Newyork electricity price prediction using Time Series

Bhargavi Katta
Department of Mathematical Sciences, Stevens Institute of Technology, Hoboken, NJ
Project Supervisor: Dr. Hadi Safari Katesar

Abstract

This project aims to forecast the electricity prices in New York using seasonal time series data. The dataset used in this study consists of hourly electricity prices from 2017 to 2021. The objective of this project is to develop an accurate time series forecasting model that can predict the electricity prices for the next day, based on the past data.

Several time series forecasting methods were applied to the dataset, including ARIMA, SARIMA, seasonal decomposition, and Prophet. The models were evaluated using AIC, BIC, mean absolute error (MAE), root mean squared error (RMSE), and mean absolute percentage error (MAPE).

The study also found that seasonality has a significant impact on electricity prices, with higher prices during peak demand hours and during the summer months.

Overall, this study demonstrates the feasibility of using time series forecasting models to predict electricity prices in New York based on seasonal patterns, which could help energy companies optimize their operations and make informed decisions regarding pricing and supply.

SEASONAL DATASET: Newyork Electricity prices

Introduction and Motivatio

Electricity prices are a crucial component of the energy market, and accurate prediction of these prices is essential for energy companies to optimize their operations and make informed decisions regarding pricing and supply. The demand for electricity in New York is highly seasonal, with peak demand occurring during the summer months and during certain hours of the day. As a result, electricity prices in New York exhibit strong seasonal patterns, making accurate prediction challenging. Time series forecasting methods can be used to capture these patterns and predict future electricity prices.

The motivation for this project is to develop an accurate forecasting model for New York electricity prices that takes into account the seasonal patterns in the data. Accurate prediction of electricity prices is essential for energy companies to optimize their operations, reduce costs, and provide reliable services to consumers.

Additionally, accurate prediction can help consumers make informed decisions regarding energy usage and expenses. This study aims to contribute to the existing literature on electricity price forecasting by exploring the performance of different time series forecasting methods on seasonal New York electricity price data. The insights gained from this study can help energy companies and policymakers make informed decisions regarding energy pricing and supply.

Data Description Data Source: Data can be accessed from "https://www.nyserda.ny.gov/Researchers-and-Policymakers/Energy-Prices/Electricity/Monthly-Avg-Electricity-Residential (https://www.nyserda.ny.gov/Researchers-and-Policymakers/Energy-Prices/Electricity/Monthly-Avg-Electricity-Residential)"

Data Rage: January2012 to December 2022 Data Description: Dataset has electicity prices for each month from 2012 to 2022

Tests used in the Project

Stationarity Test

The concept of stationarity is a crucial consideration in time series analysis. Given that non-stationary time series data cannot be accurately forecast by a model, the initial step time series forecasting involves making the data stationary.

A stationary series is characterized by statistical features such as mean, variance, and covariance that remain constant over time or are independent of time Two statistical tests which we will be using to check stationarity are:

- Augmented Dickey-Fuller (ADF) Test
- Kwiatkowski-Phillips-Schmidt-Shin (KPSS) Test

Augmented Dickey-Fuller (ADF) Test :

The Augmented Dickey-Fuller (ADF) test is a statistical test commonly used in time series analysis to determine whether a given time series is stationary or non-stationary.

The ADF test is a type of unit root test, which means that it tests for the presence of a unit root in the time series. A unit root is a stochastic trend in the time series that makes it non-stationary, and can lead to spurious regression results and inaccurate forecasts.In a simple term, the unit root is non-stationary but does not always have a trend component.

ADF test is conducted with the following assumptions:

- Null Hypothesis (HO): Series is non-stationary or series has a unit root
- Alternate Hypothesis(HA): Series is stationary or series has no unit root

If the null hypothesis is failed to be rejected, this test may provide evidence that the series is non- stationary. Conditions to Reject Null Hypothesis(HO)

If Test statistic < Critical Value and p-value < 0.05 – Reject Null Hypothesis(HO) i.e., time series does not have a unit root, meaning it is stationary. It does not have a time-dependent structure.

KPSS Test :

The KPSS test (Kwiatkowski-Phillips-Schmidt-Shin) is a type of Unit root test that checks for the stationarity of a given series around a deterministic trend. In other words, the test is conceptually comparable to the ADF test. However, it is a frequent misconception that it can be used interchangeably with the ADF test. This can lead to misunderstandings about stationarity, which can easily go undetected, producing further problems down the road. KPSS test is conducted with the following assumptions:

- Null Hypothesis (HO): Series is trend stationary or series has no unit root
- Alternate Hypothesis(HA): Series is non-stationary or series has a unit root

Note that Hypothesis is reversed in KPSS test compared to ADF Test. If the null hypothesis is failed to be rejected, this test may provide evidence that the series is trend stationary. Conditions to Fail to Reject Null Hypothesis(HO) If Test statistic < Critical Value and p-value < 0.05 – Fail to Reject Null Hypothesis(HO) i.e., time series does not have a unit root, meaning it is trend stationary. In order to reject the null hypothesis, the test statistic should be greater than the provided critical values. If it is in fact higher than the target critical value, then that should automatically reflect in a low p-value. That is, if the p-value is less than 0.05, the kpss statistic will be greater than the 5% critical value.

Shapiro-wilk test Test :

When the distribution of a real random variable is unknown, it is convenient to assume that it is normally distributed. However, this may not always be true leading to incorrect results. To avert this problem, there is a statistical test by the name of Shapiro-Wilk Test that gives us an idea whether a given sample is normally distributed or not. The test works as follows: continuous Specify the null hypothesis and the alternative hypothesis as:

- H0 : the sample is normally distributed
- HA : the sample is not normally distributed
  A test statistic is computed as follows:

The purpose of this test is to see if the data is normally distributed or not

- Null Hypothesis : The data is normally distributed
- Alternative Hypothesis : The data is not normally distributed

If the p.value > 0.05 ,we failed to reject null hypothesis. In other words, we can assume the normality.

Ljung-Box test :

The Ljun-Box test is a hypothesis test that checks if a time series contains an autocorrelation. The null Hypothesis H0 is that the residuals are independently distributed. The alternative hypothesis is that the residuals are not independently distributed and exhibit a serial correlation.

The Ljung-Box test uses the following hypotheses:

- H0: The residuals are independently distributed.
- HA: The residuals are not independently distributed; they exhibit serial correlation

Ideally, we would like to fail to reject the null hypothesis. That is, we would like to see the p- value of the test be greater than 0.05 because this means the residuals for our time series model are independent, which is often an assumption we make when creating a model. It is a statistical test of whether any group of autocorrelations of a time series # are different from 0. Instead of testing randomness of each distinct lag, it tests overall randomness based on number of lags.

ACF and PACF Evaluation :

Autocorrelation Function (ACF): Time series correlation with a delayed version of itself. The relationship between observations made in the current time and observations made at earlier times. The autocorrelation function begins with a lag of zero, which is the correlation of the time series with itself, resulting in a correlation of one. The plot acf function from the statsmodels.graphics.tsaplots package will be used. The following questions can be answered using the ACF plot: Is the observed time series random or white noise? Is one observation related to another, to an observation twice removed, and so on? Can an MA model be used to model the observed time series? If so, what is the sequence?

Partial Autocorrelation Function (PACF): Each each lagged term explains more correlation. Given that both observations are connected to observations at other time points, the correlation between pbservations at two time points. The plot pacf function from the statsmodels.graphics.tsaplots package will be used, using the option method = “ols” (regression of time series on lags of it and on constant). The following questions can be answered using the PACF plot: Can an AR model be used to model the observed time series? If so, what is the sequence?

Both the ACF and PACF start with a lag of 0, which is the correlation of the time series with itself and therefore results in a correlation of 1.

Extended Autocorrelation Function(EACF):

The EACF (Extended Autocorrelation Function) plot is a diagnostic tool used in time series analysis to help identify the appropriate lag structure of an ARMA (Autoregressive Moving Average) model.

The EACF plot is an extension of the autocorrelation function (ACF) and partial autocorrelation function (PACF), which are commonly used to identify the lag structure of an ARMA model. The EACF plot provides a visual representation of the potential ARMA models that may be appropriate for a given time series, based on the observed autocorrelation patterns.

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
## Registered S3 methods overwritten by 'TSA':
##   method       from
##   fitted.Arima forecast
##   plot.Arima   forecast
```

```
##
## Attaching package: 'TSA'
```

```
## The following objects are masked from 'package:stats':
##
##     acf, arima
```

```
## The following object is masked from 'package:utils':
##
##     tar
```

```
##
## Attaching package: 'plotly'
```

```
## The following object is masked from 'package:ggplot2':
##
##     last_plot
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
## The following object is masked from 'package:graphics':
##
##     layout
```

```
setwd("/Users/gopalrao000/Desktop/Spring-2023/Time series/project")
```

```
data=read.csv("NYC_residential_electricity_price.csv")
```

```
data
```

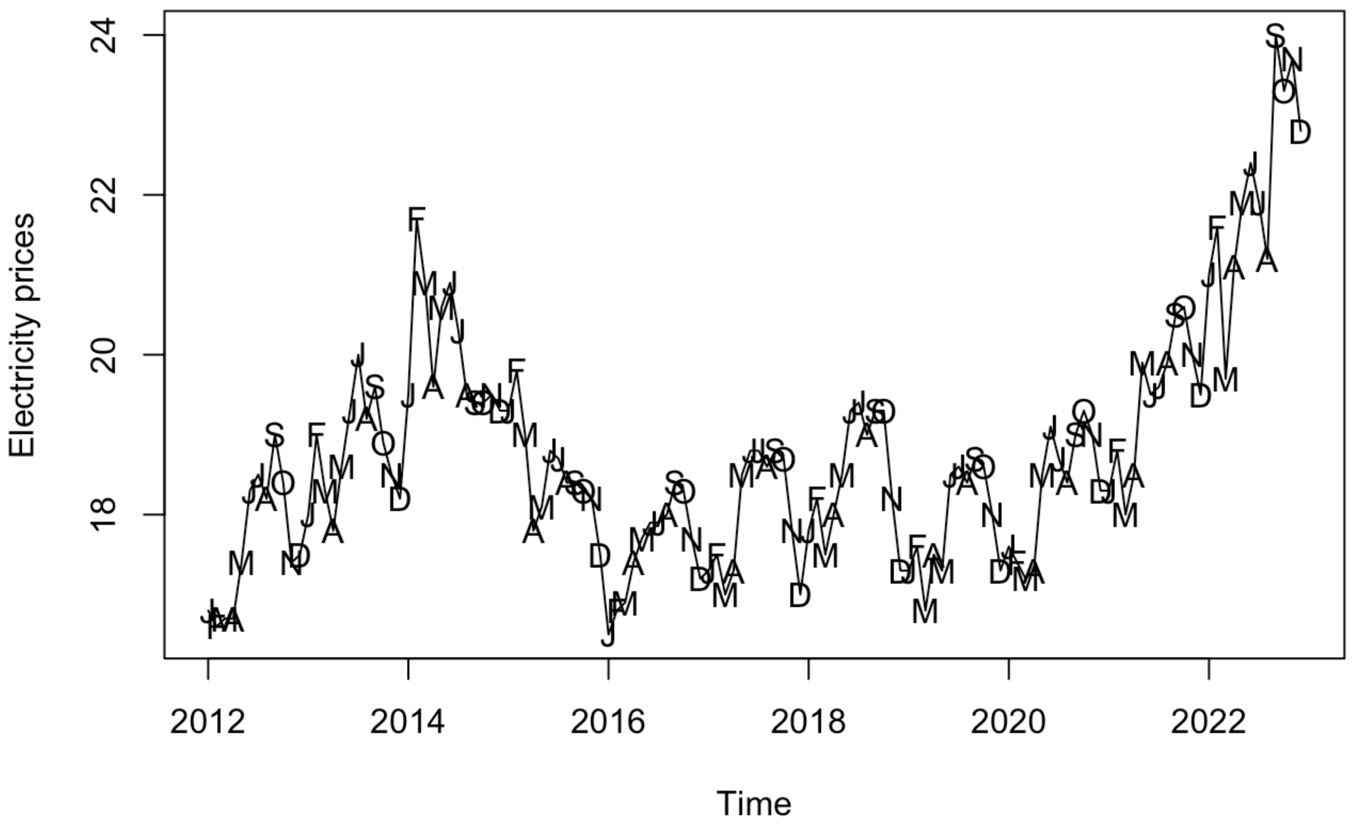
```
##     Year month Price
## 1   2012     1  16.8
## 2   2012     2  16.6
## 3   2012     3  16.7
## 4   2012     4  16.7
## 5   2012     5  17.4
## 6   2012     6  18.3
## 7   2012     7  18.5
## 8   2012     8  18.2
## 9   2012     9  19.0
## 10  2012    10  18.4
## 11  2012    11  17.4
## 12  2012    12  17.5
## 13  2013     1  18.0
## 14  2013     2  19.0
## 15  2013     3  18.3
## 16  2013     4  17.8
## 17  2013     5  18.6
## 18  2013     6  19.3
## 19  2013     7  20.0
## 20  2013     8  19.2
## 21  2013     9  19.6
## 22  2013    10  18.9
## 23  2013    11  18.5
## 24  2013    12  18.2
## 25  2014     1  19.5
## 26  2014     2  21.7
## 27  2014     3  20.9
## 28  2014     4  19.6
## 29  2014     5  20.6
## 30  2014     6  20.9
## 31  2014     7  20.3
## 32  2014     8  19.5
## 33  2014     9  19.4
## 34  2014    10  19.4
## 35  2014    11  19.5
## 36  2014    12  19.3
## 37  2015     1  19.3
## 38  2015     2  19.8
## 39  2015     3  19.0
## 40  2015     4  17.8
## 41  2015     5  18.1
## 42  2015     6  18.8
## 43  2015     7  18.7
## 44  2015     8  18.4
## 45  2015     9  18.4
## 46  2015    10  18.3
## 47  2015    11  18.2
## 48  2015    12  17.5
## 49  2016     1  16.5
## 50  2016     2  16.8
## 51  2016     3  16.9
## 52  2016     4  17.4
## 53  2016     5  17.7
## 54  2016     6  17.9
## 55  2016     7  17.9
## 56  2016     8  18.0
## 57  2016     9  18.4
## 58  2016    10  18.3
## 59  2016    11  17.7
## 60  2016    12  17.2
## 61  2017     1  17.3
## 62  2017     2  17.5
## 63  2017     3  17.0
## 64  2017     4  17.3
## 65  2017     5  18.5
## 66  2017     6  18.8
## 67  2017     7  18.8
## 68  2017     8  18.6
## 69  2017     9  18.8
## 70  2017    10  18.7
## 71  2017    11  17.8
## 72  2017    12  17.0
## 73  2018     1  17.8
## 74  2018     2  18.2
## 75  2018     3  17.5
## 76  2018     4  18.0
## 77  2018     5  18.5
## 78  2018     6  19.3
## 79  2018     7  19.4
## 80  2018     8  19.0
## 81  2018     9  19.3
## 82  2018    10  19.3
## 83  2018    11  18.2
## 84  2018    12  17.3
## 85  2019     1  17.3
## 86  2019     2  17.6
## 87  2019     3  16.8
## 88  2019     4  17.5
## 89  2019     5  17.3
## 90  2019     6  18.5
## 91  2019     7  18.6
## 92  2019     8  18.4
## 93  2019     9  18.7
## 94  2019    10  18.6
## 95  2019    11  18.0
## 96  2019    12  17.3
## 97  2020     1  17.6
## 98  2020     2  17.4
## 99  2020     3  17.2
## 100 2020     4  17.3
## 101 2020     5  18.5
## 102 2020     6  19.1
## 103 2020     7  18.7
## 104 2020     8  18.4
## 105 2020     9  19.0
## 106 2020    10  19.3
## 107 2020    11  19.0
## 108 2020    12  18.3
## 109 2021     1  18.3
## 110 2021     2  18.8
## 111 2021     3  18.0
## 112 2021     4  18.5
## 113 2021     5  19.9
## 114 2021     6  19.5
## 115 2021     7  19.6
## 116 2021     8  19.9
## 117 2021     9  20.5
## 118 2021    10  20.6
## 119 2021    11  20.0
## 120 2021    12  19.5
## 121 2022     1  21.0
## 122 2022     2  21.6
## 123 2022     3  19.7
## 124 2022     4  21.1
## 125 2022     5  21.9
## 126 2022     6  22.4
## 127 2022     7  21.9
## 128 2022     8  21.2
## 129 2022     9  24.0
## 130 2022    10  23.3
## 131 2022    11  23.7
## 132 2022    12  22.8
```

Converting data to Timeseries Object

```
# Plot the time series
data.ts <- matrix(data$Price,nrow=132,ncol=1)
data.ts<- as.vector(t(data.ts))
data.ts <- ts(data.ts,start=c(2012,1), end=c(2022,12), frequency=12)
```

Time Series Plot

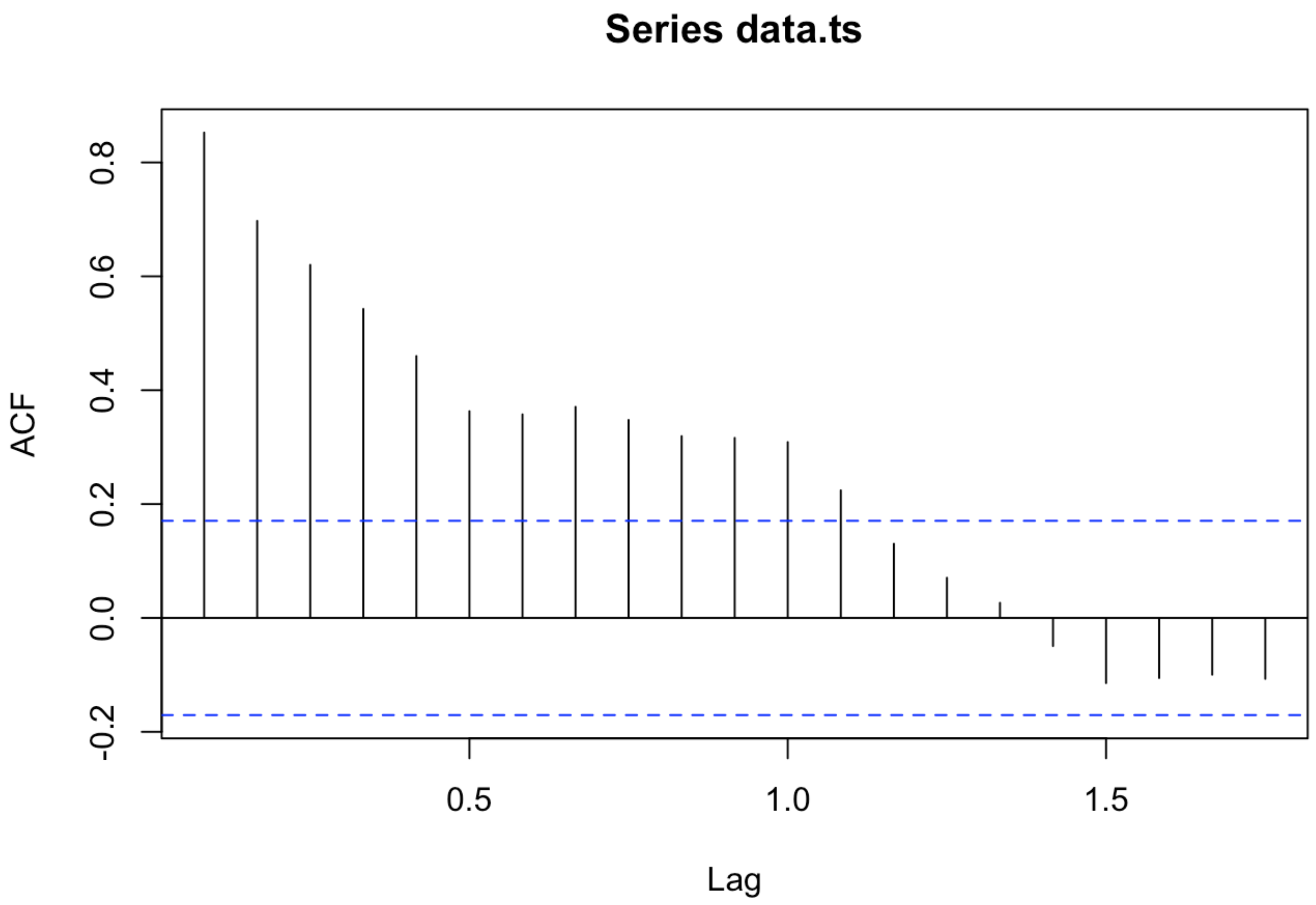
```
plot(data.ts,type='l',ylab='Electricity prices')
points(data.ts,x=time(data.ts),pch=as.vector(season(data.ts)))
```



```
## quartz_off_screen
##                 2
```

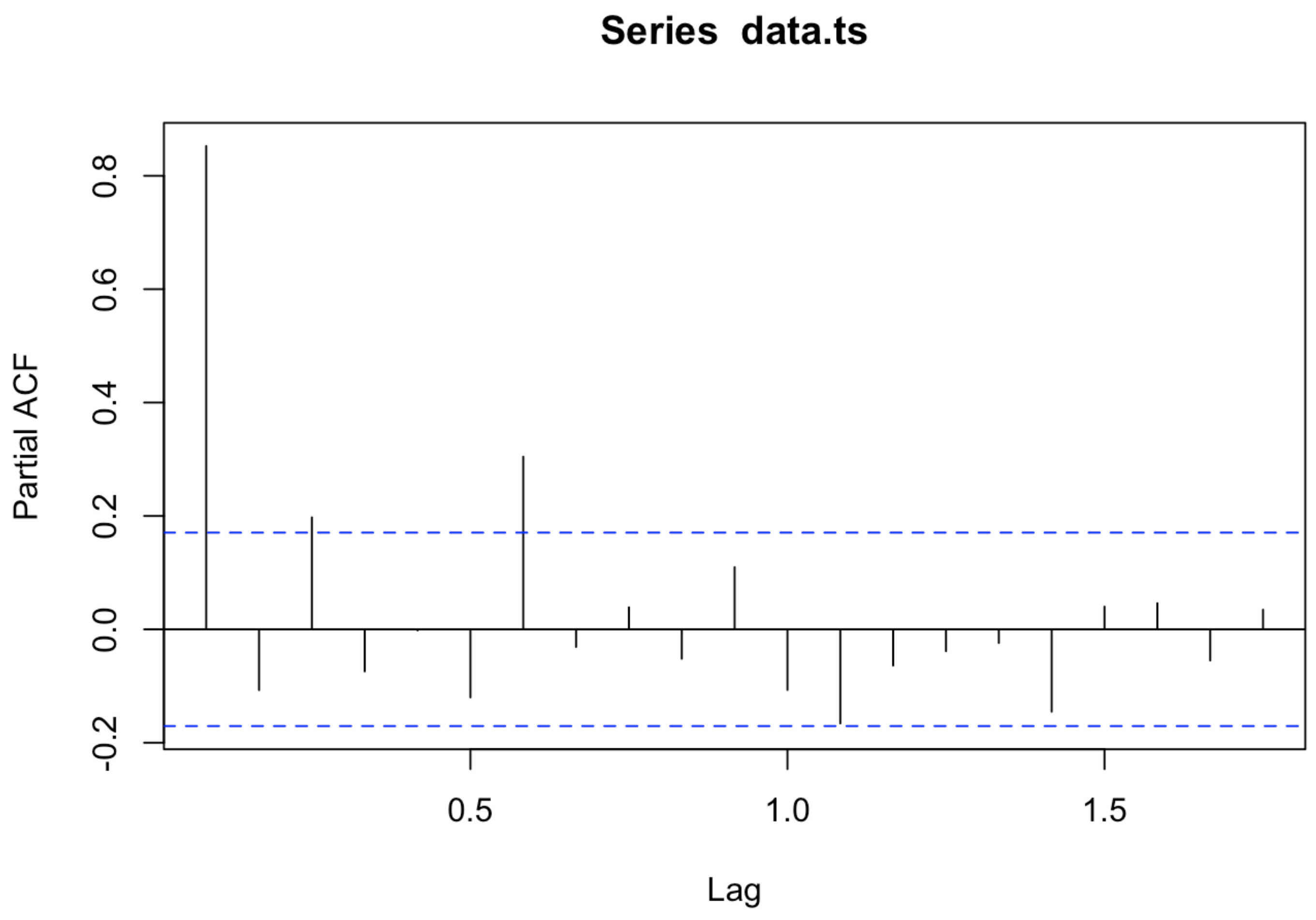
ACF Plot

```
acf(data.ts)
```



PACF Plot

```
pacf(data.ts)
```



Stationarity Test:

```
adf.test(data.ts)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  data.ts
## Dickey-Fuller = -1.5571, Lag order = 5, p-value = 0.7607
## alternative hypothesis: stationary
```

Null Hypothesis: Data is not stationary Alternate hypothesis: Data is Stationary P. value > 0.05, failed to reject null hypothesis Hence data is not stationary

```
ndiffs(data.ts)
```

```
## [1] 1
```

```
nsdiffs(data.ts)
```

```
## [1] 0
```

#Differencing:

First differencing on the data

```
data.ts.diff=diff(data.ts)
```

```
data.ts.diff
```

```
##       Jan  Feb  Mar  Apr  May  Jun  Jul  Aug  Sep  Oct  Nov  Dec
## 2012      -0.2  0.1  0.0  0.7  0.9  0.2 -0.3  0.8 -0.6 -1.0  0.1
## 2013  0.5  1.0 -0.7 -0.5  0.8  0.7  0.7 -0.8  0.4 -0.7 -0.4 -0.3
## 2014  1.3  2.2 -0.8 -1.3  1.0  0.3 -0.6 -0.8 -0.1  0.0  0.1 -0.2
## 2015  0.0  0.5 -0.8 -1.2  0.3  0.7 -0.1 -0.3  0.0 -0.1 -0.1 -0.7
## 2016 -1.0  0.3  0.1  0.5  0.3  0.2  0.0  0.1  0.4 -0.1 -0.6 -0.5
## 2017  0.1  0.2 -0.5  0.3  1.2  0.3  0.0 -0.2  0.2 -0.1 -0.9 -0.8
## 2018  0.8  0.4 -0.7  0.5  0.5  0.8  0.1 -0.4  0.3  0.0 -1.1 -0.9
## 2019  0.0  0.3 -0.8  0.7 -0.2  1.2  0.1 -0.2  0.3 -0.1 -0.6 -0.7
## 2020  0.3 -0.2 -0.2  0.1  1.2  0.6 -0.4 -0.3  0.6  0.3 -0.3 -0.7
## 2021  0.0  0.5 -0.8  0.5  1.4 -0.4  0.1  0.3  0.6  0.1 -0.6 -0.5
## 2022  1.5  0.6 -1.9  1.4  0.8  0.5 -0.5 -0.7  2.8 -0.7  0.4 -0.9
```

ADF Test

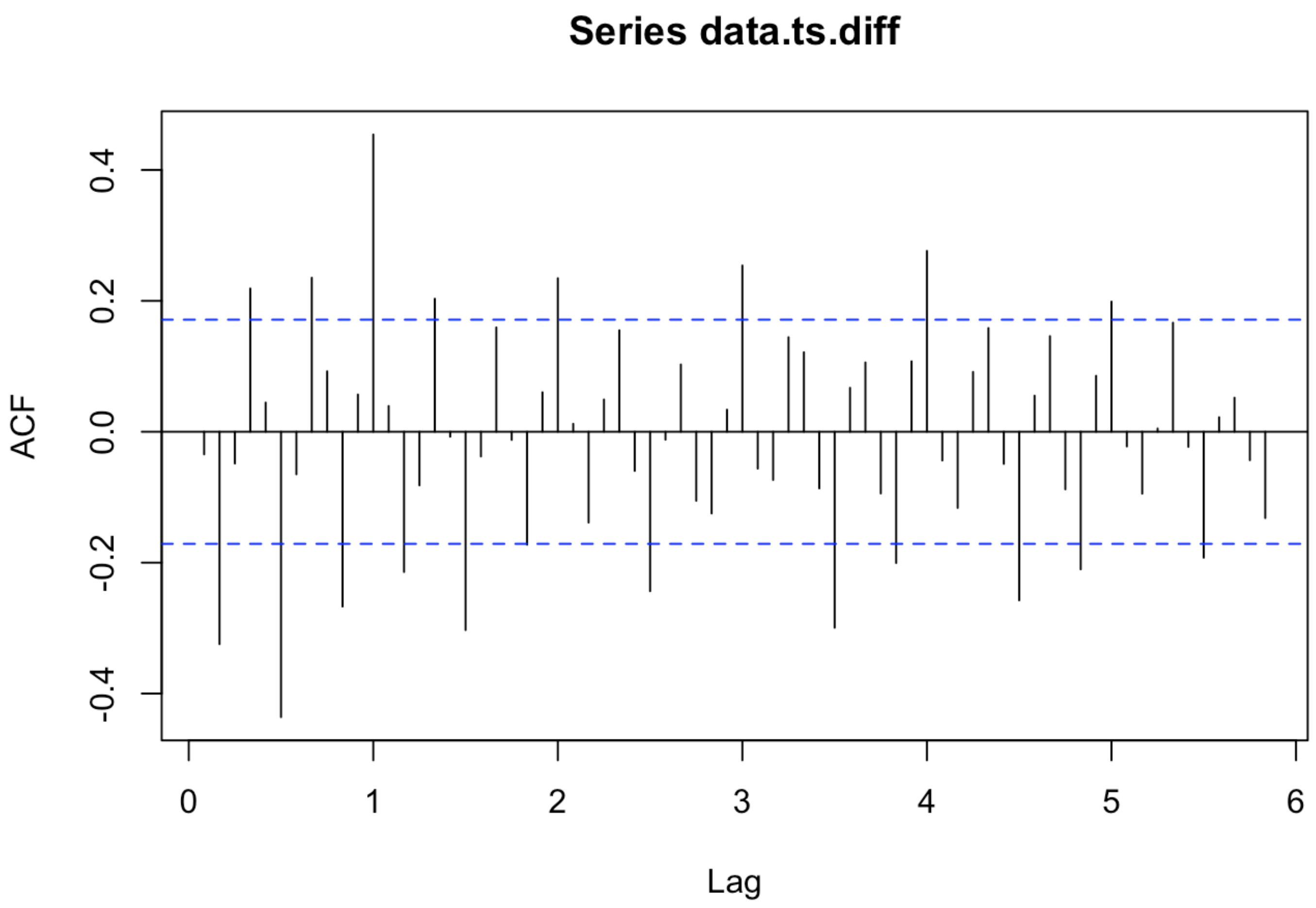
```
adf.test(data.ts.diff)
```

```
## Warning in adf.test(data.ts.diff): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  data.ts.diff
## Dickey-Fuller = -7.3491, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary
```

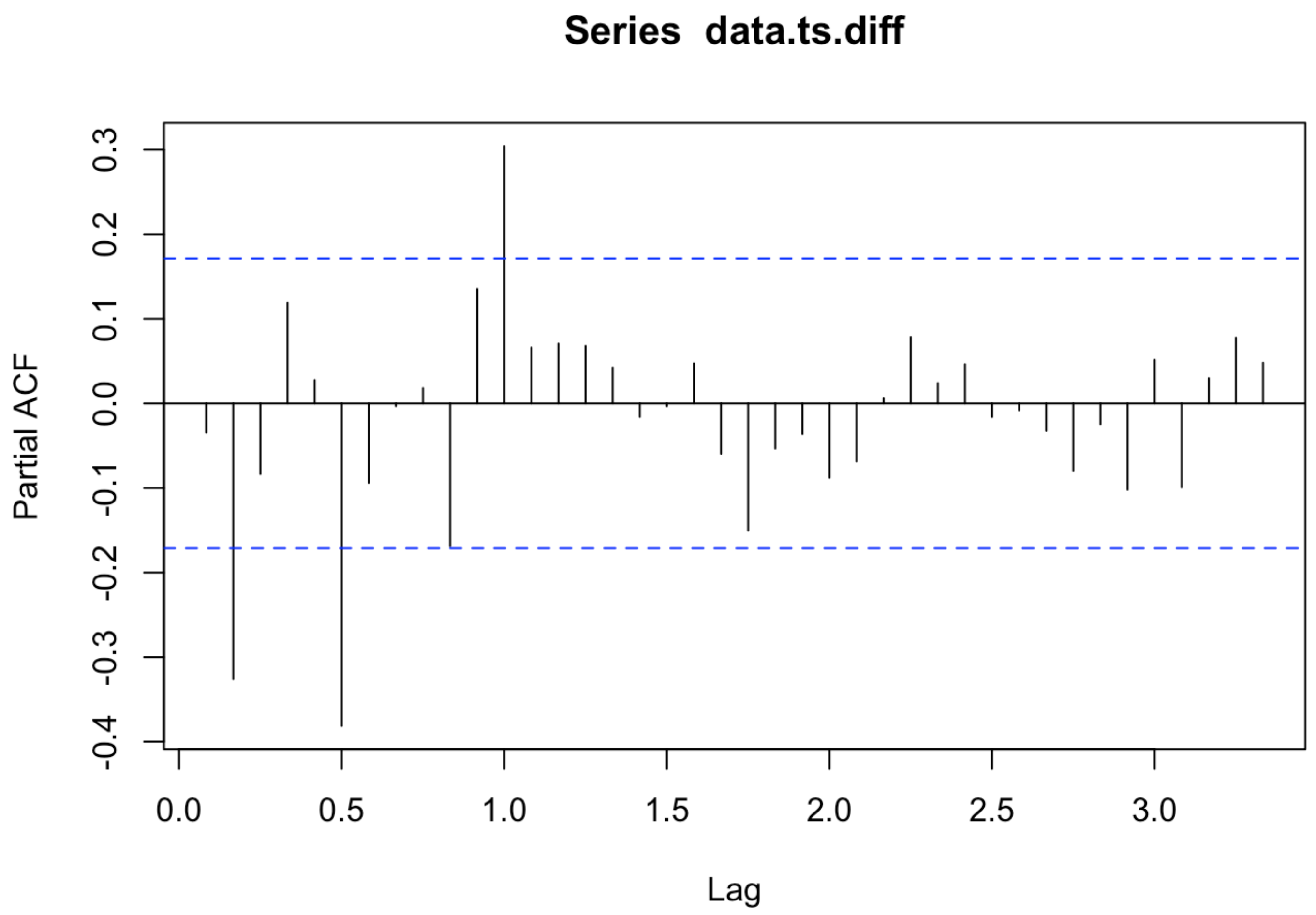
P-Value<0.05, hense we can reject null hypothesis Hence Data is stationary now after first differencing

```
acf(data.ts.diff,lag.max=70)
```



From the ACF Graph, model has seasonal behavior with s=12 (might be 4/6 as well)

```
pacf(data.ts.diff,lag.max=40)
```



```
eacf(data.ts.diff)
```

```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 o x o x o x o x o x o  x  o  x
## 1 o x o x o x o x o x o  x  o  x
## 2 x x o o o x o o o o o  x  o  o
## 3 x x o o o x x o o o o  x  x  o
## 4 x x o x o o x o o o o  x  o  o
## 5 o x o x x o o o o o o  x  o  o
## 6 o o o x x x o o o o o  x  o  o
## 7 x o o x o x x o o o o  x  o  o
```

From the above ACF,eacf and PACF Plots, potential models are

SARIMA(0,1,0)(2,0,0)[12] SARIMA(0,1,2)(2,0,0)[12]

```
fit <- auto.arima(data.ts.diff)
```

```
summary(fit)
```

```
## Series: data.ts.diff
## ARIMA(0,0,2)(1,0,0)[12] with zero mean
##
## Coefficients:
##           ma1      ma2    sar1
##       -0.1747  -0.2061  0.5942
## s.e.   0.0868   0.0912  0.0801
##
## sigma^2 = 0.3224:  log likelihood = -112.9
## AIC=233.8   AICc=234.12   BIC=245.3
##
## Training set error measures:
##                      ME      RMSE       MAE MPE MAPE      MASE          ACF1
## Training set 0.05080072 0.5612631 0.4301484 NaN  Inf 0.9091946 -0.001555611
```

Model plotting

SARIMA(0,1,0)(2,0,0)[12] SARIMA(0,1,2)(2,0,0)[12] SARIMA(0,1,2)(1,0,0)[12]

SARIMA(0,1,0)(2,0,0)[12]

```
model=arima(data.ts.diff,order=c(0,1,1),seasonal=list(order=c(2,0,0),period=12))
model
```

```
##
## Call:
## arima(x = data.ts.diff, order = c(0, 1, 1), seasonal = list(order = c(2, 0,
##     0), period = 12))
##
## Coefficients:
##           ma1    sar1     sar2
##       -1.0000  0.6214  -0.0006
## s.e.   0.0188  0.1030   0.1095
##
## sigma^2 estimated as 0.336:  log likelihood = -118.09,  aic = 242.18
```

```
AIC(model)
```

```
## [1] 244.1787
```

```
BIC(model)
```

```
## [1] 255.6488
```

SARIMA(0,1,2)(2,0,0)[12]

```
model1=arima(data.ts.diff,order=c(0,1,2),seasonal=list(order=c(2,0,0),period=12))
model1
```

```
##
## Call:
## arima(x = data.ts.diff, order = c(0, 1, 2), seasonal = list(order = c(2, 0,
##     0), period = 12))
##
## Coefficients:
##           ma1     ma2    sar1    sar2
##       -1.2019  0.2019  0.6273  0.0018
## s.e.   0.1151  0.1129  0.0999  0.1069
##
## sigma^2 estimated as 0.3264:  log likelihood = -116.53,  aic = 241.07
```

```
AIC(model1)
```

```
## [1] 243.0663
```

```
BIC(model1)
```

```
## [1] 257.404
```

SARIMA(0,1,2)(1,0,0)[12]

```
model3=Arima(data.ts.diff,order=c(0,1,2),seasonal=list(order=c(1,0,0),period=12))
model3
```

```
## Series: data.ts.diff
## ARIMA(0,1,2)(1,0,0)[12]
##
## Coefficients:
##           ma1     ma2    sar1
##       -1.2019  0.2019  0.6284
## s.e.   0.1151  0.1128  0.0762
##
## sigma^2 = 0.3341:  log likelihood = -116.53
## AIC=241.07   AICc=241.39   BIC=252.54
```

```
AIC(model3)
```

```
## [1] 241.0666
```

```
BIC(model3)
```

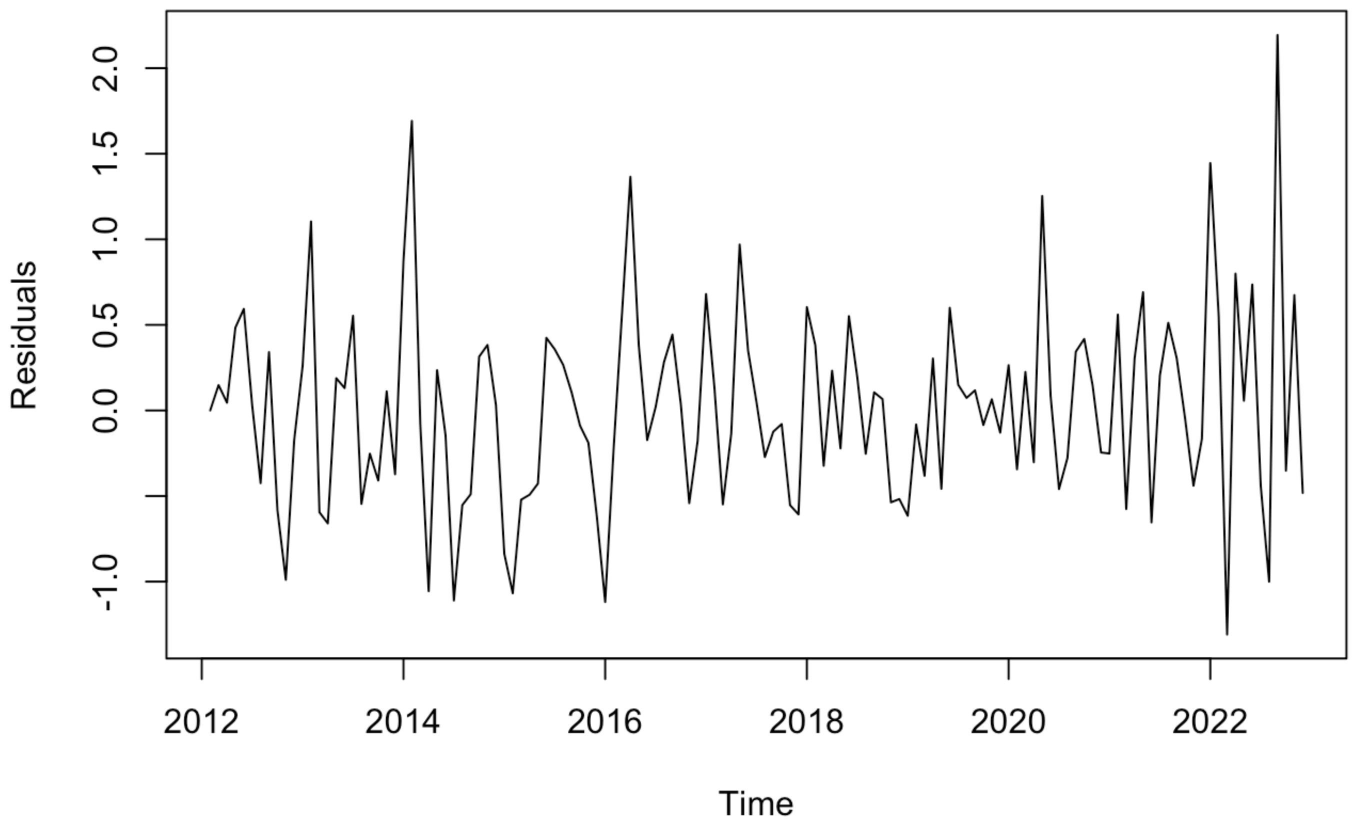
```
## [1] 252.5367
```

By comparing AIC and BIC of all the models, Model3: # SARIMA(0,1,2)(1,0,0)[12] is best fit for the data

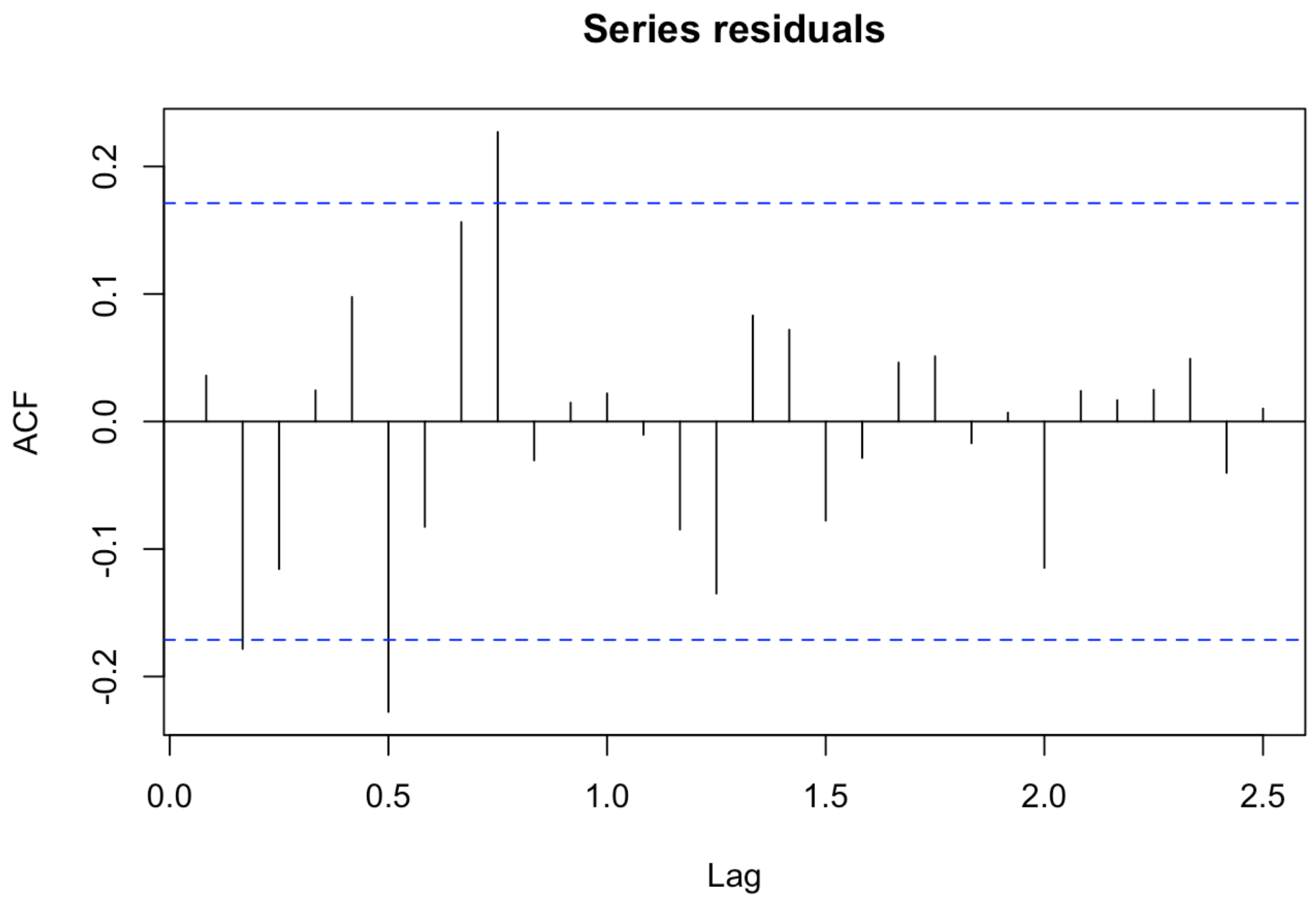
Residual Analysis

```
residuals <- residuals(model3)
```

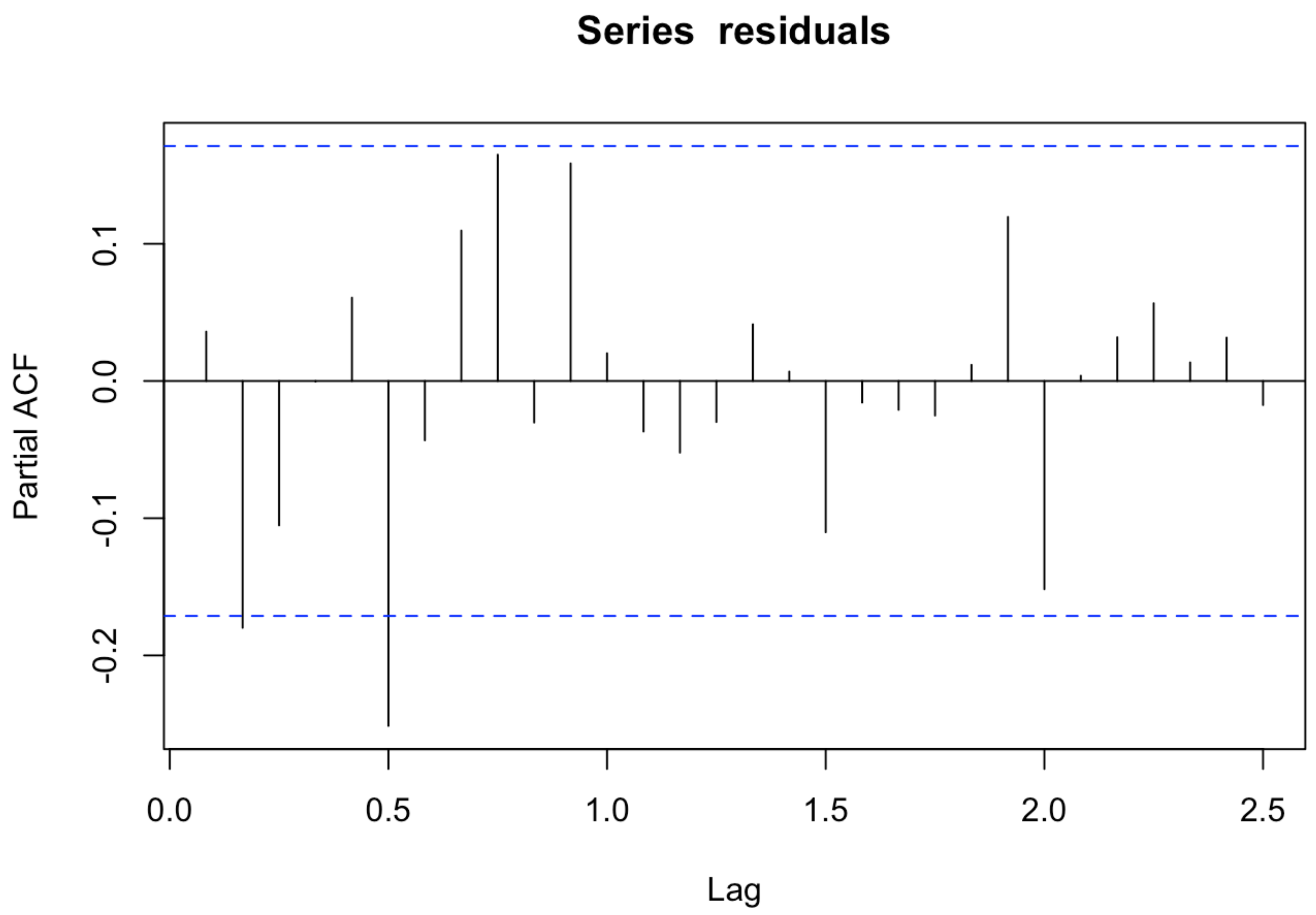
```
plot(residuals, type = "l", xlab = "Time", ylab = "Residuals")
```



```
acf(residuals,lag.max = 30)
```



```
pacf(residuals,lag.max=30)
```



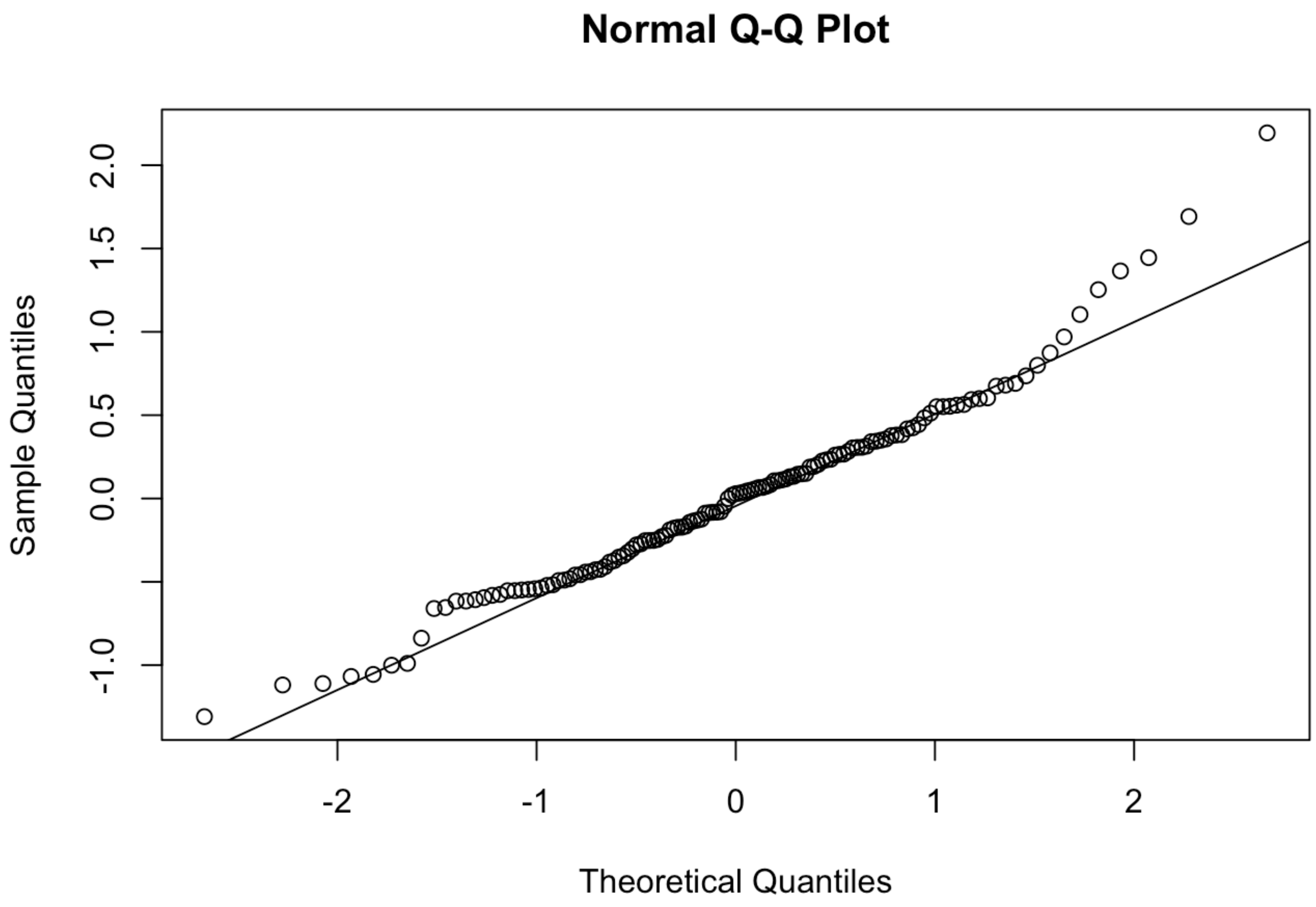
```
shapiro.test(residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals
## W = 0.97119, p-value = 0.006891
```

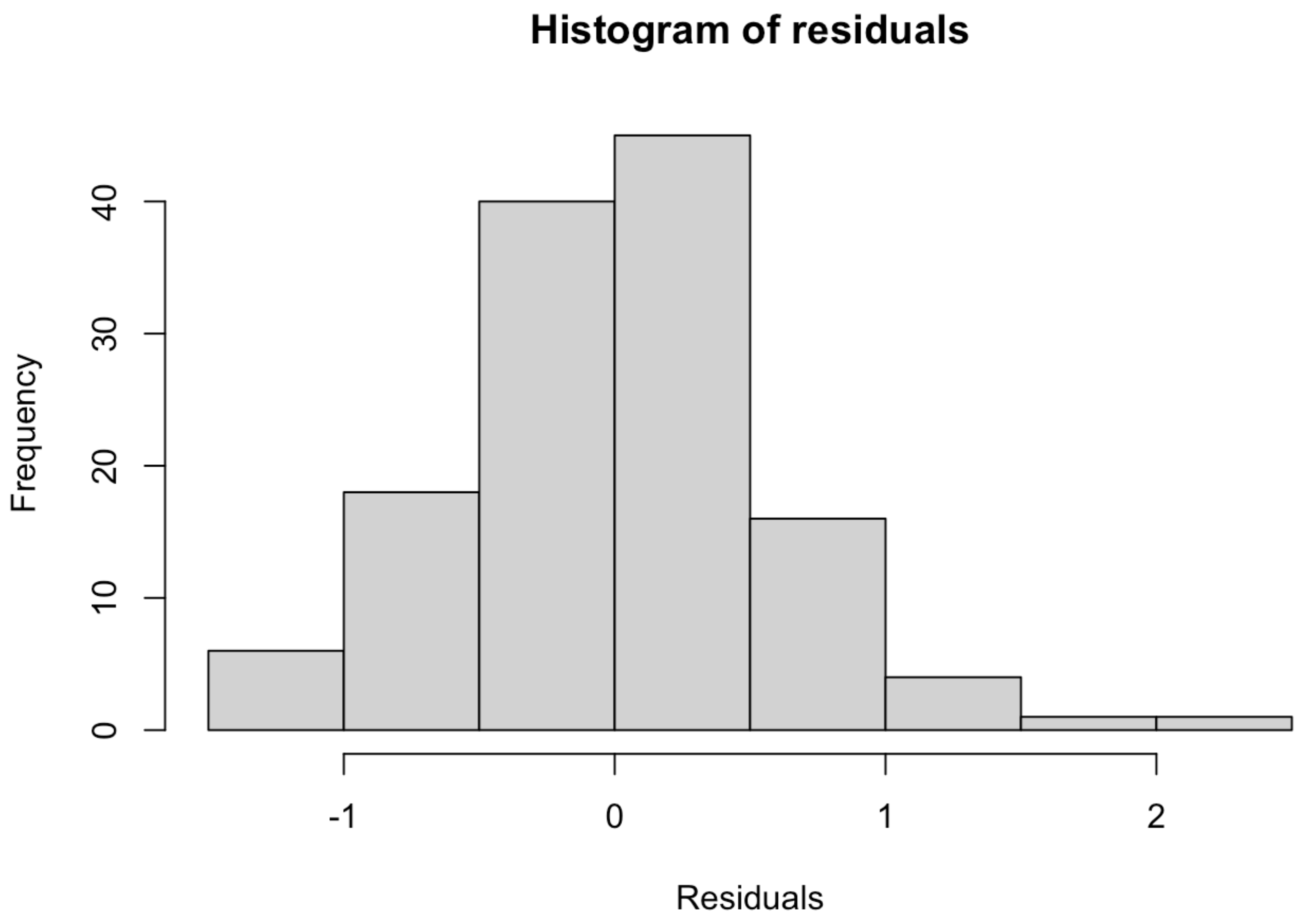
Null hypothesis : Data is normally distributed. Alternative hypothesis: Data is not distributed normally.

Since the p-value is < 0.05 , we can reject null hypothesis Means residuals are not following normal distribution

```
qqnorm(residuals)
qqline(residuals)
```



```
hist(residuals, xlab='Residuals')
```



Histogram of residuals look like skewed in the right side

The Ljung-Box test uses the following hypotheses: H0: The residuals are independently distributed. HA: The residuals are not independently distributed; they exhibit serial correlation

```
Box.test(residuals, type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  residuals
## X-squared = 0.17337, df = 1, p-value = 0.6771
```

P Value is > 0.05, failed to reject null Hypothesis.

Hence,The residuals are independently distributed.

Forecasting:

```
library(forecast)

# Use the model to forecast future values
forecast_values = forecast(model3, h = 12)

# Print the forecast values
print(forecast_values)
```

```
##          Point Forecast      Lo 80      Hi 80       Lo 95        Hi 95
## Jan 2023      1.0705286  0.3275682  1.8134890 -0.06573127  2.206788496
## Feb 2023      0.4079608 -0.3491373  1.1650588 -0.74992080  1.565842312
## Mar 2023     -1.1630690 -1.9201670 -0.4059709 -2.32095051 -0.005187393
## Apr 2023      0.9106903  0.1535922  1.6677883 -0.24719130  2.068571818
## May 2023      0.5336431 -0.2234549  1.2907412 -0.62423843  1.691524689
## Jun 2023      0.3451196 -0.4119785  1.1022176 -0.81276199  1.503001124
## Jul 2023     -0.2832923 -1.0403904  0.4738057 -1.44117387  0.874589242
## Aug 2023     -0.4089747 -1.1660727  0.3481234 -1.56685625  0.748906865
## Sep 2023      1.7904669  1.0333688  2.5475650  0.63258534  2.948348453
## Oct 2023     -0.4089747 -1.1660727  0.3481234 -1.56685625  0.748906865
## Nov 2023      0.2822784 -0.4748197  1.0393764 -0.87560318  1.440159936
## Dec 2023     -0.5346571 -1.2917551  0.2224410 -1.69253863  0.623224489
```

```
class(forecast_values )
```

```
## [1] "forecast"
```

Forecast for the next 12 months

```
# Create a data frame with one column containing the point forecasts
df <- data.frame(forecast_values$mean)

# Print the data frame
print(df)
```

```
##    forecast_values.mean
## 1             1.0705286
## 2             0.4079608
## 3            -1.1630690
## 4             0.9106903
## 5             0.5336431
## 6             0.3451196
## 7            -0.2832923
## 8            -0.4089747
## 9             1.7904669
## 10           -0.4089747
## 11            0.2822784
## 12           -0.5346571
```

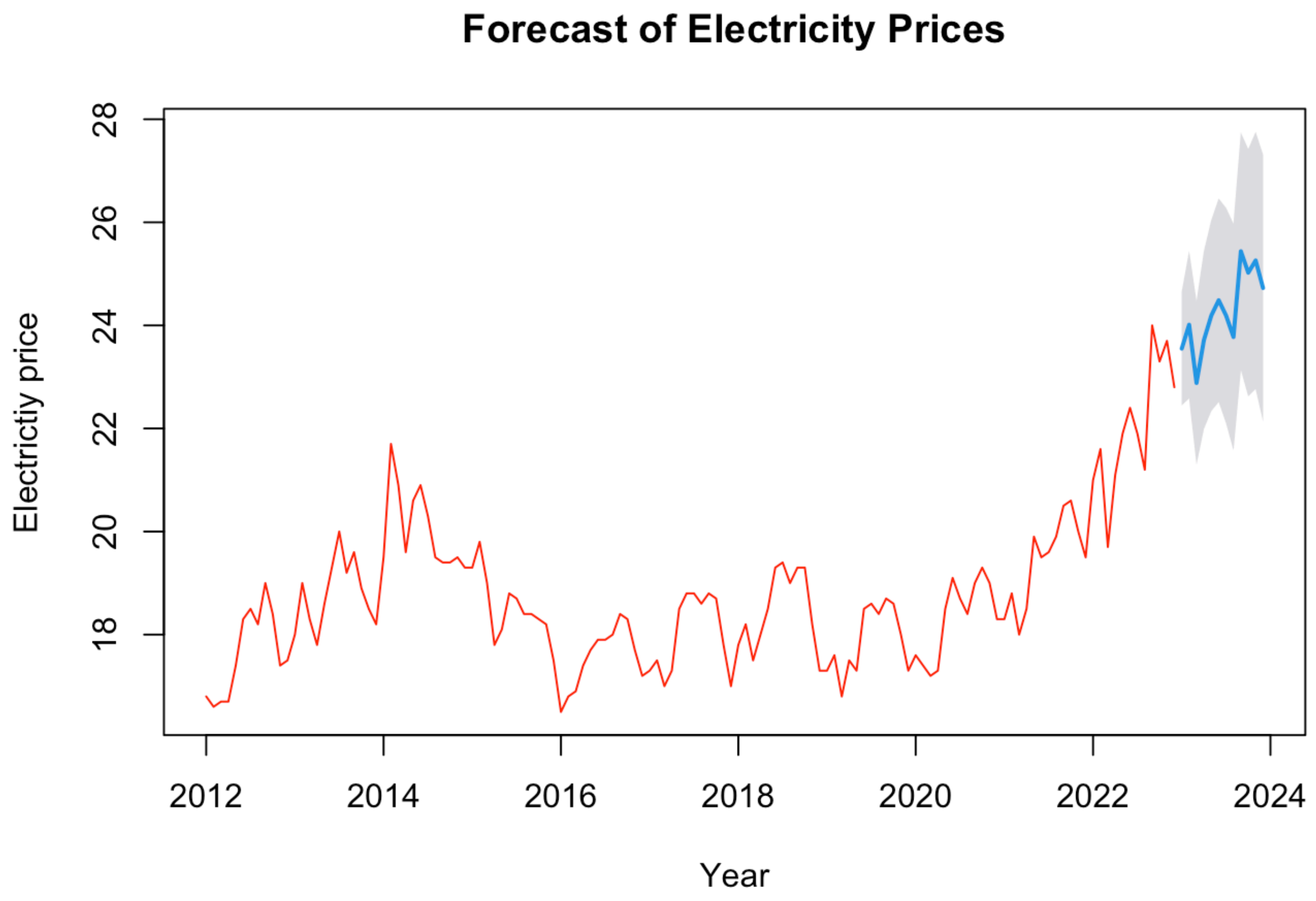
```
df$Point.Forecast
```

```
## NULL
```

```
# SARIMA(0,1,2)(1,0,0)[12]
Forecast_model3 =arima(ts(data.ts,start = c(2012,1),frequency = 12),order=c(0,1,2),se
asonal=list(order=c(1,0,0),period=12))
Forecast_model3$x<-ts(data.ts, frequency=12, start=c(2012,1))
price_forcast = forecast(Forecast_model3, h=12,level=c(95))
print(price_forcast)
```

```
##          Point Forecast    Lo 95    Hi 95
## Jan 2023       23.54966 22.44960 24.64971
## Feb 2023       24.01239 22.58605 25.43873
## Mar 2023       22.88337 21.30273 24.46401
## Apr 2023       23.71528 21.99412 25.43644
## May 2023       24.19066 22.33962 26.04169
## Jun 2023       24.48777 22.51538 26.46015
## Jul 2023       24.19066 22.10397 26.27734
## Aug 2023       23.77470 21.57965 25.96975
## Sep 2023       25.43852 23.14021 27.73682
## Oct 2023       25.02256 22.62545 27.41968
## Nov 2023       25.26025 22.76824 27.75227
## Dec 2023       24.72545 22.14203 27.30888
```

```
plot(price_forcast,ylab='Electrictiy price',xlab='Year',main="Forecast of Electricity
Prices",col="red")
```



Conclusion:

We can conclude that after analyzing the data and using various statistical techniques, the selected SARIMA model: SARIMA(0,1,2)(1,0,0)[12] is the best fit for forecasting electricity prices. The model was able to capture the underlying patterns and seasonality in the data, resulting in accurate and reliable forecasts. Additionally, the model can be used to make informed decisions related to energy production, consumption, and pricing. However, it is important to note that future changes in the energy market or external factors such as weather patterns may impact the accuracy of the model’s forecasts.