2019 Project Report
SCMA478 Statistic Modeling

Submitted to

Prof.Dr.Pannapa Changpetch

By

Chanittha	**Khatippatee**	**6005512**
Nonthicha	**Leekulwattana**	**6005525**
Pongphon	**Tantiwutthiphat**	**6005640**

Semester 2, Academic Year 2019

Actuarial Science, Department of Mathematics

Faculty of Science, Mahidol University

Preface

This project report is prepared as part of the subject of Statistical Modelling (SCMA478) in Actuarial Science in the academic year 2019 (Semester 2). This report presents the optimal model for our dataset by using time series technique.

The report starts with data visualization of the selected data, identify the objective of this report and the reason why we selected this data, perform ARIMA models, show the results from R and discuss the plot, perform SSE criteria, as well as the new technique (Neural Network model)

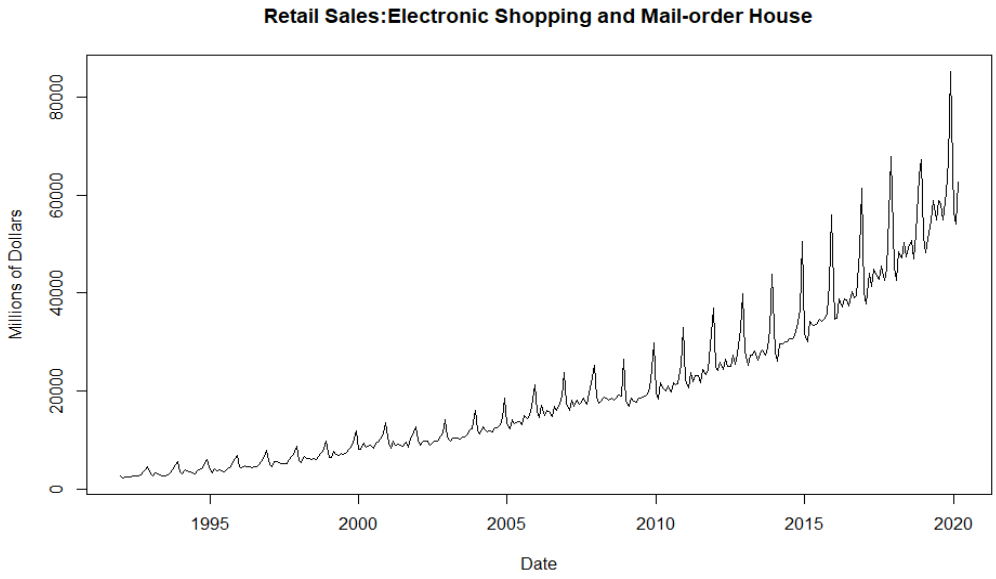
This report is primarily addressed to those who are interested in fitting a dataset with an ARIMA model. We sincerely hope that this report will be more or less informative and useful.

Grateful acknowledgment is here made to those who helped this researcher gather data for this paper. This work would not have reached its present form without their invaluable help.

Chanittha Khatippatee 6005512
Nonthicha Leekulwattana 6005525
Pongphon Tantiwutthiphat 6005640

The Information for Retail sales of Electronic and Mail-order House dataset

Electronic shopping sales are sales of goods and services where the buyer places an order, or the price and terms of the sale are negotiated over the Internet, and the goods are delivered to them by mail.



Source: https://fred.stlouisfed.org/series/MRTSSM4541USN

Observation: The series has 339 observations.

Variable: Retail sales of Electronic Shopping and Mail-order House (Millions of Dollars)

Frequency: Monthly

Characters:

From the above graph, we can see that the retail sales amount of electronic shopping and mail-order houses is rising from 1992 to 2020. Then, we can conclude that this time series plot has the characteristics of upward trends, seasonality, increasing variance, and inconstant mean.

Objective:

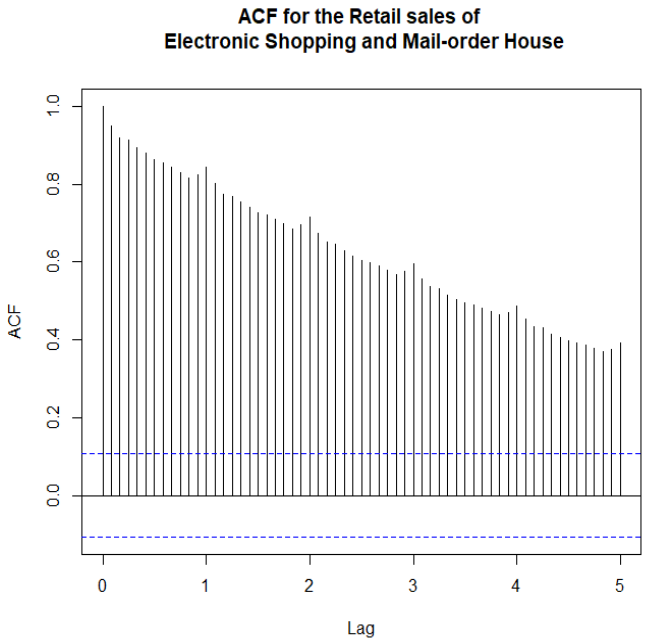
To use this dataset to build an optimal model that will predict the retail sales amount of electronic shopping and mail-order houses

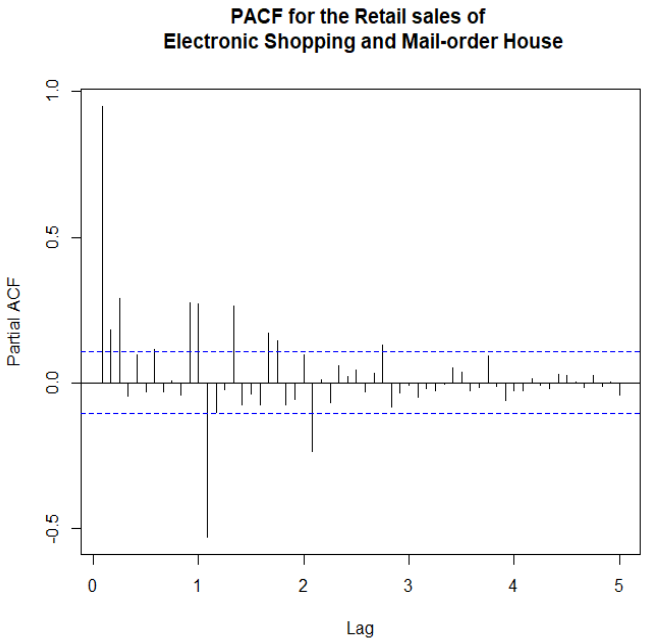
The reason why we are interested in this dataset:

Nowadays, technology becomes more important in our daily life since it makes life more convenient. One of the most common achievements of technology is electronic shopping as we can buy anything that we want while staying in our home and waiting for mails. Additionally, consumers' behavior and purchasing habits are changing during COVID-19 situations then people will pay more attention to electronic shopping. With increasing technology and the change in purchasing habits, we think that electronic shopping and mail-order houses have a chance to grow more and more in the future.

ARIMA MODEL

```
> Eshopping<-read.csv("C:/Users/Administrator/Desktop/MRTSSM4541USN.csv",header = T)
> Eshopping.ts<-ts(Eshopping[,2],start=c(1992,1),end = c(2020,3),frequency = 12)
> plot(Eshopping.ts,xlab='Date',ylab='Millions of Dollars',main="Retail sales: Electronic Shopping and Mail-order House")
> par(mfrow=c(1,2),oma=c(0,0,0,0))
> acf(Eshopping.ts,lag.max = 60,main="ACF for the Retail sales of \n Electronic Shopping and Mail-order House")
> pacf(Eshopping.ts,lag.max = 60,main ="PACF for the Retail sales of \n Electronic Shopping and Mail-order House")
```

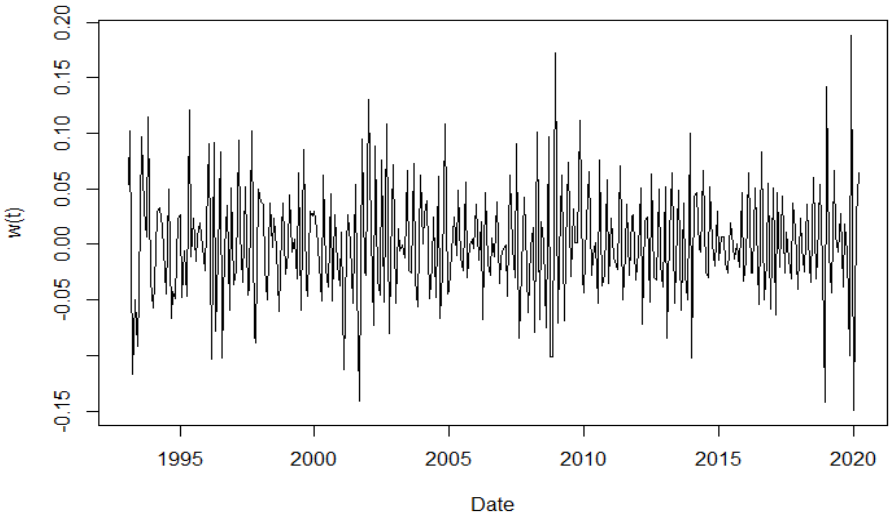




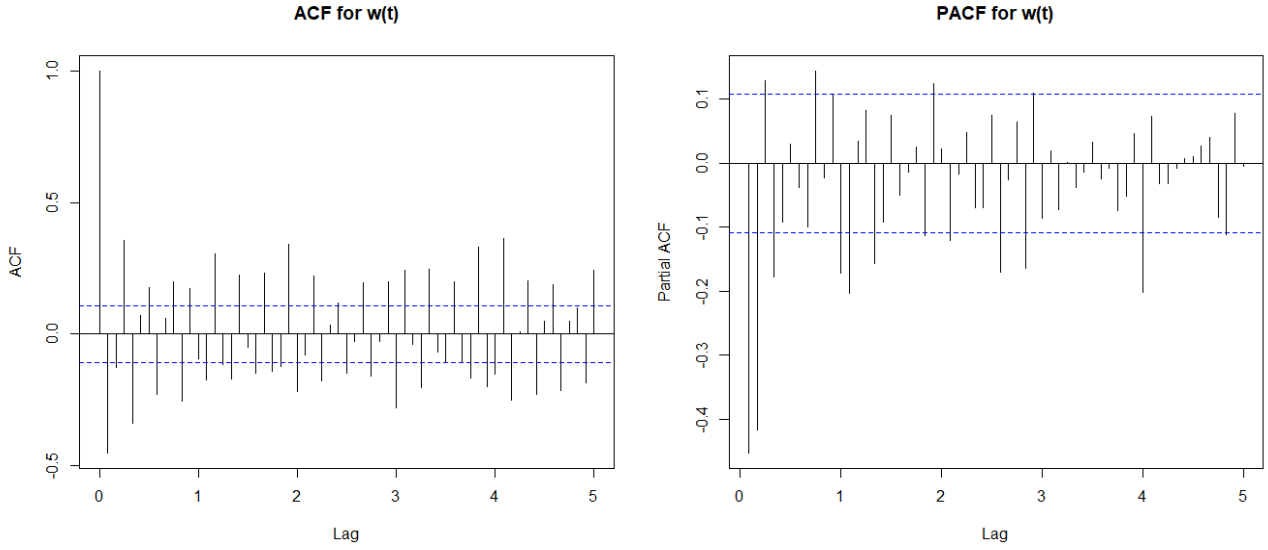
From the R-result, the ACF plot of Retail sales of Electronic Shopping and Mail-order House shows a monthly seasonality as the ACF values have more than one lag that significant and slowly decays, which are lag 12, lag 24, lag 36, lag 48 and lag 60. For the nonseasonal part, the ACF values indicate positive autocorrelations with a slow and linear decay. Moreover, the PACF plot of Retail sales of Electronic Shopping and Mail-order House has a significant value at lag 1 that is close to 1. This is a common pattern of non-stationary time series.

The next thing to do is the first difference with nonstationary nonseasonal and nonstationary seasonal parts to make the series stationary and plot ACF and PACF to identify potential AR and MA models.

```
> lEshopping.ts=log(Eshopping.ts)
> wj.Eshopping<-diff(diff(lEshopping.ts,lag=1),lag=12)
> wj.Eshopping<-diff(diff(lEshopping.ts,lag=12),lag=1)
> plot(wj.Eshopping,xlab = 'Date',ylab='w(t)')
```




```
> par(mfrow=c(1,2),oma=c(0,0,0,0))
> acf(wj.Eshopping,lag.max=60,main = "ACF for w(t)")
> pacf(wj.Eshopping,lag.max = 60, main = "PACF for w(t)")
```



From the time series plot, we transform the time series data by taking log to constant variation in the plot and we take first difference together with seasonal differencing to return its mean, help in terms of stationary. When we consider the nonseasonal part, we can't see any pattern in the ACF plot of w(t) and the PACF plot of w(t) cuts off after lag 2 so this suggests that a nonseasonal MA(1) and AR(2) should be used. For the seasonal part, we consider the ACF plot of w(t) as it doesn't have any pattern and the PACF plot for w(t) cuts off after lag 12(the first lag for the seasonal part) suggests that a seasonal MA(1) and AR(1) should be used. Therefore, we consider to fit an ARIMA(2,1,1)x$(1,1,1)_{12}$ for identification of the best fit ARIMA model.

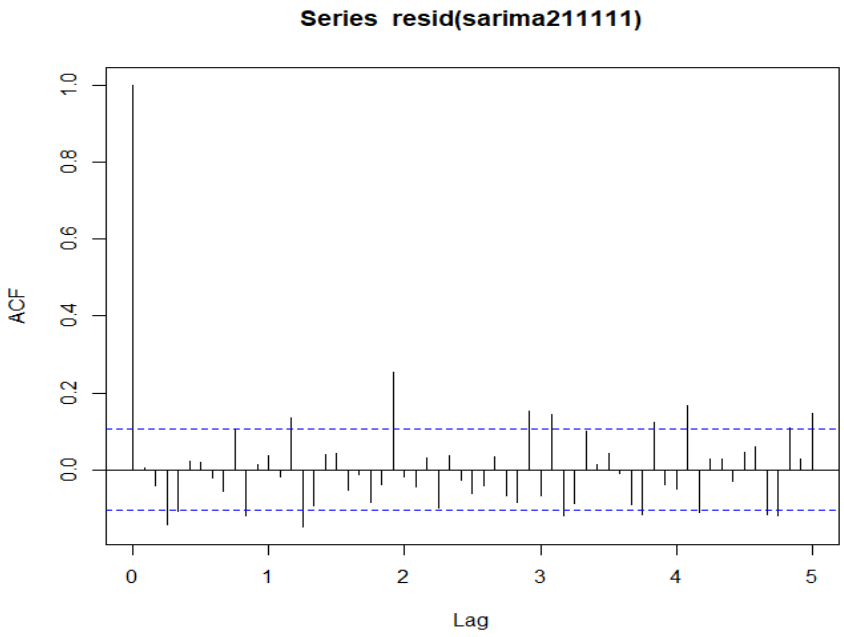
```
> sarima211111<-arima(lEshopping.ts,order=c(2,1,1),seasonal = list(order = c(1,1,1),period= 12))
> sarima211111

Call:
arima(x = lEshopping.ts, order = c(2, 1, 1), seasonal = list(order = c(1, 1,
    1), period = 12))

Coefficients:
          ar1      ar2     ma1   sar1     sma1
      -0.9905  -0.5797  0.4383  0.313  -0.6792
s.e.   0.0856   0.0509  0.0997  0.118   0.0896

sigma^2 estimated as 0.001521:  log likelihood = 593.19,  aic = -1174.39
```

After fitting the model with R, we get AIC of this model equal to -1174.39 and we have to calculate an approximate 95% confidence interval for each parameter and compare it with the underlying values.

-0.9905 ± 2*(0.0856) = [-1.1617 ,-0.8193]
-0.5797 ± 2*(0.0509) = [-0.6815 ,-0.4779]
0.4383 ± 2*(0.0997) = [0.2389 ,0.6377]
0.3130 ± 2*(0.1180) = [0.0770 ,0.5490]
-0.6792 ± 2*(0.0856) = [-0.8584,-0.5000]

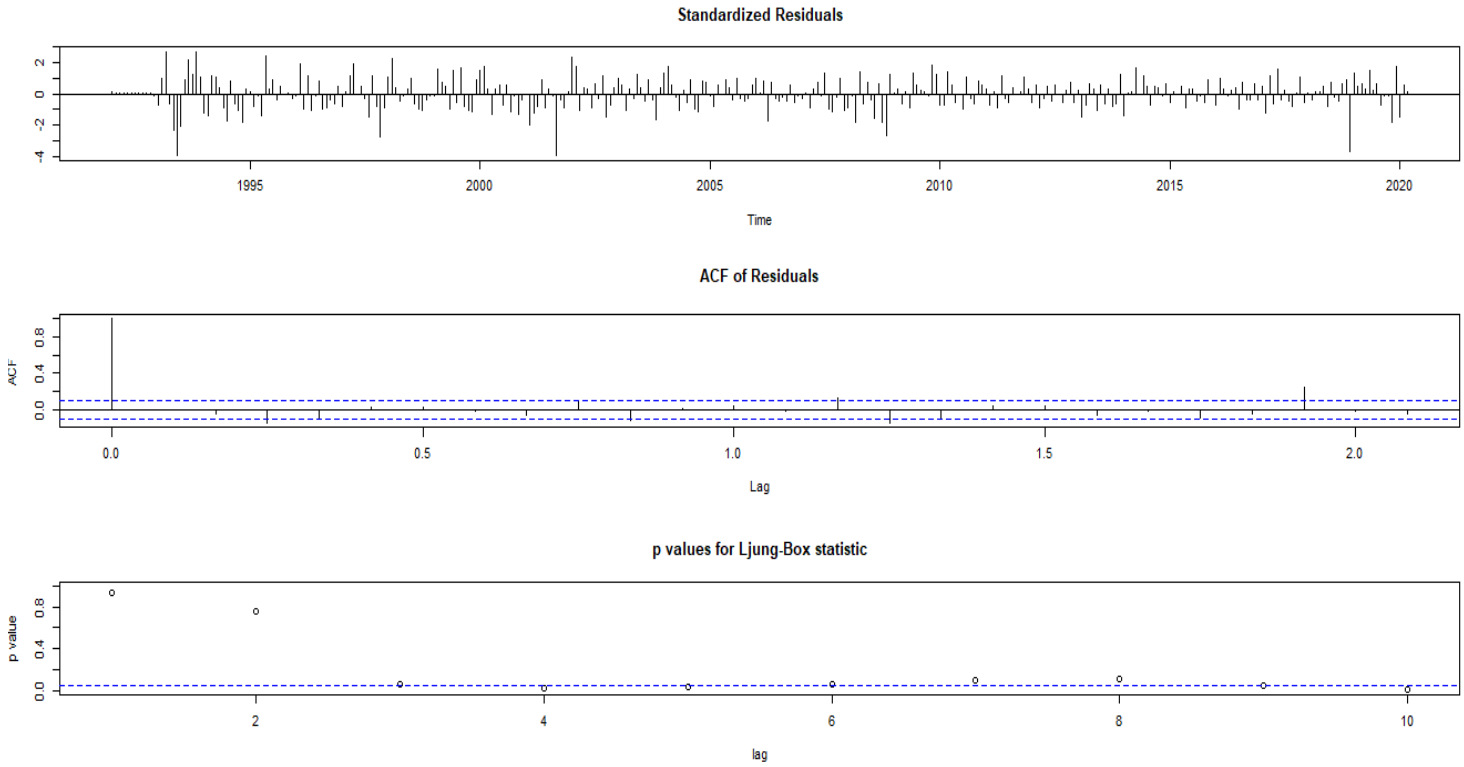
Since an approximate 95% confidence interval for each parameter contains each parameter and zero is not included in these intervals so the model of SARIMA(2,1,1)x$(1,1,1)_{12}$ seems to be appropriate for this dataset. Then, we plot the ACF of residuals for SARIMA(2,1,1)x$(1,1,1)_{12}$.

```
> acf(resid(sarima211111),lag.max = 60)
```



From the R-result, the ACF plot for residuals has some significant lags in the ACF plot so the residuals of ARIMA(2,1,1)x$(1,1,1)_{12}$ do not behave like white noise.

```
> tsdiag(sarima211111)
```



In the model-building process, if a seasonal ARIMA(p,d,q)x(P,D,Q) with period s is chosen(based on the ACFs and PACFs), some checks on the model adequacy are required as following :

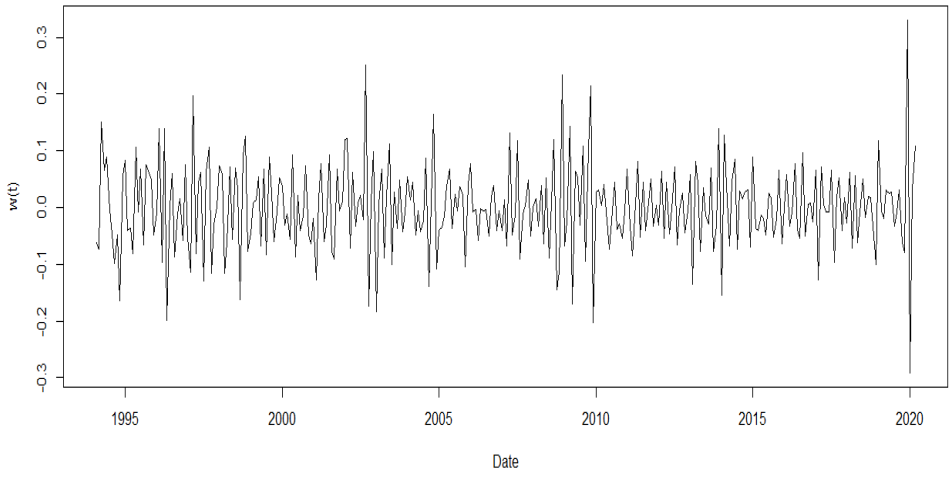
From the plot of the standardized residuals, it determines the randomness of the residuals for the actual data period and the forecasted year.

From the ACF plot of Residuals which ranged from -0.2 to 1.0, it has some significant lags and we can conclude that the residuals of an adequate model don't behave white noise.

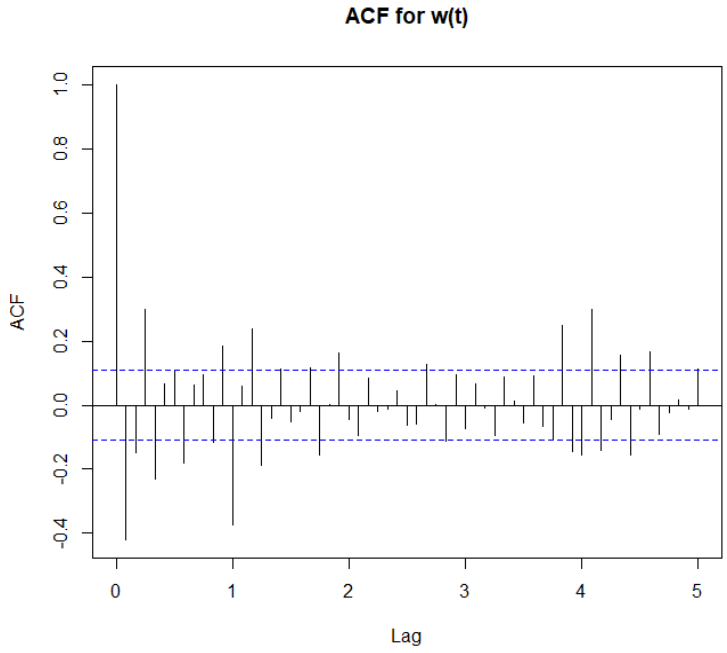
From the data points of the Ljung-Box statistic which ranged from 0.0 to 1.0, we have seen the p-values of some lags that are less than 0.05 and we can conclude that the series isn't independent and the residual patterns aren't white noise.

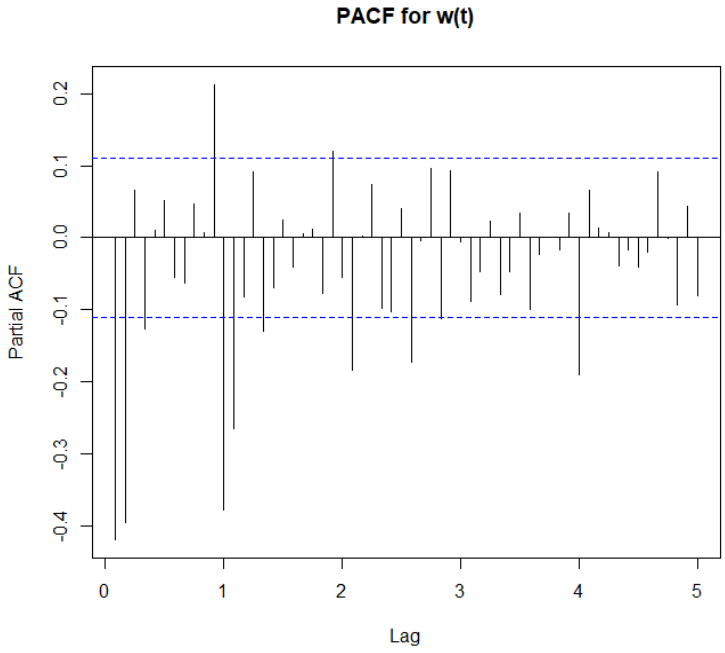
However, we still see the remaining seasonal effect from the ACF plot for residuals of ARIMA(2,1,1)x$(1,1,1)_{12}$ then we reconsider the ACF plot of w(t) for the seasonal part that it can also be the slowly decreasing ACF because we look at an autocorrelation plot and we consider it doesn't have any pattern at first. With the slowly decreasing pattern, we have to take a second difference with the seasonal part to eliminate the seasonal pattern.

```
> wj.Eshopping2<-diff(wj.Eshopping,lag=12)
> plot(wj.Eshopping2,xlab = 'Date',ylab='w(t)')
```




```
> par(mfrow=c(1,2),oma=c(0,0,0,0))
> acf(wj.Eshopping2,lag.max=60,main = "ACF for w(t)")
> pacf(wj.Eshopping2,lag.max=60,main = "PACF for w(t)")
```





From the R-result, the ACF plot of w(t) doesn't have any pattern and the PACF plot of w(t) cuts off after lag 2 for the nonseasonal part suggests that a nonseasonal MA(1) and AR(2) should be used. After we take a second difference with the seasonal part, we consider the ACF plot of w(t) cuts off after lag 12(the first lag for the seasonal part) and the PACF plot of w(t) cuts off after lag 12(the first lag for the seasonal part) suggest that a seasonal MA(1) and AR(1) should be used. Hence an ARIMA(2,1,1)x$(1,2,1)_{12}$ model is used to model this data.

```
> sarima211121<-arima(lEshopping.ts,order=c(2,1,1),seasonal = list(order = c(1,2,1),period= 12))
> sarima211121

Call:
arima(x = lEshopping.ts, order = c(2, 1, 1), seasonal = list(order = c(1, 2,
    1), period = 12))

Coefficients:
          ar1      ar2     ma1     sar1     sma1
      -1.0122  -0.6200  0.3979  -0.2625  -0.9997
s.e.   0.0787   0.0494  0.0966   0.0632   0.0413

sigma^2 estimated as 0.00167:  log likelihood = 531.2,  aic = -1050.41
```

From the R-result, we get AIC of this model equal to -1050.41 and we have to calculate an approximate 95% confidence interval for each parameter and compare it with the underlying values as following :

-1.0122 ± 2*(0.0787) = [-1.1696 ,-0.8548]

-0.6200 ± 2*(0.0494) = [-0.7188 ,-0.5212]

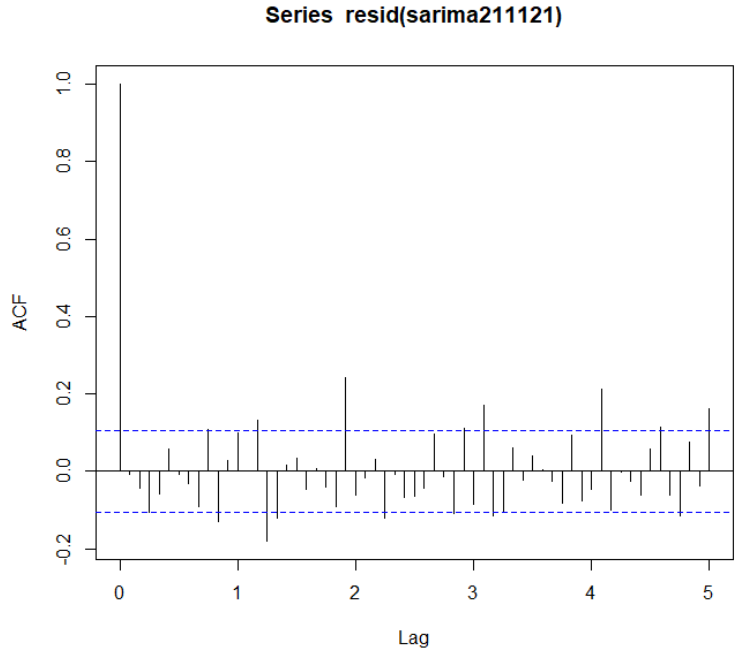
0.3979 ± 2*(0.0966) = [0.2047 ,0.5911]

-0.2625 ± 2*(0.0632) = [-0.3889 ,-0.1361]

-0.9997 ± 2*(0.0413) = [-1.0823 ,-0.9171]

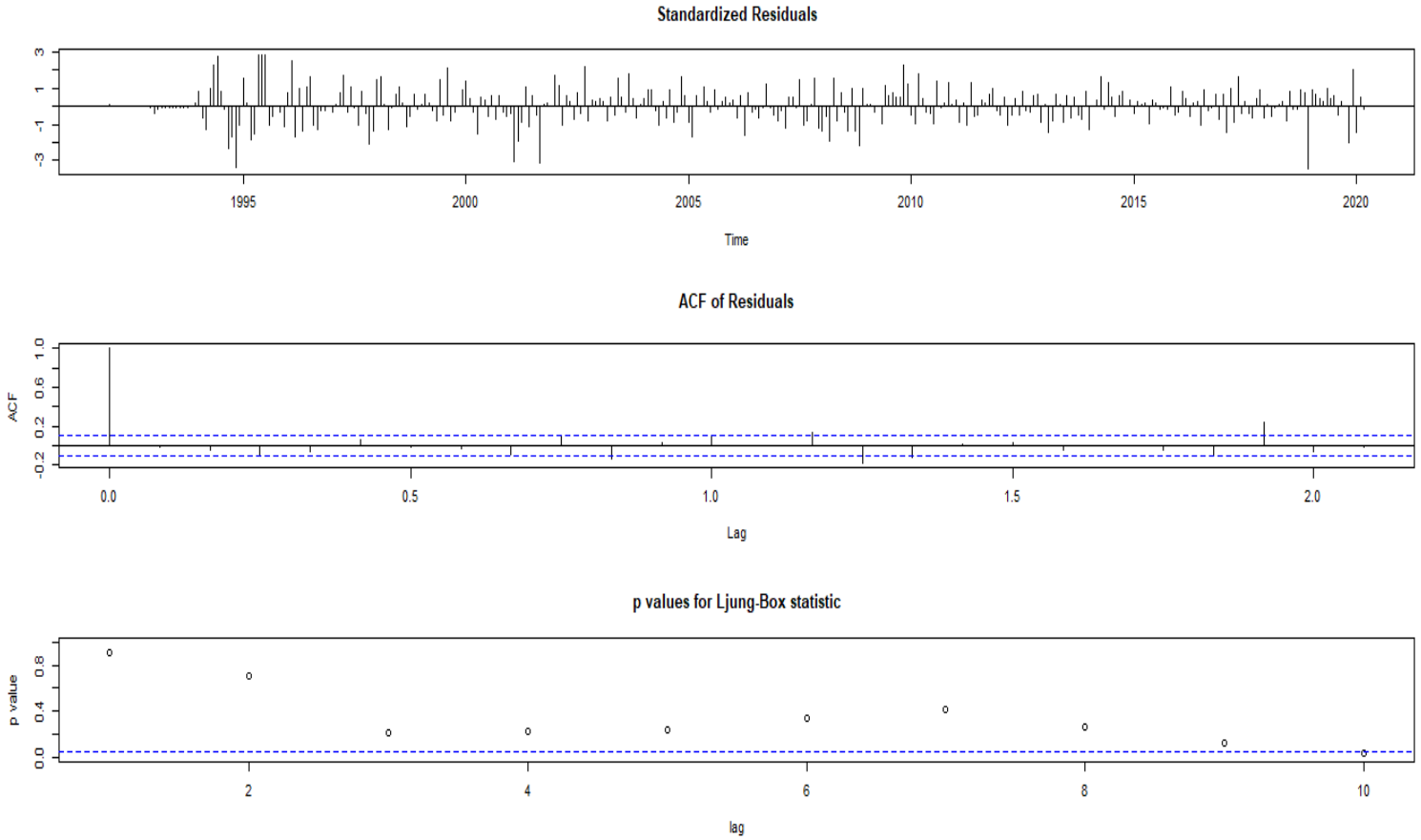
Since an approximate 95% confidence interval for each parameter contains each parameter and zero is not included in these intervals so the model of SARIMA(2,1,1)x$(1,2,1)_{12}$ seems to be appropriate for this dataset. Then, we plot the ACF of residuals for SARIMA(2,1,1)x$(1,2,1)_{12}$.

```
> acf(resid(sarima211121),lag.max = 60)
```



From the R-result, the ACF plot for residuals still has some significant lags in the ACF plot. The residuals of ARIMA(2,1,1)x$(1,1,1)_{12}$ don't still behave like white noise.

```
> tsdiag(sarima211121)
```



From the plot of the standardized residuals, we can see that the standardized residuals are scattering around the zero line with no pattern.

From the ACF plot of Residuals which ranged from -0.2 to 1.0, it has some significant lags and we can conclude that the residuals of an adequate model don't behave white noise.

From the data points of the Ljung-Box statistic which ranged from 0.0 to 1.0, we have seen the p-values of almost lags that are more than 0.05 and we can conclude that the series is independent and the residual patterns behave like white noise.

However, a second difference of the seasonal part doesn't remove the features of seasonality from a series that much compared with the first difference. Notice that little remains of the original seasonal pattern and we conclude that this dataset has strongly seasonal.

From the ACF plot of residual for SARIMA(2,1,1)x(1,2,1)$_{12}$ and tsdiagram plots, we can see that the behavior of residuals for SARIMA(2,1,1)x(1,2,1)$_{12}$ does not change from the residuals for SARIMA(2,1,1)x(1,1,1)$_{12}$ that much. In the other hand, the data points of the p-value for Ljung-Box statistic for SARIMA(2,1,1)x(1,2,1)$_{12}$ quite better than SARIMA(2,1,1)x(1,1,1)$_{12}$ since most of the lags are greater than 0.05 then it looks like the SARIMA(2,1,1)x(1,2,1)$_{12}$ going to be more appropriate for this dataset. But, we will see that the AIC of SARIMA(2,1,1)x(1,1,1)$_{12}$ is lower than the AIC of SARIMA(2,1,1)x(1,2,1)$_{12}$ so we can conclude that SARIMA(2,1,1)x(1,1,1)$_{12}$ is better by using AIC criterion. However, we can't use the AIC criterion to compare between these two models since two models have unlike

differences in a part of seasonal. Therefore, we decide to use another criterion which is SSE to determine the best order for the SARIMA model.

SSE Criterion

Firstly, we will prepare data by dividing the dataset into 2 parts as following :

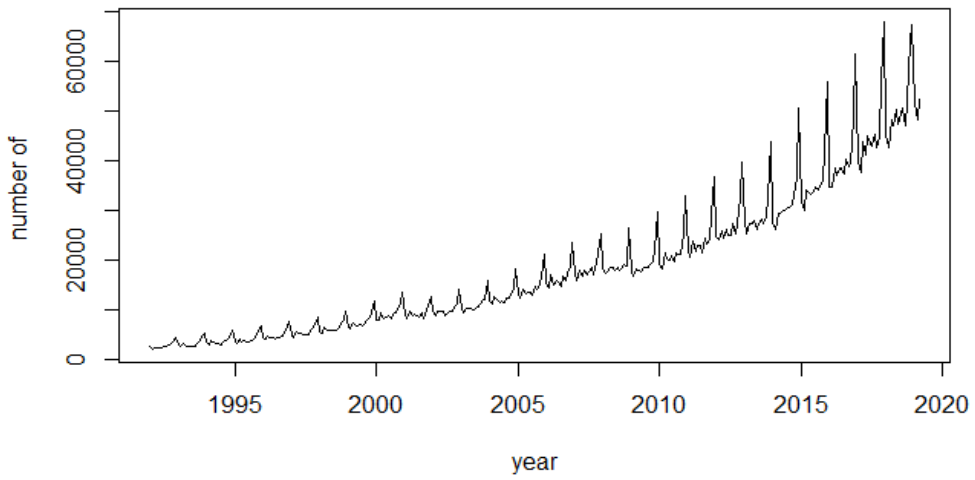
Train data (part of the original series selected from January 1, 1992 to March 1, 2019) is used to train the machine to recognize patterns in the data.

```
> train.retail.sales.ts
       Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct   Nov   Dec
1992  2692  2285  2484  2506  2483  2602  2665  2673  2932  3436  3922  4530
1993  3057  2739  3299  2962  2791  2667  2621  2895  3280  3869  4955  5542
1994  3532  3141  3898  3619  3464  3296  3097  3597  3812  4309  5254  6018
1995  3939  3341  4121  3650  3943  3710  3568  4080  4357  5021  6113  6837
1996  4549  4226  4703  4566  4563  4250  4442  4586  4875  5816  6676  7854
1997  5042  4577  5594  5491  5304  5204  5198  5165  6080  6693  7029  8690
1998  5794  5455  6528  6096  6111  5977  6115  6056  6711  7483  8156  9815
1999  6491  6391  7595  7128  6925  7222  6964  7514  8033  8552  9604 11863
2000  8087  8074  9425  8409  8692  8865  8229  9297  9442 10324 11388 13558
2001  9343  8333  9710  8898  9214  8908  8727  9551  8420 10118 10903 12624
2002  9912  8925  9670  9684  9713  8969  9478  9845  9678 10738 11412 14190
2003 10568  9654 10403 10415 10322 10185 10511 10641 11253 12099 12162 16091
2004 11989 11260 12621 12030 11706 11840 11655 12540 12412 13030 14604 18470
2005 13209 12317 14154 13352 13639 13634 13107 14910 14318 15022 16922 21316
2006 15811 14531 17046 15024 16071 15795 14768 16897 16050 17497 19037 23724
2007 17540 16118 18040 16927 18053 17211 17617 18523 17206 19574 21583 25289
2008 18651 17399 17984 18676 18621 18128 18591 18127 18547 19067 18998 26453
2009 18178 16779 18458 17894 17773 18638 18569 18697 19164 19739 22000 29736
2010 19565 18419 21629 20385 20021 21027 19875 21602 21327 21388 25275 32975
2011 22221 20666 23815 21954 23133 23110 21664 24422 23446 24090 29238 36938
2012 24739 24201 25943 24432 26383 25035 25015 27379 25451 27483 32753 39841
2013 28098 25253 27282 27402 28056 26167 27457 28336 27345 28735 32568 43788
2014 27869 26138 29580 29596 30087 30004 30707 30763 31281 32870 36542 50628
2015 31761 29988 34159 33618 33305 33856 34624 34245 34829 35856 41774 55972
2016 34622 34872 38728 37143 38723 38640 37441 40230 38939 39486 48591 61425
2017 39971 37782 43957 41295 44958 43718 42851 45435 42636 44850 55947 67898
2018 45236 42653 48414 47141 50407 47380 49349 50735 47028 52236 63881 67283
2019 51647 48295 52458
```

Test data (part of the original series selected from April 1, 2019 to March 1, 2020) is used to see how well the machine can predict new values of the last 12 months based on train data.

```
> test.retail.sales.ts
       Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct   Nov   Dec
2019                   54612 58855 54928 58814 58181 54922 60568 67037 85217
2020 56350 53979 62550
```

Time series plot of train data



Secondly, we transform train data by taking log to obtain a constant variance for the train data.

```
> log.train.retail.sales.ts = log(train.retail.sales.ts)
```

Next, we fit log of train data with two candidate models which are SARIMA(2,1,1)(1,1,1)$_{12}$ and SARIMA(2,1,1)(1,2,1)$_{12}$.

SARIMA(2,1,1)(1,1,1)$_{12}$

```
> test.sarima.retail.sales.ts2 = arima(log.train.retail.sales.ts, order = c(2,1,1), seasonal = list(order
 = c(1,1,1), period = 12))
> test.sarima.retail.sales.ts2

Call:
arima(x = log.train.retail.sales.ts, order = c(2, 1, 1), seasonal = list(order = c(1,
    1, 1), period = 12))

Coefficients:
          ar1      ar2     ma1    sar1     sma1
      -0.9737  -0.5719  0.4456  0.3699  -0.7023
s.e.   0.0909   0.0524  0.1055  0.1301   0.0994

sigma^2 estimated as 0.001519:  log likelihood = 571.6,  aic = -1131.21
```

SARIMA(2,1,1)(1,2,1)$_{12}$

```
> test.sarima.retail.sales.ts3 = arima(log.train.retail.sales.ts, order = c(2,1,1), seasonal = list(order
 = c(1,2,1), period = 12))
> test.sarima.retail.sales.ts3

Call:
arima(x = log.train.retail.sales.ts, order = c(2, 1, 1), seasonal = list(order = c(1,
    2, 1), period = 12))

Coefficients:
          ar1      ar2     ma1    sar1     sma1
      -0.9882  -0.6053  0.4092  -0.216  -0.9996
s.e.   0.0837   0.0512  0.1015   0.066   0.0515

sigma^2 estimated as 0.001673:  log likelihood = 510.48,  aic = -1008.97
```

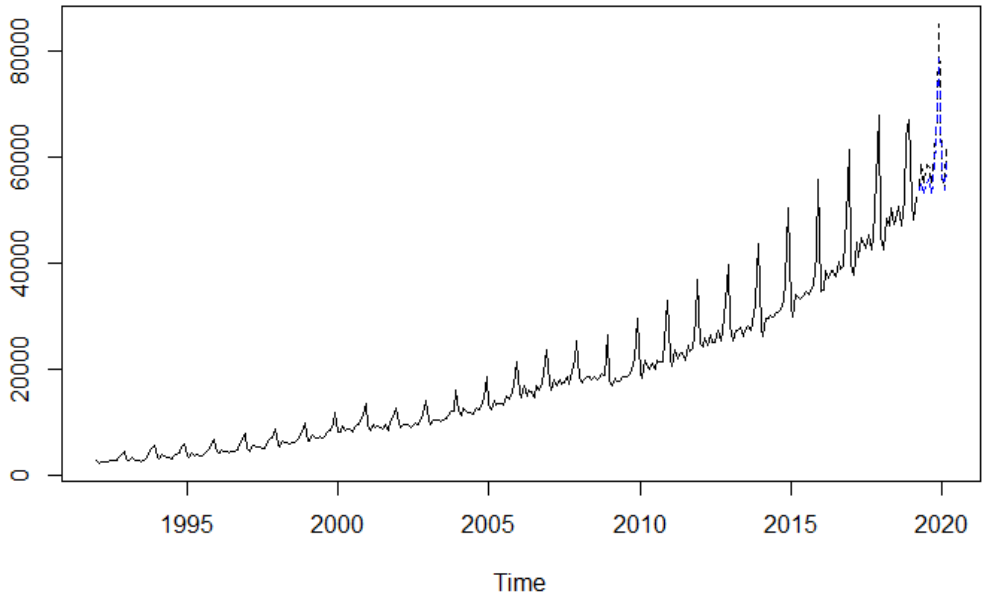
After we fit 2 models with the log of train data, we will predict the value for the period from April 1, 2019 to March 1, 2020. Moreover, we have to take an exponential function to get the prediction from the original data.

From SARIMA(2,1,1)(1,1,1)$_{12}$

```
> predict.test.sarima.retail.sales.ts2 = exp(predict(test.sarima.retail.sales.ts2, 12)$pred)
> predict.test.sarima.retail.sales.ts2
          Jan      Feb      Mar      Apr      May      Jun      Jul      Aug      Sep      Oct
2019                               53992.36 55373.11 52820.33 55268.71 56089.49 53158.77 58018.33
2020 57314.54 53748.53 59354.22
          Nov      Dec
2019 69798.14 78815.28
2020
```

A plot of actual (Black) compare to predicted value (Blue)

```
> ts.plot(cbind(window(train.retail.sales.ts, start = 1992),
+               test.retail.sales.ts,
+               predict.test.sarima.retail.sales.ts2), lty = c(1,2,2), col = c("Black","Black", "Blue"))
```

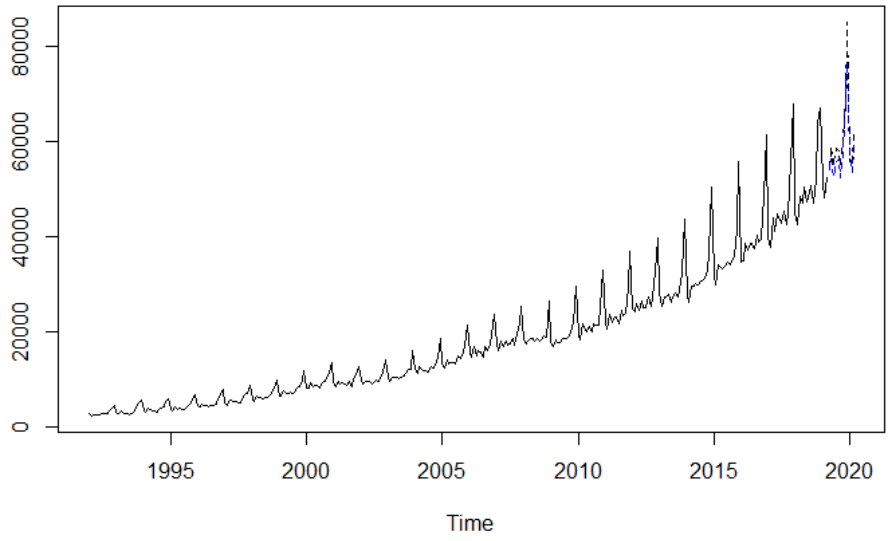


From SARIMA(2,1,1)(1,2,1)$_{12}$

```
> predict.test.sarima.retail.sales.ts3 = exp(predict(test.sarima.retail.sales.ts3, 12)$pred)
> predict.test.sarima.retail.sales.ts3
          Jan      Feb      Mar      Apr      May      Jun      Jul      Aug      Sep      Oct
2019                                 54059.73 56314.25 52805.40 55767.15 56437.74 52411.66 57729.55
2020 57289.44 53644.52 59040.91
          Nov      Dec
2019 70275.97 76753.36
2020
```

The plot of actual (Black) compare to predicted value (Blue)

```
> ts.plot(cbind(window(train.retail.sales.ts, start = 1992),
+               test.retail.sales.ts,
+               predict.test.sarima.retail.sales.ts3), lty = c(1,2,2), col = c("Black","Black", "Blue"))
```



Lastly, we find SSE for each model to compare the difference between actual value and predicted value for the last 12 months. The smaller the difference, it means the predicted value is closer to the actual value and that model will be more useful for prediction.

From SARIMA(2,1,1)(1,1,1)

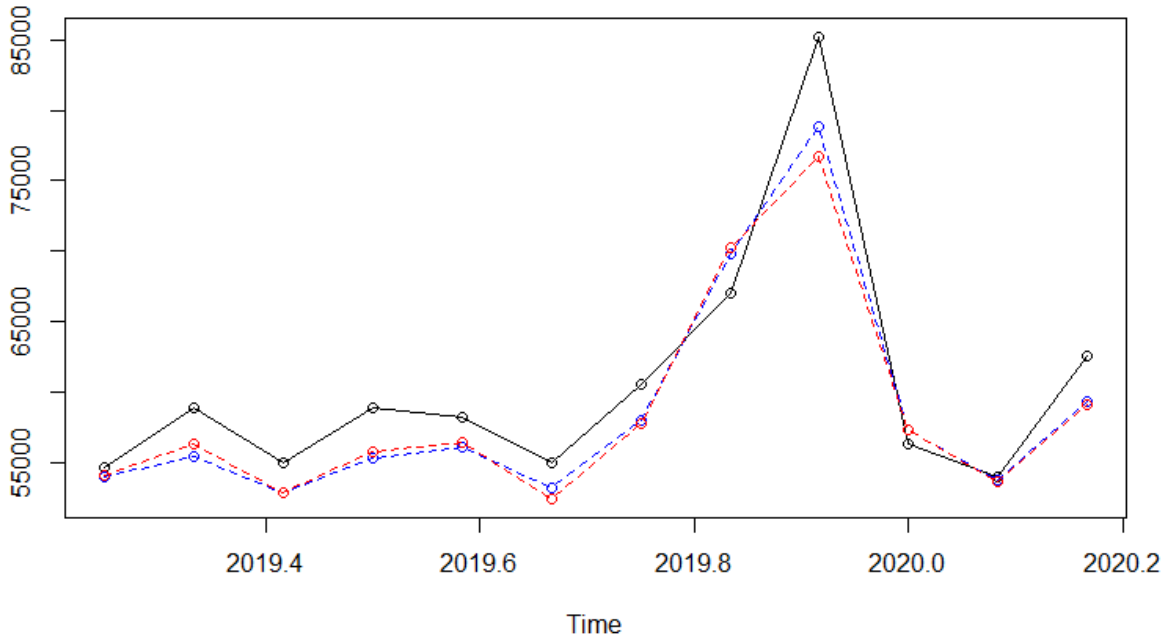
```
> sum.sq.error.test.sarima.retail.sales.ts2 = sum((predict.test.sarima.retail.sales.ts2 - test.retail.sal
es.ts)^2)
> sum.sq.error.test.sarima.retail.sales.ts2
[1] 103305400
```

From SARIMA(2,1,1)(1,2,1)$_{12}$

```
> sum.sq.error.test.sarima.retail.sales.ts3 = sum((predict.test.sarima.retail.sales.ts3 - test.retail.sal
es.ts)^2)
> sum.sq.error.test.sarima.retail.sales.ts3
[1] 133378870
```

Plot of actual (Black) compare to predicted value (Blue) of SARIMA(2,1,1)(1,1,1)$_{12}$ and predicted value (Red) of SARIMA(2,1,1)(1,2,1)$_{12}$

```
> ts.plot(cbind(test.retail.sales.ts,
+               predict.test.sarima.retail.sales.ts2,
+               predict.test.sarima.retail.sales.ts3
+               ), lty = c(1,2,2), col = c("Black", "Blue", "Red"), type = 'o')
```



From the value prediction for last 12 periods, SSE of SARIMA(2,1,1)(1,1,1)$_{12}$ which is 103305400 is less than SSE of SARIMA(2,1,1)(1,2,1)$_{12}$ which is 133378870. So, we can conclude that SARIMA(2,1,1)(1,1,1)$_{12}$ is the best model based on SSE criterion.

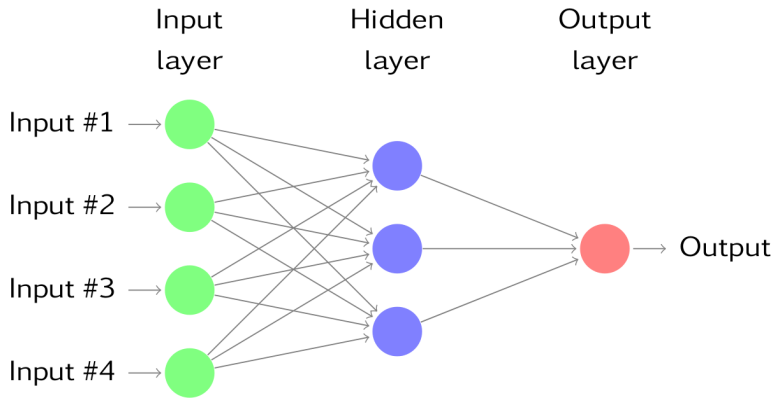
By taking the first difference for this dataset, the residual does not behave like white noise that much and it still has the remaining seasonal effect. Even if we take the second difference for a seasonal part, the residuals still behave in the same way as the first difference and there is some seasonal effect left.

In conclusion, SARIMA(2,1,1)x(1,1,1)$_{12}$ is the best ARIMA model for this dataset. However, we think that there will be other models that can reduce the seasonal effect which appropriates with this dataset more than the ARIMA model.

New Technique (Neural Network model)

Neural networks, also called Artificial neural networks, are simple mathematical models that imitate biological activity in the brain allowing complex nonlinear relationships between the response variable and its predictors. It can be used in prediction and classification.

The structure of Neural Network can be illustrated as a network of nodes (neurons) organized in 3 main layers which are input, hidden and output layers.



A picture of a neural network model with four inputs and one hidden layer with three hidden neurons.This is known as a multilayer feed-forward network.

Input and Output mechanism for each layer

the input layer, consists of nodes that simply accept predictors' values and their output is the same as the input.
The hidden layer contains nodes that each one receives all inputs from the previous layers (output values of input layers) are combined using a weighted linear combination.

For example, the inputs (#1 to #4 from input layer) into hidden neuron j are combined linearly to give

$$z_j = b_j + \sum_{i=1}^{4} w_{i,j} x_i$$

The weights b_j , also called bias of node j, and $w_{i,j}$ are randomly initialized which are usually in the range of [- 0.05, 0.05] that represent a state of no knowledge, similar to a model with no predictors and then these weights are updated (“learned”) using the observed data.

The result is then modified by a nonlinear function, also called activation or transformation function, before being output. The usual choice of activation function is such as a linear function, an exponential function and in this case a sigmoid/logistic function which is the most popular one,

$$s(z) = \frac{1}{1 + e^{-z}},$$

This is used to be the input for the next layer which in this case, the output layer. The advantage of using a sigmoid/logistic function is the ability to reduce the effect and make the model robust to extreme input values (outliers) since it has a squashing effect on very small or very large values but is almost linear in the range where the value of the function is between 0.1 and 0.9.

For an output layer, obtains input value from the hidden layer. It then applied the same function as the hidden layer i.e. weighted sum of inputs with applying a certain function, usually, for the output layer is a sigmoid function for categorical response and linear function for numerical response and then output the predicted value of the response.

As the weights are randomly initialized, there is an element of randomness in the predictions produced by a neural network. So, the neural network model is usually trained several times using different random starting values, and results of prediction are averaged.

Preprocessing the data

For neural networks to achieve best performance, a different transformation method should be properly applied to the predictors and response variable for matching with a different activation function. When using a logistic activation function, these variables should be scaled to a [0, 1] interval or for the hyperbolic tangent function, it is usually better to scale predictors to a [-1, 1] scale. However, it is acceptable to simply standardize predictors.

Training the model

For training the model i.e. estimating the weight b_j and $w_{i,j}$ that lead to the best predictive model. The process for computing neural network output is repeated for all records in the training set. For each record, the model produces a prediction which is then computed with the actual outcome value and the difference is the error,

$$e_k = (y_k - \hat{y}_k)$$

The estimation process uses the errors iteratively to update the estimated weights. The error of the output node is then distributed across all nodes in the hidden layer. Each of these is then used for updating the weights in each node.

The most popular method for using model errors to update weights is back propagation algorithm. Errors are computed from the output layer back to the hidden layers.

The error associated with output node k is computed by

$$err_k = \hat{y}_k(1-\hat{y}_k)(y_k - \hat{y}_k)$$

The weights are then updated as follow,

$$\theta_j^{new} = \theta_k^{old} + l \times err_j,$$

$$w_{i,j}^{new} = w_{i,j}^{new} + l \times err_j,$$

Where l is a learning rate, so called weight decay, a parameter which is a constant ranging typically in [0, 1], which controls the amount of change in weights from one iteration to the next and prevent the weights ($w_{i,j}$ and b_j) growing too large.
Two methods for updating the weights are case updating and batch updating.
For case updating, which the weights are updated after each record is run through the network until all records in the training set are used (This is called one epoch or iteration).

For batch updating, the entire training set is run through the network before each weights update. The error (err_k) in the updating equation is the sum of the errors from all records.
In practice, case updating tends to yield more accurate results than batch updating.

The most common condition to stop the weights updating is

- When the new weights are only incrementally different from those of the preceding iteration.
- When the limit on the number of runs is reached.

Avoiding overfitting

A weakness which often occurred easily and ignored is that the neural network model overfit the train data causing the error rate on validation data (and most critical, on new data) to be too large. Therefore, It is important to limit the number of training iterations. Overfitting can be detected by examining the performance on the validation and observing when it starts becoming progressively worse, while the training set performance is still improving. The point of minimum validation error is a good indicator of the best number of iterations for training, and the weights at that stage are likely to provide the best error rate in new data.

Required User input

We need to decide on a network architecture i.e. parameters setups. The usual procedure is to make proper guesses using past experience and to do several trial-and-error runs on different setups. As of now, no automatic method seems clearly superior to the trial-and-error approach. A few general guidelines for choosing an architecture follow,

- the number of hidden layers: The most popular choice is single (one) hidden layer and usually sufficient to capture even very complex relationships between the predictors.
- the number (size) of hidden layers: The rule of thumb is to start with p (number of predictors) nodes and gradually decrease or increase while checking for overfitting.
- the number of output nodes: for a numerical outcome, typically a single output node is used.
- the choice of predictors: Since neural networks are highly dependent on the quality of their input, the choice of predictors should be done carefully, using domain knowledge, variable selection, and dimension reduction techniques.
- learning rate (weight decay): it is used to avoid overfitting, by down-weighting new information. The suggested approach is a rule of thumb of setting ⌊ (learning rate) = 1/(current number of iteration) i.e. schedule learning rate technique.

Evaluating predictive performance

The predictive performance of neural networks for numerical response can be assessed by metrics including MAE, MAPE and RMSE based on test (validation) data. This also benefits in comparing metrics based on the validation data and based on the training data for detecting overfitting which extreme differences can be indicative of overfitting.

The predictive accuracy performance is not the same as goodness-of-fit which determines the level of fitness of a model with train data which is measured by, for example, residual analysis.

When comparing models with predictive accuracy it is common practice to split the data set into two parts, training and test data, where the training data is used to estimate any model's

parameter and test data is used to evaluate its accuracy. The test data could provide a reliable indication of performance in forecast on new data since, in a sense, it is not used in parameter estimation.

The timeline illustrates the period used training (earlier period) and test (later period) data

The size of the test set depends on the size of the dataset and desired period to forecast, usually about 20% of the total size of the dataset and is required to be at least equal to the forecasting period.

Moreover, visualization is another important tool in evaluating and comparing by examining time plots of the actual and predicted series which can help clarify the model performance and hint toward possible improvement.

The following points should be noted when selecting a model.

- A model that fits the training data well will not necessarily forecast well.
- A perfect fit can always be obtained by using a model with enough parameters.
- Overfitting a model to data is just as bad as failing to identify a systematic pattern in the data.

The benchmark performance: Naive Benchmark
Although it is desirable to implement complicated forecasting methods, we must evaluate their added value compared to a very simple approach, the naive forecast, for non-seasonal series and also for seasonal series, the seasonal naive forecast should be included.
A good model should outperform the naive model in terms of predictive performance.

Predictive accuracy measures:
The prediction error for record k is the difference between its actual outcome value and its predicted outcome,

$$e_{T+h} = y_{T+h} - \hat{y}_{T+h|T},$$

where the training data is given by $\{y_1, ..., y_T\}$ and the test data is given by $\{y_{T+1}, y_{T+2}, \ldots\}$
it means the unpredictable part of an observation. Note that prediction errors are different from residuals in two ways. First, residuals are calculated on the training set while forecast errors are calculated on the test set. Second, residuals are based on one-step forecasts while forecast errors can involve multi-step forecasts.

A popular numerical measures of predictive accuracy by summarising the forecast errors are Mean absolute error (MAE) and Root mean squared error (RMSE)

$$\text{Mean absolute error: MAE} = \text{mean}(|e_t|),$$
$$\text{Root mean squared error: RMSE} = \sqrt{\text{mean}(e_t^2)}.$$

The MAE is popular as it is easy to both understand and compute. A forecast method that minimizes the MAE will lead to forecasts of the median while minimizing the RMSE will lead to

forecasts of the mean. As a result, the RMSE is also widely used, despite being more difficult to interpret.

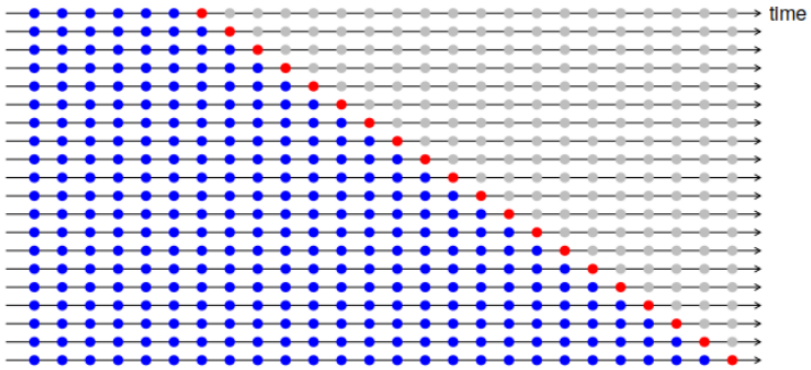
Mean absolute percentage error (MAPE)
Measures based on percentage errors have various disadvantages such as value being infinite or undefined if $y_t = 0$ for any t, and having extreme value when y_t is close to zero.

$$\text{Mean absolute percentage error: MAPE} = \text{mean}(|p_t|).$$

It is included here only because it is widely used regardless of its various disadvantages.

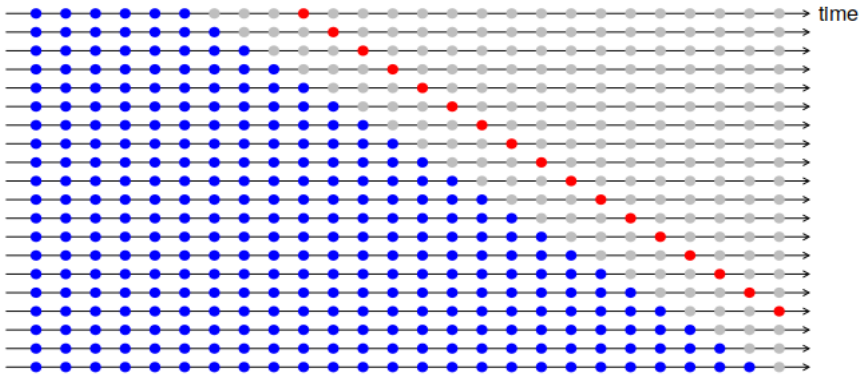
A more sophisticated version of training/test sets with predictive performance metrics is time series cross-validation. In this technique, there are a series of test sets, each consisting of a single observation. The corresponding training set consists only of observations that occurred prior to the observation that forms the test set. So, no future observations can be used in constructing the forecast. Since it is not possible to obtain a reliable forecast based on a small training set, the earliest observations are not considered as test sets.

The following diagram illustrates the series of training and test sets, where the blue observations form the training sets, and the red observations form the test sets.



The forecast accuracy is computed by averaging over the test sets. This procedure is sometimes known as “evaluation on a rolling forecasting origin” because the “origin” at which the forecast is based rolls forward in time.

With time series forecasting, one-step forecasts may not be as relevant as multi-step forecasts. In this case, the cross-validation procedure based on a rolling forecasting origin can be modified to allow multi-step errors to be used. Suppose that we are interested in models that produce good 4-step-ahead forecasts. Then, the corresponding diagram is shown below.



A good way to choose the best forecasting model is to find *the model with the smallest RMSE computed using time series cross-validation*.

Advantages and weakness of neural networks
the broadly accepted advantages of neural networks and worth mentioned are

- The most prominent advantages of neural networks is their good predictive performance
- High tolerance to noisy data and the ability to capture highly complicated relationship between the predictors and response variable

Several consideration and dangers that should be kept in mind when using neural networks

- Although they are capable of generalizing from a set of examples, extrapolation is still a serious weakness, If the network sees only records in a certain range, its prediction outside this range can be completely invalid.
- Required a careful consideration to identify key predictions
- The extreme flexibility of the neural network relies on having sufficient data for training purposes.
- the risk of obtaining weights that lead to a local optimum rather than global optimum, in the sense that the weights converge to values that do not provide the best fit to the training data.
- Neural network is relatively heavy on computation time, required a longer runtime than other methods which grows greatly when the number of predictors is increased

<u>Time series forecasting with neural network - Neural Network AutoRegressive (NNetAR) model</u>

For applying neural networks with time series data *the lagged values of the time series can be used as the inputs* (predictors) and maintain other characteristics of the neural network.

In this project, we decide to use feed-forward networks with one hidden layer with the lagged values as input implemented in a “nnetar” function of the “forecast” package because it is powerful enough to match our data in addition, it is dedicatedly developed for time series forecasting by using neural network which purposely accepts lagged values input and simple to use with allowing some model customization.
The model from nnetar function can be denoted by NNAR(p, P, k)$_m$ model which has lagged inputs y_{t-1}, y_{t-2}, …, y_{t-p} and y_{t-m}, y_{t-2m}, …, y_{t-Pm} and k neurons in the hidden layer. This is similar to using lagged values in a linear autoregression ARIMA(p,0,0)(P,0,0)$_m$ without the restrictions on the parameters to ensure stationarity. For example, an NNAR(3, 1, 2)$_{12}$ model has inputs y_{t-1}, y_{t-2}, y_{t-3} and y_{t-12} with two neurons in the hidden layer.

When it comes to forecasting, the network is applied iteratively. For forecasting one step ahead, we simply use the available historical inputs. For forecasting two steps ahead, we use the one-step forecast as an input, along with the historical data. This process proceeds until we have computed all the required forecasts.

Preparing Data

Partitioning data
We will prepare data by partitioning the dataset into 2 parts which are training and test sets which are similar to the one used in the ARIMA model but without any modification to achieve a stationary series as in ARIMA prior to feeding into the model.

Preprocessing the data
For simplicity, We decide to standardize predictors which can be done automatically by setting the parameter in function nnetar, scale.inputs = TRUE. Then, inputs are scaled by subtracting the column means and dividing by their respective standard deviations before feeding into the model (standardizing predictors). Moreover, the result of prediction, the output would be also automatically scaled back to the original units of that outcome variable.

Training (Fitting) the model

In Neural Network setting up the model parameters is one of the most important parts and has played a great role on model performance. We select two candidate models with a different parameter set up which are the model with default parameters and the model with tuned parameters since we have learned the capability of the package that it can automatically determine some important model's parameters so we decided to leave it as default value. However, we suspect that adjusting a model's parameters is still required to help the model better capture insight of the data and then better performance. Hence, we try to customize it as possible as the package would allow.

After we have considered techniques used for adjusting the Neural Network model's parameters along with package capability and limitation in customization model, we decided to focus on these parameters :

p : number of non-seasonal predictors lags used as inputs.
P : number of seasonal lags predictors used as inputs.
size : number of nodes in the hidden layer.
repeats : number of networks to fit with different random starting weights. These are then averaged when producing forecasts.
scale.inputs : If TRUE, standardizing predictors
decay : parameter for weight decay.
maxit : maximum number of iterations.

Fitted model with default parameters

We input train data and leave all parameters as default which are suggested by the packages

```
> fit = nnetar(train.retail.sales.ts)
> fit
Series: train.retail.sales.ts
Model:  NNAR(2,1,2)[12]
Call:   nnetar(y = train.retail.sales.ts)

Average of 20 networks, each of which is
a 3-2-1 network with 11 weights
options were - linear output units

sigma^2 estimated as 802699
```

fit	list [15] (S3: nnetar)	List of length 15
x	integer [327] (S3: ts)	2692 2285 2484 2506 2483 2602 ...
m	double [1]	12
p	double [1]	2
P	double [1]	1
scalex	list [2]	List of length 2
center	double [1]	18498.25
scale	double [1]	14205.43
size	double [1]	2
subset	integer [327]	1 2 3 4 5 6 ...
model	list [20] (S3: nnetarmodels)	List of length 20
nnetargs	list [0]	List of length 0
fitted	double [327] (S3: ts)	NA NA NA NA NA NA ...
residuals	double [327] (S3: ts)	NA NA NA NA NA NA ...
lags	double [3]	1 2 12
series	character [1]	'train.retail.sales.ts'
method	character [1]	'NNAR(2,1,2)[12]'
call	language	nnetar(y = train.retail.sales.ts)
[[1]]	symbol	`nnetar`
y	symbol	`train.retail.sales.ts`

These suggested parameters value are p = 2, P = 1, size = 2, repeats = 20, decay = 0 and maxit = 100

Fitted model with tuned parameters

For parameters selection, we have to justify the value of p and P which is derived from analyzing the time series based on our assumption that the neural network model can generalize patterns of data so, the effect of non-stationary and seasonal is eliminated so, we choose lagged inputs similar to ARIMA model which input is non-stationary and no seasonal.
For decay, maxit and scale.input is chosen based on trial-and-error approach and choose combination which give lowest value of RMSE on the test set

```
> nnetar.train.retail.sales.ts <- nnetar(train.retail.sales.ts, p = 2, P = 1, size = 3, decay = 1/750, ma
xit = 500, scale.inputs = TRUE)
> nnetar.train.retail.sales.ts
Series: train.retail.sales.ts
Model:  NNAR(2,1,3)[12]
Call:   nnetar(y = train.retail.sales.ts, p = 2, P = 1, size = 3, scale.inputs = TRUE,
    decay = 1/750, maxit = 500)

Average of 20 networks, each of which is
a 3-3-1 network with 16 weights
options were - linear output units  decay=0.001333333

sigma^2 estimated as 773409
```

nnetar.train.retail.sales.ts	list [15] (S3: nnetar)	List of length 15
x	integer [327] (S3: ts)	2692 2285 2484 2506 2483 2602 ...
m	double [1]	12
p	double [1]	2
P	double [1]	1
scalex	list [2]	List of length 2
center	double [1]	18498.25
scale	double [1]	14205.43
size	double [1]	3
subset	integer [327]	1 2 3 4 5 6 ...
model	list [20] (S3: nnetarmodels)	List of length 20
nnetargs	list [2]	List of length 2
decay	double [1]	0.001333333
maxit	double [1]	500
fitted	double [327] (S3: ts)	NA NA NA NA NA NA ...
residuals	double [327] (S3: ts)	NA NA NA NA NA NA ...
lags	double [3]	1 2 12
series	character [1]	'train.retail.sales.ts'
method	character [1]	'NNAR(2,1,3)[12]'
call	language	nnetar(y = train.retail.sales.ts, p = 2, P = 1, size = 3, scale.inputs = TRUE, ...
[[1]]	symbol	`nnetar`
y	symbol	`train.retail.sales.ts`

In conclusion, The selected value of each parameter is p = 2, P = 1, size = 3, repeats = 20, decay = 0.00133 and maxit = 500.

Prediction performance
Prediction for the period from April 1, 2019 to March 1, 2020 (last 12 months)

Prediction from model with default parameters

```
> fcast = forecast(fit, h=12)
> fcast
          Jan      Feb      Mar      Apr      May      Jun      Jul      Aug      Sep      Oct
2019                                    53264.40 56670.85 53539.15 55650.83 56987.29 53185.74 58467.07
2020 57881.52 54659.01 58672.80
          Nov      Dec
2019 67241.85 69095.28
2020
```

Prediction from model with tuned parameters

```
> fcast.nnetar.train.retail.sales.ts <- forecast(nnetar.train.retail.sales.ts, h=12) #predicted value
> fcast.nnetar.train.retail.sales.ts
          Jan      Feb      Mar      Apr      May      Jun      Jul      Aug      Sep      Oct
2019                                    53420.78 56851.19 53742.19 55759.06 57197.31 53383.08 58561.31
2020 58160.97 54576.23 58779.87
          Nov      Dec
2019 67115.53 68721.77
2020
```

The benchmark performance: Naive and Seasonal naive model

We compare the prediction result of these two fitted models with the prediction of naive and seasonal naive models to validate that they are better than the benchmark in terms of predictive performance.

Prediction of Naive model

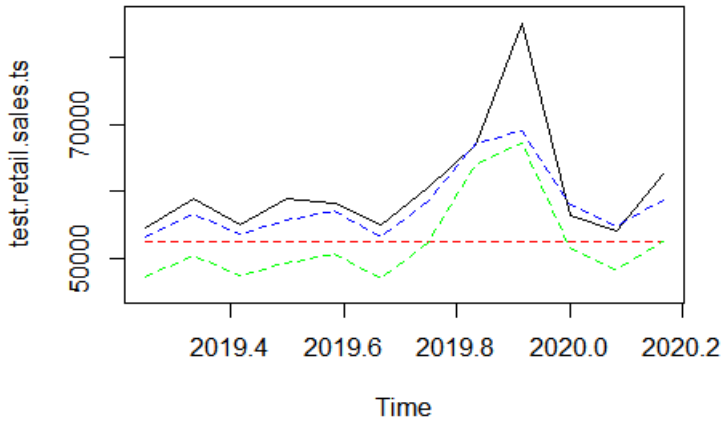
```
> naive.train.retail.sales.ts = naive(train.retail.sales.ts, h = 12)
> naive.train.retail.sales.ts
         Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
Apr 2019          52458 47271.73 57644.27 44526.28 60389.72
May 2019          52458 45123.51 59792.49 41240.86 63675.14
Jun 2019          52458 43475.11 61440.89 38719.86 66196.14
Jul 2019          52458 42085.46 62830.54 36594.57 68321.43
Aug 2019          52458 40861.15 64054.85 34722.14 70193.86
Sep 2019          52458 39754.28 65161.72 33029.34 71886.66
Oct 2019          52458 38736.42 66179.58 31472.65 73443.35
Nov 2019          52458 37789.01 67126.99 30023.72 74892.28
Dec 2019          52458 36899.19 68016.81 28662.85 76253.15
Jan 2020          52458 36057.57 68858.43 27375.71 77540.29
Feb 2020          52458 35257.08 69658.92 26151.47 78764.53
Mar 2020          52458 34492.23 70423.77 24981.73 79934.27
```

Prediction of Seasonal Naive model

```
> snaive.train.retail.sales.ts = snaive(train.retail.sales.ts, h = 12)
> snaive.train.retail.sales.ts
         Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
Apr 2019          47141 43883.49 50398.51 42159.07 52122.93
May 2019          50407 47149.49 53664.51 45425.07 55388.93
Jun 2019          47380 44122.49 50637.51 42398.07 52361.93
Jul 2019          49349 46091.49 52606.51 44367.07 54330.93
Aug 2019          50735 47477.49 53992.51 45753.07 55716.93
Sep 2019          47028 43770.49 50285.51 42046.07 52009.93
Oct 2019          52236 48978.49 55493.51 47254.07 57217.93
Nov 2019          63881 60623.49 67138.51 58899.07 68862.93
Dec 2019          67283 64025.49 70540.51 62301.07 72264.93
Jan 2020          51647 48389.49 54904.51 46665.07 56628.93
Feb 2020          48295 45037.49 51552.51 43313.07 53276.93
Mar 2020          52458 49200.49 55715.51 47476.07 57439.93
```

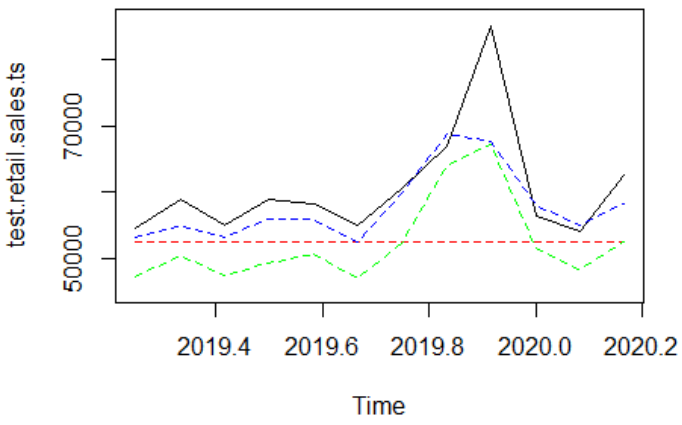
Time series plot of test (actual) data (black) compare to the prediction result of fitted model with default parameters (blue) compare to naive (green) and seasonal Naive model (red)

```
> plot(test.retail.sales.ts, ylim = c(45000, 86000)) #plot actual value
> lines(naive.train.retail.sales.ts$mean, lty = 2, col = "red") #plot naive
> lines(snaive.train.retail.sales.ts$mean, lty = 2, col = "green") #plot snaive
> lines(fcast$mean, lty = 2, col = "Blue") #plot model forecast
```



Time series plot of test data (black) compare to prediction result of fitted model with tuned parameters (blue) compare to naive (green) and seasonal naive model (red)

```
> plot(test.retail.sales.ts, ylim = c(45000, 86000)) #plot actual value
> lines(naive.train.retail.sales.ts$mean, lty = 2, col = "red") #plot naive
> lines(snaive.train.retail.sales.ts$mean, lty = 2, col = "green") #plot snaive
> lines(fcast.nnetar.train.retail.sales.ts$mean, lty = 2, col = "Blue") #plot model forecast
```



We can clearly see that both two fitted models can significantly outperform both Naive and Seasonal Naive models. However, we could not see the difference between these fitted models. Therefore, We use various measures of prediction error to compare model performance to select the best model.

Accuracy Evaluation metrics

```
> accuracy.fcast = accuracy(fcast$mean, test.retail.sales.ts)
> accuracy.fcast
                ME      RMSE       MAE       MPE      MAPE       ACF1 Theil's U
Test set 2558.1 5042.875 2960.831 3.633411 4.347281 -0.361177 0.5292724
```

```
> accuracy.fcast.nnetar.train.retail.sales.ts = accuracy(fcast.nnetar.train.retail.sales.ts$mean, test.re
tail.sales.ts)
> accuracy.fcast.nnetar.train.retail.sales.ts
                ME     RMSE      MAE      MPE     MAPE       ACF1 Theil's U
Test set 2478.643 5111.139 2893.098 3.474303 4.213861 -0.3506536 0.5345482
```

We use all 3 evaluation metrics which are RMSE, MAE, and MAPE to select the best model. From the result, we interpret based on evaluation metrics of the model with tuned parameters can slightly outperform the model with default parameters in terms of MAE and MAPE. So, we can conclude that the fitted model with tuned parameters has slightly probability of better prediction performance than the fitted model with default parameters on forecasting.

We have seen that even our NNetAR model could considerably capture the non-seasonal part of series but it could not capture the pattern of seasonal part of series which has a very high peak around November well enough. We suspect this might be the result of the networks having insufficient records of a seasonal part of the series because of the imbalanced distribution of the target variable.

However, We might be able to develop a more robust model for this dataset by considering use resampling technique for seasonal part of series or develop model with other packages that allow more flexible model customization including advance setting learning rate and optimization method of model which would be more accurate with suggested method and give more way to develop the model.

In conclusion, the prediction performance of best NNetAR$(2,1,1)_{12}$ is significantly lower than best ARIMA$(2,1,1)(1,1,1)_{12}$ model since we can clearly see from the time series plot of actual and predicted data showing ARIMA model is significantly better in capture the pattern of test data. So, we conclude that the ARIMA$(2,1,1)(1,1,1)_{12}$ model is the best fitted model for Retail sales of Electronic Shopping and Mail-order House dataset.