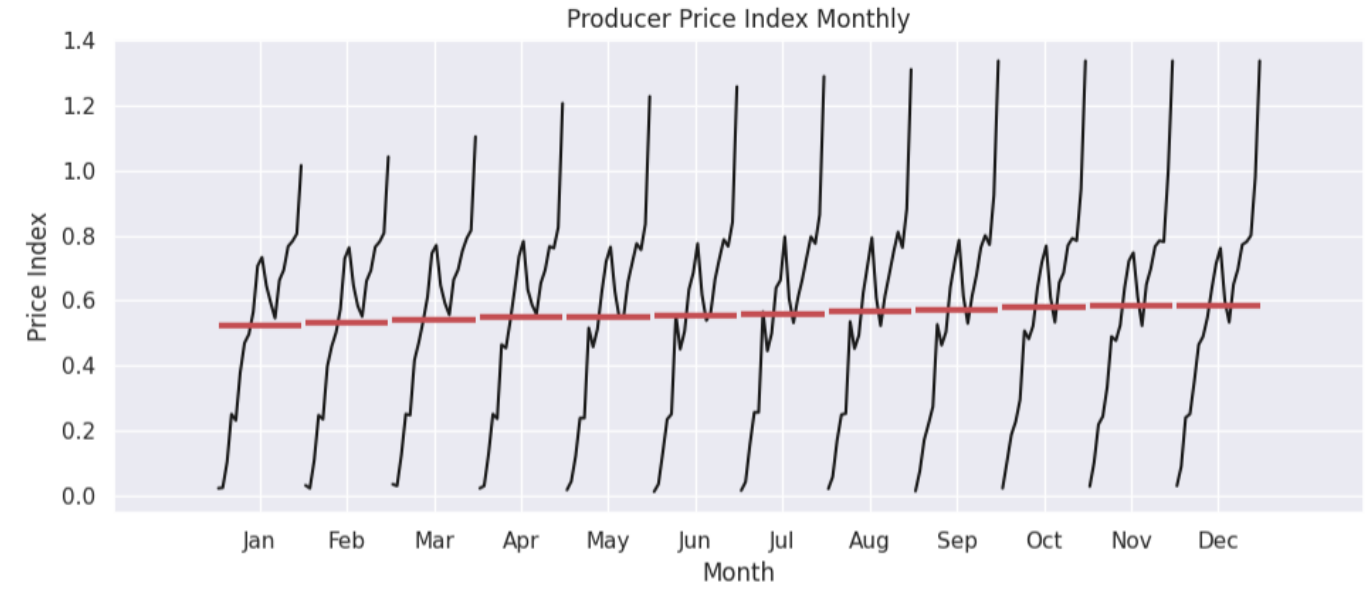
Appendix 4.5

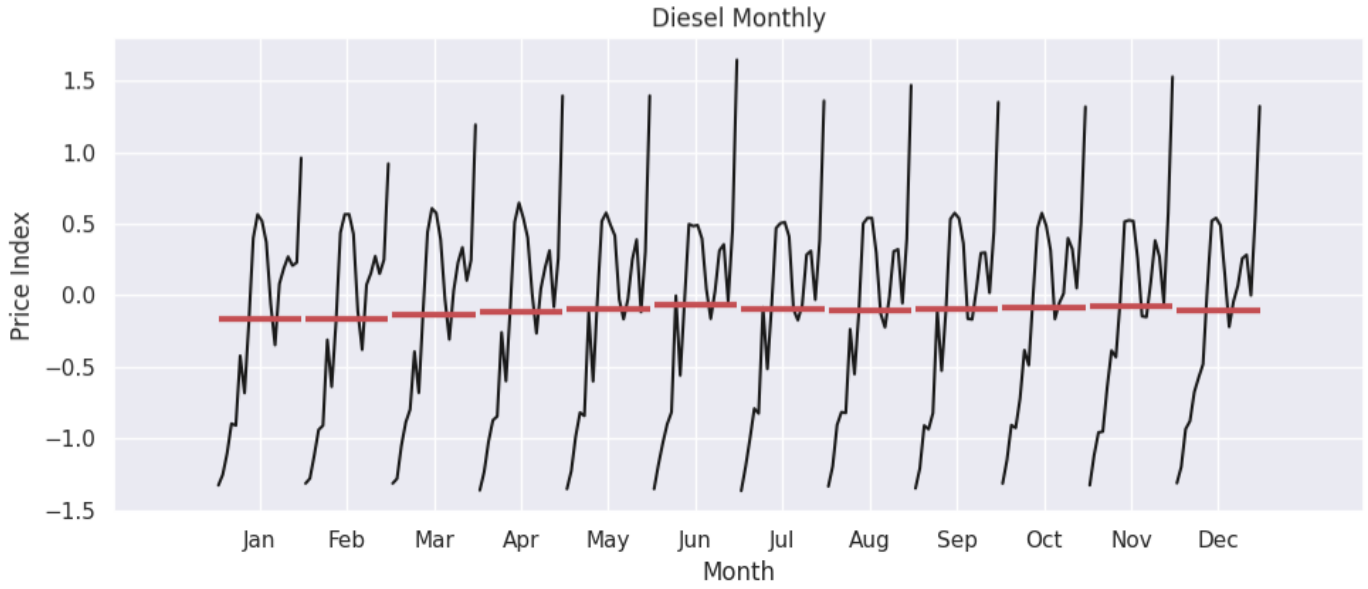

Official Core Consumer Price Index Monthly
Price Index
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Month


Energy Price Index Monthly
Price Index
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Month


Food Price Index Monthly
Price Index
1.2
1.0
0.8
0.6
0.4
0.2
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Month

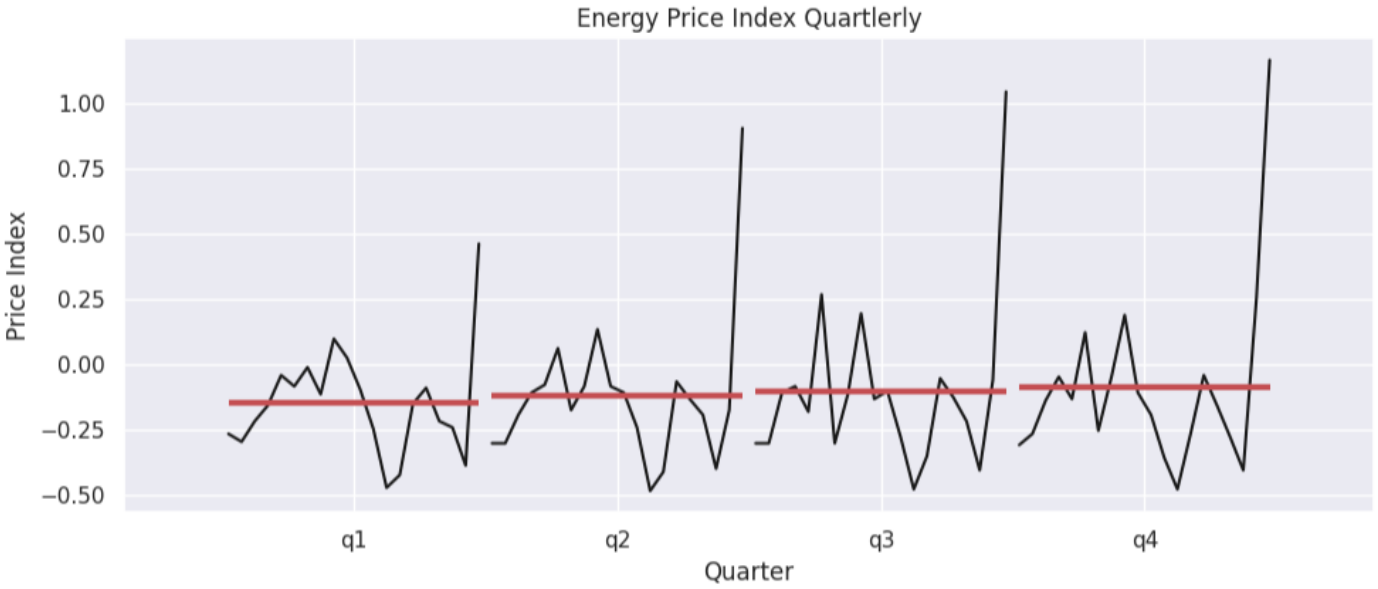

Headline Consumer Price Index Monthly
Price Index
1.0
0.8
0.6
0.4
0.2
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Month


Producer Price Index Monthly
Price Index
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Month

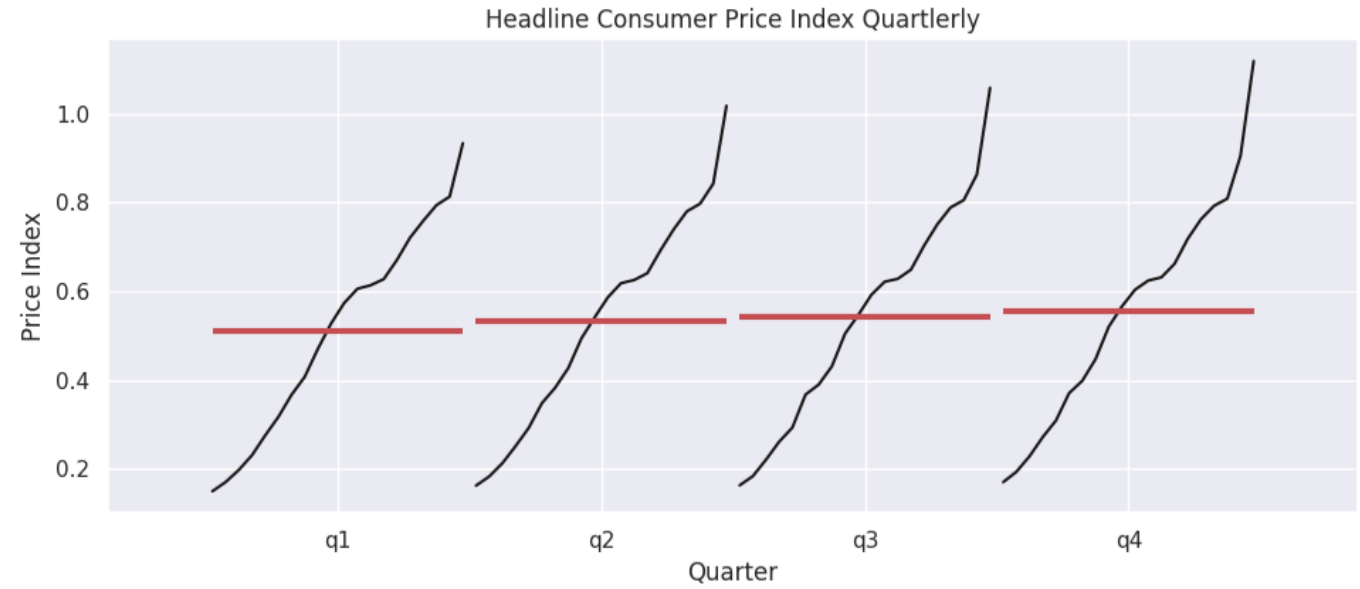

Diesel Monthly
Price Index
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Month

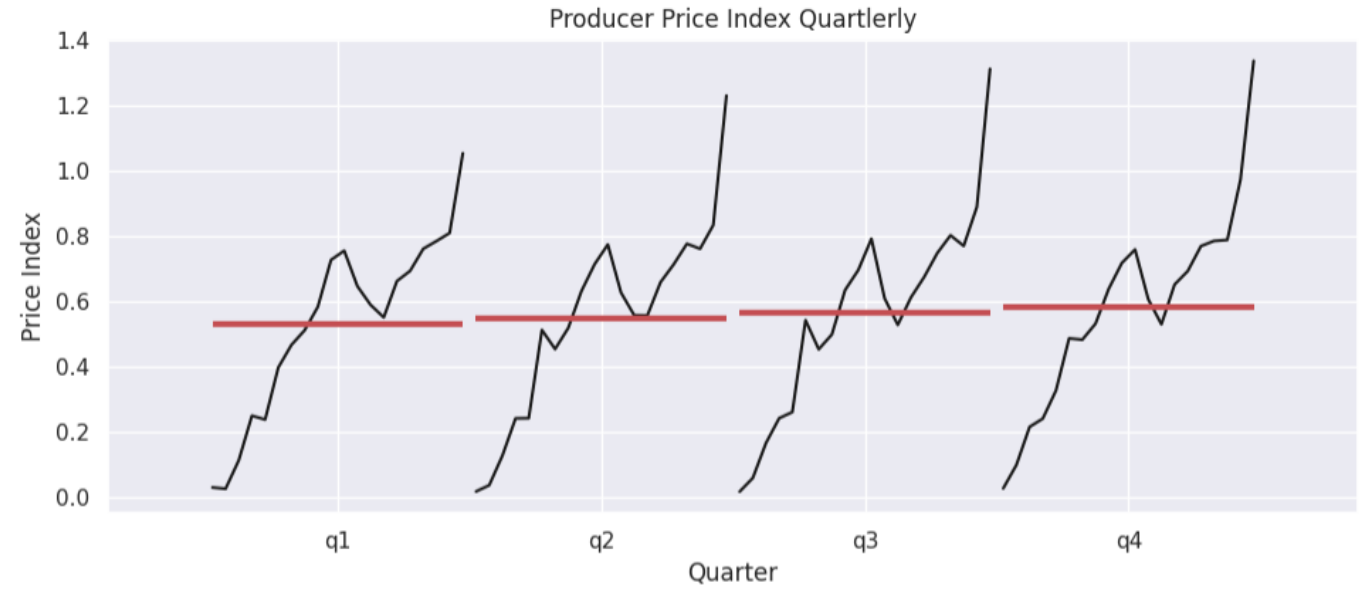
Appendix 4.6

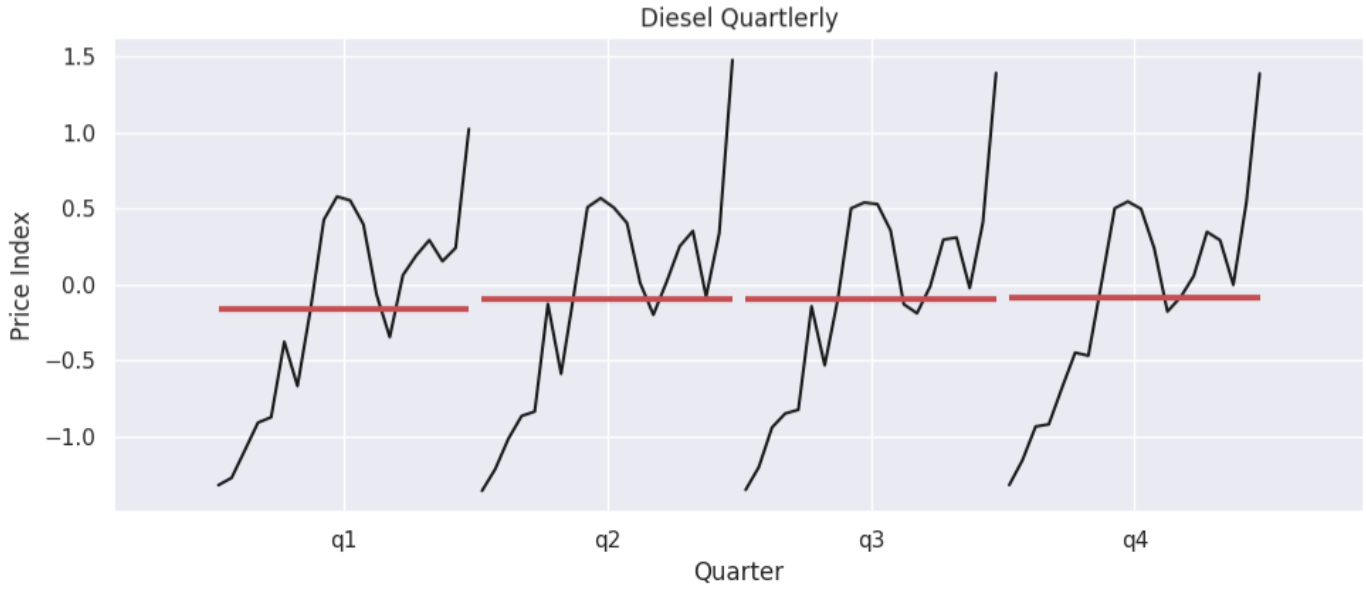

Official Core Consumer Price Index Quartlerly
Price Index
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
q1
q2
q3
q4
Quarter


Energy Price Index Quartlerly
Price Index
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
q1
q2
q3
q4
Quarter

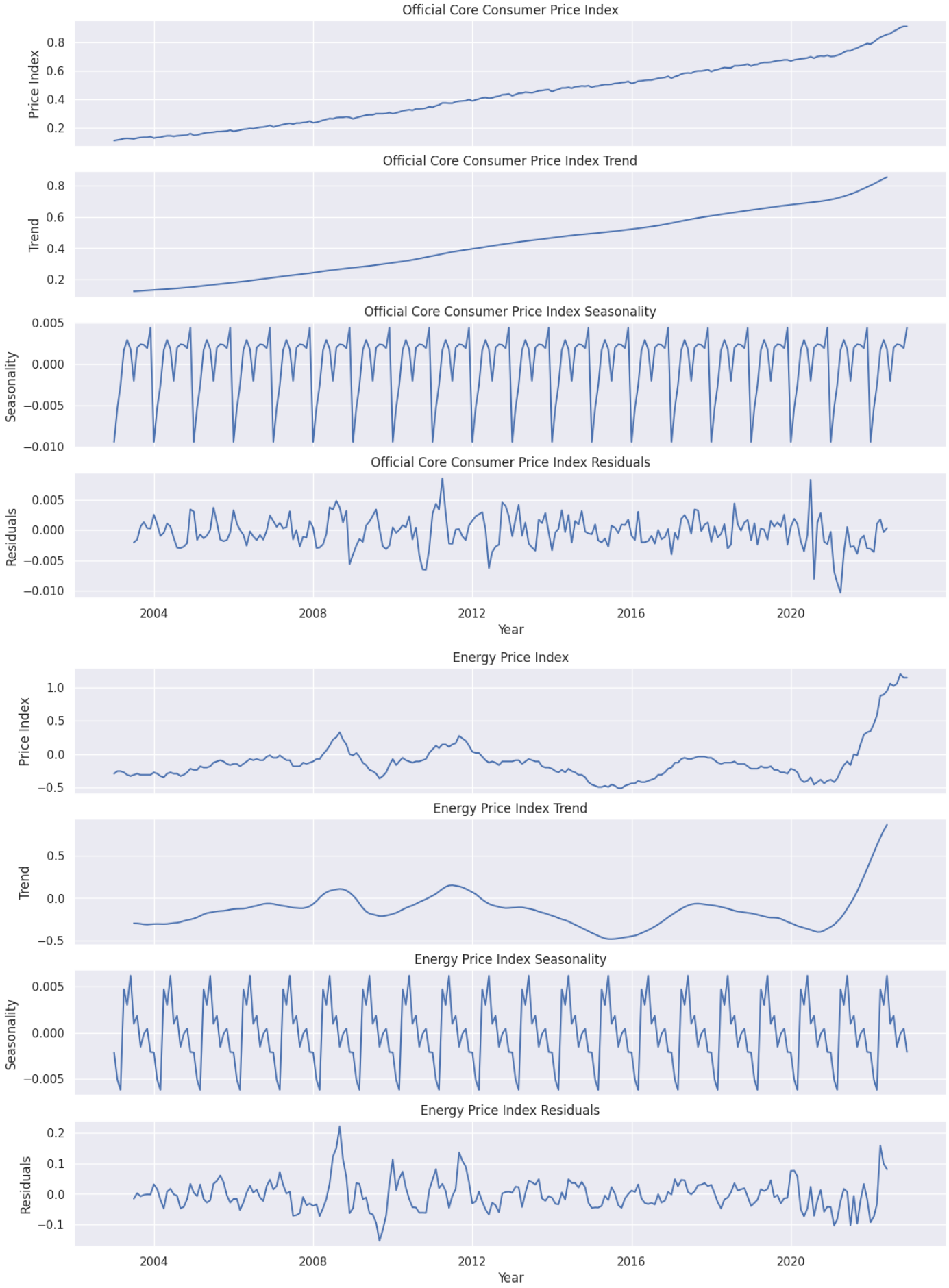

Food Price Index Quartlerly
Price Index
1.2
1.0
0.8
0.6
0.4
0.2
q1
q2
q3
q4
Quarter

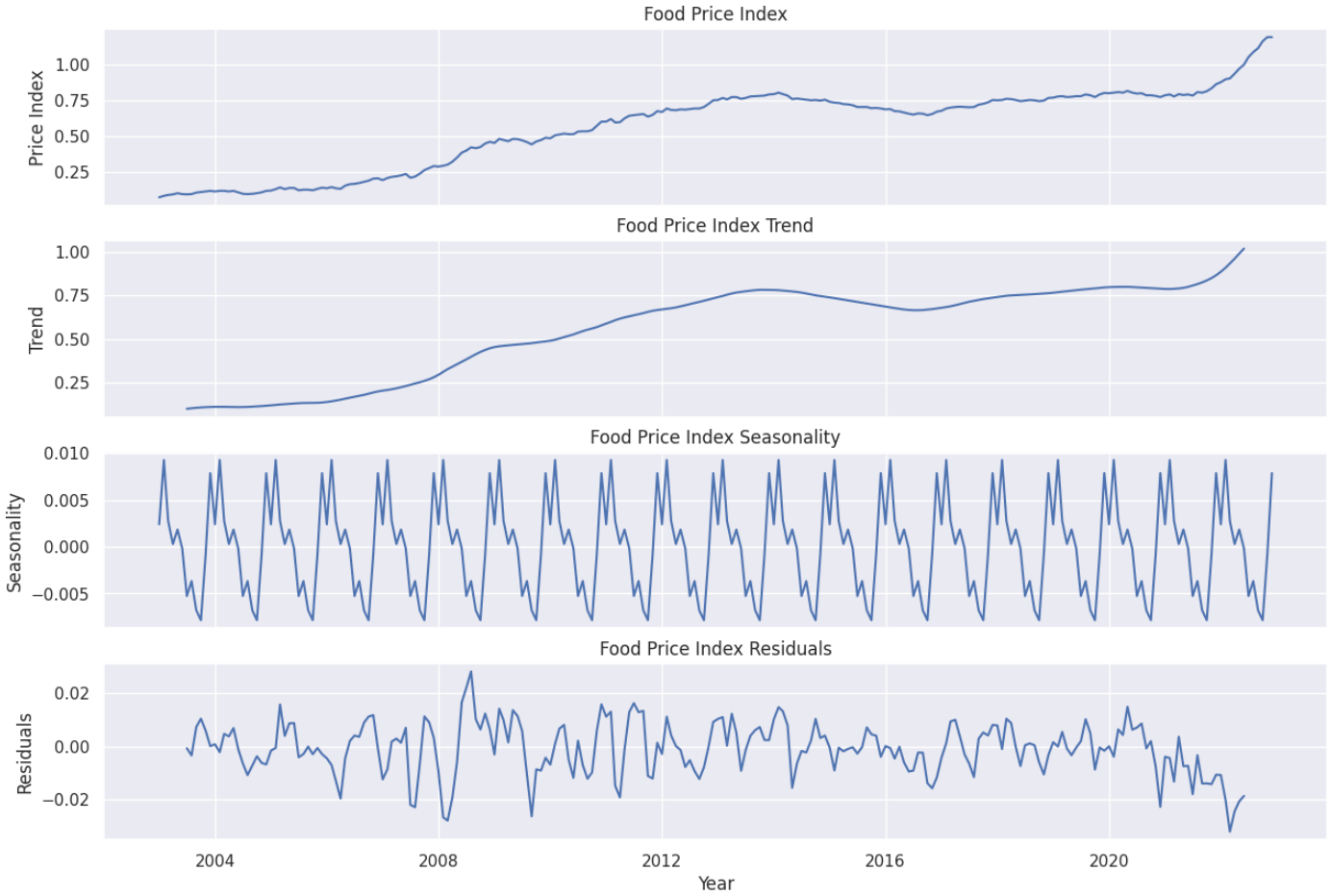

Headline Consumer Price Index Quartlerly
Price Index
1.0
0.8
0.6
0.4
0.2
q1
q2
q3
q4
Quarter

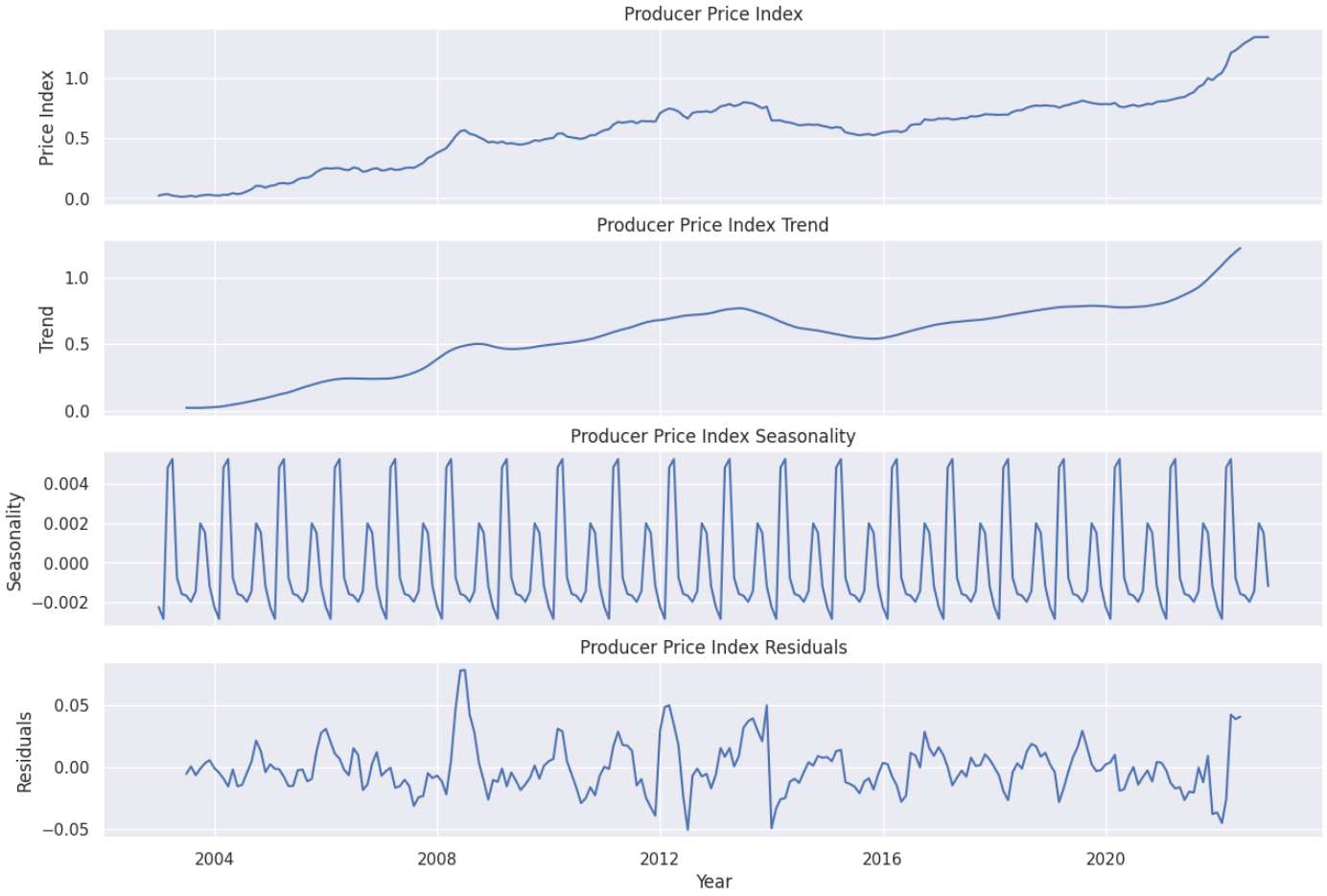

Producer Price Index Quartlerly
Price Index
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
q1
q2
q3
q4
Quarter

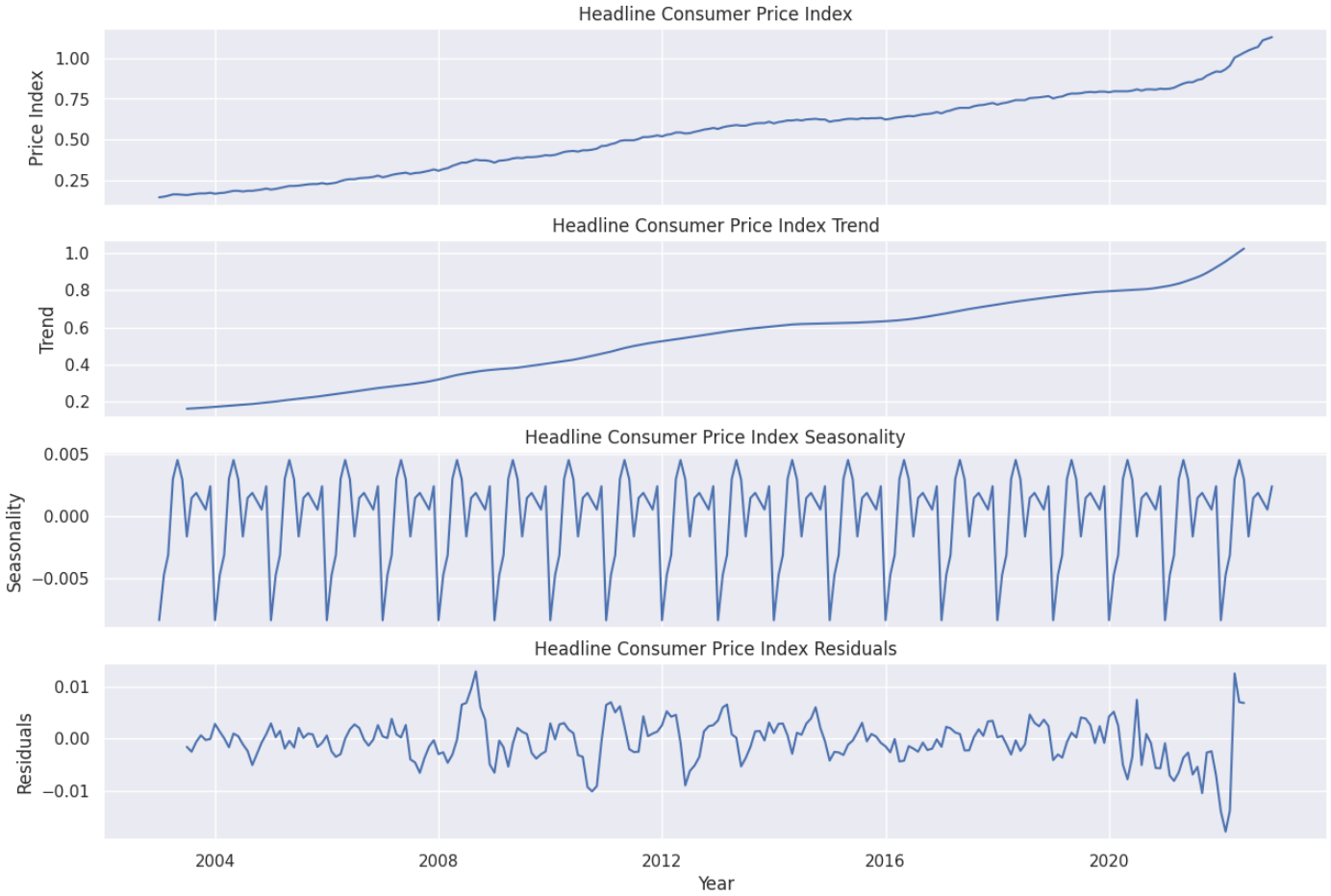

Diesel Quartlerly
Price Index
1.5
1.0
0.5
0.0
−0.5
−1.0
q1
q2
q3
q4
Quarter

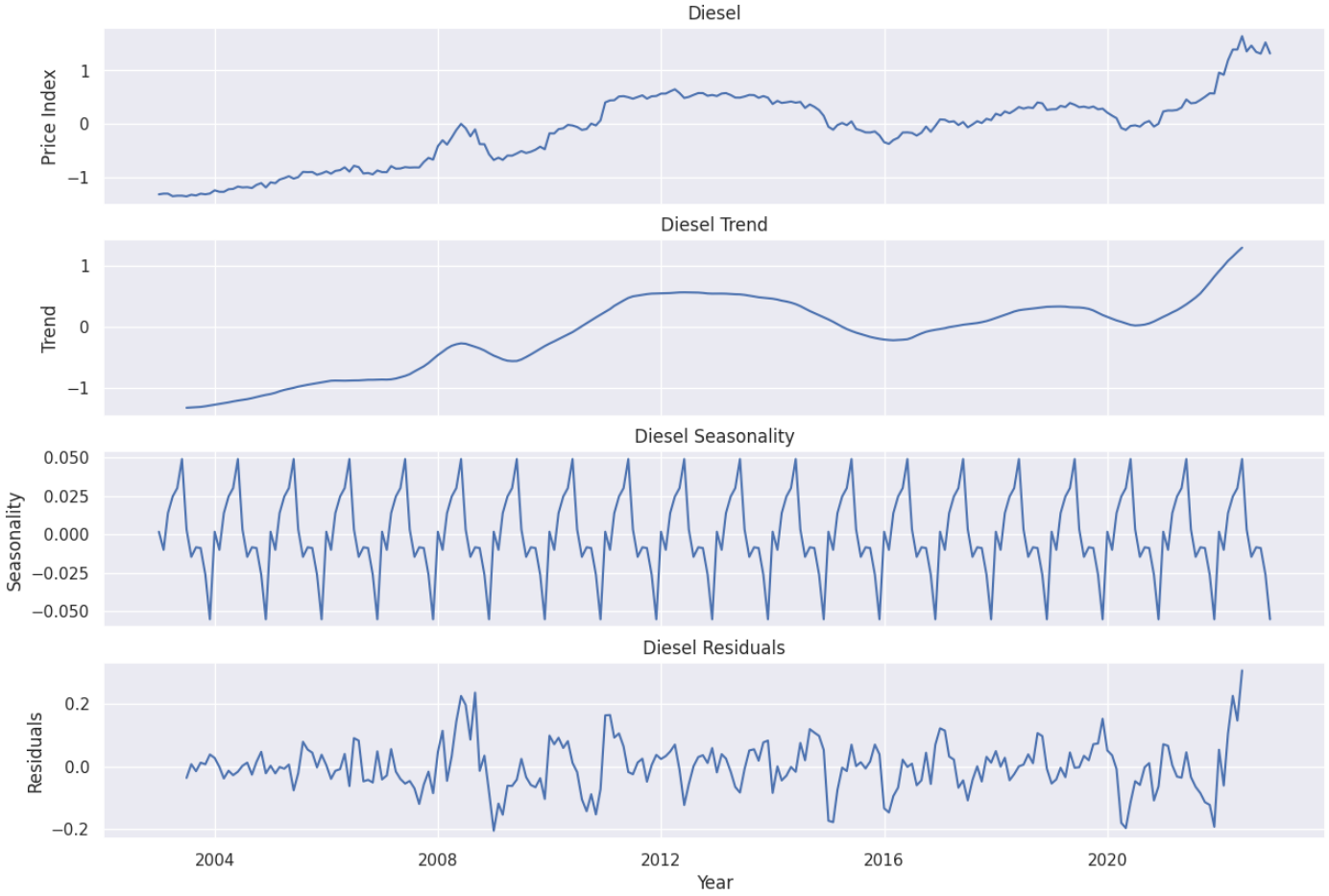
Appendix 4.7


Official Core Consumer Price Index
Price Index
0.8
0.6
0.4
0.2
Official Core Consumer Price Index Trend
Trend
0.8
0.6
0.4
0.2
Official Core Consumer Price Index Seasonality
Seasonality
0.005
0.000
−0.005
−0.010
Official Core Consumer Price Index Residuals
Residuals
0.005
0.000
−0.005
−0.010
2004
2008
2012
2016
2020
Year
Energy Price Index
Price Index
1.0
0.5
0.0
−0.5
Energy Price Index Trend
Trend
0.5
0.0
−0.5
Energy Price Index Seasonality
Seasonality
0.005
0.000
−0.005
Energy Price Index Residuals
Residuals
0.2
0.1
0.0
−0.1
2004
2008
2012
2016
2020
Year


Food Price Index
Price Index
1.00
0.75
0.50
0.25
Food Price Index Trend
Trend
1.00
0.75
0.50
0.25
Food Price Index Seasonality
Seasonality
0.010
0.005
0.000
−0.005
Food Price Index Residuals
Residuals
0.02
0.00
−0.02
2004
2008
2012
2016
2020
Year

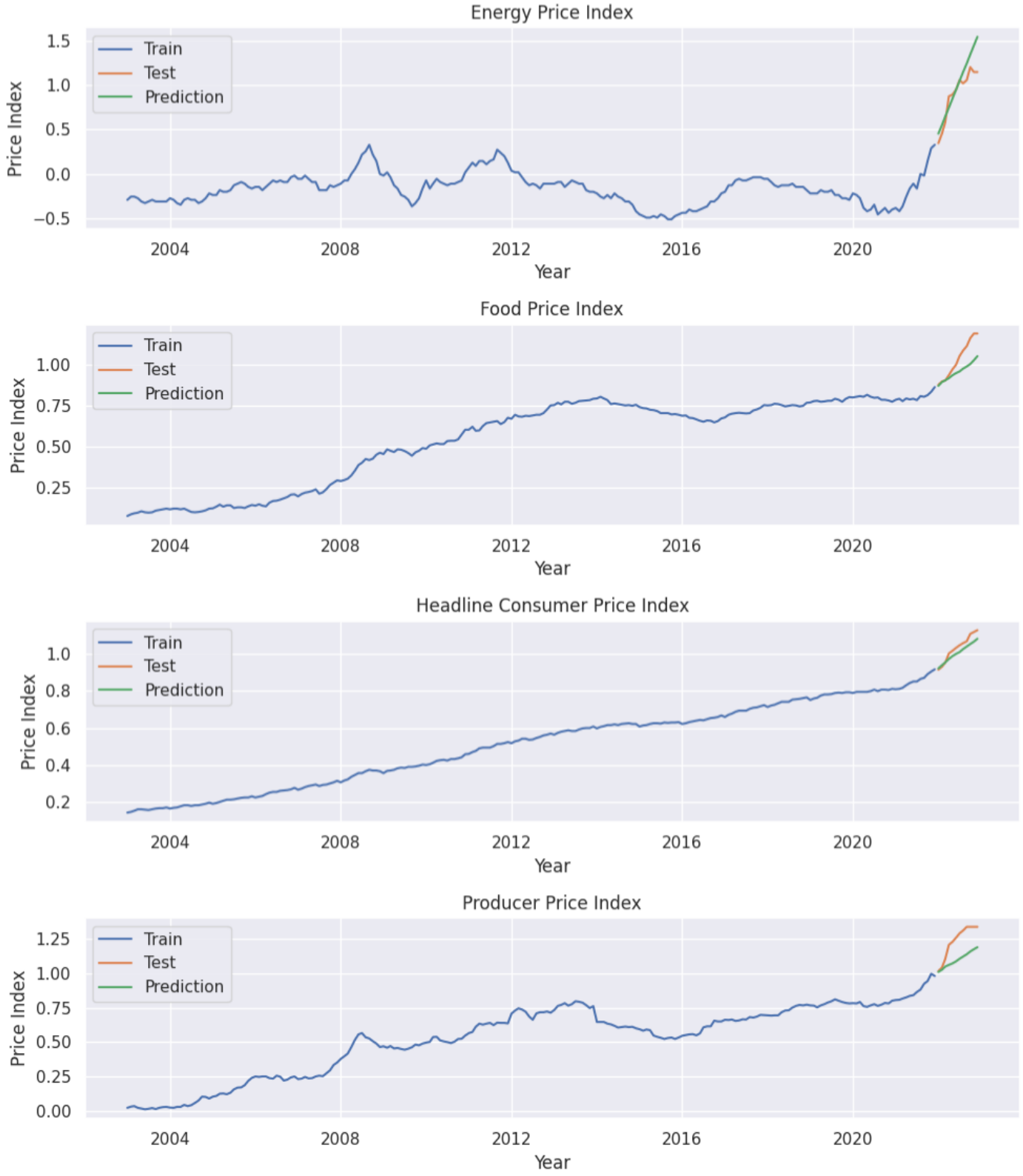

Producer Price Index
Price Index
1.0
0.5
0.0
Producer Price Index Trend
Trend
1.0
0.5
0.0
Producer Price Index Seasonality
Seasonality
0.004
0.002
0.000
−0.002
Producer Price Index Residuals
Residuals
0.05
0.00
−0.05
2004
2008
2012
2016
2020
Year

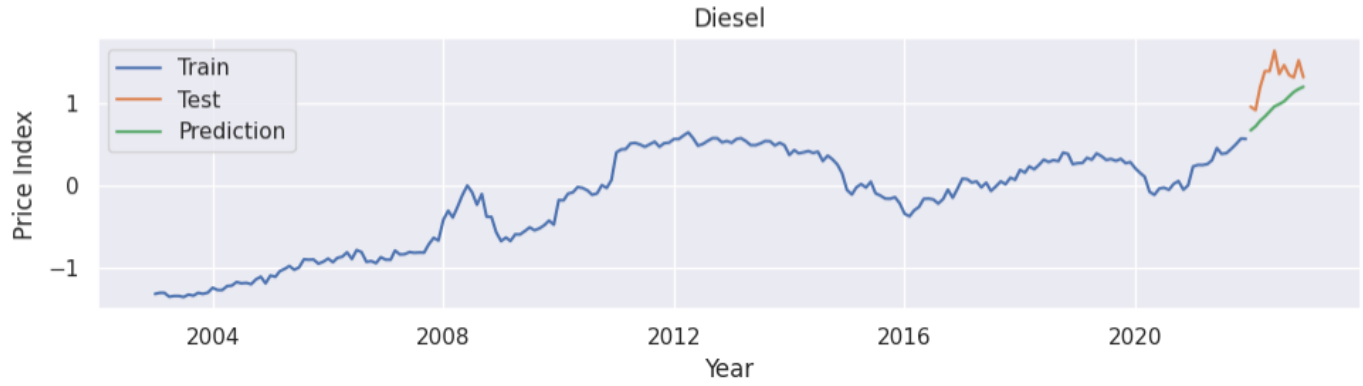

Headline Consumer Price Index
Price Index
1.00
0.75
0.50
0.25
Headline Consumer Price Index Trend
Trend
1.0
0.8
0.6
0.4
0.2
Headline Consumer Price Index Seasonality
Seasonality
0.005
0.000
−0.005
Headline Consumer Price Index Residuals
Residuals
0.01
0.00
−0.01
2004
2008
2012
2016
2020
Year


Diesel
Price Index
1
0
−1
Diesel Trend
Trend
1
0
−1
Diesel Seasonality
Seasonality
0.050
0.025
0.000
−0.025
−0.050
Diesel Residuals
Residuals
0.2
0.0
−0.2
2004
2008
2012
2016
2020
Year

Appendix 5

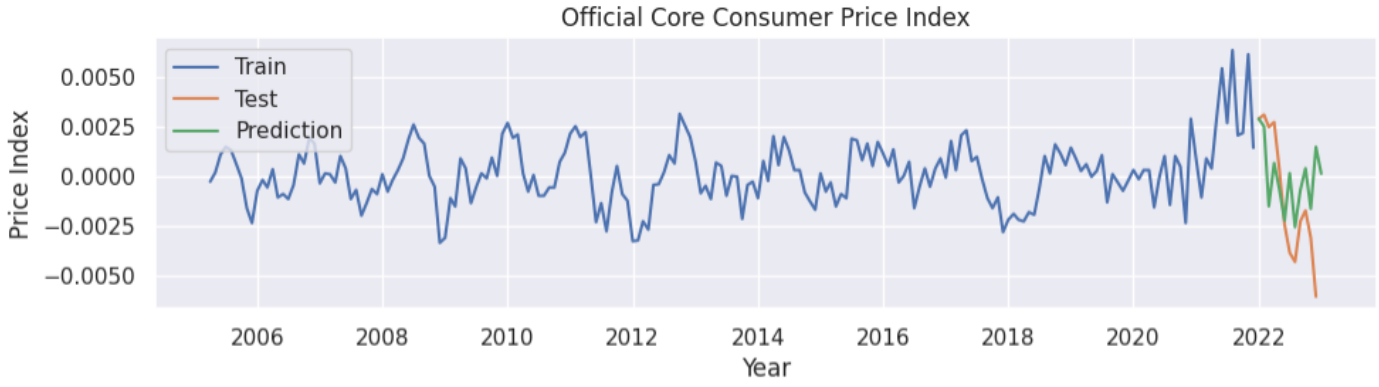
Appendix 5.1

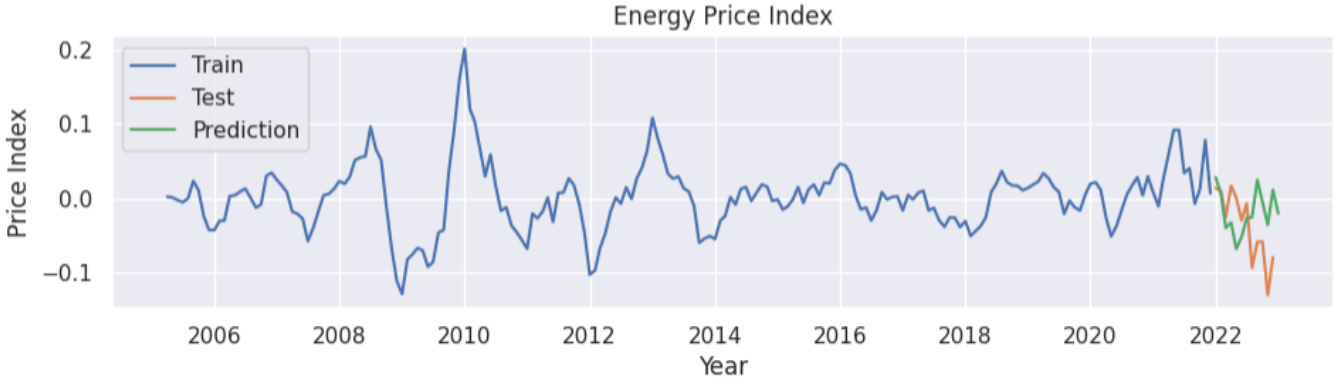


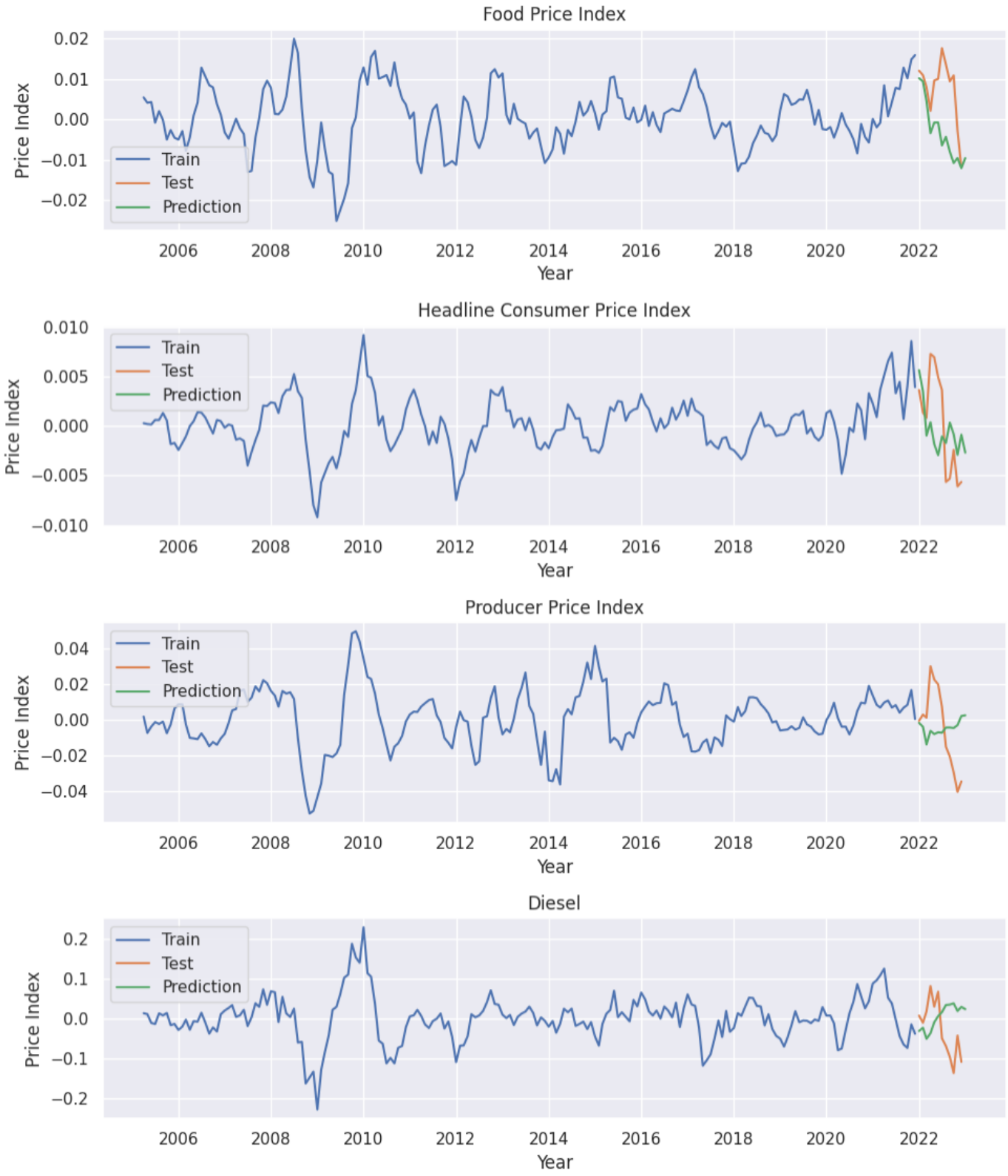

Diesel
Train
Test
Prediction
Price Index
1
0
−1
2004
2008
2012
2016
2020
Year


Official Core Consumer Price Index
Train
Test
Prediction
Price Index
0.8
0.6
0.4
0.2
2004
2008
2012
2016
2020
Year

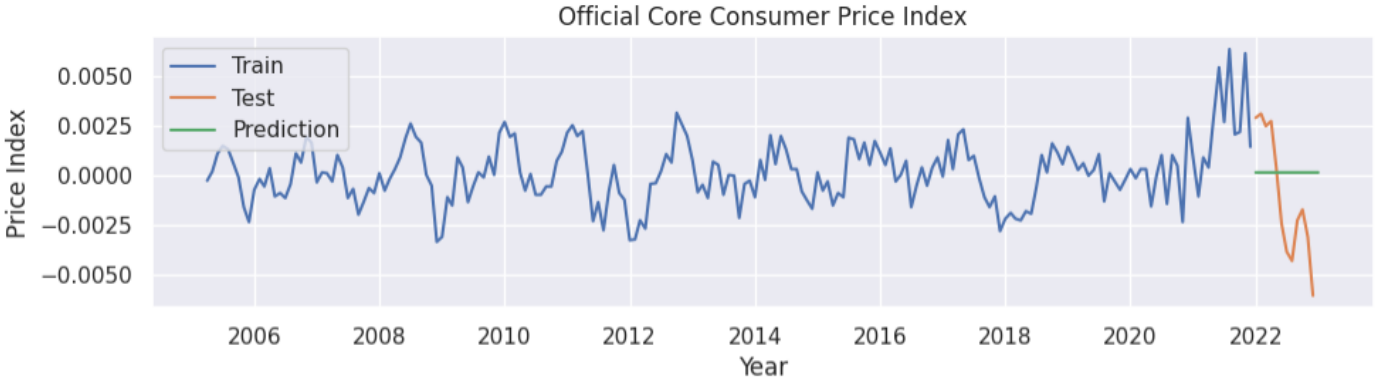
Appendix 5.2

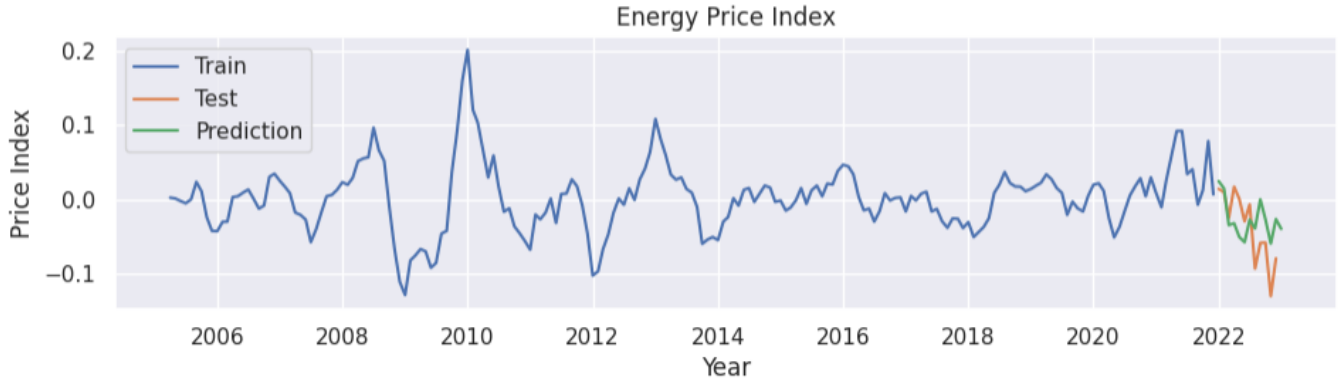

Official Core Consumer Price Index
Train
Test
Prediction
Price Index
0.0050
0.0025
0.0000
−0.0025
−0.0050
2006
2008
2010
2012
2014
2016
2018
2020
2022
Year


Energy Price Index
Train
Test
Prediction
Price Index
0.2
0.1
0.0
−0.1
2006
2008
2010
2012
2014
2016
2018
2020
2022
Year


Food Price Index
Price Index
0.02
0.01
0.00
−0.01
−0.02
Train
Test
Prediction
2006
2008
2010
2012
2014
2016
2018
2020
2022
Year
Headline Consumer Price Index
Price Index
0.010
0.005
0.000
−0.005
−0.010
Train
Test
Prediction
2006
2008
2010
2012
2014
2016
2018
2020
2022
Year
Producer Price Index
Price Index
0.04
0.02
0.00
−0.02
−0.04
Train
Test
Prediction
2006
2008
2010
2012
2014
2016
2018
2020
2022
Year
Diesel
Price Index
0.2
0.1
0.0
−0.1
−0.2
Train
Test
Prediction
2006
2008
2010
2012
2014
2016
2018
2020
2022
Year

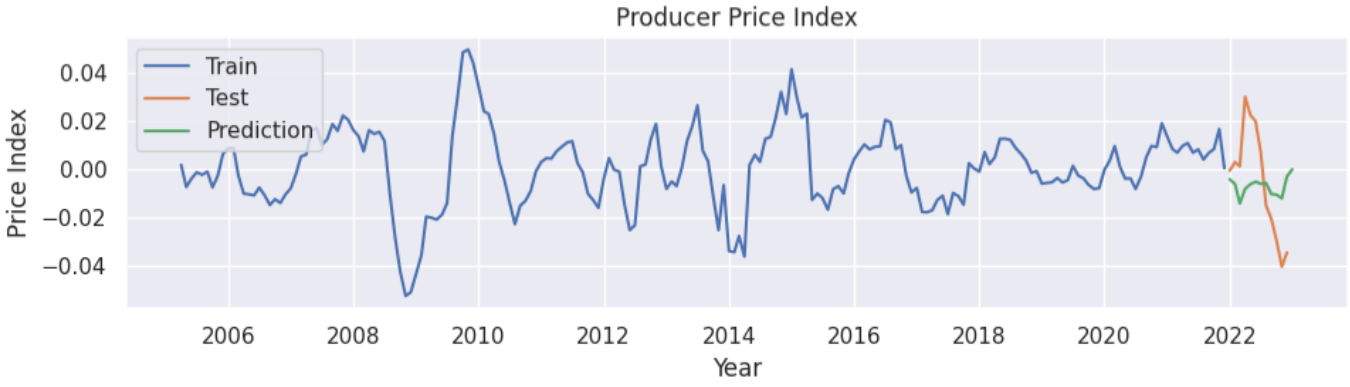
Appendix 5.3

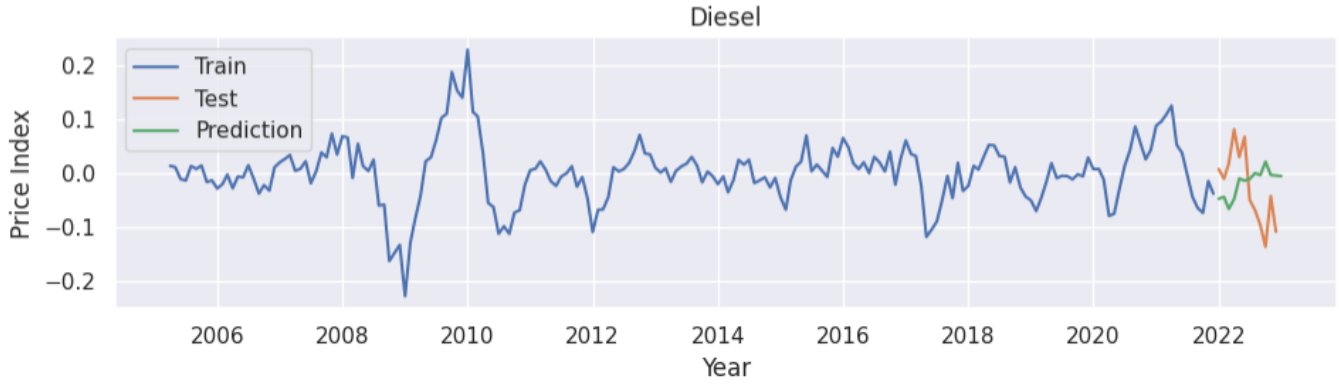

Official Core Consumer Price Index
Train
Test
Prediction
Price Index
0.0050
0.0025
0.0000
−0.0025
−0.0050
2006
2008
2010
2012
2014
2016
2018
2020
2022
Year


Energy Price Index
Train
Test
Prediction
Price Index
0.2
0.1
0.0
−0.1
2006
2008
2010
2012
2014
2016
2018
2020
2022
Year

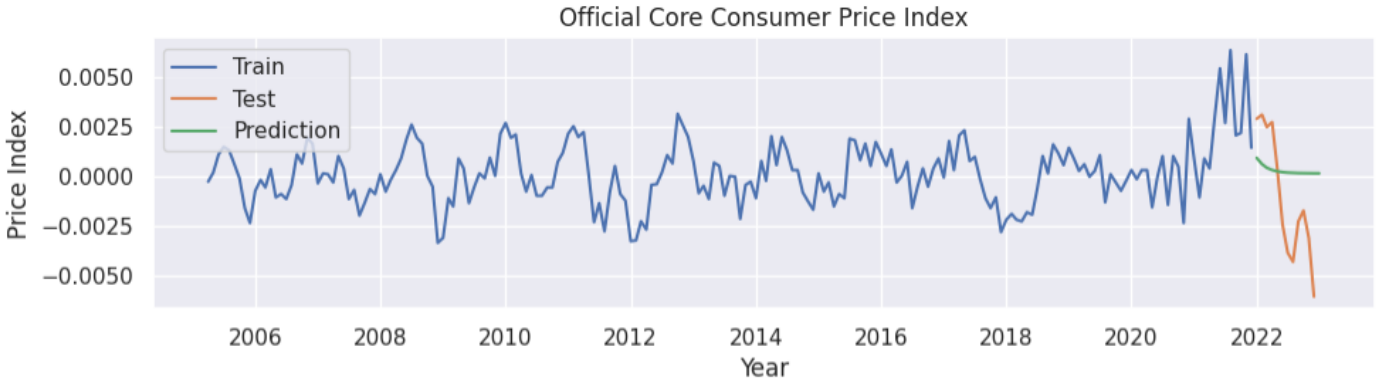

Food Price Index
Train
Test
Prediction
Price Index
0.02
0.01
0.00
−0.01
−0.02
2006
2008
2010
2012
2014
2016
2018
2020
2022
Year

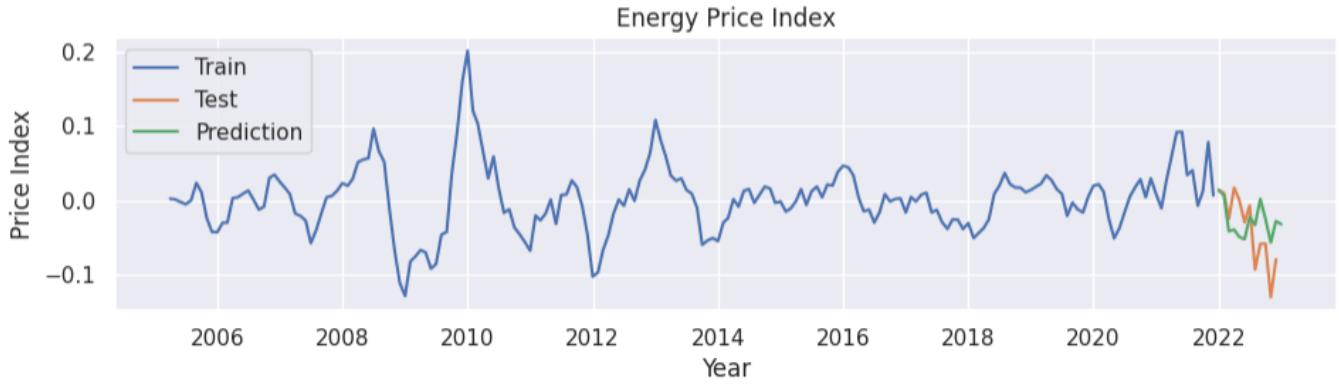

Headline Consumer Price Index
Train
Test
Prediction
Price Index
0.010
0.005
0.000
−0.005
−0.010
2006
2008
2010
2012
2014
2016
2018
2020
2022
Year

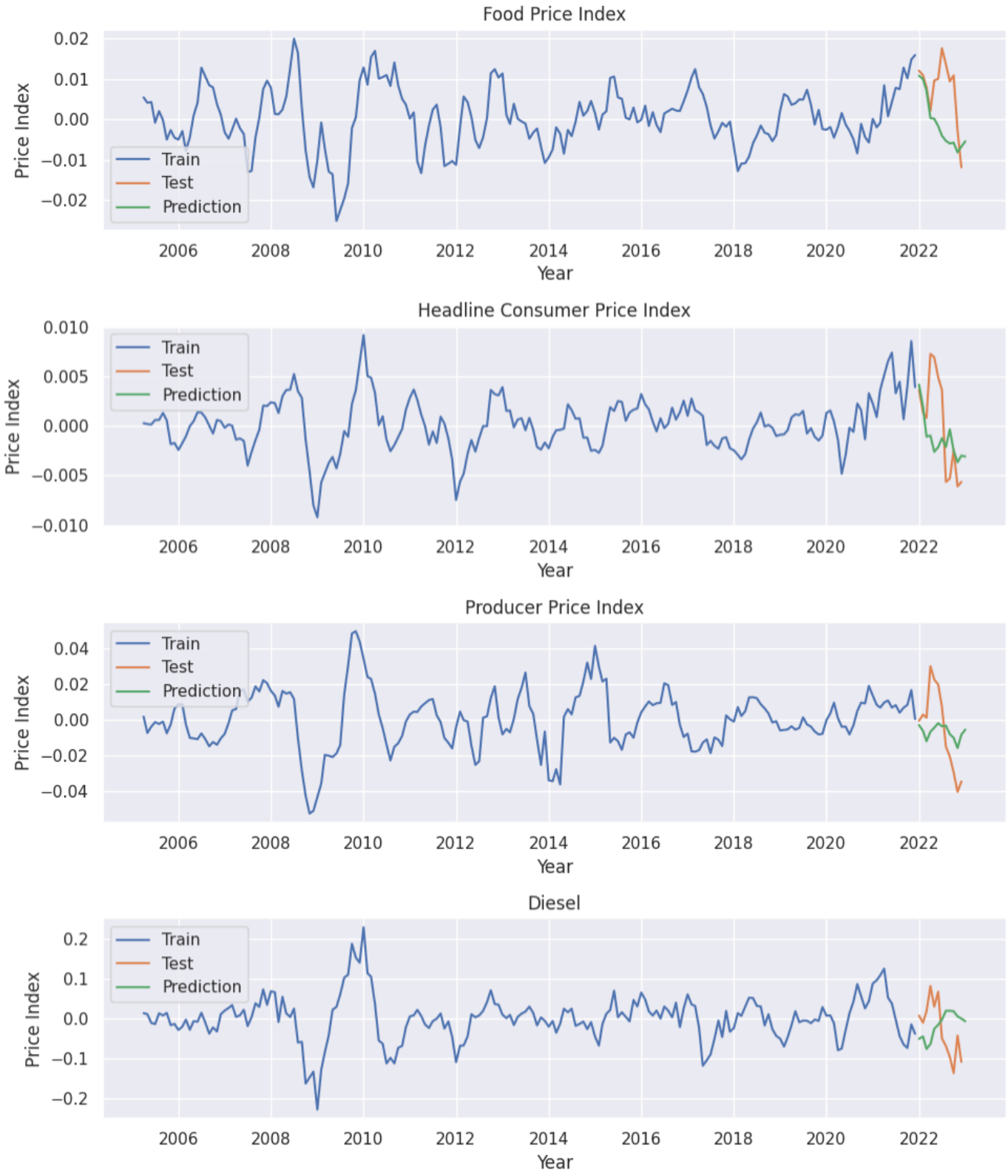

Producer Price Index
Train
Test
Prediction
Price Index
0.04
0.02
0.00
−0.02
−0.04
2006
2008
2010
2012
2014
2016
2018
2020
2022
Year


Diesel
Train
Test
Prediction
Price Index
0.2
0.1
0.0
−0.1
−0.2
2006
2008
2010
2012
2014
2016
2018
2020
2022
Year

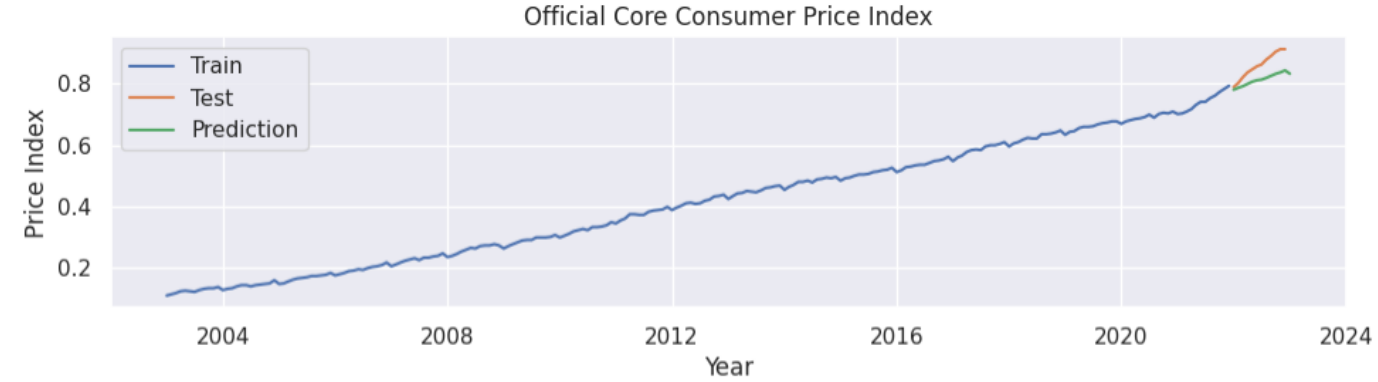
Appendix 5.4


Official Core Consumer Price Index
Train
Test
Prediction
Price Index
0.0050
0.0025
0.0000
−0.0025
−0.0050
2006
2008
2010
2012
2014
2016
2018
2020
2022
Year


Energy Price Index
Train
Test
Prediction
Price Index
0.2
0.1
0.0
−0.1
2006
2008
2010
2012
2014
2016
2018
2020
2022
Year


Food Price Index
Price Index
Year
Train
Test
Prediction
Headline Consumer Price Index
Producer Price Index
Diesel

Appendix 5.5

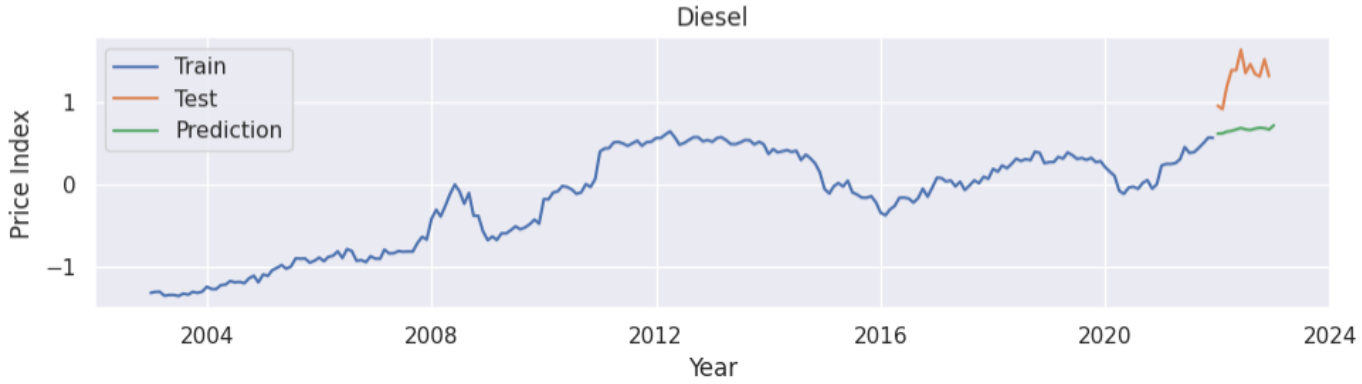

Official Core Consumer Price Index
Train
Test
Prediction
Price Index
0.2
0.4
0.6
0.8
2004
2008
2012
2016
2020
2024
Year


Energy Price Index
Train
Test
Prediction
Price Index
−0.5
0.0
0.5
1.0
2004
2008
2012
2016
2020
2024
Year

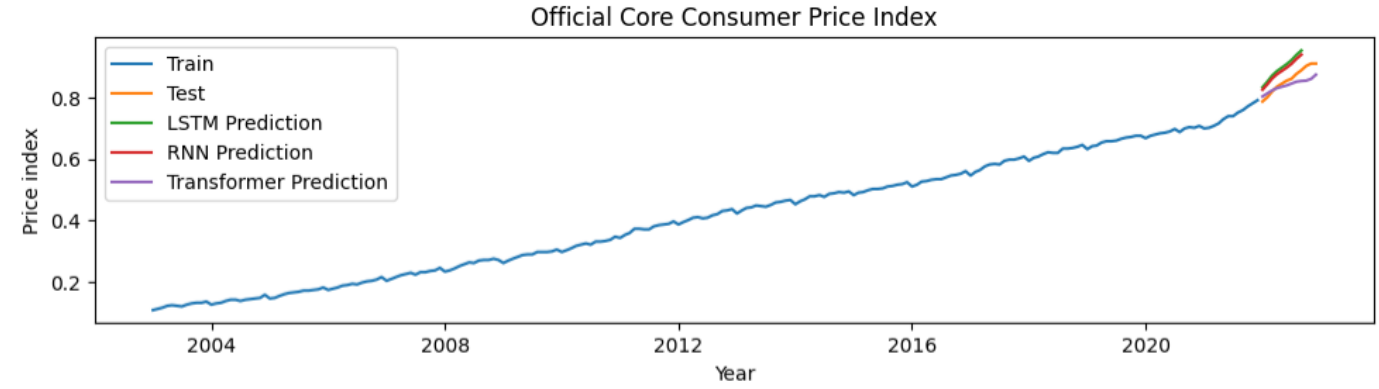

Food Price Index
Train
Test
Prediction
Price Index
0.25
0.50
0.75
1.00
2004
2008
2012
2016
2020
2024
Year

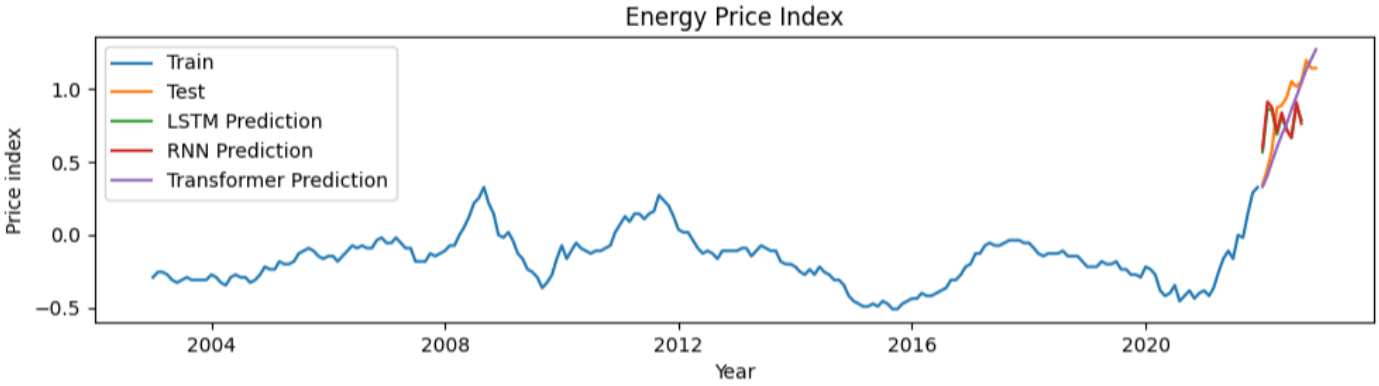

Headline Consumer Price Index
Train
Test
Prediction
Price Index
0.2
0.4
0.6
0.8
1.0
2004
2008
2012
2016
2020
2024
Year

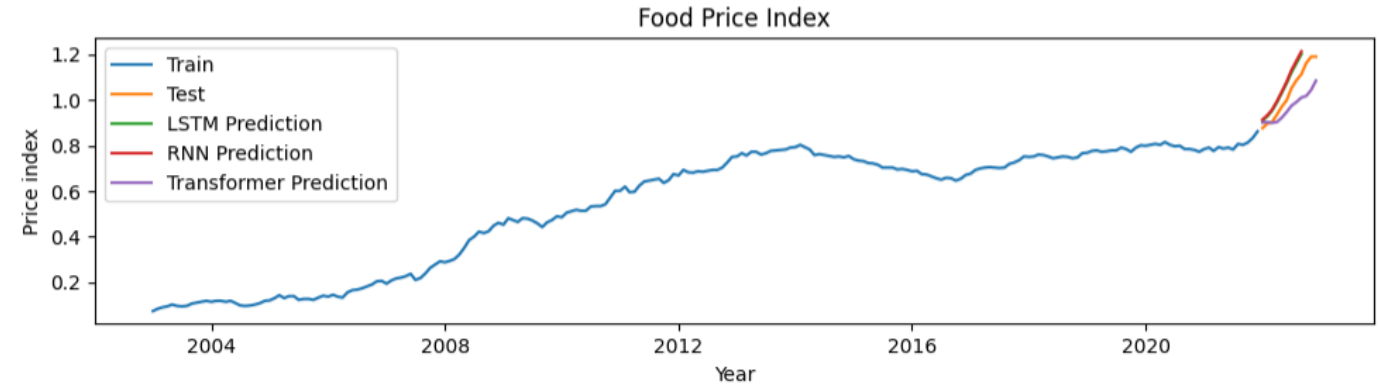

Producer Price Index
Train
Test
Prediction
Price Index
1.25
1.00
0.75
0.50
0.25
0.00
2004
2008
2012
2016
2020
2024
Year

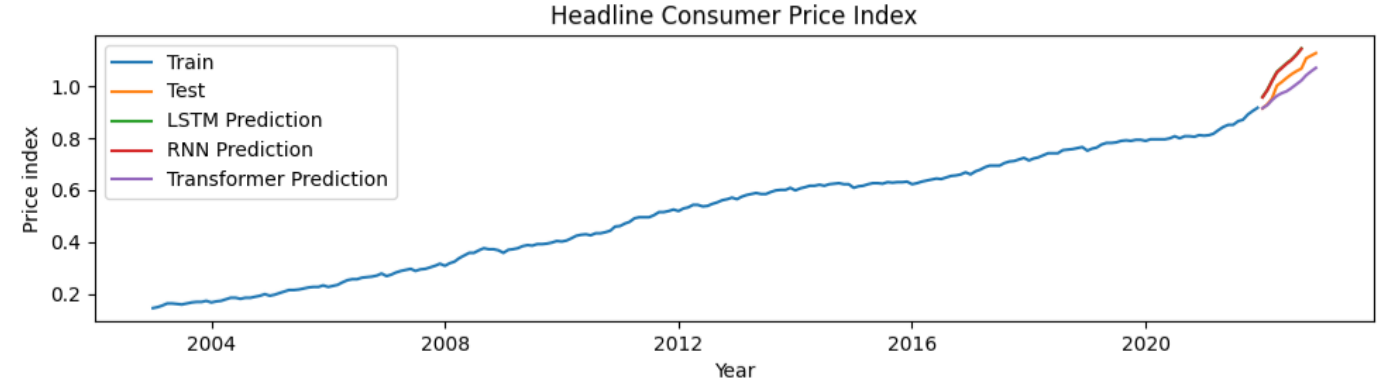

Diesel
Train
Test
Prediction
Price Index
1
0
−1
2004
2008
2012
2016
2020
2024
Year

Appendix 5.6

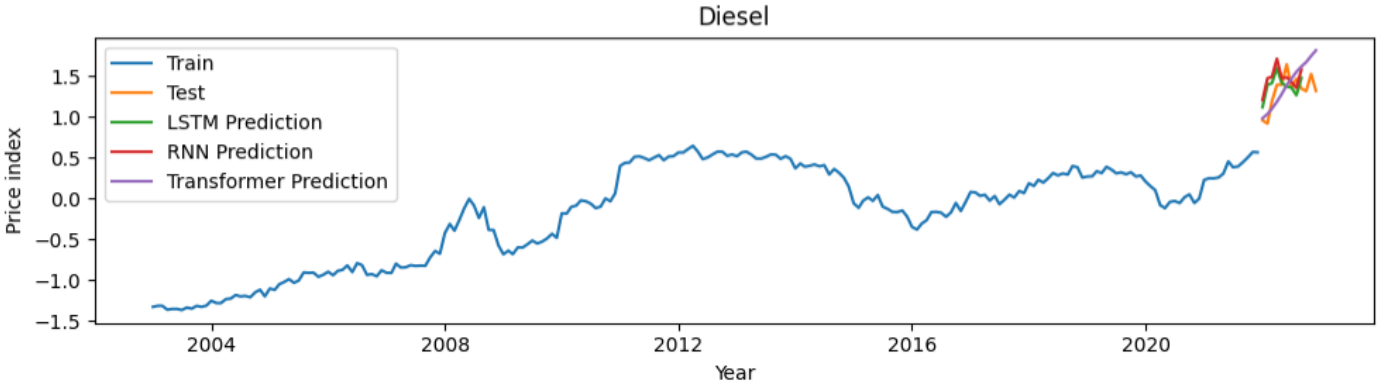

Official Core Consumer Price Index
Train
Test
LSTM Prediction
RNN Prediction
Transformer Prediction
Price index
0.8
0.6
0.4
0.2
2004
2008
2012
2016
2020
Year


Energy Price Index
Train
Test
LSTM Prediction
RNN Prediction
Transformer Prediction
Price index
1.0
0.5
0.0
−0.5
2004
2008
2012
2016
2020
Year

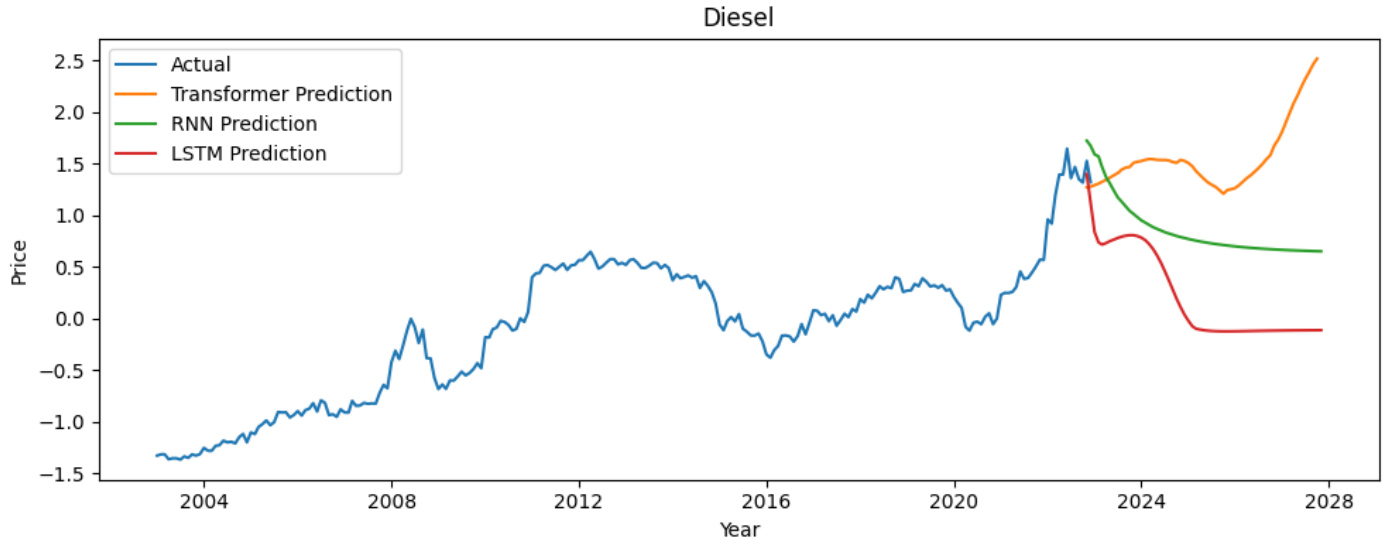

Food Price Index
Train
Test
LSTM Prediction
RNN Prediction
Transformer Prediction
Price index
Year
2004
2008
2012
2016
2020


Headline Consumer Price Index
Train
Test
LSTM Prediction
RNN Prediction
Transformer Prediction
Price index
Year
2004
2008
2012
2016
2020


Producer Price Index
Train
Test
LSTM Prediction
RNN Prediction
Transformer Prediction
Price index
Year
2004
2008
2012
2016
2020

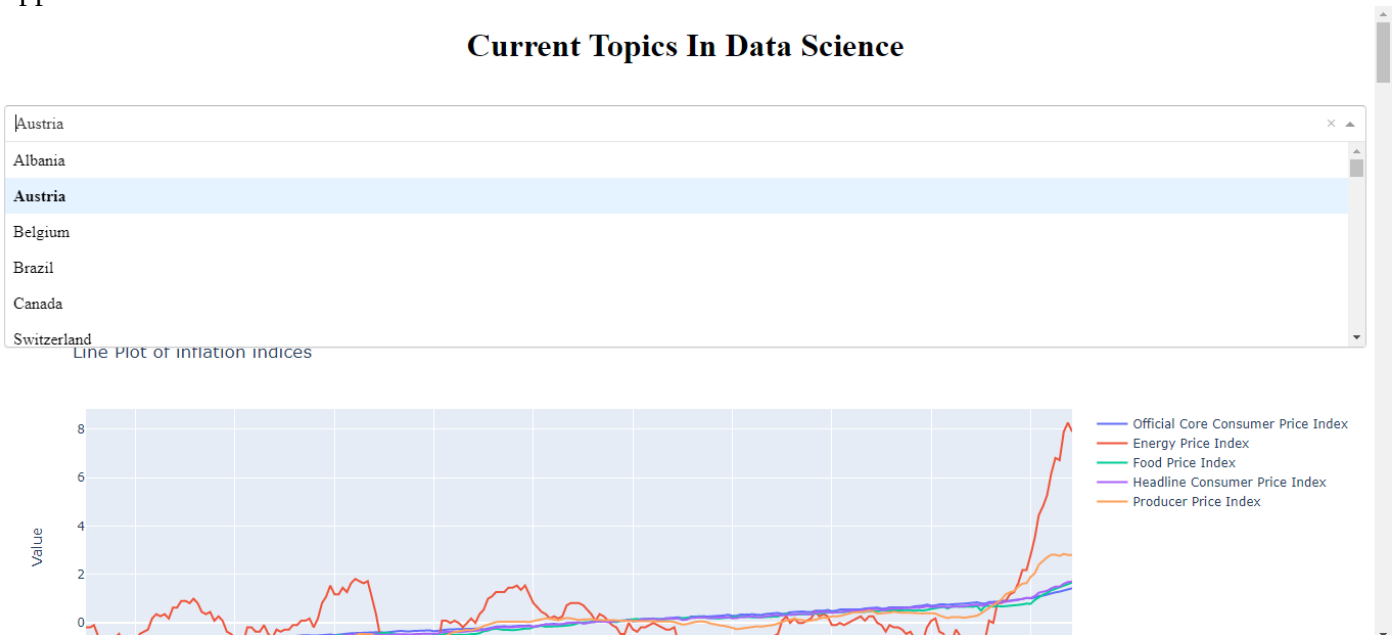

Diesel
Train
Test
LSTM Prediction
RNN Prediction
Transformer Prediction
Price index
Year
2004
2008
2012
2016
2020

Appendix 5.7



Appendix 6

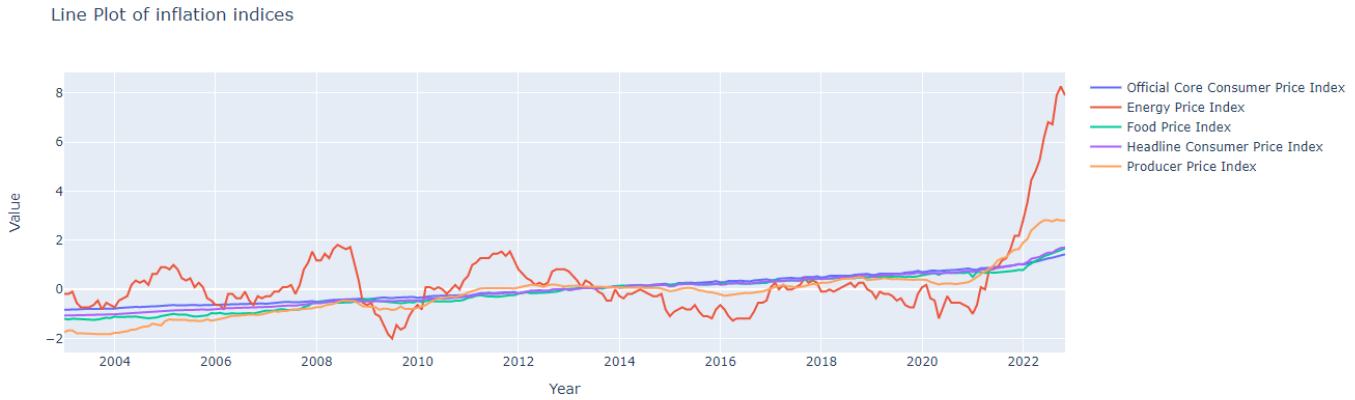
Appendix 6.1



Appendix 6.2

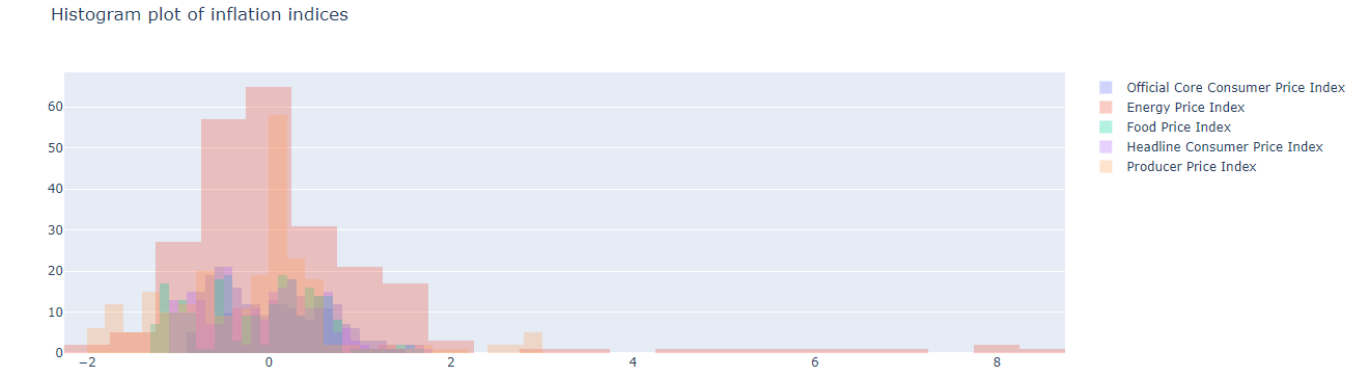
Country Selected is : Austria

Select year range to view Line Plot plots for Inflation indices



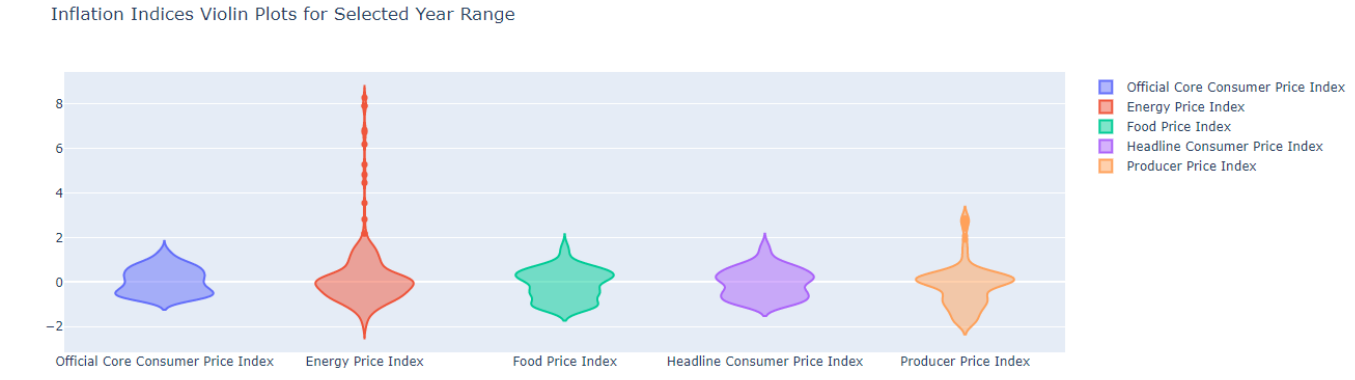
Appendix 6.3

Select year range to view Histogram plots for Inflation indices



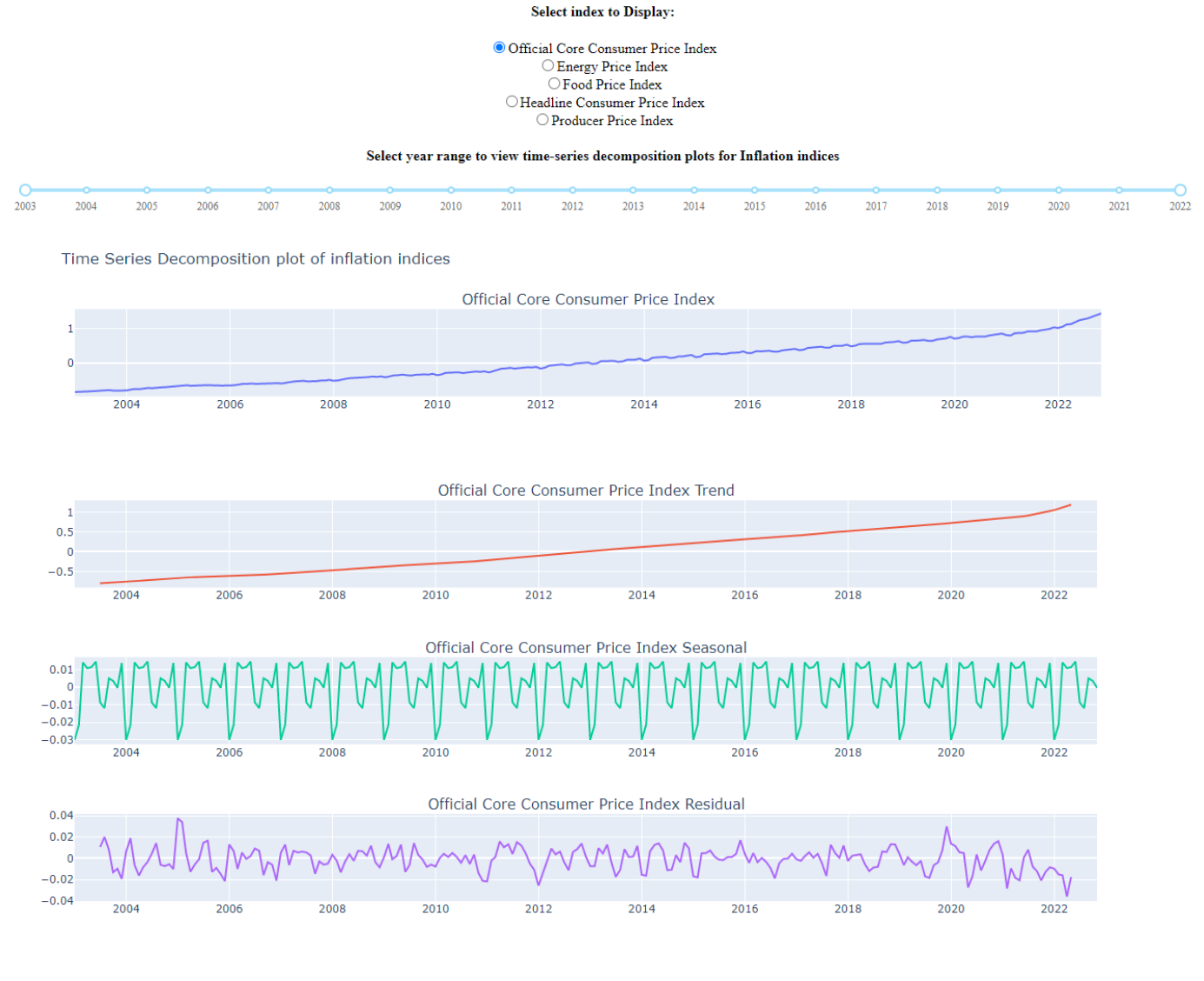
Appendix 6.4

Select year range to view violin plots for Inflation indices

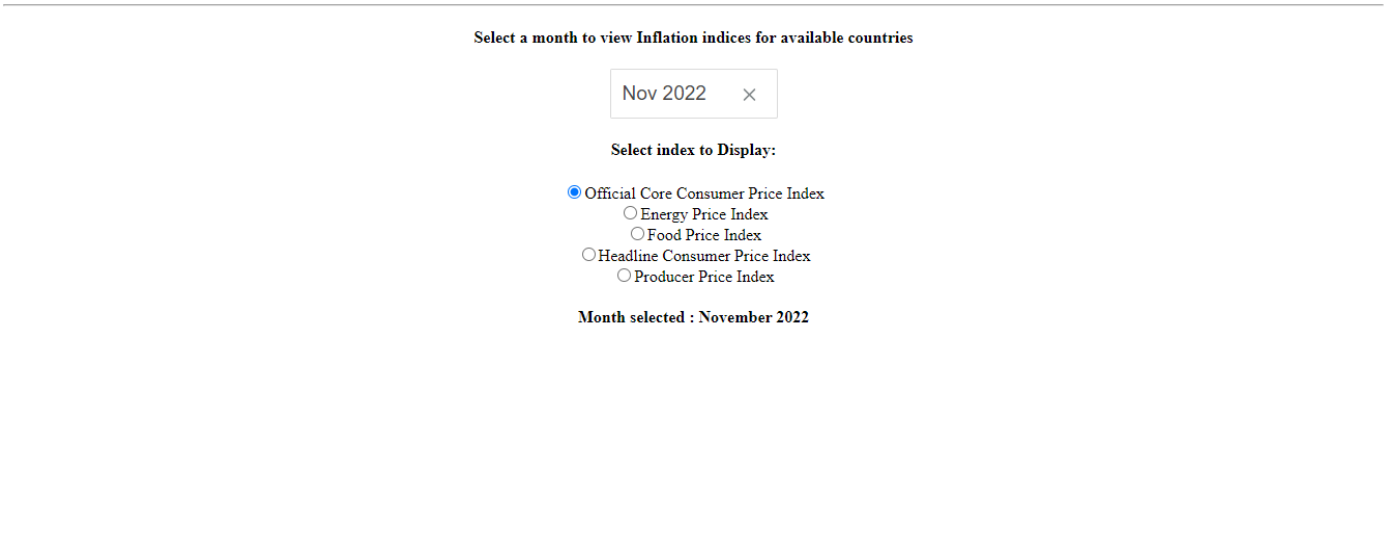


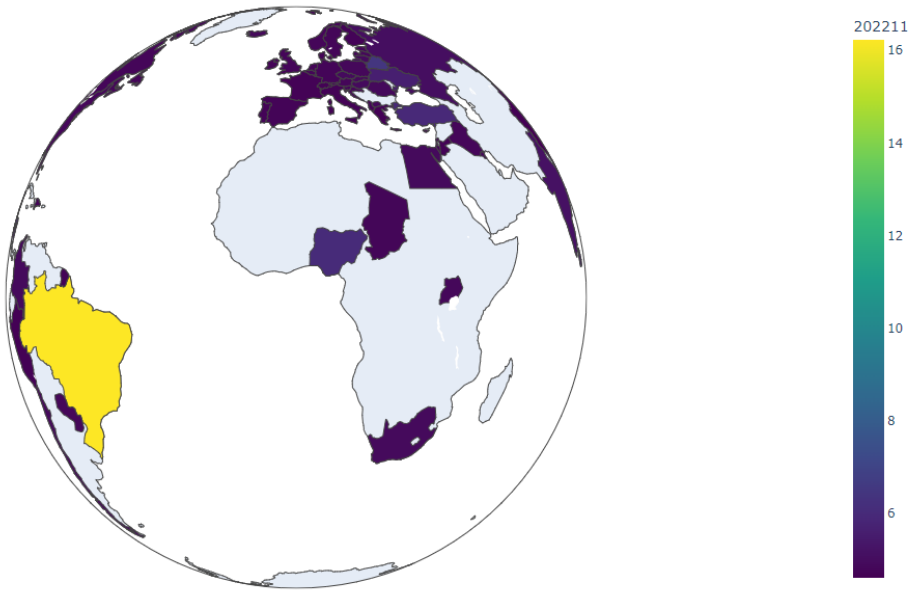
Appendix 6.5

Appendix 6.6


Select index to Display:
Official Core Consumer Price Index
Energy Price Index
Food Price Index
Headline Consumer Price Index
Producer Price Index
Select year range to view time-series decomposition plots for Inflation indices
2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020 2021 2022
Time Series Decomposition plot of inflation indices
Official Core Consumer Price Index
Official Core Consumer Price Index Trend
Official Core Consumer Price Index Seasonal
Official Core Consumer Price Index Residual

Appendix 6.7


Select a month to view Inflation indices for available countries
Nov 2022
Select index to Display:
Official Core Consumer Price Index
Energy Price Index
Food Price Index
Headline Consumer Price Index
Producer Price Index
Month selected : November 2022



Appendix 6.8

Inflation predictions

Select index to Display:

- (●) Official Core Consumer Price Index
- ( ) Energy Price Index
- ( ) Food Price Index
- ( ) Headline Consumer Price Index
- ( ) Producer Price Index

Select Start and End months:

Nov 2022 → Jan 2024 ×

Select precision level:

High ×

Start Month : November 2022, End Month : January 2024, Precision : high