Predicting CPI: A Time Series Analysis

Nam Tran, Juliette Patterson, and Ryan Polonski

2023

Table of Contents

[01. Executive Summary 2](#_Toc135144671)

[02. Introduction 3](#_Toc135144672)

[03. Data and Preprocessing 4](#_Toc135144673)

[3.1 Data Description 4](#_Toc135144674)

[3.2 Label: CPI 5](#_Toc135144675)

[3.3 Predictors 10](#_Toc135144676)

[04. Statistical Test 12](#_Toc135144677)

[4.1 White Noise 12](#_Toc135144678)

[4.2 ADF 13](#_Toc135144679)

[4.3 KPSS 14](#_Toc135144680)

[05. Method 15](#_Toc135144681)

[5.1 Splitting the Data 15](#_Toc135144682)

[5.2 Base Model: ARIMA (1,2,1) 15](#_Toc135144683)

[5.3 Other Versions of ARIMA 16](#_Toc135144684)

[5.4 Random Forest 16](#_Toc135144685)

[5.5 Long Short-Term Memory 19](#_Toc135144686)

[06. Analyses And Results 20](#_Toc135144687)

[6.1 Statistical Discussions 20](#_Toc135144688)

[6.2 Result’s Discussion 21](#_Toc135144689)

[07. Conclusion 23](#_Toc135144690)

[References 25](#_Toc135144691)

[Appendix A 26](#_Toc135144692)

# 01. Executive Summary

The project’s purpose is to create a time series analysis to predict the Consumer Price Index (CPI). Specifically, the Consumer Price Index of Urban Consumers (CPI-U) is the direct figure used to measure inflation. Since a prediction model can normally predict one label, predicting CPI values allows us to calculate both monthly and yearly inflation. The business insight we are trying to achieve is to accurately forecast inflation rates. By doing this, we can provide valuable information on inflation including trends, causes, and effects. Additionally, predicting CPI trends can be crucial to businesses and the economy. Making informed decisions related to inflation, economic policy, and financial planning is pivotal to the performance of the economy.

Through the usage of Python, we attempt to develop a time series model accurately predicting the CPI. Using historical data along with external factors such as Unemployment Rate, Wages and Salaries, Exchange Rates, and other relevant economic metrics, our goal is to build a comprehensive model that can forecast CPI values while proving to be accurate.

Exploring different time series techniques, we conducted several preprocessing steps to analyze our data, such as label decomposition and stationarity tests. We implemented several different models such as ARIMA, ARIMA using different moving averages and autoregressions, Random Forest, and Long Short-Term Memory (LSTM). Furthermore, the Consumer Price Index data had to be transformed using second-order differencing. Given this transformation, we focused on improving the results of our models using Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Square Error (RMSE). The LSTM provided the best predictive performance on the second-order differenced data, but the optimized random forest model with the label’s lags incorporated has the least RMSE and MAE when reversing back to the monthly inflation rate with the original CPI data. Our results will thoroughly be explained in section five, Method, and section six, Analyses and Results.

The goal of this project is to exhibit our knowledge and understanding of conducting a time series analysis while also focusing on predicting a key measure of inflation. Possessing the capability to predict the CPI would be extremely useful for monetary policy, business planning, financial planning, among other economic decision-making activities. It will also provide great insight into inflation trends and patterns based on key economic factors.

# 02. Introduction

Inflation is the long-term, steadily rising level of prices for most products and services across the economy. Each unit of currency can purchase less than it used to when inflation is present. The Consumer Price Index (CPI) is frequently used to measure it, among other things.

The Consumer Price Index is an indicator of inflation, and it records the average price changes of goods and services over time. The Bureau of Labor Statistics introduced the concept in the United States in 1913. Initially developed to aid in understanding how price changes affected the purchasing power of the average consumer, it tracks variations in the cost of a set basket of goods and services that are representative of normal household spending in urban areas. There are two common versions of the CPI, one being the CPI for all urban consumers (CPI-U) and the other is CPI for urban wage earners and clerical workers (CPI-W). The CPI-U is designed to measure price changes faced by urban consumers, who represent 93% of the U.S. population.

The CPI metric has been improved throughout time to represent shifts more precisely in consumer behavior and market dynamics. Governments, companies, and individuals utilize the CPI to make educated decisions about financial planning, contracts, and economic policy. It helps evaluate changes in the cost of living and offers a useful picture of inflation trends.

Many nations use different iterations of the CPI to keep track of inflation and direct monetary policy. It is still a crucial tool for comprehending how price fluctuations affect both daily living and the economy. CPI can be used as an economic indicator. It is the most widely used measure of inflation and deflation. Changes in this figure affect nearly every market and person in the United States. Therefore, it is an indicator of the effectiveness of government policy. Business executives, labor leaders, and other private citizens use the index as a guide in making economic decisions.

For this project we will be working with historical CPI data, and we have extracted 16 predictors (See Table 1). All data was collected from 1970 and cut to focus on 2010-2020. Our goal is to accurately create a time series model to predict the Consumer Price Index and give more insight into economic and inflation trends.

# 03. Data and Preprocessing

## 3.1 Data Description

The dataset consists of historical economic data from 1970-present. The label (dependent variable) is the Consumer Price Index. We carefully and meticulously picked sixteen predictor variables, as shown in Table 1. The dataset consists of numerical variables only. All the variables were taken from the Federal Reserve Economic Data (FRED)[[1]](#footnote-1), including the CPI data compiled. After carefully considering the underlying structure of the data, we decided to build models using the period 2010-2020. Additionally, when building our dataset, we focused on economic variables that we thought would have the most impact on inflation and the Consumer Price Index. Our data was compiled with the purpose of accurately predicting CPI with meaningful features.

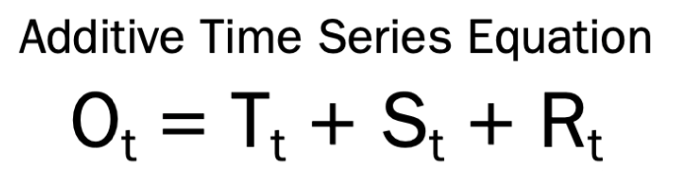


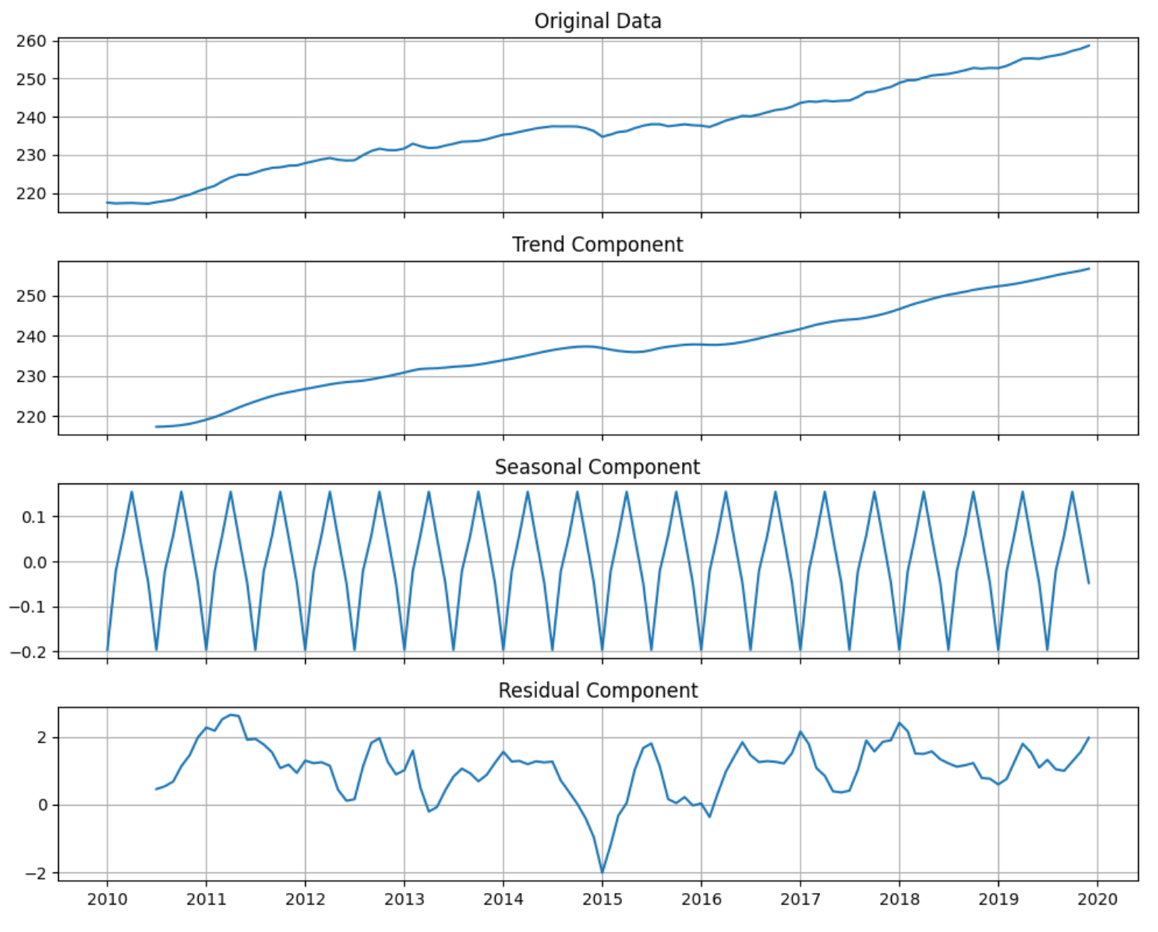
**Table 1**

## 3.2 Label: CPI

### 3.2.1 Decomposition

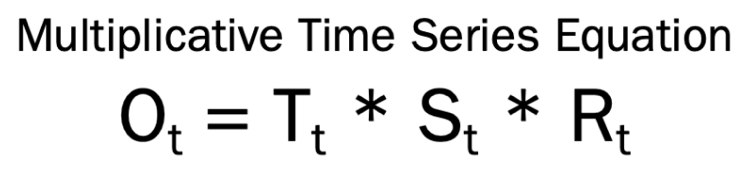
Additive and Multiplicative decomposition are two different approaches to analyzing and breaking a time series down into three components, trend, seasonality, and residuals. Additive decomposition is more appropriate when values are increasing by the same amount. Multiplicative decomposition is more appropriate when values are increasing by different values. Both approaches were performed on the label, CPI.

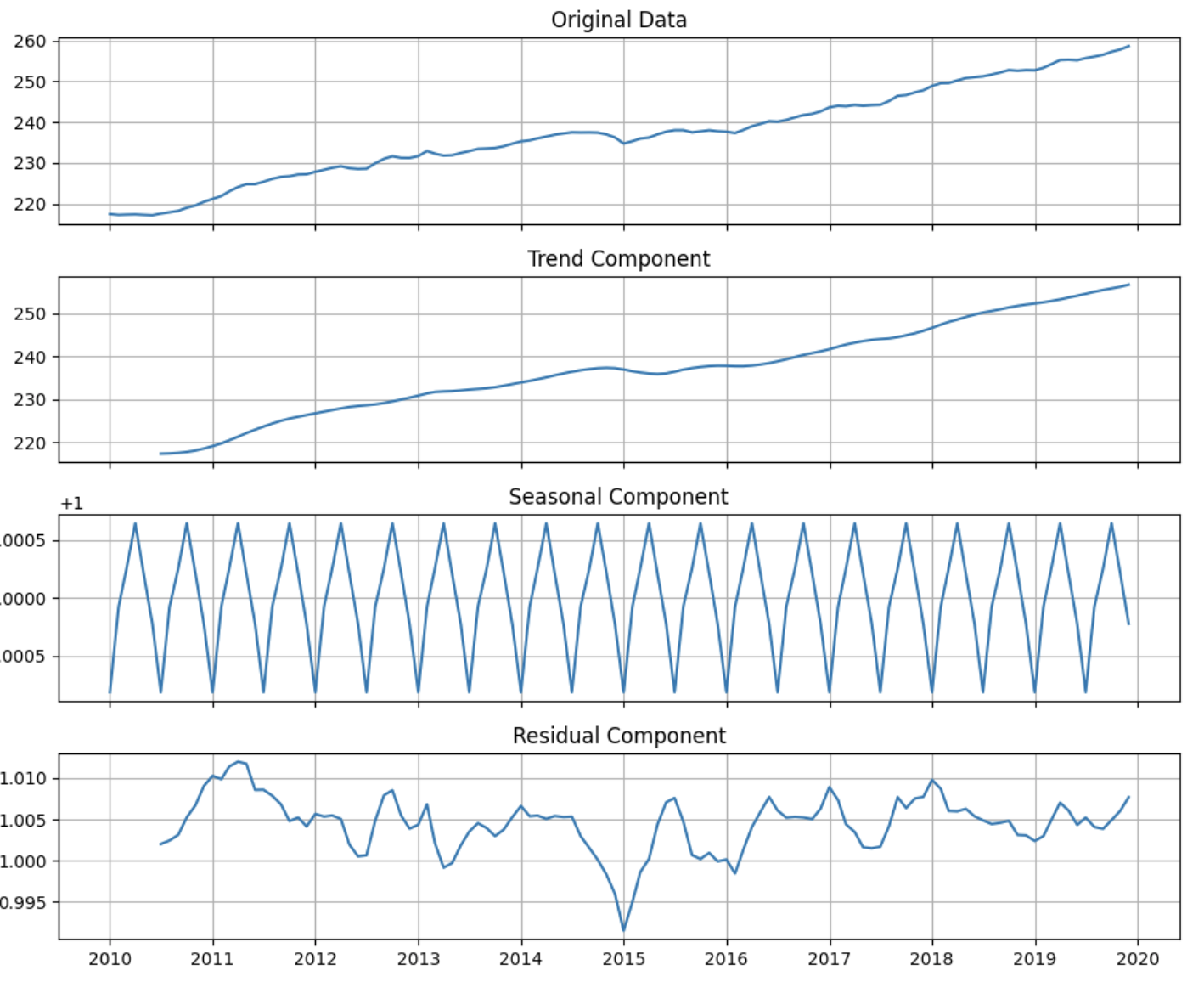
[[2]](#footnote-2)



**Figure 1.1**

First, we decomposed the Consumer Price Index into trend, seasonal, and residual components using the additive method, as shown in Figure 1.1. The trend component appears to be the most significant, given that it mirrors the original data the most. Additionally, the seasonal component exhibits perfect seasonality. Since the seasonal component was removed from the original data, we can visualize cyclical trends better. The seasonal component is a regularly repeating pattern. In this decomposition method, it appears that the variance in the early series is smaller than in the latter series.



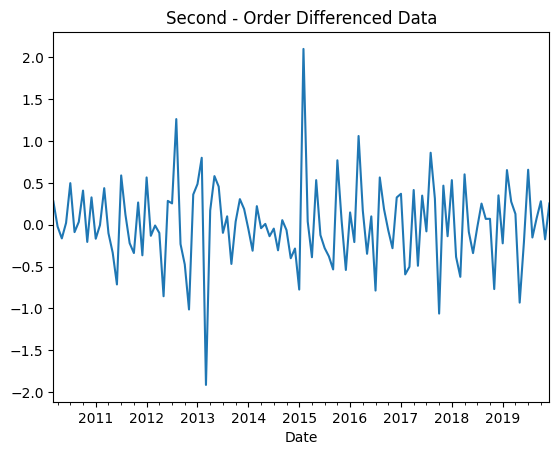
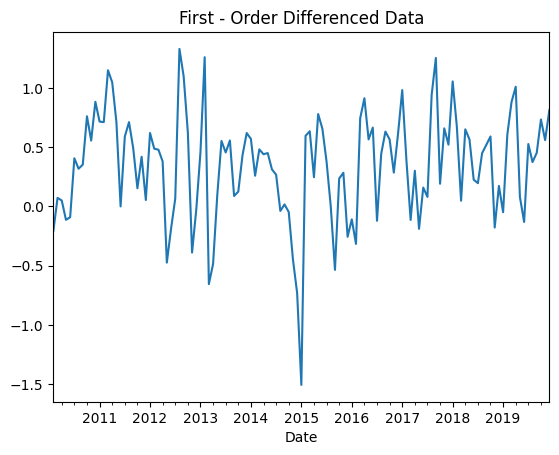


**Figure 1.2**

Once applying the additive decomposition, we moved on with the multiplicative method, as shown in Figure 1.2. In this method, the results are like that of the additive method. Therefore, the trend component is the most significant since it mimics the data almost exactly. Additionally, the seasonality of the multiplicative method is a regularly repeating pattern.

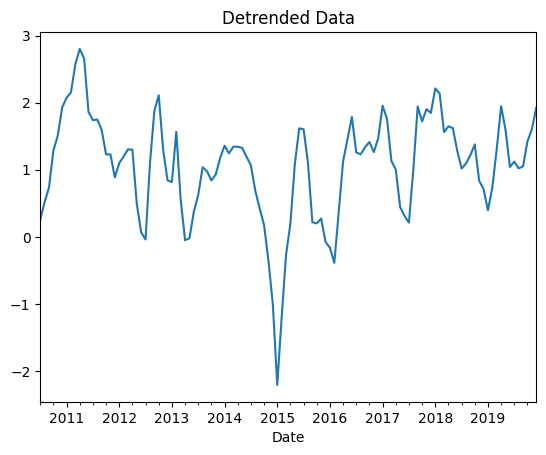
After trying multiple periods/frequencies, we decided to use a period of six to decompose the CPI index as it results in the perfect seasonal component. As we can see from the original data and the seasonal component, seasonality does not have a significant effect on the CPI, even after multiple decomposition periods we conducted. Intuitively, prices often tend to increase year over year and rarely drop back to the previous year’s level. As a result, we can conclude that the seasonal component is insignificant in the current data. However, the upward trend of the data seems to be very consistent, linear, and smooth. The additive and multiplicative decomposition show that the trend component is the most important component in the CPI index. The trend component has the largest range of y-axis values, so in each respective equation it holds the most impact.

### 3.2.2 Trend Analysis and Differenced Data



**Figure 2.1**

Since we have decomposed the label to get a deeper understanding of the series' components, we move on to data differentiating and detrending. Differencing is performed by calculating the differences between the Consumer Price Index’s consecutive values. By doing this, our goal is to eliminate any trends or patterns in the data, making it stationary and easier to analyze. Differentiating allows us to observe changes over time and improve our models and forecasting by subtracting each data point from the one prior. There are different types of differentiating the data, and we will perform first-order differencing and second-order differencing. The first-order loses the first value of data while second-order loses the first two values of data. To perform second-order differencing, first-order must be completed. Figure 2.1 shows the results of first and second-order differencing on the label. After further investigation of the rolling mean and standard deviations of these values, we concluded that second-order differencing generated a more stable rolling mean and standard deviation compared to first-order differencing (See Appendix A, Graph 1).



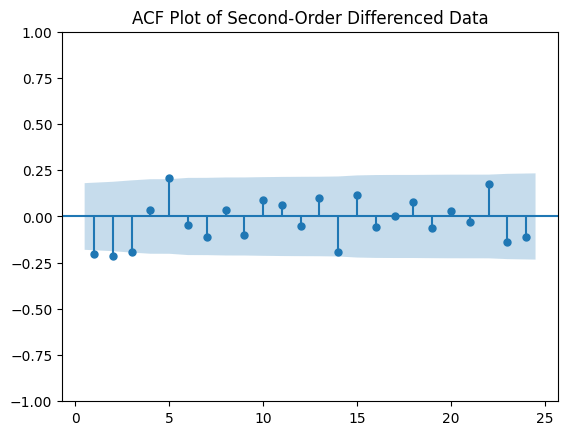
**Figure 2.2**

Detrending data is the process of removing the trend component from the original data. In other words, detrended data is computed by subtracting the trend values from the actual values. Detrending helps to analyze and identify any underlying variations, making it easier to recognize patterns. Since we used a period of six to smooth the data, the function uses a centered moving average with a window size of six to smooth the trend component. That is, six periods prior to the current value are used. As a result, we lose six observations in using label detrending, compared to only losing the first and second value in differenced data. Figure 2.2 shows the results of the detrended data. The rolling mean and standard deviation of the detrended data showed inconsistent results; therefore, the results were not stable.

After further investigation of the three transformed methods, we concluded that the mean and variance of the three data are not constant over time. Therefore, of the three methods, second-order differencing appears to be the most stationary, and thus we will move forward with second-order differencing to represent the label.

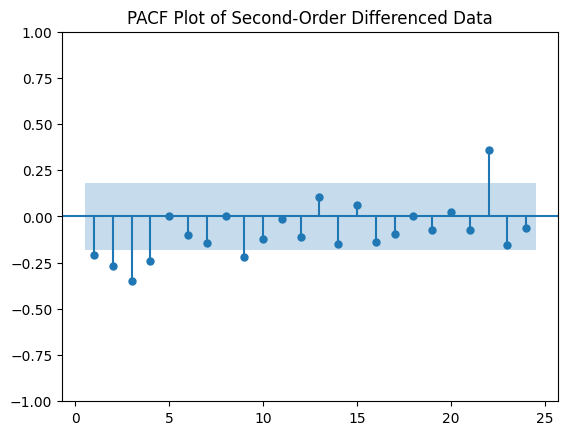
### 3.2.3 Lag Analysis

To identify the useful lag variables (previous values), we can use the autocorrelation function (ACF) and the Partial Autocorrelation Function (PACF) plots. The main difference between ACF and PACF is that ACF measures the total correlation between a time series and its lagged values, while PACF measures the direct correlation between a time series and its lagged values after removing the effect of the correlations with the intervening observations. ACF is primarily used to determine the moving-average (MA) component, while PACF is used to determine the autoregressive (AR) component.



**Figure 3.1**

The ACF plot shown in Figure 3.1 tells us that the label is correlated with its lagged values up to the third period. The shaded area is the significance level in both plots. If a lagged variable appears to be out of the shaded area, it is significantly correlated with the label.

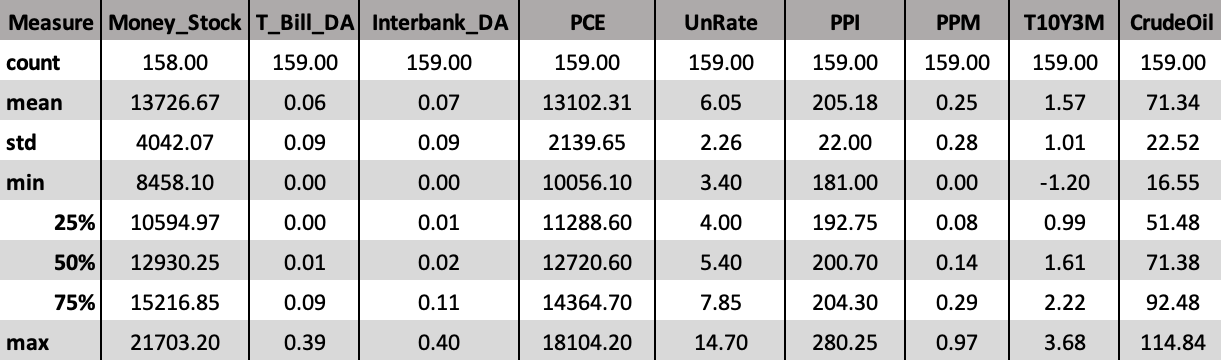


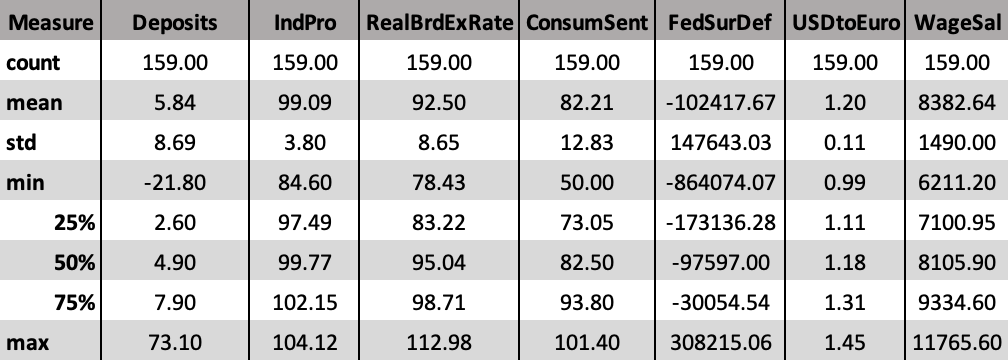
**Figure 3.2**

Figure 3.2 shows the PACF plot, and from this plot we can gather that the label is directly correlated with the first four lagged values, as well as lags nine and twenty-two. It is important to note that we cannot be sure that lag twenty-two is substantially significant as it shows on the graph due to the small size of the data.

## 3.3 Predictors

### 3.3.1 Statistical Description

**Table 2.1**

**Table 2.2**

Tables 2.1 and 2.2 show the statistical summary of the predictor variables used. There are sixteen total variables, which are all described in Table 1. As shown, these variables contain key economic metrics and values that we considered to be important or impactful to the Consumer Price Index. Money\_Stock, PCE, CrudeOil, RealBrdExRate, USDtoEuro, Deposits, and WageSal all follow an abnormal distribution. T\_Bill\_DA, Interbank\_DA, UnRate, PPI, and PPM appear to be right skewed. T10Y3M has a relatively normal distrbution. IndPro, ConsumSent, and FedSurDef all appear to be left skewed. Given the statistics of the raw data for the selected predictors, we will move forward with data cleaning including treating outliers and normalizing the predictors.

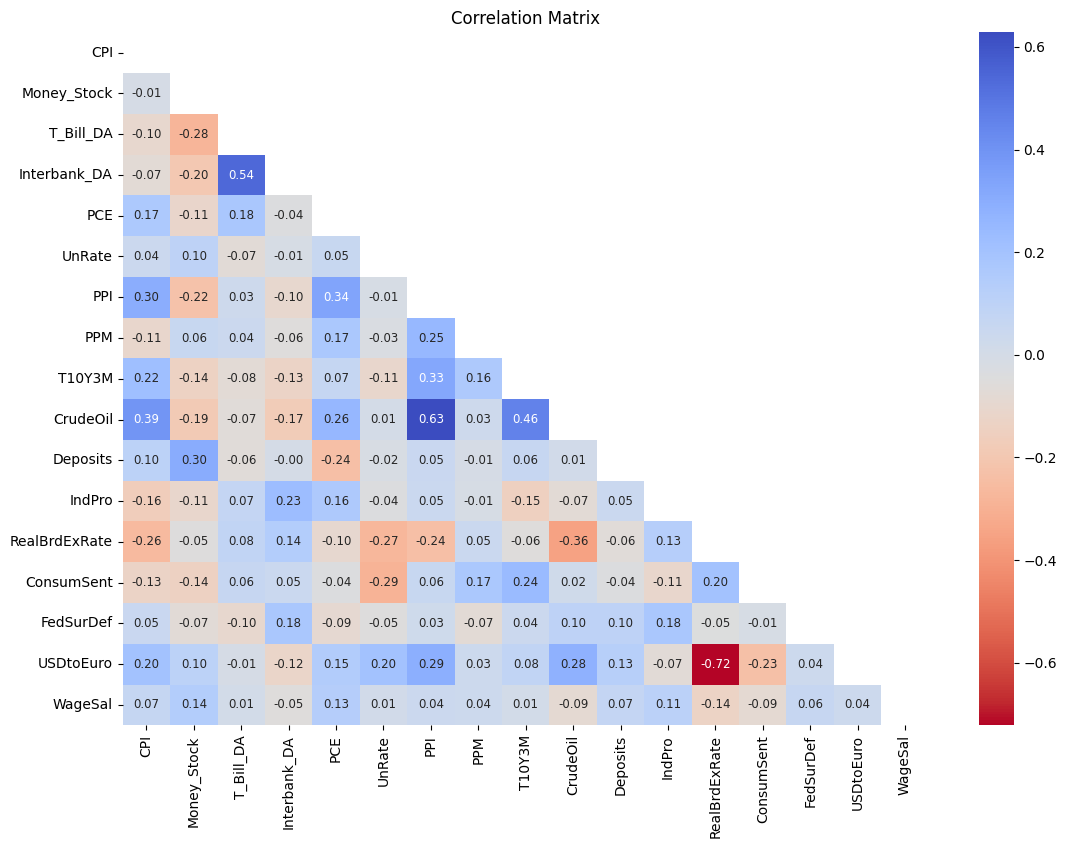
### 3.3.2 Predictor Cleaning

Since we have completed data difference on CPI and decided to move forward with second-order differencing, it makes sense to transform the predictors to at least first-order differencing as well. Additionally, we would like to see how the change in these variables affect the movement of the label.

While the predictors are deemed to be of equal importance, they are scaled differently. Once we explored and visualized the predictors, we treated the outliers using the Interquartile Range Method. In other words, we created an upper bound and a lower bound to get rid of values below or above this range. By doing this, we were able to eliminate the skewness of the predictors.

With the cleaned predictors, we moved on to normalization. Using the min-max scaler technique, we were able to normalize the predictors to a particular range. The min-max scaler method uses the minimum and maximum values of each respective predictor to scale the data.

### 3.3.3 Correlation Analysis



**Figure 4**

Figure 4 shows a correlation matrix of the cleaned predictors and the label. Most features are moderately or weakly correlated with CPI. In an economic sense, they should have a strong correlation with the label; however, since both the label and the predictors have been differentiated, the strong correlation no longer holds. As shown in Figure 4, Money\_Stock and FedSurDef are found to have small correlation with the label, though they still might be important economic factors to inflation and Consumer Price Index. In addition, since correlation measures only linear relationships, non-linear relationships between predictors and the label can be significant and useful for prediction.

# 04. Statistical Test

## 4.1 White Noise

Before we attempted to go any further or build our model, we ran a white noise check on our data. If a time series has white noise, it means that the series is a random sequence of numbers and has no predictive ability. White noise has no underlying structure or relationships between its data points, and any attempt to build a model for predicting future values is likely to be unsuccessful. There are some statistical tests we can use to check if data shows white noise or not, in our case we used the Ljung-Box test.

The Ljung-Box test is done to check if autocorrelation is present in the data. If there is no autocorrelation in the data, then there might be white noise. For the test we have a Null and Alternative Hypothesis, and the Null Hypothesis states that the data is independently distributed, meaning there is no autocorrelation, while the Alternative Hypothesis claims that the data is not independently distributed and there is autocorrelation in the data. Our desired result is to reject the Null Hypothesis and prove there is some level of autocorrelation in our data.

The Ljung-Box test is an iterative process that tests each current value against each lag. The p-value of each test ran is then compiled into a list that is created to store the values. The last step is to then check these lists to see if all the p-values are less than our confidence level of 0.05. If this is the case, then we have proven some level of autocorrelation between the current values and their lags and there is no white noise. The results from our Ljung-Box test implied that the time series is not white noise. The series illustrates that there is a correlation between the current values and the lagged values.

## 4.2 ADF

The Augmented Dickey-Fuller (ADF) test is used to statistically verify whether the data being used is stationary or not. For a time series data to be stationary, the data must have a constant mean, constant variance, constant autocorrelation structure, and no seasonal components over time. The Null Hypothesis for this test claims that the data is not stationary while the Alternative Hypothesis states that the series is stationary. A significance level is defined to conclude whether to accept or reject the Null or Alternative Hypothesis, and this level is generally a p-value of 0.05, which we use in our test. To confirm whether is it stationary or not, we need a p-value that is lower than the significance level to reject the Null Hypothesis. Additionally, the critical values should be greater than the ADF statistics.

|  |  |
| --- | --- |
| **Metric** | **Value** |
| ADF Statistics | **-**5.7738 |
| P-value | 0.0000 |
| Critical Values | ‘1%’: -3.4924  ‘5%’: -2.8887  ‘10%’: -2.5813 |

**Table 3**

Given that we moved forward with the second-order differenced data for the label, we conducted the ADF test on this data. The results from the ADF test are shown in Table 3, and from the results we can gather that the data is stationary. Given a p-value lower than 0.05, we can move forward and reject the Null Hypothesis; therefore, the Alternative Hypothesis stands true. Furthermore, the critical values reported are greater than the ADF statistics, confirming the stationarity of the data. Although the results prove that our data is stationary, the ADF test is not always reliable because it is sensitive to data size. Additionally, the ADF test assumes a linear relationship between variables, and significant autocorrelation can also affect the results. Given the setbacks of the ADF test, we will utilize the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test to confirm whether our data is stationary or not. If the results of the KPSS test contradict the results from the ADF test, further investigation is needed.

## 4.3 KPSS

The Kwiatkowski, Phillips, Schmidt, and Shin test (KPSS) is a non-parametric statistical test used to analyze whether a time series data is stationary around a linear trend or mean or not. Like the ADF test we are looking to prove that our data’s statistical properties are constant over time and contain no seasonal components. The Null Hypothesis for this test claims that the data is stationary while the Alternative Hypothesis states that the data is not stationary. As with the ADF test, we want our data to be stationary and to prove so we need to accept the Null Hypothesis. This is done by establishing a level of significance, which will be a p-value of 0.05, and using it to draw our conclusions from the test. The critical values for this test should also be greater than the test statistics to enforce the rejection of the Alternative Hypothesis.

|  |  |
| --- | --- |
| **Metric** | **Value** |
| KPSS Statistics | 0.107 |
| P-value | 0.100 |
| Critical Values | ‘1%’: 0.739  ‘2.5%’: 0.574  ‘5%’: 0.463  ‘10%’: 0.347 |

**Table 4**

The KPSS test was done on the second-order differenced data, with the results for the test shown in Table 4. Our conclusion from these results is the same as the ADF test, in that our data is stationary. The level of significance we set at 0.05 is lower than the test's p-value of 0.1, and thus we can reject the Alternative Hypothesis and accept the Null Hypothesis. In addition to the p-value’s significance, the critical values for the test are also greater than the KPSS statistics, further enforcing our conclusions. Given that the results of the KPSS test do not contradict the ADF test we can confirm that our data is in fact stationary.

# 05. Method

While utilizing the programming software Python, we will explore different methods to implement our time series models. Our data is compiled from the years 2010-2020, and we will split our data into training, validation, and test sets. For our modeling, we will use the Autoregressive Integrated Moving Average (ARIMA) method. Furthermore, we will explore models using the Random Forest and Long Short-Term Memory (LSTM) methods.

## 5.1 Splitting the Data

The initial seven years of the dataset, spanning from 2010-03-01 to 2016-12-01, is the training period. The validation set covers the following two years, from 2017-01-01 to 2018-12-01. Lastly, the final year of the dataset, from 2019-01-01 to 2019-12-01, serves as the test set.

We divided the original data into these three distinct subsets to achieve specific goals: the training set is used to build and train the model, the validation set aids in hyperparameter tuning and prevent overfitting, while the test set offers an unbiased evaluation of the model's performance, ensuring that the model can generalize well on unseen data. The test should always be kept separate from the training and validation sets so that the model’s performance is not artificially inflated. As a result, this partitioning approach reduces over-optimization and fosters the development of a more dependable model.

## 5.2 Base Model: ARIMA (1,2,1)

For our analysis in predicting the Consumer Price Index (CPI), we have chosen the ARIMA (1,2,1) model as our base model to serve as a benchmark against other machine learning methods. ARIMA is an algorithm that is traditionally well known for its capability to forecast time series data. In this case, our ARIMA (1,2,1) is composed of three components:

1) Autoregressive (AR) order: 1 – This indicates the model includes one lag term, and the current value has a linear relationship with its most recent lag.

2) Integrated (I): 2 – the data is applied differencing twice to make the data’s mean and variance more stable. Differencing is the process of subtracting successive data from one another. Though in the code script the integrated component is shown as 0, we have manually converted the original data to second-order differenced data. This effectively makes the model an ARIMA (1,2,1) model.

3) Moving average (MA) order: 1 – meaning that the model includes one lagged error term.

By using this model as a reference, we aim to compare its forecasting performance with that of other advanced machine learning techniques to determine the best method for predicting the CPI as the comparison will allow us to see how other models perform relative to a strong baseline model.

## 5.3 Other Versions of ARIMA

Upon examining the Autocorrelation Function plot (see Figure 3.1), we have found significant autocorrelation for the first three lags. This suggests the potential usefulness of including any of the three lagged errors in our model. In addition, the Partial Autocorrelation Function plot (see Figure 3.2) reveals that the first four lags of the current value display a statistically significant autocorrelation with the current value. This illustrates that having any of the first three lags as an AR term might potentially enhance the model’s predictive performance.

To capture insights from the ACF and PACF plots, we iteratively incorporate these AR and MA components into the ARIMA model and assess its performance using Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE). In addition to these error metrics, we also use other model selection methods such as the Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC) to determine the best model. These criteria balance the model's fit with its complexity, and a lower value in either suggests a better model. Therefore, the ideal model would be the one that has the lowest RMSE, MAE, AIC, and BIC values.

## 5.4 Random Forest

Guided by the insights from an article by Leo Grinzstaijn, Edouard Oyallon, and Gael Varoquaux, which highlighted the superior performance of tree-based models over deep learning in modeling tabular data, we have decided to explore tree-based models for our task of predicting the target variable (Grinsztajn et al.). Our dataset's tabular format provides an ideal environment for such models.

In this section of our project, we will engage in the construction of several Random Forest models, each characterized by default settings and tunned hyperparameters. This strategy aims to maximize the model's performance potential through this ensemble machine learning technique. In addition to this, we are interested in investigating the impact of incorporating lag components on the model's predictive capabilities. We will be creating two distinct sets of Random Forest models - one set will incorporate lag components, while the other will not. This comparative approach will allow us to evaluate the potential benefits of integrating lag variables into our models.

### 5.4.1 Random Forest without Lag Components

We will be constructing Random Forest models with default settings and tunned hyperparameters. The rationale for doing this is to see if adjusting hyperparameters would lead to an enhanced predictive power between the two models with the same set of predictors. Additionally, we would like to see how it generalizes on unseen data.

**Default settings**

The first Random Forest model we created uses all features without any lag components from the label. We also use the default settings for Random Forest in Scikit – Learn when modeling this dataset (see table 5.1).

**Tunned Hyperparameters**

There are two common tunning methods in Scikit – Learn: Grid Search and Random Search. Both aim to find the best hyperparameter combination to optimize the performance of machine learning models. GridSearchCV performs an exhaustive search over a specified search space as it tries all possible combinations of hyperparameters, and it guarantees to find the best hyperparameters combination within the specified parameter grid. Meanwhile, RandomizedSearchCV uses a fixed number of random combinations from the specified parameter grid. By doing so, random search reduces the computational costs compared to GridSearchCV while still having a decent chance of finding near optimal hyperparameters.

In our case, since our data and the search space are relatively small, and we have sufficient computational resources, GridSearchCV was adopted to find the best hyperparameter combinations. Table 5.1 shows the default settings of the hyperparameters and the tuned settings that were used in the Random Forest without any lag components.

|  |  |  |  |
| --- | --- | --- | --- |
| **Hyperparameter** | **Definition** | **Default Setting** | **Tuned Setting** |
| n\_estimators | The number of trees in the forest. | 100 | 200 |
| criterion | The function is to measure the quality of a split. | Gini | Gini |
| max\_depth | The maximum depth of the trees. | None | 10 |
| min\_samples\_split | The minimum number of samples required to split a node. | 2 | 5 |
| min\_samples\_leaf | The minimum number of samples required to be at a leaf node. | 1 | 4 |
| max\_features | The number of features to consider when splitting a node. | Auto | Sqrt |
| bootstrap | Whether to bootstrap samples when building the trees. | False | True |
| oob\_score | Whether to compute out-of-bag (oob) scores on the training set. | False | False |

**Table 5.1**

### 5.4.2 Random Forest with Lag Components

In the second set of Random Forest models, we will be incorporating lag components as predictors for the Consumer Price Index (CPI). From the insights retrieved from ARIMA models, we found that the ARIMA model with three autoregressive components has the most robust performance compared to other models. Hence, we decided to include three lags of the label as predictors for the Consumer Price Index (CPI).

Like the first set of Random Forest, two versions of Random Forest models (Default Settings vs Tunned Hyperparameters) are built to evaluate how they generalize on unseen data. The tuning process also uses the same search space as in the first tuned Random Forest model without lags. Table 5.2 shows the tuned settings with the incorporated lag components.

|  |  |  |
| --- | --- | --- |
| **Hyperparameter** | **Default Setting** | **Tuned Setting** |
| n\_estimators | 100 | 500 |
| criterion | Gini | Gini |
| max\_depth | None | 10 |
| min\_samples\_split | 2 | 5 |
| min\_samples\_leaf | 1 | 2 |
| max\_features | Auto | Sqrt |
| bootstrap | False | False |
| oob\_score | False | False |

**Table 5.2**

## 5.5 Long Short-Term Memory

Long Short-Term Memory (LSTM) is a specialized form of Recurrent Neural Network (RNN). It is exceptionally good at handling time series data, making it an ideal candidate for predicting our target variable. Uniquely, LSTM can capture long-term dependencies within a data sequence. Its sophisticated algorithm can selectively retain or discard information over extended periods, enabling it to learn effectively from the temporal relationships among data points in a sequence.

In our analysis and the context of LSTM model, the concept of "timesteps" assumes a crucial role. Timesteps essentially represent the number of time periods considered when making predictions. While a timestep of 1 might not be the norm in LSTM models, in our scenario, considering the inclusion of three lags of the target variable, we have opted for a timestep of 1. Given the nature of the Consumer Price Index, which tends to fluctuate in sync with predictors in the model, a timestep of 1 ensures that predictions are based directly on the most recent macroeconomic changes.

Each increment by one unit in the timesteps effectively adds a set of features to the model's dimensions, thereby escalating its complexity. This, in turn, could potentially trigger the "curse of dimensionality", a phenomenon where an increase in the dimensionality of the data significantly complicates the computational task and may lead to overfitting, which is something we strive to avoid in our model construction.

To obtain the optimal predictive performance for the LSTM, we performed a grid search on a predefined list of hyperparameters. Those being tuned are the batch size, the dropout rate, and the number of hidden nodes in the LSTM layer which are shown in Table 6. We did not have to seek the optimal epoch because “Callbacks” and “EarlyStopping” have been enabled in the train and search process. The model would automatically stop the training process and the optimal hyperparameters are restored if the validation loss will not reduce after five epochs, as “patience” in “EarlyStopping” were set to five. Besides, the Adaptive Moment Estimation (Adam) optimization algorithm with a learning rate of 0.001 is used to update the network weights after each batch. After all the combinations of hyperparameters were tried, these were the ones that resulted in the model with the lowest validation loss, and therefore the best performance on the validation data.

|  |  |
| --- | --- |
| **Metric** | **Optimal Value** |
| Batch Size | 32 |
| Hidden Nodes | 10 |
| Dropout Rate | 0.2 |

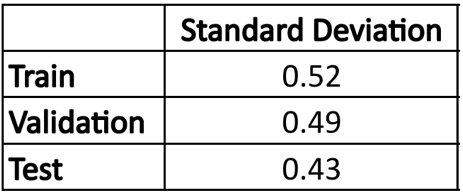
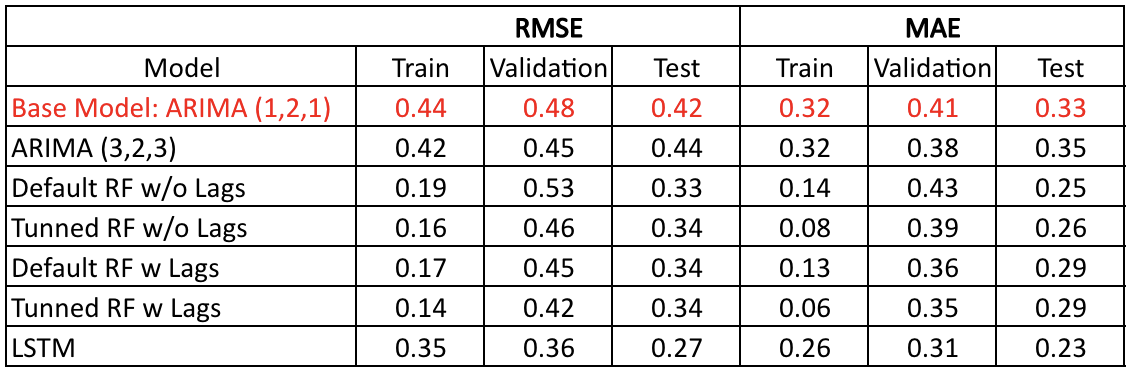
**Table 6**

# 06. Analyses And Results

## 6.1 Statistical Discussions

Two evaluation metrics used in evaluating model’s performance are RMSE and MAE. Mean Absolute Error tells us on average how much our predictions are off without considering the direction, while RMSE gives more weight to larger errors because they are squared before averaged. In other words, MAE gives an equal weight to all errors while RMSE is more sensitive to outliers. This is important especially when our validation and test sets are quite volatile.

Deciding which metrics to focus on depends on the economic periods. For instance, if we are predicting the CPI for a time of economic volatility, where sudden changes are likely to occur and large errors in these predictions could have significant consequences, RMSE is a good choice. On the other hand, MAE could be a great choice to observe for periods of relative economic stability, where outliers are less common, and we treat both small and large errors the same. However, in most cases, it’s often a good idea to look at both metrics to get a comprehensive view of the model’s performance.



**Table 7**   **Table 8**

Table 7 and Table 8 show the results of our models within each set. From these results we can gather that all models demonstrated a solid ability to forecast patterns that exceed the inherent variability in the corresponding dataset, as shown in their RMSE and MAE scores falling below the dataset’s standard deviation.

Despite most models only showing a slight improvement in the RMSE on the validation dataset in comparison to the benchmark model, their significantly better performance on the test set implies that they are better at extracting underlying patterns and relationships that the benchmark model has failed to capture. An exception to this is the LSTM model, which surpassed all other models in performance.

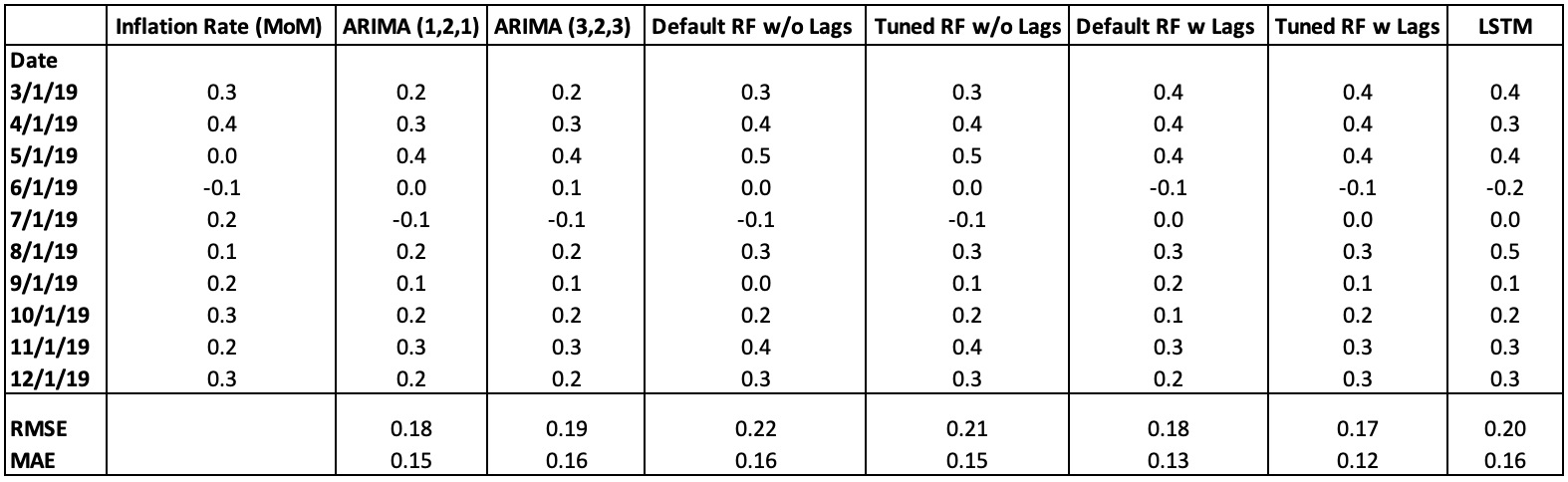
Regarding ARIMA models, after systematically adding and evaluating each component, we have found that the ARIMA (3,2,3) model offers the best predictive performance based on our chosen evaluation metrics. This model, despite its superior MAE and RMSE on the validation set, fell short of the benchmark ARIMA (1,2,1) model when applied to the test set. In addition, the AIC and BIC of ARIMA (1,2,1) report lower values, indicating that the base model is preferred as it balances model fit and complexity in the best way. Apart from that, results from all ARIMA models also reveal that lagged error terms are not statistically significant due to their high p-values. Therefore, we can conclude that the MA components are not helpful in predicting the label.

Although the Random Forest models without lags using default settings from Scikit-Learn were found to have a poor predictive performance on the validation dataset in terms of both RMSE and MAE, it performs surprisingly well on the test set. This model slightly outperformed the tuned Random Forest model that used the same features, and even performed better than Random Forest models that incorporated label’s lags into their feature set. Hence, we can see that although adding label’s lag to the set of features does improve predictive power on the validation set, it does not generalize very well on unseen data.

It’s particularly noteworthy that the Long Short-Term Memory model consistently outperforms all other models with the lowest RMSE and MAE. This could be due to the algorithm’s capabilities to effectively learn and leverage the temporal patterns between features and lags of the label.

## 6.2 Result’s Discussion

In our training process, we use labels in the form of second-order differences, which, while effective for modeling, might pose difficulties for interpretation. To make the results more comprehensible and intuitive, we have decided to reverse the differenced data back to the original CPI data, and then we calculated the monthly inflation rates. This solution offers a better way of understanding the result and makes our predictions readily relatable to grasp for a broader audience.



**Table 9**

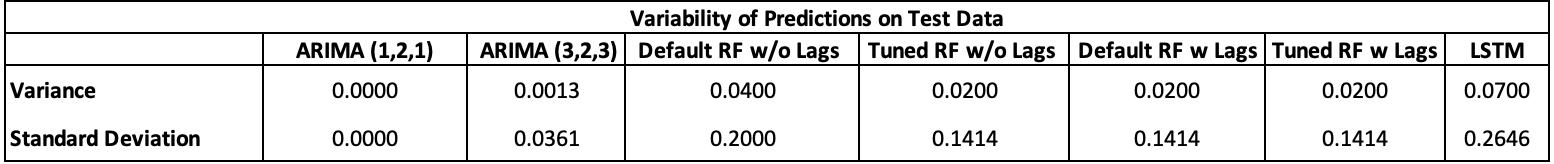
According to Table 9 we can see that the tuned Random Forest with label’s lags shows the lowest RMSE and MAE in predicting the month-over-month inflation rate in 2019. It is very surprising that inflation rates extracted from Long Short-Term Memory’s second-order difference results underperform Random Forest with lags included, and even both ARIMA models. Though this underperformance comes unexpectedly, there are some possible explanations for this phenomenon.

First, since we back transformed our differenced data to its original form in comparison with the actual month over month inflation rate, any predictions errors that were present in the differenced data will propagate through the reverse process. This could potentially lead to the magnification of errors, which could have caused the deterioration in performance when we evaluate the model on the original scale.

Secondly, it could be due to the difference in the magnitude of values. Since the scale and variability of our data in its original form and its second-order difference form are massively different, small relative errors or high variance in the prediction can become large absolute errors when we back transform to the original scale. As we can see in Table 10, variance in LSTM predictions on the test set are considerably higher than in the results produced by other models.

Lastly, the issue is with differencing time series. Differencing can remove trends and seasonality, which helps simplify the structure of the data. Therefore, when we reverse the differencing process, these components are reintroduced. Though LSTM is well known for its capability for capturing complex patterns and dependencies in data, the insufficient data in our case prevents it from having a longer timestep to do so. This, unfortunately, leads to the deterioration of performance when we back transform it to the original scale.

For models that are built using Random Forest and ARIMA, their prediction’s variance, as shown In Table 10, appears to be significantly less than the LSTM. Given the fact that the Consumer Price Index has an increasing trend, and it does not deviate much from the trend, and a smaller variance in differenced data when reversing to original data will result in rates that are more like the actual rates.



**Table 10**

# 07. Conclusion

The Consumer Price Index is the most commonly used measure when analyzing and understanding inflation. The goal of our project is to accurately create a time series model to forecast the CPI. Possessing the capability to forecast CPI provides valuable information about economic instances and decisions made on a regular basis.

After analyzing the trends and lags of our data, we preprocessed the label and the sixteen predictors we felt were most influential to the CPI. Upon completion, we split our data into three sets: Test, Train, and Validation. Our base model used the ARIMA method with an autoregressive order of 1, an integration of 2, and a moving average of 1. We then tested other versions of ARIMA.

Additionally, we utilized the Random Forest method, with and without label’s lagged components, to forecast the CPI. Our Random Forest model had the most accurate score on the original data that was back-transformed from the second-order difference data to the original data. This model exhibited the closest prediction to the monthly inflation rates.

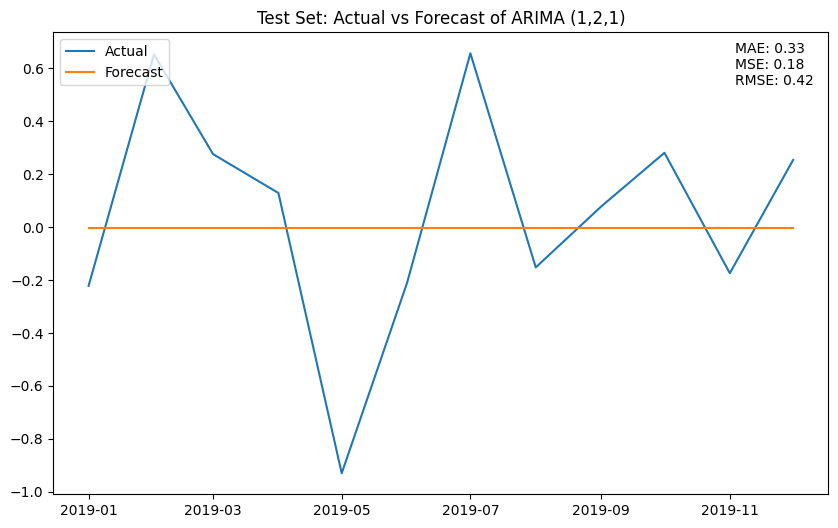
Another method that was explored and utilized was the Long Short-Term Memory method. This model presented the best predictive performance on the transformed CPI. The LSTM model consistently achieved the best performance, surpassing all other models’ RMSE and MAE results. This is likely because the algorithm effectively learns and utilizes patterns among features and the label’s lags.

As we move forward, the significance of this subject cannot be overstated, particularly in light of the current economic environment where inflation has reached a staggering 40-year high. Expanding our data sources and investigating more advanced models and reliable indicators could help enhance the precisions of our predictions even further. Establishing a robust method for forecasting the Consumer Price Index (CPI) will not only strengthen the economic framework and efficiency of governments, but also guide businesses to adjust their pricing strategies, ensuring profitability and competitiveness in the marketplace.

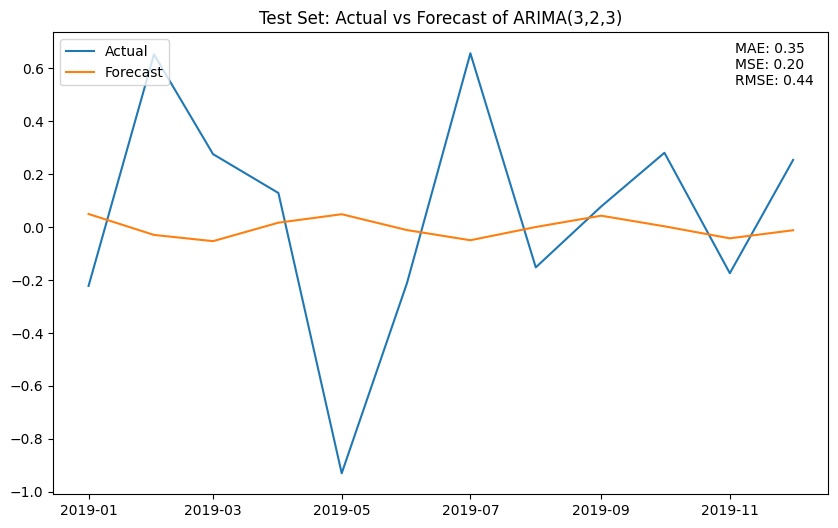
# References

Grinsztajn, Léo, et al. "Why do tree-based models still outperform deep learning on tabular data?" Cornell University, 18 Jul. 2022 https://arxiv.org/abs/2207.08815, undefined. Accessed 14 Apr. 2023.

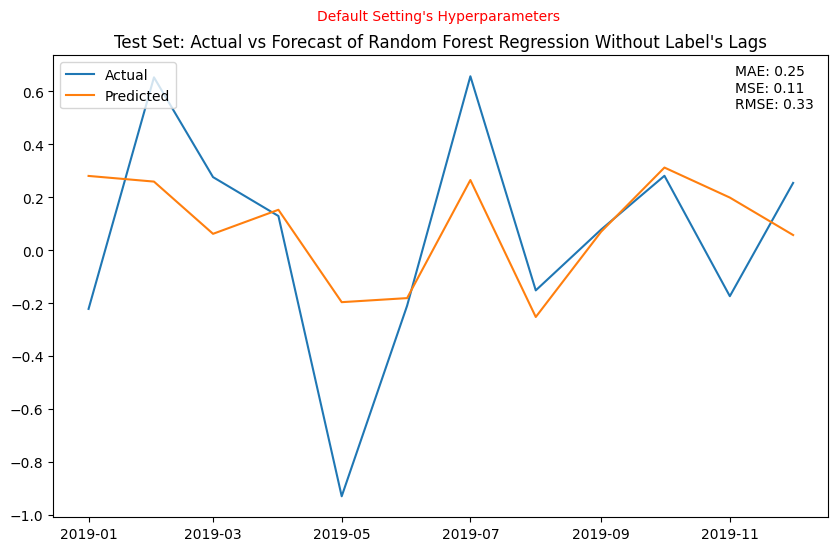
# Appendix A

**Graph 1**

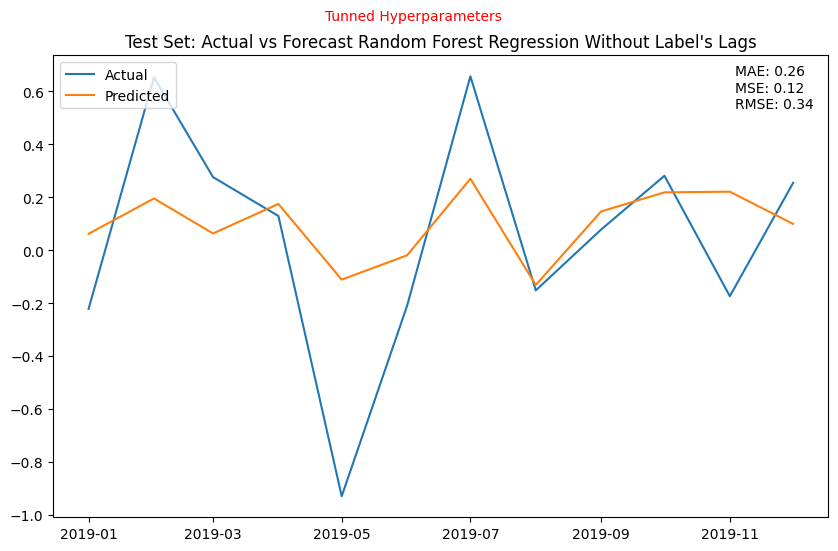
**Graph 2**



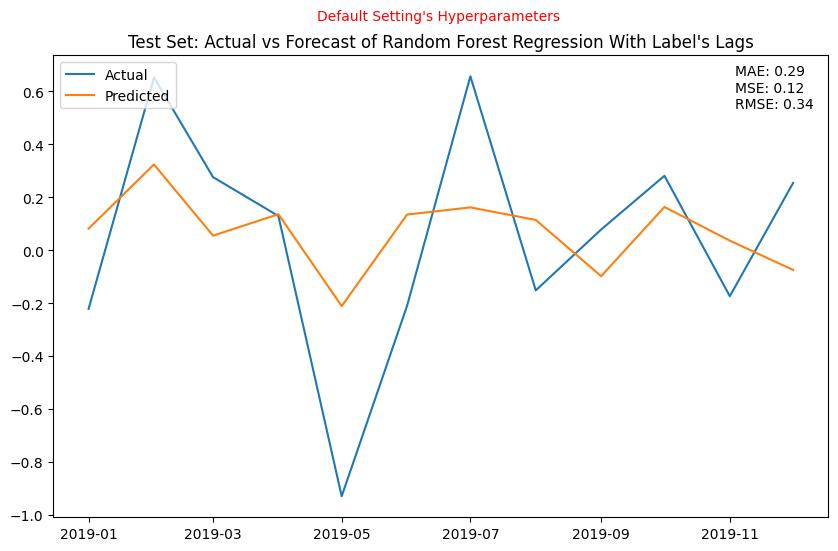
**Graph 3**



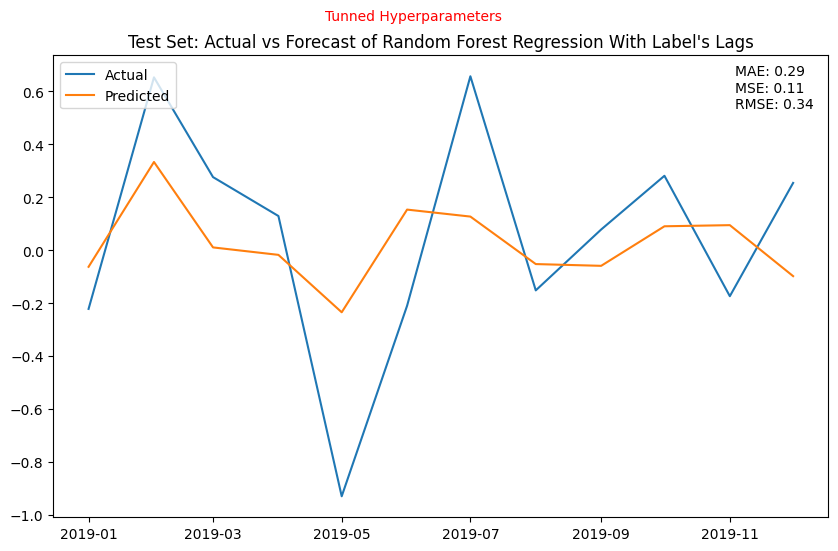
**Graph 4**



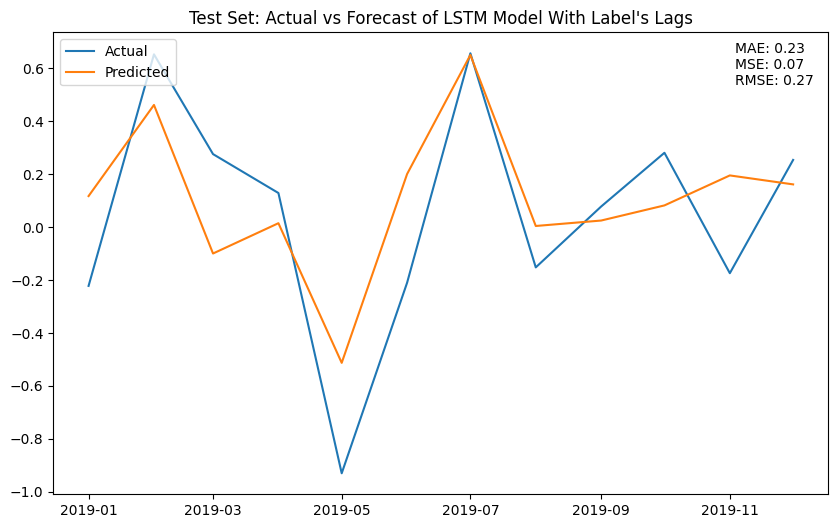
**Graph 5**

****

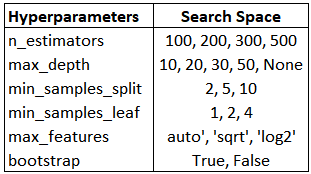
**Graph 6**



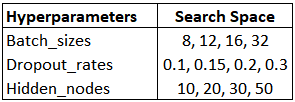
**Graph 7**



**Graph 8**



**Search Space: Random Forest**



**Search Space: Long-Short Term Memory**

**A picture containing text, screenshot, font

Description automatically generated**

1. <https://fred.stlouisfed.org> [↑](#footnote-ref-1)
2. Ot is the output, Tt is the trend component, St is the seasonality component, Rt is the residual component, and t is a variable representing a particular point in time. [↑](#footnote-ref-2)