2022

Assignment 2 Report

MSCI570: Forecasting and Predictive Analytics

Student Id: 35493311

Table of Contents

Table of Figures

Table of tables

Executive Summary

For the dataset, we applied proven forecasting models to forecast the 2 weeks' data for the ATM transactions in the given dataset. Below are the forecasted values for them. When analyzing the data, most of the transactions are between 17 to 26. The maximum transaction that happened was 69 but there are very few numbers having such high transactions. After we applied the forecasting models to the dataset the transactions for these 2 weeks data are in the range of 17 to 26 based on the forecasting model output which is in line with the actual dataset. Also, would like to comment on the part where for some days of the week there are a high number of transactions that occurred forex. Wed, Thud, Fri, Sat. Also, during some specific months the number of transactions was higher than the usual transaction this means that more cash had been withdrawn on those specific months. The dataset has 735 observations and out of those 20 were missing observations, out of 20, 12 were from spec the of the day of the week which is Sat. While running the different forecasting models, I created some parameters which will take care of the day of the week frequency. Hence, I critically analyzed the significance related to the day of the week as well. The predicted values for the next 2 weeks signify that this many transactions would Kiley to happen in these 2 weeks. There would not be a significant difference with actual transactions. In 95 percent of cases, these transactions are likely to happen.

Introduction

The data set associated with ATM transactions happened during the period 18 Mar 1996 - 22 Mar 1998 in England. As in the last assignment, we explore the time series of the data set NN5-093 and did describe the various finding associated with the exploration. We found out the strong presence of weekly seasonality. This report aims to highlight and finalize the forecasting model applicable to the data set “NN5-093”. This process aims to apply various models starting with seasonal naive, arithmetic means, ETS, ARIMA, Neural network, and finally Multiple regression models. I will compare the model accuracy with the out sample of the data set along with the rolling origin and after comparing various models based on this result, will apply the most accurate model to the complete data set to forecast future 2 weeks data. Before we start processing concerning forecasting, we need to critically analyze the data and make it consistent throughout the dataset so that forecasting processes could be done effectively. In the next section, we address those and then we will progress with applying the forecasting processes.

Time Series Components Analysis

When we decompose this time series in the previous report in a detailed manner, we found out the strong presence of weekly seasonality. One thing would like to highlight here, when we plotted the distribution of the complete data set, we could see it is right-skewed but not normally distributed which could be a bit difficult to further explore and analyze. The reason for rightly skewed could be the large presence of outliers plus NAs data. To lower the impact of the season on the current dataset we applied various methods and analyse what impact could it do on the output if we replace it with one specific method. Below are the specifics -

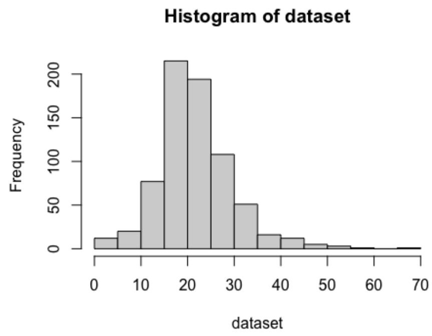

Figure 1. Histogram of the original dataset

Handling missing data

We have 20 NA present in the data set, among those seems like 12 NA entries are from specifically when there was a Sunday and 5 are when there was a Friday. There seems to be not usual on Sundays and Fridays.

I tried various methods such as na_seasplit, random, mean, median, and na_interpolation and compare the mean values of the data set after imputing in the data set with the below methods and found out the na interpolation method has the lowest mean after imputing the NAS with interpolation method and has lowest RMSE of irregular components found through decomposition function R compared with others. I believe lowest the RMSE of irregular components makes better the time series as it will reduce errors in the dataset. Hence, we will go forward with an interpolation method.

Handling Outliers

The dataset also consist consists of outliers along with missing values. Now the question which we will be discussing next hasn't been highlighted in previous findings. Now the question arises, should we impute the missing values first with an appropriate algorithm or should we handle the outliers first? If we handle the outliers first by replacing them with a median value of the data set, (as outliers' values don't impact median) we will be able to reduce some of the noise and make the series better streamlined. But in this case, we are introducing the bias element in the data set which may not turn reasonable down the line interns of forecasting. If we calculate the RMSE of the irregular components after first replacing the outliers with median value and then replacing the missing values with the interpolation method it would be around 4.12, but interestingly by visual inspection of decomposing time series we can see that it added a lot of irregular components to the time series though reduced the RMSE. If we replace the NA values first then handle the outliers with median values RSME of irregular components is 4.69 but it had smooth the outliers, we can see in decompose plot. In this latter approach, while calculating the replacement for missing values it considers outliers value which essentially makes the replacement value slightly greater but on the positive side it's not adding the bias to the time series which could be a good side of it. So though the in approach 1 where replacement of outlier happens first and then missing values RMSE is less than compared to the other approach I will still prepare to go with another approach where replacement of missing value would happen first and then outlier because it will not add bias and nor smooth the irregular components and will be in taking consideration the actual time series instead of fitted outliers.

After replacing the missing values with the interpolation method and reducing the impact of outliers by replacing them with the median of the dataset, if we plot the distribution, we can see it is now normally distributed, so we are in better shape to proceed further.

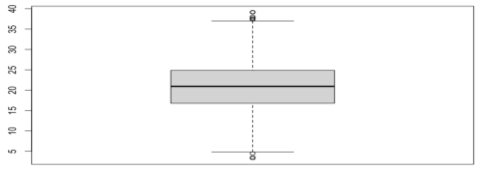

Figure 2. Box plot of the transformed dataset.

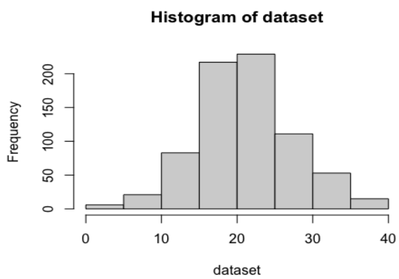

Figure 3. Histogram after replacing missing values and outliers

Preparing for train and test dataset

To test different methods and models on the time series, we split the original dataset into a train set and test set. The dataset has 735 observations. In the initial analysis we found that by taking the training dataset as 70% of the total dataset, the result of the application of forecasting methods is not consistent with the actual dataset in terms of finding the pattern in the series. Forex, if we take 70% dataset as a training dataset, we may not find multiplicative models as an appropriate model to this dataset but if we choose the training dataset as 95% of the total (which is not ideal but as per the dataset behavior), it's giving consistent result. Hence train dataset is having 721 observations and the test dataset will be 14 observations.

To correctly gauge the accuracy of applied forecasting models, I will be using the rolling origin method where the origin will not be constant while producing the forecast. By assessing the result through rolling origin, I would assume we will get better picture interns of the forecasting models' behavior.

Error measures we will be using are mean absolute percentage error and root mean square error because of their scale independence and absolute value feature for checking the performance of the forecasting model.

Various Model Implementation

Arithmetic Mean Model

We will start with the Arithmetic Mean model which uses the recent value as a forecast value for the future. Below is the forecast plot using the Arithmetic Mean Model –

There are still some residuals present in error measured against the test data set and the forecasted values. If we plot the scatter plot of the error measured, we could see it is hugely distributed and we can spot some data lying in the middle of the plot linearly early.

If we measure the various errors concerning the test data set of this model, we built on the training dataset. Below is the summary –

	ME	MSE	MAE	MAPE	RMSE
Arithmetic Mean Model Error Measures	2.939781	51.93375	5.404399	21.89850	7.206507
Arithmetic Mean Model Error Measures with Rolling origin	2.386622	65.11877	6.618608	30.88187	8.069620

Simple Moving Average

We are using here a simple moving average of order 8. This seems to be a better model than the previous arithmetic mean model. There is no impact on ACF/PACF graph compared to the arithmetic mean Residuals are greatly reduced in the SMA compared with the arithmetic mean method. Below is the summary of the error measured

	ME	MSE	MAE	MAPE	RMSE
Arithmetic Mean Model Error Measures	2.9397806	51.93375	5.404399	21.89850	7.206507
Arithmetic Mean Model Error Measures with Rolling origin	2.3866223	65.11877	6.618608	30.88187	8.069620
Simple Moving Average Model Error Measures	0.7713281	43.88639	5.089309	22.65706	6.624680
Simple Moving Average Model Error Measures with Rolling origin	-2.0499725	74.41465	7.180229	39.88180	8.626393

Naive method

To start with benchmarking, the naïve method could be a good fit. Below is the result of the naïve model if we compare it with the test dataset, with a rolling origin.

	ME	MSE	MAE	MAPE	RMSE
Naive Model Error Measures	8.0446382	108.0076	8.574435	32.07602	10.39267
Naive Model Error Measures with Rolling origin	-0.5113447	126.9699	9.327126	47.99214	11.26809

Seasonal Naive method

The naive method would be a good benchmark to start with. There is no impact on ACF/PACF graph compared to the previous 2 methods Below is the summary of the error measured.

	ME	MSE	MAE	MAPE	RMSE
Naive Model Error Measures	8.0446382	108.00764	8.574435	32.07602	10.392672
Naive Model Error Measures with Rolling origin	-0.5113447	126.96988	9.327126	47.99214	11.268091
Seasonal Naive Model Error Measures	1.7781994	72.80482	5.926392	24.16777	8.532574
Seasonal Naive Model Error Measures with Rolling origin	-1.9789531	135.99723	9.019689	49.60457	11.661785

Exponential Smoothing

Exponential Smoothing Additive Error, No Trend and Additive Seasonality Model (A, N, A)

Based on the initial analysis we conclude that this dataset has no trend pattern. Based on this judgment below discussed model below would be a good start to implement the exponential smoothing. There is a definitive seasonality component presence present in the dataset so we would go with additive seasonality. Hence, the additive Error, No Trend, and Additive Seasonality Model (A, N, A) model could be a good benchmarking model to start the forecasting.

	ME	MSE	MAE	MAPE	RMSE
Naive Model Error Measures	8.0446382	108.00764	8.574435	32.07602	10.392672
Naive Model Error Measures with Rolling origin	-0.5113447	126.96988	9.327126	47.99214	11.268091
Seasonal Naive Model Error Measures	1.7781994	72.80482	5.926392	24.16777	8.532574
Seasonal Naive Model Error Measures with Rolling origin	-1.9789531	135.99723	9.019689	49.60457	11.661785
ETS ANA Model Error Measures	2.4526088	33.25859	4.648521	19.28735	5.767026
ETS ANA Model Error Measures with Rolling origin	-1.6127738	67.17263	6.556565	37.33562	8.195891

Exponential Smoothing Additive Error, Additive Trend and Additive Seasonality Model (A, A, A)

Though the dataset has a very insignificant trend pattern it could be worth applying the trend factor to consider the fact their trend pattern if consider the dataset has multi seasonality in it. This means that if this dataset consists of multi-seasonality, i.e. yearly and weekly, trend patterns should be included to accommodate this. Hence we will apply the below model with Additive Error, Additive Trend, and Additive Seasonality Model.

	ME	MSE	MAE	MAPE	RMSE
Naive Model Error Measures	8.0446382	108.00764	8.574435	32.07602	10.392672
Naive Model Error Measures with Rolling origin	-0.5113447	126.96988	9.327126	47.99214	11.268091
Seasonal Naive Model Error Measures	1.7781994	72.80482	5.926392	24.16777	8.532574
Seasonal Naive Model Error Measures with Rolling origin	-1.9789531	135.99723	9.019689	49.60457	11.661785
ETS ANA Model Error Measures	2.4526088	33.25859	4.648521	19.28735	5.767026
ETS ANA Model Error Measures with Rolling origin	-1.6127738	67.17263	6.556565	37.33562	8.195891
ETS AAA Model Error Measures	2.3749237	32.88965	4.611196	19.17630	5.734950
ETS AAA Model Error Measures with Rolling origin	-1.8099943	68.70708	6.599205	37.81265	8.288974

Exponential Smoothing Multiplicative Error, Additive Trend and Multiplicative Seasonality Model (M, A, M)

If we see the ATM transactions are in increasing order in the dataset in terms of their value, we can spot this with the help of decomposing graph as well. This suggests such we may need to consider the multiplicative models as well. As there is strong weekly seasonality, multiplicative seasonality with error could the possible choice of the model here. Also, I presume that, as there is a strong presence of seasonality and its evolving if you see 1997 and 1998 observations, multiplicative models with error and seasonality would be a good fit here, and to accommodate this additive trend could be the possible choice of trend.

	ME	MSE	MAE	MAPE	RMSE
Naive Model Error Measures	8.0446382	108.00764	8.574435	32.07602	10.392672
Naive Model Error Measures with Rolling origin	-0.5113447	126.96988	9.327126	47.99214	11.268091
Seasonal Naive Model Error Measures	1.7781994	72.80482	5.926392	24.16777	8.532574
Seasonal Naive Model Error Measures with Rolling origin	-1.9789531	135.99723	9.019689	49.60457	11.661785
ETS ANA Model Error Measures	2.4526088	33.25859	4.648521	19.28735	5.767026
ETS ANA Model Error Measures with Rolling origin	-1.6127738	67.17263	6.556565	37.33562	8.195891
ETS AAA Model Error Measures	2.3749237	32.88965	4.611196	19.17630	5.734950
ETS AAA Model Error Measures with Rolling origin	-1.8099943	68.70708	6.599205	37.81265	8.288974
ETS MAM Model Error Measures	0.8830109	28.06036	4.275528	19.03858	5.297203
ETS MAM Model Error Measures with Rolling origin	-2.4611235	74.36137	6.757129	39.84965	8.623304

Exponential Smoothing Multiplicative Error, No Trend and Additive Seasonality Model (M, N, A)

As mentioned in the above comment, multiplicative could be the potential choice for error and as the dataset does not show the strong presence of ta rend, we have considered no trend and additive seasonality in this model.

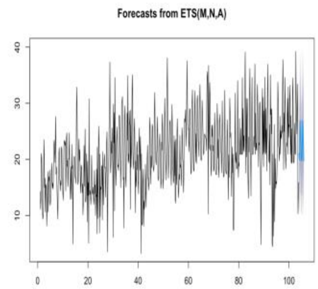

Figure 4. Forecast of ETS (M, N, A) Model

	ME	MSE	MAE	MAPE	RMSE
Naive Model Error Measures	8.0446382	108.00764	8.574435	32.07602	10.392672
Naive Model Error Measures with Rolling origin	-0.5113447	126.96988	9.327126	47.99214	11.268091
Seasonal Naive Model Error Measures	1.7781994	72.80482	5.926392	24.16777	8.532574
Seasonal Naive Model Error Measures with Rolling origin	-1.9789531	135.99723	9.019689	49.60457	11.661785
ETS ANA Model Error Measures	2.4526088	33.25859	4.648521	19.28735	5.767026
ETS ANA Model Error Measures with Rolling origin	-1.6127738	67.17263	6.556565	37.33562	8.195891
ETS AAA Model Error Measures	2.3749237	32.88965	4.611196	19.17630	5.734950
ETS AAA Model Error Measures with Rolling origin	-1.8099943	68.70708	6.599205	37.81265	8.288974
ETS MAM Model Error Measures	0.8830109	28.06036	4.275528	19.03858	5.297203
ETS MAM Model Error Measures with Rolling origin	-2.4611235	74.36137	6.757129	39.84965	8.623304
ETS MNA Model Error Measures	1.2515850	29.35334	4.371057	19.18990	5.417873
ETS MNA Model Error Measures with Rolling origin	-1.7120639	64.58501	6.470804	37.12767	8.036480

Optimized method using ETS Function

If we run the dataset through the ETS ZZZ model, the ETS function suggested the ETS Multiplicative Error, No Trend, and Multiplicative Seasonality (M, N, M) model as the right candidate for the forecasting. I could not argue against this as the best candidate because this model has the lowest AIC, BIC, mean absolute percentage error with the static origin, Root Mean Square Error with the static origin, Mean absolute percentage error with the rolling origin, and Root Mean Square Error with rolling origin. Please refer to the table

	ME	MSE	MAE	MAPE	RMSE
Naive Model Error Measures	8.0446382	108.00764	8.574435	32.07602	10.392672
Naive Model Error Measures with Rolling origin	-0.5113447	126.96988	9.327126	47.99214	11.268091
Seasonal Naive Model Error Measures	1.7781994	72.80482	5.926392	24.16777	8.532574
Seasonal Naive Model Error Measures with Rolling origin	-1.9789531	135.99723	9.019689	49.60457	11.661785
ETS ANA Model Error Measures	2.4526088	33.25859	4.648521	19.28735	5.767026
ETS ANA Model Error Measures with Rolling origin	-1.6127738	67.17263	6.556565	37.33562	8.195891
ETS AAA Model Error Measures	2.3749237	32.88965	4.611196	19.17630	5.734950
ETS AAA Model Error Measures with Rolling origin	-1.8099943	68.70708	6.599205	37.81265	8.288974
ETS MAM Model Error Measures	0.8830109	28.06036	4.275528	19.03858	5.297203
ETS MAM Model Error Measures with Rolling origin	-2.4611235	74.36137	6.757129	39.84965	8.623304
ETS MNA Model Error Measures	1.2515850	29.35334	4.371057	19.18990	5.417873
ETS MNA Model Error Measures with Rolling origin	-1.7120639	64.58501	6.470804	37.12767	8.036480
ETS Optimised Model Error Measures	1.2939608	30.15806	4.463584	19.64800	5.491636
ETS Optimised Model Error Measures with Rolling origin	-2.0555994	72.45337	6.769067	39.37497	8.511954

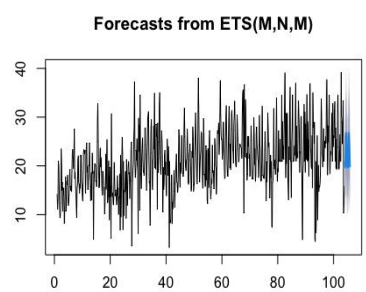

Figure 5. Forecast of ETS (M, N, M) Model

Table 1. Summary table of various ETS models

Model	AIC	AICc	BIC	P-Value	Lag	MAPE	RMSE	Rolling origin MAPE	Rolling origin RMSE
ETS Optimsed	7021.421	7021.73	7067.227	4.75E-05	2 lags outside	19.648	5.491636	39.37497	8.511954
ETS ANA	7051.535	7051.845	7097.342	7.48E-06	2 lags outside	19.28735	5.767026	37.33562	8.195891
ETS AAA	7056.605	7057.046	7111.573	5.94E-07	3 lags outside	19.1763	5.73495	37.81265	8.288974
ETS(M,A,M)	7024.776	7025.216	7079.743	1.54E-05	2 lags outside	19.03858	5.297203	39.84965	8.623304
ETS(M,N,A)	7029.001	7029.311	7074.807	1.01E-05	2 lags outside	19.1899	5.417873	37.12767	8.03648

Recommended Model using the exponential smoothing

As per the below summary table, **ETS Optimized model (Multiplicative Error, No Trend and Multiplicative Seasonality (M, N, M) model)** has the lowest AIC value 7021.421 and the lowest BIC value 7067.227 if compare with other models. Also, if we compare the Mean absolute percentage error with the static origin, Root Mean Square Error with the tactic origin, mean absolute percentage error with the rolling origin, and Root Mean Square Error with rolling origin values with other models, these values are very similar to other models. Based on this result the optimized model which is suggested by ETS function "R" is a good fit for forecasting this dataset. The next best candidate for the model would be ETS (Multiplicative Error, No Trend and Additive Seasonality (M, N, A) model) since the AIC value is 2nd lowest 7029.001 with BIC as 7074.807. Also, if compare the error measures MAPE is 2nd lowest along with RMSE. This model has done better in rolling origin error measures by being the lowest in the table values. Not only these values but as this dataset has strong seasonality with no trend, with possible multi seasonality factor, it makes sense to use this model.

ARIMA/SARIMA models

For ARIMA models to be fitted, we need to make sure the time series is stationary. Our time series is not stationary as it contains weekly seasonality. But when we did KPSS/ADF test, test results were inconclusive. But as we know it contains seasonality hence, we can reply on visual inspection that the time series is not stationary. To make it stationary, we will try to first-order differencing, and below is the plot of residuals-

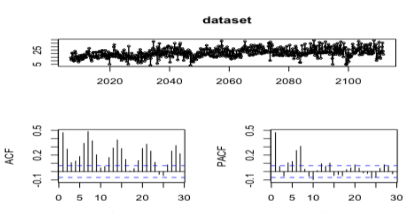

Figure 6. ACF/PACF Plot of the original dataset

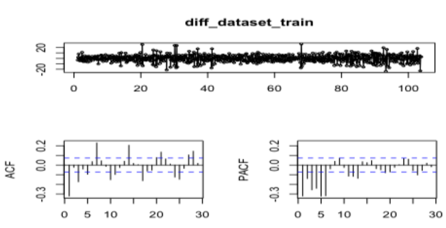

Figure 7. ACF/PACF Plot of 1st order differenced dataset

We can see after first order differencing; the time series does not become stationary. but when I did the KPSS/ADF test on 1st order difference time series, seems like results are saying it has become stationary.

Now as after the 1st differencing it seems like the data has become stationary, we will proceed with the identification of the ARIMA model. To do that, we will analyze the ACF and PACF plot as below and will establish an appropriate model.

As the PACF cuts off after lag 7 that means Autoregressive (AR) and ACF does not die away hence ARIMA Model – Non-Stationary would be a good suit also that means we need differencing.

In general, PACF cuts off after the value p means AR value, in our case it's lag 7 and in general, ACF cuts off after the value q that means MA value, in our case, it’s not cutting off but exponentially decaying. Seasonal AR models are preferred in this case.

Below are models I think could be a potential choice for a good fit in forecasting based on the above discussion. While finding the potential models I consider the Ljung- Box test to evaluate whether the residual lags have any co-correlation between them. Also, I checked whether the residuals are under the significance level of 5 % in the ACF/PACF plot which essentially means that the residuals have no or insignificant correlation between them and there is an insignificant presence of any pattern.

If we compare the AIC, BIC values along with the error measures against the test dataset, the ARIMA model is the recommended model among stall this.

If we see the residuals for the ARIMA(7,1,1)(0,1,3)[7], ARIMA(2,0,2)(0,1,2)[7], ARIMA(2,0,2)(1,1,1)[7] and ARIMA(1,1,3)(2,1,2)[7] lags can be seen outside the significance level which means lags could be related to the previous ones but this 4 model does have only one lag outside the significance level which can be ignored to some extent if everything falls under. The model ARIMA (7,1,1) (0,1,3) [7] has an AR value of 7 which is significantly large and could not be the right choice to proceed as the parameters are unnecessarily increased. Hence, we will not select ARIMA (7,1,1) (0,1,3) [7]. ARIMA (1,1,3) (2,1,2) [7] for this model the AIC, BIC values are greater if we compare with others with residual lag outside the significance level hence, we will not proceed with this model. O output of the remaining 4 models, ARIMA(3,0,2)(1,1,2)[7] and ARIMA(4,0,2)(3,1,1)[7] model does not have any residual lag outside the significance level which is a good sign to proceed but the model ARIMA(4,0,2)(3,1,1)[7] has greater AIC, BIC, MAPE, MAPE with the rolling origin, RMSE, RMSE with rolling origin than other 3 hence we will not proceed with this. Out of the remaining 3, ARIMA (3,0,2) (1,1,2) [7] has the lowest AIC along with MAPE and has no residuals lags outside the significance level hence this would be a good candidate forecasting.

Below is the summary matrix for both the Naïve and Seasonal models and the various ARIMA models taken into consideration.

	ME	MSE	MAE	MAPE	RMSE
Naive Model Error Measures	8.0446382	108.00764	8.574435	32.07602	10.392672
Naive Model Error Measures with Rolling origin	-0.5113447	126.96988	9.327126	47.99214	11.268091
Seasonal Naive Model Error Measures	1.7781994	72.80482	5.926392	24.16777	8.532574
Seasonal Naive Model Error Measures with Rolling origin	-1.9789531	135.99723	9.019689	49.60457	11.661785

Table 2. Summary table of various ARIMA models

Model	AIC Value	AICc Value	BIC Value	Ljung-Box test (P-Value)	Lag	MAPE	RMSE	Rolling origin MAPE	Rolling origin RMSE
ARIMA(7,1,1)(0,1,3)[7]	4300.07	4300.52	4354.91	0.4424	1 lag outside	17.07091	4.859094	36.80672	8.051207
ARIMA(2,0,2)(0,1,2)[7]	4299.36	4299.52	4331.35	0.1621	1 lag outside	16.91682	5.143028	34.75201	7.775084
ARIMA(2,0,2)(1,1,1)[7]	4298.92	4299.07	4330.91	0.1707	1 lag outside	16.93098	5.166875	34.96024	7.836527
ARIMA(1,1,3)(2,1,2)[7]	4309.83	4310.08	4350.95	0.2775	1 lag outside	17.42067	5.007937	37.76944	8.233554
ARIMA(3,0,2)(1,1,2)[7]	4297.04	4297.29	4338.17	0.1344	No lag outside	16.80678	5.223571	35.29591	7.863515
ARIMA(4,0,2)(3,1,1)[7]	4299.9	4300.27	4350.18	0.07902	No lag outside	17.01671	5.177261	35.5316	7.914115
Auto ARIMA	4398.57	4398.73	4430.63	0.0003046	multiple lags outside	21.9497	6.37433	38.71362	8.608693

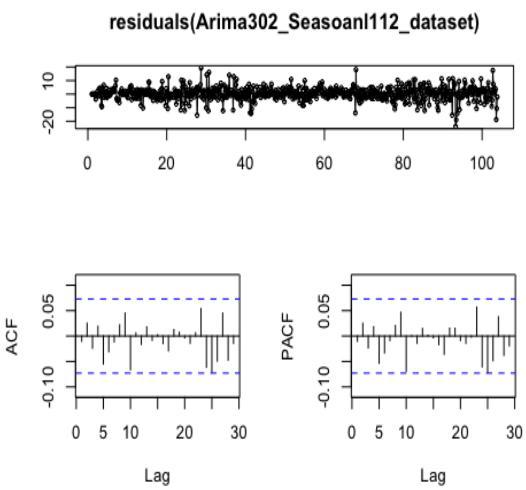

Figure 8. Residuals of ARIMA (3,0,2) (1,1,2) Model

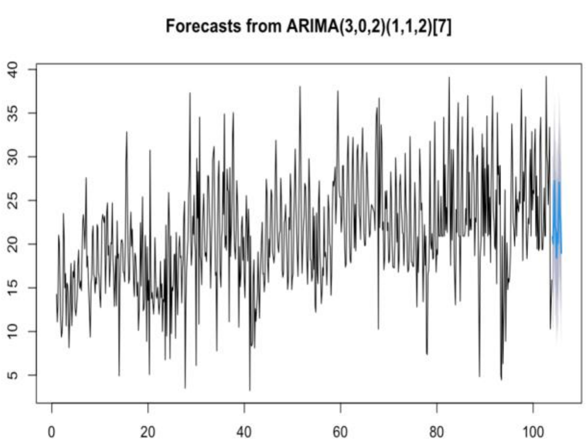

Figure 9.Forecast of ARIMA (3,0,2) (1,1,2) Model

Optimized ARIMA Model

If we run the dataset on auto-arima function, it has recommended ARIMA (2,1,2) (0,0,2)[7] model. if we analyze the model, it has an AIC value of 4398.57, BIC value of 4430.63, and fails the Ljung-Box test of autocorrelations of a time series which means that residuals could be related to previous lags. If we are carefully the residuals plot, then you would see there are numerous lags outside the significance level which is not a sign of stationary. Also, MAPE and RMSE are comparatively higher than the previously discussed model so we will not be proceeding further with the automatically built model.

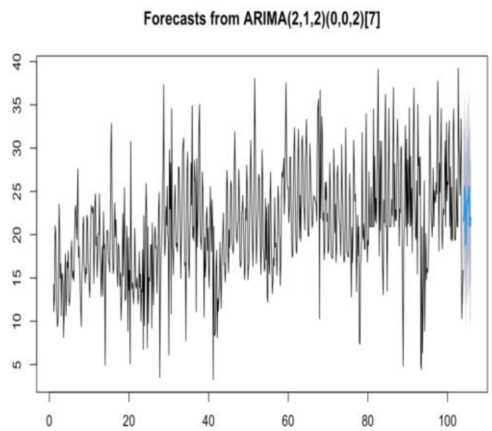

Figure 10. Forecast of auto arima model

Regression

In the regression model, I considered the below model variables to build the regression model to use for forecasting.

1- Lag values of the dataset

2- Seasonal dummies

3- Trend variable

Simple regression –

In simple regression, I considered the dataset values against the own values as a variable and the estimated value came as 20.9912 and std error as 0.2351.

Multiple Regression model –

In multiple regression model mainly below models could be created-

1: Multiple regression model with only lag variables

2: Multiple regression model with only seasonal dummies variables

3: Multiple regression model with only seasonal trend variable

4: Multiple regression model with lag and seasonal dummies variables

5: Multiple regression model with lag, seasonal dummies, and trend variables

Starting with 1st model where considered only lag variable, I added all the lag variables and the

Adjusted R-squared: 0.3661 was the value with for some of the variable's p-value was greater than 0.05. These many variables potentially add the multi-collinearity and may be useful and hence removing these and checking the adjusted R squared values against them, If we add the L7 lag value, the value of the adjusted r-square is increasingly significant along with L1 and L2 lags dataset. If check the VIF factor it's coming under 5 which means good to go.

For the 2nd model if considered all the seasonal dummies the Adjusted R-squared: 0.1631 only and potential multicollinearity effect and same with the only trend as it was not a significant variable.

Hence for the 4th type of model seasonal dummies, now we will check how can we improve the model by adding the seasonal dummies. For now, if we keep all the seasonal dummies we created as Mon to Sat in the regression model just created by using the lags dataset, the adjusted R squared values have increased to 0.4132. But this could add the multi-collinearity effect as well. If remove the variables which could be having similar significance, it may become a better model. Hence by doing this, we remove the L3, L5, and L6 lag values from the model. Now the new model consists of L1, L2, L4, L7 lag values along with 6 seasonal dummies of the day of week/. When checking the p-value for all the variables, the p-value for Mon and Tue variable is higher than the 5% significance level and hence we will remove the Mon and Tue variable. This model is giving an adjusted R squared value of 0.4157.

Hence now considered the 5th model which potentially increases the adjusted R squared, If you add the trend variable into the model, it will increase further the adjusted R squared value to 0.4262 and the p-values for all the variables are falling under the 5 % significance level. I checked the VIF factor, all are coming under the 6 which means we are not having a multicollinearity effect. The model which could be the good fit is **(Transactions ~ L1_dataset + L2_dataset + L4_dataset + L7_dataset + Wed + Thu + Fri + Sat + dataset trend).** This has an AIC of 4271.513 and a BIC of 4321.792. Also, the adjusted R squared value is 0.4262. Automatically built model by the "lm" function in "R" is giving the model as (**Transactions ~ L1_dataset + L2_dataset + L4_dataset + L7_dataset + L12_dataset + L14_dataset + Wed + Thu + Fri + Sat)** when we run the algorithm in both directions, i.e. Forward and Backward. This model has an AIC of value as 4241.827 and a BIC value of 4296.559 which is quite lower than the identified model as above. But the Adjusted R-squared: 0.4146 has a lower value than the above model which could be a downside to this model.

Hence the good-fit candidate for the regression model is **(Transactions ~ L1_dataset + L2_dataset + L4_dataset + L7_dataset + Wed + Thu + Fri + Sat)** where Transactions is the dataset time series and L1, L2, L4, and L7 are the lag variable of the dataset along with Wed, Thu, Fri and Sat are the dummy variables for the day of the week.

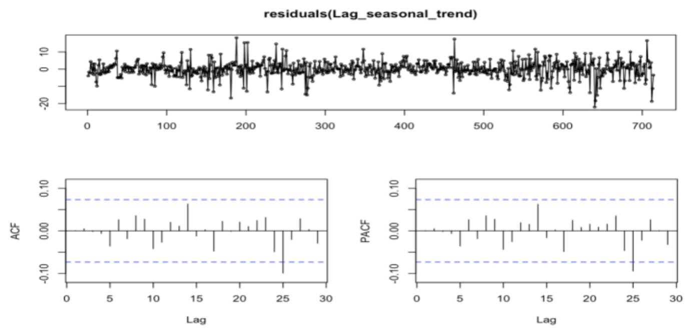

Figure 11. Residuals of Lag Seasonal Trend Regression Model

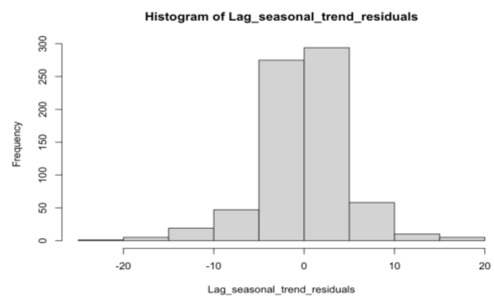

Figure 12.Histogram of residuals of Lag Seasonal Trend Model

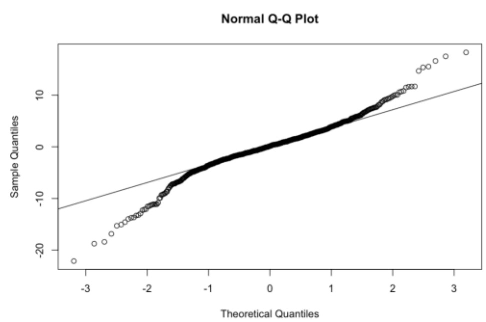

Figure 13. Normal Q-Q plot of lag seasonal trend regression model

Model selection

For the dataset, we critically analyzed the different models as follows: Arithmetic Mean, Simple Moving Average, Naive, Seasonal Naive, ETS ANA, ETS AAA, ETS MAM, ETS MNA, ETS Optimized, ARIMA (7,1,1) (0,1,3)[7], ARIMA(2,0,2)(0,1,2)[7],ARIMA(2,0,2)(1,1,1)[7], ARIMA (1,1,3) (2,1,2) [7], ARIMA(3,0,2) (1,1,2) [7], ARIMA (4,0,2) (3,1,1) [7], Auto ARIMA, Simple Regression, Multiple regression with lag variables, Multiple regression with seasonal dummies variables, Multiple regression with lag and seasonal dummies variables, Multiple regression with lag, seasonal dummies and trend variables. Based on the output measured against the test dataset, the error matrix final recommended models are: **ARIMA (3,0,2) (1,1,2) [7] and ETS Multiplicative Error, No Trend, and Multiplicative Seasonality.** The reason for selection is because when the checked the error measures with the test dataset, it seems that this model gives a better result if compare with other models. The MAPE and RMSE have the lowest value amongst all other models also when checked with rolling origin the MAPE and RMSE outperform other models.

Model	MAPE	RMSE
ARIMA(3,0,2)(1,1,2)[7]	16.80678	5.223571
Auto ARIMA	21.9497	6.37433
ETS Optimsed	19.648	5.491636
ETS(M,N,A)	19.1899	5.417873
Lag seasonal trend	20.39466	5.038351
Automatically selected regression model	19.88181	5.236775

Conclusion

The dataset has 735 observations in it, some of the observations were missed and some were considered as outliers when we checked. To apply forecasting models, I had to impute the missing values and outliers' replacement was done with median values as there was less impact on the median. I split the dataset into the train and test dataset as this will allow us to build the model based on the training dataset and we can test the performance on the test dataset. I applied various models to name a few, ETS various models, ARIMA models, Regression models and performance of those were evaluated against the factors like AIC/BIC values if comparing the forecasting models of the same family, error measures with the respect to test dataset if comparing the forecasting model with a different family. Out of various errors, mean percentage absolute error and root mean squared error was selected for evaluating the performance of these forecasting models because of the scale-independent and absolute value production feature.

After critically analyzing the various models, the best candidates to produce the forecast for the given dataset are the ETS Multiplicative Error, No Trend and Multiplicative Seasonality, and ARIMA (3,0,2) (1,1,2) [7]. The reasons for selection are various but one main reason is the performance of this model on the test dataset of this model. When we test these models on the test dataset, the errors produced were the lowest amongst all. Based on this forecasting, I produced the forecast for the next 14 values i.e., 2 weeks of data for the ATM transactions. *(Please refer to appendix 1)*

References

1: https://www.google.com/search?q=Ljung-Box+test&rlz=1C5CHFA_enGB984GB984&oq=Ljung-Box+test&aqs=chrome..69i57j35i39i362l5j46i39i362j35i39i362l2.396j0j7&sourceid=chrome&ie=UTF-8

2: https://www.statisticshowto.com/variance-inflation-factor/

Appendix

1: The forecast values for the next 14 days:

Day	Seasonal naïve	ETS Optimised (M,N,M)	ARIMA(3,0,2)(1,1,2)[7]	Lag Seasonal Trend
1	23.18517	22.84422	22.94607	20.92464
2	23.64545	21.54037	22.1582	20.78181
3	21.02841	24.4178	24.02311	24.4036
4	29.87901	30.10594	29.06416	30.0352
5	35.34982	32.88944	27.01717	27.36673
6	23.40873	23.60674	23.82758	22.11571
7	18.99001	19.27782	19.74083	18.62717
8	23.18517	22.84422	23.01988	20.09892
9	23.64545	21.54037	21.60296	19.91072
10	21.02841	24.4178	24.52623	25.27453
11	29.87901	30.10594	28.75775	28.93069
12	35.34982	32.88944	25.83235	26.69369
13	23.40873	23.60674	23.48966	25.70733
14	18.99001	19.27782	19.79805	21.22148

R Script:

Assignment 2
Student Id 35493311

```
# Library and dataset import ---------------------------------------------

#Importing libraries
library("forecast")
library("tsutils")
library("imputES")
library("tseries")
library("readxl")
library("xts")
```

```
library("seastests")
library("tinytex")
library("tsibble")
library("dplyr")
library("outliers")
library("moments")
library("VIM")
library("naniar")
library("ggplot2")
library("imputeTS")
library("knitr")
library("regclass")



#Import sdata
dataset <- read_excel("Assignment 1 Data.xls")

#colnames(data) <- c("Date","Transactions")
colnames(dataset) <- c("Transactions")

#Converting data to time series
dataset <- ts(dataset, frequency = 7, start = c(1996,77))

# Histogram of whole distribution before cleaning the data set
hist(dataset)

# Missing values and outliers distribution -------------------------------------------
```

```
# Missing values distribution
ggplot_na_distribution(dataset)
ggplot_na_intervals(dataset)
ggplot_na_gapsize(dataset)

#Filing missing values

# Approach 1 -Replace with corresponding day of week data
which_na(dataset)

# Approach 2 - Replace with interpolation method
dataset <- na_interpolation(dataset)

summary(dataset)
plot(dataset)
which_na(dataset)

#daily <- as.xts(unlist(list(dataset1)),order.by=as.Date(data$Date))
#plot(daily, type = "l", col = "BLACK")

# Box plot the outliers
boxplot(dataset)
# Check how many are the outliers
boxplot.stats(dataset)

# Statistical test for outliers
grubbs.test(dataset,type = 11, opposite = FALSE, two.sided = TRUE)
chisq.out.test(dataset)
```

```
# Find the outliers in the series

out <- boxplot.stats(dataset)$out

out

out_ind <- which(dataset %in% c(out))

out_ind

dataset[c(163,167,196,235,265,283,538,557,558,559,561,562,563,565,586,587,592,593,594,600

        ,621,628,649,670,684,691,698,711,712,726)]


#Replacing the outliers with median values

series_median = median(dataset)

series_median

dataset[c(163,167,196,235,265,283,538,557,558,559,561,562,563,565,586,587,592,593,594,600

        ,621,628,649,670,684,691,698,711,712,726)] = series_median



# Summary of dataset

summary(dataset)


# Box plot the outliers after replacing previously identified outliers with series median

boxplot(dataset)

# Check how many are the outliers after replacing previously identified outliers with series median

boxplot.stats(dataset)


# Length of data set

length(dataset)
```

```
# Frequency of dataset
frequency(dataset)

# Visualize the time series of data sES
plot(dataset)

# Histogram of whole distribution
hist(dataset)

#Test for trend
trendtest(dataset)

#Checking for seasonality
isSeasonal(dataset, freq = 365)
isSeasonal(dataset, freq= 7)
isSeasonal(dataset, test = "combined")

seastests::welch(dataset)

#Decomposition of the dataset 3 using decomp function
decomp_dataset <- decomp(dataset,outplot = TRUE)

# Plot the season
seasplot(dataset,m = 7)

#ACF and PACF analysis on the entire dataset
tsdisplay(dataset)
```

```
# Perform KPSS and ADF test on time series
kpss.test(dataset)
adf.test(dataset)

# Split into training and test ------------------------------------------

# Find the total number of observations
dataset_length <- length(dataset)
# Write down size of training set
dataset_train_length <- 721

# Split into training and test
dataset_train_length<- 721
dataset_train <- ts(dataset[1:dataset_train_length], frequency = 7)
length(dataset_train)
dataset_test <- dataset[(dataset_train_length+1):length(dataset)]
length(dataset_test)

#Setting the horizon
h=14

# Rolling origin -------------------------------------------------------

# Set horizon and number of rolling origins
H <- 14 #42 # 14 # 32
origins <- 14 #14
dataset_length <- length(dataset)
```

```
dataset_rolling_train_length <- dataset_length - H - origins + 1

dataset_rolling_test_length <- H + origins - 1

dataset_rolling_train <- ts(dataset[1:dataset_rolling_train_length],
                      frequency=frequency(dataset),
                      start=start(dataset))

dataset_rolling_test <- dataset[(dataset_rolling_train_length+1):dataset_length]



# Arithmetic Mean model --------------------------------------------------


#Arithmetic mean model

Arithmetic_mean_dataset<- mean(dataset_train)

Forecast_Arithmetic_mean_dataset<- forecast(Arithmetic_mean_dataset,h=h)$mean

Forecast_Arithmetic_mean_dataset

#Error Measures for Arithmetic mean model

Arithmetic_errors_dataset<- dataset_test - Forecast_Arithmetic_mean_dataset

Arithmetic_ME_dataset<- mean(Arithmetic_errors_dataset) #Mean error

Arithmetic_MSE_dataset<- mean(Arithmetic_errors_dataset^2) #Mean squared error

Arithmetic_MAE_dataset<- mean(abs(Arithmetic_errors_dataset)) #Mean absolute error

Arithmetic_MAPE_dataset<- 100 * mean(abs(Arithmetic_errors_dataset)/dataset_test)
#Mean absolute percentage error

Arithmetic_RMSE_dataset<- sqrt(mean(Arithmetic_errors_dataset^2)) #Root mean
squared error


## Rolling origin for Arithmetic mean


dataset_rolling_forecasts_Arithmeticmean <- matrix(NA, nrow=origins, ncol=H)

dataset_rolling_holdout_Arithmeticmean <- matrix(NA, nrow=origins, ncol=H)

colnames(dataset_rolling_forecasts_Arithmeticmean) <- paste0("horizon",c(1:H))
```

```
rownames(dataset_rolling_forecasts_Arithmeticmean) <- paste0("origin",c(1:origins))

dimnames(dataset_rolling_holdout_Arithmeticmean) <-
dimnames(dataset_rolling_forecasts_Arithmeticmean)


for(i in 1:origins)

{

  # Create a ts object out of the dataset data

  dataset_rolling_train_set <- ts(dataset[1:(dataset_rolling_train_length+i-1)],

                              frequency=frequency(dataset),

                              start=start(dataset))


  # Write down the holdout values from the test set

  dataset_rolling_holdout_Arithmeticmean[i,] <- dataset_rolling_test[i-1+(1:H)]


  # Produce forecasts and write them down

  dataset_rolling_forecasts_Arithmeticmean[i,] <-
forecast(mean(dataset_rolling_train_set),h=H)$mean

}


## MAPE for Rolling origin of Arithmetic mean

colMeans(abs(dataset_rolling_holdout_Arithmeticmean -
dataset_rolling_forecasts_Arithmeticmean))

Rolling_errors_dataset_Arithmetic<- dataset_rolling_holdout_Arithmeticmean -
dataset_rolling_forecasts_Arithmeticmean

Rolling_ME_errors_dataset_Arithmetic<- mean(Rolling_errors_dataset_Arithmetic) #Mean
error

Rolling_MSE_errors_dataset_Arithmetic<- mean(Rolling_errors_dataset_Arithmetic^2)
#Mean squared error

Rolling_MAE_errors_dataset_Arithmetic<- mean(abs(Rolling_errors_dataset_Arithmetic))
#Mean absolute error
```

```
Rolling_MAPE_dataset_Arithmetic<- 100 *
mean(abs(Rolling_errors_dataset_Arithmetic)/dataset_rolling_holdout_Arithmeticmean)

Rolling_RMSE_errors_dataset_Arithmetic<-
sqrt(mean(Rolling_errors_dataset_Arithmetic^2)) #Root mean squared error



# Create summary table for error measure of Arithmetic Mean Model

Summary_stats <-
matrix(c(Arithmetic_ME_dataset,Arithmetic_MSE_dataset,Arithmetic_MAE_dataset,Arithme
tic_MAPE_dataset,Arithmetic_RMSE_dataset), ncol=5, byrow=TRUE)

Summary_stats <-
rbind(Summary_stats,c(Rolling_ME_errors_dataset_Arithmetic,Rolling_MSE_errors_dataset_
Arithmetic,Rolling_MAE_errors_dataset_Arithmetic,Rolling_MAPE_dataset_Arithmetic,Rolli
ng_RMSE_errors_dataset_Arithmetic))

colnames(Summary_stats) <- c('ME','MSE','MAE','MAPE','RMSE')

rownames(Summary_stats) <- c('Arithmetic Mean Model Error Measures','Arithmetic Mean
Model Error Measures with Rolling origin')

Summary_stats <- as.table(Summary_stats)

Summary_stats

names(Summary_stats) <- c("Arithmetic Mean Model Error Measures","Arithmetic Mean
Model Error Measures with Rolling origin")

knitr::kable(Summary_stats)



# Simple Moving Average Model --------------------------------------------------


#Fitting a Simple average model

SMA_dataset <- ma(dataset_train, order=8, centre=FALSE)


SMA_no_NAs_dataset <- SMA_dataset[!is.na(SMA_dataset)]

Forecast_SMA_dataset <- ts(rep(SMA_no_NAs_dataset[length(SMA_no_NAs_dataset)],h),
frequency=7)
```

```
#Calculating the error measures for Simple average model

SMA_errors_dataset <- dataset_test - Forecast_SMA_dataset

SMA_ME_dataset <- mean(SMA_errors_dataset) #Mean error

SMA_MSE_dataset <- mean(SMA_errors_dataset^2) #Mean squared error

SMA_MAE_dataset<- mean(abs(SMA_errors_dataset)) #Mean absolute error

SMA_MAPE_dataset <- 100 * mean(abs(SMA_errors_dataset)/dataset_test) #Mean absolute
percentage error

SMA_RMSE_dataset<- sqrt(mean(SMA_errors_dataset^2)) #Root mean squared error


## Rolling origin for SMA


dataset_rolling_forecasts_SMA <- matrix(NA, nrow=origins, ncol=H)

dataset_rolling_holdout_SMA <- matrix(NA, nrow=origins, ncol=H)

colnames(dataset_rolling_forecasts_SMA) <- paste0("horizon",c(1:H))

rownames(dataset_rolling_forecasts_SMA) <- paste0("origin",c(1:origins))

dimnames(dataset_rolling_holdout_SMA) <- dimnames(dataset_rolling_forecasts_SMA)


for(i in 1:origins)

{

 # Create a ts object out of the dataset data

 dataset_rolling_train_set <- ts(dataset[1:(dataset_rolling_train_length+i-1)],

                                 frequency=frequency(dataset),

                                 start=start(dataset))


 # Write down the holdout values from the test set

 dataset_rolling_holdout_SMA[i,] <- dataset_rolling_test[i-1+(1:H)]
```

```
  # Produce forecasts and write them down

  dataset_rolling_forecasts_SMA[i,] <- forecast(ma(dataset_rolling_train_set, order=8,
centre=FALSE),h=H)$mean

}


## MAPE for Rolling origin of SMA

Rolling_errors_dataset_SMA<- dataset_rolling_holdout_SMA - dataset_rolling_forecasts_SMA

Rolling_ME_dataset_SMA<- mean(Rolling_errors_dataset_SMA) #Mean error

Rolling_MSE_dataset_SMA<- mean(Rolling_errors_dataset_SMA^2) #Mean squared error

Rolling_MAE_dataset_SMA<- mean(abs(Rolling_errors_dataset_SMA)) #Mean absolute
error

Rolling_MAPE_dataset_SMA<- 100 *
mean(abs(Rolling_errors_dataset_SMA)/dataset_rolling_holdout_SMA)

Rolling_RMSE_dataset_SMA<- sqrt(mean(Rolling_errors_dataset_SMA^2)) #Root mean
squared error


# Create summary table for error measure of SMA Model


Summary_stats <-
rbind(Summary_stats,c(SMA_ME_dataset,SMA_MSE_dataset,SMA_MAE_dataset,SMA_MAPE_
dataset,SMA_RMSE_dataset))

Summary_stats <-
rbind(Summary_stats,c(Rolling_ME_dataset_SMA,Rolling_MSE_dataset_SMA,Rolling_MAE_d
ataset_SMA,Rolling_MAPE_dataset_SMA,Rolling_RMSE_dataset_SMA))

colnames(Summary_stats) <- c('ME','MSE','MAE','MAPE','RMSE')

rownames(Summary_stats) <- c('Arithmetic Mean Model Error Measures','Arithmetic Mean
Model Error Measures with Rolling origin','Simple Moving Average Model Error
Measures','Simple Moving Average Model Error Measures with Rolling origin')

Summary_stats <- as.table(Summary_stats)

Summary_stats

names(Summary_stats) <- c("Arithmetic Mean Model Error Measures","Arithmetic Mean
Model Error Measures with Rolling origin","Simple Moving Average Model Error
Measures","Simple Moving Average Model Error Measures with Rolling origin")
```

```
knitr::kable(Summary_stats)

# Naive Model -----------------------------------------------------------

# Naive model
Naive_method_dataset <- naive(dataset_train, h=h)
Forecast_naive_dataset <- Naive_method_dataset$mean
plot(Naive_method_dataset)
#Calculating the error measures for Naive
Naive_errors_dataset<- dataset_test - Forecast_naive_dataset
Naive_ME_dataset <- mean(Naive_errors_dataset) #Mean error
Naive_MSE_dataset <- mean(Naive_errors_dataset^2) #Mean squared error
Naive_MAE_dataset<- mean(abs(Naive_errors_dataset)) #Mean absolute error
Naive_MAPE_dataset <- 100 * mean(abs(Naive_errors_dataset)/dataset_test) #Mean
absolute percentage error
Naive_RMSE_dataset<- sqrt(mean(Naive_errors_dataset^2)) #Root mean squared error


## Rolling origin for Naive

dataset_rolling_forecasts_Naive <- matrix(NA, nrow=origins, ncol=H)
dataset_rolling_holdout_Naive <- matrix(NA, nrow=origins, ncol=H)
colnames(dataset_rolling_forecasts_Naive) <- paste0("horizon",c(1:H))
rownames(dataset_rolling_forecasts_Naive) <- paste0("origin",c(1:origins))
dimnames(dataset_rolling_holdout_Naive) <- dimnames(dataset_rolling_forecasts_Naive)
for(i in 1:origins)
{
  # Create a ts object out of the dataset data
```

```
  dataset_rolling_train_set <- ts(dataset[1:(dataset_rolling_train_length+i-1)],
                          frequency=frequency(dataset),
                          start=start(dataset))

  # Write down the holdout values from the test set
  dataset_rolling_holdout_Naive[i,] <- dataset_rolling_test[i-1+(1:H)]

  # Produce forecasts and write them down
  dataset_rolling_forecasts_Naive[i,] <- naive(dataset_rolling_train_set,h=H)$mean
}

## MAPE for Rolling origin of Naive
Rolling_errors_dataset_Naive<- dataset_rolling_holdout_Naive - dataset_rolling_forecasts_Naive
Rolling_ME_dataset_Naive<- mean(Rolling_errors_dataset_Naive) #Mean error
Rolling_MSE_dataset_Naive<- mean(Rolling_errors_dataset_Naive^2) #Mean squared error
Rolling_MAE_dataset_Naive<- mean(abs(Rolling_errors_dataset_Naive)) #Mean absolute error
Rolling_MAPE_dataset_Naive<- 100 * mean(abs(Rolling_errors_dataset_Naive)/dataset_rolling_holdout_Naive)
Rolling_RMSE_dataset_Naive<- sqrt(mean(Rolling_errors_dataset_Naive^2)) #Root mean squared error

# Create summary table for error measure of Naive Model
Summary_table <- matrix(c(Naive_ME_dataset,Naive_MSE_dataset,Naive_MAE_dataset,Naive_MAPE_dataset,Naive_RMSE_dataset), ncol=5, byrow=TRUE)
Summary_table <- rbind(Summary_table,c(Rolling_ME_dataset_Naive,Rolling_MSE_dataset_Naive,Rolling_MAE_dataset_Naive,Rolling_MAPE_dataset_Naive,Rolling_RMSE_dataset_Naive))
colnames(Summary_table) <- c('ME','MSE','MAE','MAPE','RMSE')
```

```
rownames(Summary_table) <- c('Naive Model Error Measures','Naive Model Error Measures with Rolling origin')

Summary_table <- as.table(Summary_table)

Summary_table

names(Summary_table) <- c("Naive Model Error Measures","Naive Model Error Measures with Rolling origin")

knitr::kable(Summary_table)


# Seasonal Naive Model ----------------------------------------------------


#Seasonal Naive model

SNaive_method_dataset <- snaive(dataset_train, h=h)

Forecast_Snaive_dataset <- SNaive_method_dataset$mean

plot(SNaive_method_dataset)

#Calculating the error measures for Naive

SNaive_errors_dataset<- dataset_test - Forecast_Snaive_dataset

SNaive_ME_dataset <- mean(SNaive_errors_dataset) #Mean error

SNaive_MSE_dataset <- mean(SNaive_errors_dataset^2) #Mean squared error

SNaive_MAE_dataset<- mean(abs(SNaive_errors_dataset)) #Mean absolute error

SNaive_MAPE_dataset <- 100 * mean(abs(SNaive_errors_dataset)/dataset_test) #Mean absolute percentage error

SNaive_RMSE_dataset<- sqrt(mean(SNaive_errors_dataset^2)) #Root mean squared error


## Rolling origin for SNaive


dataset_rolling_forecasts_SNaive <- matrix(NA, nrow=origins, ncol=H)

dataset_rolling_holdout_SNaive <- matrix(NA, nrow=origins, ncol=H)

colnames(dataset_rolling_forecasts_SNaive) <- paste0("horizon",c(1:H))

rownames(dataset_rolling_forecasts_SNaive) <- paste0("origin",c(1:origins))
```

```
dimnames(dataset_rolling_holdout_SNaive) <- dimnames(dataset_rolling_forecasts_SNaive)
for(i in 1:origins)
{
  # Create a ts object out of the dataset data
  dataset_rolling_train_set <- ts(dataset[1:(dataset_rolling_train_length+i-1)],
                                  frequency=frequency(dataset),
                                  start=start(dataset))

  # Write down the holdout values from the test set
  dataset_rolling_holdout_SNaive[i,] <- dataset_rolling_test[i-1+(1:H)]

  # Produce forecasts and write them down
  dataset_rolling_forecasts_SNaive[i,] <- snaive(dataset_rolling_train_set,h=H)$mean
}

## MAPE for Rolling origin of SNaive
Rolling_errors_dataset_SNaive<- dataset_rolling_holdout_SNaive -
dataset_rolling_forecasts_SNaive
Rolling_ME_dataset_SNaive<- mean(Rolling_errors_dataset_SNaive) #Mean error
Rolling_MSE_dataset_SNaive<- mean(Rolling_errors_dataset_SNaive^2) #Mean squared
error
Rolling_MAE_dataset_SNaive<- mean(abs(Rolling_errors_dataset_SNaive)) #Mean absolute
error
Rolling_MAPE_dataset_SNaive<- 100 *
mean(abs(Rolling_errors_dataset_SNaive)/dataset_rolling_holdout_SNaive)
Rolling_RMSE_dataset_SNaive<- sqrt(mean(Rolling_errors_dataset_SNaive^2)) #Root mean
squared error

# Create summary table for error measure of Seasonal Naive Model
```

Summary_table<-
rbind(Summary_table,c(SNaive_ME_dataset,SNaive_MSE_dataset,SNaive_MAE_dataset,SNaive_MAPE_dataset,SNaive_RMSE_dataset))

Summary_table<-
rbind(Summary_table,c(Rolling_ME_dataset_SNaive,Rolling_MSE_dataset_SNaive,Rolling_MAE_dataset_SNaive,Rolling_MAPE_dataset_SNaive,Rolling_RMSE_dataset_SNaive))

colnames(Summary_table) <- c('ME','MSE','MAE','MAPE','RMSE')

rownames(Summary_table) <- c('Naive Model Error Measures','Naive Model Error Measures with Rolling origin', 'Seasonal Naive Model Error Measures', 'Seasonal Naive Model Error Measures with Rolling origin')

Summary_table <- as.table(Summary_table)

Summary_table

names(Summary_table) <- c("Naive Model Error Measures","Naive Model Error Measures with Rolling origin","Seasonal Naive Model Error Measures","Seasonal Naive Model Error Measures with Rolling origin")

knitr::kable(Summary_table)

Forecast using Seasonal Naive --

SNaive_method_dataset <- snaive(dataset, h=h)

Forecast_Snaive_dataset <- SNaive_method_dataset$mean

Forecast_Snaive_dataset

Exponential Smoothing ---

Exponential Smoothing ZZZ ---

Calculate an Optimized ES Method using ETS()

ETS_optimised_dataset <- ets(dataset_train, model = "ZZZ")

Check the AIC

ETS_optimised_dataset

Coefficient of ETS optimized method

```
coef(ETS_optimised_dataset)

checkresiduals(ETS_optimised_dataset)


#Forecating the ETS optimized model

Forecast_ETS_optimised_dataset <- forecast(ETS_optimised_dataset, h=h)

plot(Forecast_ETS_optimised_dataset)


# Error check for Forecast for ETS optimized

ETS_optimised_dataset_errors <- dataset_test - (Forecast_ETS_optimised_dataset$mean)

ETS_optimised_ME_dataset<- mean(ETS_optimised_dataset_errors) #Mean error

ETS_optimised_MSE_dataset<- mean(ETS_optimised_dataset_errors^2) #Mean squared error

ETS_optimised_MAE_dataset<- mean(abs(ETS_optimised_dataset_errors)) #Mean absolute error

ETS_optimised_MAPE_dataset<- 100 * mean(abs(ETS_optimised_dataset_errors)/dataset_test) #Mean absolute percentage error

ETS_optimised_RMSE_dataset<- sqrt(mean(ETS_optimised_dataset_errors^2)) # Root mean squared error


ETS_optimised_MAPE_dataset

ETS_optimised_RMSE_dataset



## Rolling origin for ETS optimized


dataset_rolling_forecasts_ETS_optimised<- matrix(NA, nrow=origins, ncol=H)

dataset_rolling_holdout_ETS_optimised <- matrix(NA, nrow=origins, ncol=H)

colnames(dataset_rolling_forecasts_ETS_optimised) <- paste0("horizon",c(1:H))

rownames(dataset_rolling_forecasts_ETS_optimised) <- paste0("origin",c(1:origins))
```

```
dimnames(dataset_rolling_holdout_ETS_optimised) <-
dimnames(dataset_rolling_forecasts_ETS_optimised)

for(i in 1:origins)

{

  # Create a ts object out of the dataset data

  dataset_rolling_train_set <- ts(dataset[1:(dataset_rolling_train_length+i-1)],

                                  frequency=frequency(dataset),

                                  start=start(dataset))


  # Write down the holdout values from the test set

  dataset_rolling_holdout_ETS_optimised[i,] <- dataset_rolling_test[i-1+(1:H)]


  # Produce forecasts and write them down

  dataset_rolling_forecasts_ETS_optimised[i,] <-
forecast(ets(dataset_rolling_train_set,"ZZZ"),h=H)$mean

}


## MAPE for Rolling origin of ETS optimized

Rolling_errors_dataset_ETS_optimised<- dataset_rolling_holdout_ETS_optimised -
dataset_rolling_forecasts_ETS_optimised

Rolling_ME_dataset_ETS_optimised<- mean(Rolling_errors_dataset_ETS_optimised) #Mean
error

Rolling_MSE_dataset_ETS_optimised<- mean(Rolling_errors_dataset_ETS_optimised^2)
#Mean squared error

Rolling_MAE_dataset_ETS_optimised<- mean(abs(Rolling_errors_dataset_ETS_optimised))
#Mean absolute error

Rolling_MAPE_dataset_ETS_optimised<- 100 *
mean(abs(Rolling_errors_dataset_ETS_optimised)/dataset_rolling_holdout_ETS_optimised)

Rolling_RMSE_dataset_ETS_optimised<-
sqrt(mean(Rolling_errors_dataset_ETS_optimised^2)) #Root mean squared error
```

```
# Create summary table for error measure of ETA Optimised Model

Summary<-
rbind(Summary,c(ETS_optimised_ME_dataset,ETS_optimised_MSE_dataset,ETS_optimised_
MAE_dataset,ETS_optimised_MAPE_dataset,ETS_optimised_RMSE_dataset))

Summary<-
rbind(Summary,c(Rolling_ME_dataset_ETS_optimised,Rolling_MSE_dataset_ETS_optimised,
Rolling_MAE_dataset_ETS_optimised,Rolling_MAPE_dataset_ETS_optimised,Rolling_RMSE_d
ataset_ETS_optimised))

colnames(Summary) <- c('ME','MSE','MAE','MAPE','RMSE')

rownames(Summary) <- c('Naive Model Error Measures','Naive Model Error Measures
with Rolling origin', 'Seasonal Naive Model Error Measures', 'Seasonal Naive Model Error
Measures with Rolling origin', ' ETS ANA Model Error Measures', ' ETS ANA Model Error
Measures with Rolling origin',

             ' ETS AAA Model Error Measures', ' ETS AAA Model Error Measures with
Rolling origin',

             ' ETS MAM Model Error Measures', ' ETS MAM Model Error Measures with
Rolling origin',

             ' ETS MNA Model Error Measures', ' ETS MNA Model Error Measures with
Rolling origin',

             ' ETS Optimised Model Error Measures', ' ETS Optimised Model Error Measures
with Rolling origin')

Summary <- as.table(Summary)

Summary

names(Summary) <- c("Naive Model Error Measures","Naive Model Error Measures with
Rolling origin","Seasonal Naive Model Error Measures","Seasonal Naive Model Error
Measures with Rolling origin"

            ,"ETS ANA Model Error Measures","ETS ANA Model Error Measures with Rolling
origin"

            ,"ETS AAA Model Error Measures","ETS AAA Model Error Measures with Rolling
origin",

            ,"ETS MAM Model Error Measures","ETS MAM Model Error Measures with
Rolling origin",

            ,"ETS MNA Model Error Measures","ETS MNA Model Error Measures with Rolling
origin",
```

```
             ,"ETS Optimised Model Error Measures","ETS Optimised Model Error Measures with Rolling origin")

knitr::kable(Summary)

# Forecast using Best ETS ------------------------------------------------

ETS_optimised_dataset <- ets(dataset, model = "ZZZ")


#Forecating the ETS optimized model

Forecast_ETS_optimised_dataset <- forecast(ETS_optimised_dataset, h=h)

Forecast_ETS_optimised_dataset


## Exponential Smoothing ANA ----------------------------------------------


# Fit a model using ETS(A,N,A):

ETS_ANA_opt_dataset <- ets(dataset_train, model = "ANA")

# Check the AIC

ETS_ANA_opt_dataset

#Finding the coefficients

coef(ETS_ANA_opt_dataset)

#Forecating the ANA model

Forecast_ETS_ANA_opt_dataset <- forecast(ETS_ANA_opt_dataset, h=h)

plot(Forecast_ETS_ANA_opt_dataset)

checkresiduals(ETS_ANA_opt_dataset)

# Error check for Forecast for ETS(A,N,A)

ETS_ANA_opt_dataset_errors <- dataset_test - forecast(ETS_ANA_opt_dataset, h=h)$mean

ETS_ANA_ME_dataset<- mean(ETS_ANA_opt_dataset_errors) #Mean error

ETS_ANA_MSE_dataset<- mean(ETS_ANA_opt_dataset_errors^2) #Mean squared error

ETS_ANA_MAE_dataset<- mean(abs(ETS_ANA_opt_dataset_errors)) #Mean absolute error

ETS_ANA_MAPE_dataset<- 100 * mean(abs(ETS_ANA_opt_dataset_errors)/dataset_test)
#Mean absolute percentage error
```

```
ETS_ANA_RMSE_dataset<- sqrt(mean(ETS_ANA_opt_dataset_errors^2)) # Root mean squared error


ETS_ANA_MAPE_dataset

ETS_ANA_RMSE_dataset

## Rolling origin for ETS ANA


dataset_rolling_forecasts_ETS_ANA <- matrix(NA, nrow=origins, ncol=H)

dataset_rolling_holdout_ETS_ANA <- matrix(NA, nrow=origins, ncol=H)

colnames(dataset_rolling_forecasts_ETS_ANA) <- paste0("horizon",c(1:H))

rownames(dataset_rolling_forecasts_ETS_ANA) <- paste0("origin",c(1:origins))

dimnames(dataset_rolling_holdout_ETS_ANA) <- dimnames(dataset_rolling_forecasts_ETS_ANA)


for(i in 1:origins)

{

  # Create a ts object out of the dataset data

  dataset_rolling_train_set <- ts(dataset[1:(dataset_rolling_train_length+i-1)],

                               frequency=frequency(dataset),

                               start=start(dataset))


  # Write down the holdout valuETS from the test set

  dataset_rolling_holdout_ETS_ANA[i,] <- dataset_rolling_test[i-1+(1:H)]


  # Produce forecasts and write them down

  dataset_rolling_forecasts_ETS_ANA[i,] <- forecast(ets(dataset_rolling_train_set,"ANA"),h=H)$mean

}
```

MAPE for Rolling origin of ETA ANA

```
Rolling_errors_dataset_ETS_ANA<- dataset_rolling_holdout_ETS_ANA -
dataset_rolling_forecasts_ETS_ANA

Rolling_ME_dataset_ETS_ANA<- mean(Rolling_errors_dataset_ETS_ANA) #Mean error

Rolling_MSE_dataset_ETS_ANA<- mean(Rolling_errors_dataset_ETS_ANA^2) #Mean
squared error

Rolling_MAE_dataset_ETS_ANA<- mean(abs(Rolling_errors_dataset_ETS_ANA)) #Mean
absolute error

Rolling_MAPE_dataset_ETS_ANA<- 100 *
mean(abs(Rolling_errors_dataset_ETS_ANA)/dataset_rolling_holdout_ETS_ANA)

Rolling_RMSE_dataset_ETS_ANA<- sqrt(mean(Rolling_errors_dataset_ETS_ANA^2)) #Root
mean squared error


Rolling_MAPE_dataset_ETS_ANA

Rolling_RMSE_dataset_ETS_ANA


# Create summary table for error measure of ETA ANA Model


Summary<-
rbind(Summary,c(ETS_ANA_ME_dataset,ETS_ANA_MSE_dataset,ETS_ANA_MAE_dataset,ETS
_ANA_MAPE_dataset,ETS_ANA_RMSE_dataset))

Summary<-
rbind(Summary,c(Rolling_ME_dataset_ETS_ANA,Rolling_MSE_dataset_ETS_ANA,Rolling_MA
E_dataset_ETS_ANA,Rolling_MAPE_dataset_ETS_ANA,Rolling_RMSE_dataset_ETS_ANA))

colnames(Summary) <- c('ME','MSE','MAE','MAPE','RMSE')

rownames(Summary) <- c('Naive Model Error Measures','Naive Model Error Measures
with Rolling origin', 'Seasonal Naive Model Error Measures', 'Seasonal Naive Model Error
Measures with Rolling origin', ' ETS ANA Model Error Measures', ' ETS ANA Model Error
Measures with Rolling origin'

                    )

Summary <- as.table(Summary)

Summary
```

names(Summary) <- c("Naive Model Error Measures","Naive Model Error Measures with Rolling origin","Seasonal Naive Model Error Measures","Seasonal Naive Model Error Measures with Rolling origin"

,"ETS ANA Model Error Measures","ETS ANA Model Error Measures with Rolling origin")

knitr::kable(Summary)

Exponential Smoothing AAA ---

Fit a model using ETS(A,A,A):

ETS_AAA_opt_dataset <- ets(dataset_train, model= "AAA")

Check the AIC

ETS_AAA_opt_dataset

#Finding the coefficients

coef(ETS_AAA_opt_dataset)

#Forecating the ANA model

Forecast_ETS_AAA_opt_dataset <- forecast(ETS_AAA_opt_dataset, h=h)

plot(Forecast_ETS_AAA_opt_dataset)

checkresiduals(ETS_AAA_opt_dataset)

Error check for Forecast for ETS(A,A,A)

ETS_AAA_opt_dataset_errors <- dataset_test - (Forecast_ETS_AAA_opt_dataset$mean)

ETS_AAA_ME_dataset<- mean(ETS_AAA_opt_dataset_errors) #Mean error

ETS_AAA_MSE_dataset<- mean(ETS_AAA_opt_dataset_errors^2) #Mean squared error

ETS_AAA_MAE_dataset<- mean(abs(ETS_AAA_opt_dataset_errors)) #Mean absolute error

ETS_AAA_MAPE_dataset<- 100 * mean(abs(ETS_AAA_opt_dataset_errors)/dataset_test)

ETS_AAA_RMSE_dataset<- sqrt(mean(ETS_AAA_opt_dataset_errors^2)) # Root mean squared error

ETS_AAA_MAPE_dataset

```
ETS_AAA_RMSE_dataset

## Rolling origin for ETS AAA


dataset_rolling_forecasts_ETS_AAA <- matrix(NA, nrow=origins, ncol=H)

dataset_rolling_holdout_ETS_AAA <- matrix(NA, nrow=origins, ncol=H)

colnames(dataset_rolling_forecasts_ETS_AAA) <- paste0("horizon",c(1:H))

rownames(dataset_rolling_forecasts_ETS_AAA) <- paste0("origin",c(1:origins))

dimnames(dataset_rolling_holdout_ETS_AAA) <-
dimnames(dataset_rolling_forecasts_ETS_AAA)

for(i in 1:origins)

{

  # Create a ts object out of the dataset data

  dataset_rolling_train_set <- ts(dataset[1:(dataset_rolling_train_length+i-1)],

                                  frequency=frequency(dataset),

                                  start=start(dataset))


  # Write down the holdout values from the test set

  dataset_rolling_holdout_ETS_AAA[i,] <- dataset_rolling_test[i-1+(1:H)]


  # Produce forecasts and write them down

  dataset_rolling_forecasts_ETS_AAA[i,] <-
forecast(ets(dataset_rolling_train_set,"AAA"),h=H)$mean

}


## MAPE for Rolling origin of ETS AAA

Rolling_errors_dataset_ETS_AAA<- dataset_rolling_holdout_ETS_AAA -
dataset_rolling_forecasts_ETS_AAA

Rolling_ME_dataset_ETS_AAA<- mean(Rolling_errors_dataset_ETS_AAA) #Mean error

Rolling_MSE_dataset_ETS_AAA<- mean(Rolling_errors_dataset_ETS_AAA^2) #Mean
squared error
```

Rolling_MAE_dataset_ETS_AAA<- mean(abs(Rolling_errors_dataset_ETS_AAA)) #Mean absolute error

Rolling_MAPE_dataset_ETS_AAA<- 100 * mean(abs(Rolling_errors_dataset_ETS_AAA)/dataset_rolling_holdout_ETS_AAA)

Rolling_RMSE_dataset_ETS_AAA<- sqrt(mean(Rolling_errors_dataset_ETS_AAA^2)) #Root mean squared error

Rolling_MAPE_dataset_ETS_AAA

Rolling_RMSE_dataset_ETS_AAA

Create summary table for error measure of ETA AAA Model

Summary<- rbind(Summary,c(ETS_AAA_ME_dataset,ETS_AAA_MSE_dataset,ETS_AAA_MAE_dataset,ETS_AAA_MAPE_dataset,ETS_AAA_RMSE_dataset))

Summary<- rbind(Summary,c(Rolling_ME_dataset_ETS_AAA,Rolling_MSE_dataset_ETS_AAA,Rolling_MAE_dataset_ETS_AAA,Rolling_MAPE_dataset_ETS_AAA,Rolling_RMSE_dataset_ETS_AAA))

colnames(Summary) <- c('ME','MSE','MAE','MAPE','RMSE')

rownames(Summary) <- c('Naive Model Error Measures','Naive Model Error Measures with Rolling origin', 'Seasonal Naive Model Error Measures', 'Seasonal Naive Model Error Measures with Rolling origin', ' ETS ANA Model Error Measures', ' ETS ANA Model Error Measures with Rolling origin',

' ETS AAA Model Error Measures', ' ETS AAA Model Error Measures with Rolling origin')

Summary <- as.table(Summary)

Summary

names(Summary) <- c("Naive Model Error Measures","Naive Model Error Measures with Rolling origin","Seasonal Naive Model Error Measures","Seasonal Naive Model Error Measures with Rolling origin"

,"ETS ANA Model Error Measures","ETS ANA Model Error Measures with Rolling origin"

,"ETS AAA Model Error Measures","ETS AAA Model Error Measures with Rolling origin")

```
knitr::kable(Summary)

## Exponential Smoothing MAM -------------------------------------------

# Fit a model using ETS(M,A,M):
ETS_MAM_dataset <- ets(dataset_train, model = "MAM")
# Check the AIC
ETS_MAM_dataset
#Finding the coefficients
coef(ETS_MAM_dataset)
#Forecating the ANA model
Forecast_ETS_MAM_dataset <- forecast(ETS_MAM_dataset, h=h)
plot(Forecast_ETS_MAM_dataset)
checkresiduals(ETS_MAM_dataset)

# Error check for Forecast for ETS(A,N,M)
ETS_MAM_dataset_errors <- dataset_test - (Forecast_ETS_MAM_dataset$mean)
ETS_MAM_ME_dataset<- mean(ETS_MAM_dataset_errors) #Mean error
ETS_MAM_MSE_dataset<- mean(ETS_MAM_dataset_errors^2) #Mean squared error
ETS_MAM_MAE_dataset<- mean(abs(ETS_MAM_dataset_errors)) #Mean absolute error
ETS_MAM_MAPE_dataset<- 100 * mean(abs(ETS_MAM_dataset_errors)/dataset_test)
#Mean absolute percentage error
ETS_MAM_RMSE_dataset<- sqrt(mean(ETS_MAM_dataset_errors^2)) # Root mean squared
error

ETS_MAM_MAPE_dataset
ETS_MAM_RMSE_dataset
## Rolling origin for ETS MAM
```

```
dataset_rolling_forecasts_ETS_MAM <- matrix(NA, nrow=origins, ncol=H)

dataset_rolling_holdout_ETS_MAM <- matrix(NA, nrow=origins, ncol=H)

colnames(dataset_rolling_forecasts_ETS_MAM) <- paste0("horizon",c(1:H))

rownames(dataset_rolling_forecasts_ETS_MAM) <- paste0("origin",c(1:origins))

dimnames(dataset_rolling_holdout_ETS_MAM) <-
dimnames(dataset_rolling_forecasts_ETS_MAM)

for(i in 1:origins)

{

 # Create a ts object out of the dataset data

 dataset_rolling_train_set <- ts(dataset[1:(dataset_rolling_train_length+i-1)],

                frequency=frequency(dataset),

                start=start(dataset))


 # Write down the holdout values from the test set

 dataset_rolling_holdout_ETS_MAM[i,] <- dataset_rolling_test[i-1+(1:H)]


 # Produce forecasts and write them down

 dataset_rolling_forecasts_ETS_MAM[i,] <-
forecast(ets(dataset_rolling_train_set,"MAM"),h=H)$mean

}


## MAPE for Rolling origin of ETS MAM

Rolling_errors_dataset_ETS_MAM<- dataset_rolling_holdout_ETS_MAM -
dataset_rolling_forecasts_ETS_MAM

Rolling_ME_dataset_ETS_MAM<- mean(Rolling_errors_dataset_ETS_MAM) #Mean error

Rolling_MSE_dataset_ETS_MAM<- mean(Rolling_errors_dataset_ETS_MAM^2) #Mean
squared error

Rolling_MAE_dataset_ETS_MAM<- mean(abs(Rolling_errors_dataset_ETS_MAM)) #Mean
absolute error
```

Rolling_MAPE_dataset_ETS_MAM<- 100 * mean(abs(Rolling_errors_dataset_ETS_MAM)/dataset_rolling_holdout_ETS_MAM)

Rolling_RMSE_dataset_ETS_MAM<- sqrt(mean(Rolling_errors_dataset_ETS_MAM^2)) #Root mean squared error

Rolling_MAPE_dataset_ETS_MAM

Rolling_RMSE_dataset_ETS_MAM

Create summary table for error measure of ETA MAM Model

Summary<- rbind(Summary,c(ETS_MAM_ME_dataset,ETS_MAM_MSE_dataset,ETS_MAM_MAE_dataset,ETS_MAM_MAPE_dataset,ETS_MAM_RMSE_dataset))

Summary<- rbind(Summary,c(Rolling_ME_dataset_ETS_MAM,Rolling_MSE_dataset_ETS_MAM,Rolling_MAE_dataset_ETS_MAM,Rolling_MAPE_dataset_ETS_MAM,Rolling_RMSE_dataset_ETS_MAM))

colnames(Summary) <- c('ME','MSE','MAE','MAPE','RMSE')

rownames(Summary) <- c('Naive Model Error Measures','Naive Model Error Measures with Rolling origin', 'Seasonal Naive Model Error Measures', 'Seasonal Naive Model Error Measures with Rolling origin', ' ETS ANA Model Error Measures', ' ETS ANA Model Error Measures with Rolling origin',

' ETS AAA Model Error Measures', ' ETS AAA Model Error Measures with Rolling origin',

' ETS MAM Model Error Measures', ' ETS MAM Model Error Measures with Rolling origin')

Summary <- as.table(Summary)

Summary

names(Summary) <- c("Naive Model Error Measures","Naive Model Error Measures with Rolling origin","Seasonal Naive Model Error Measures","Seasonal Naive Model Error Measures with Rolling origin"

,"ETS ANA Model Error Measures","ETS ANA Model Error Measures with Rolling origin"

,"ETS AAA Model Error Measures","ETS AAA Model Error Measures with Rolling origin",

```
                ,"ETS MAM Model Error Measures","ETS MAM Model Error Measures with Rolling origin")

knitr::kable(Summary)


## Exponential Smoothing MNA ---------------------------------------------

# Fit a model using ETS(M,N,A):

ETS_MNA_dataset <- ets(dataset_train, model = "MNA")

# Check the AIC

ETS_MNA_dataset

#Finding the coefficients

coef(ETS_MNA_dataset)

#Forecating the MNA model

Forecast_ETS_MNA_dataset <- forecast(ETS_MNA_dataset, h=h)

plot(Forecast_ETS_MNA_dataset)

checkresiduals(ETS_MNA_dataset)


# Error check for Forecast for ETS(M,N,A)

ETS_MNA_dataset_errors <- dataset_test - (Forecast_ETS_MNA_dataset$mean)

ETS_MNA_ME_dataset<- mean(ETS_MNA_dataset_errors) #Mean error

ETS_MNA_MSE_dataset<- mean(ETS_MNA_dataset_errors^2) #Mean squared error

ETS_MNA_MAE_dataset<- mean(abs(ETS_MNA_dataset_errors)) #Mean absolute error

ETS_MNA_MAPE_dataset<- 100 * mean(abs(ETS_MNA_dataset_errors)/dataset_test) #Mean absolute percentage error

ETS_MNA_RMSE_dataset<- sqrt(mean(ETS_MNA_dataset_errors^2)) # Root mean squared error



## Rolling origin for ETS MNA
```

```
dataset_rolling_forecasts_ETS_MNA <- matrix(NA, nrow=origins, ncol=H)

dataset_rolling_holdout_ETS_MNA <- matrix(NA, nrow=origins, ncol=H)

colnames(dataset_rolling_forecasts_ETS_MNA) <- paste0("horizon",c(1:H))

rownames(dataset_rolling_forecasts_ETS_MNA) <- paste0("origin",c(1:origins))

dimnames(dataset_rolling_holdout_ETS_MNA) <-
dimnames(dataset_rolling_forecasts_ETS_MNA)

for(i in 1:origins)

{

 # Create a ts object out of the dataset data

 dataset_rolling_train_set <- ts(dataset[1:(dataset_rolling_train_length+i-1)],

                  frequency=frequency(dataset),

                  start=start(dataset))


 # Write down the holdout values from the test set

 dataset_rolling_holdout_ETS_MNA[i,] <- dataset_rolling_test[i-1+(1:H)]


 # Produce forecasts and write them down

 dataset_rolling_forecasts_ETS_MNA[i,] <-
forecast(ets(dataset_rolling_train_set,"MNA"),h=H)$mean

}


## MAPE for Rolling origin of ETS MNA

Rolling_errors_dataset_ETS_MNA<- dataset_rolling_holdout_ETS_MNA -
dataset_rolling_forecasts_ETS_MNA

Rolling_ME_dataset_ETS_MNA<- mean(Rolling_errors_dataset_ETS_MNA) #Mean error

Rolling_MSE_dataset_ETS_MNA<- mean(Rolling_errors_dataset_ETS_MNA^2) #Mean
squared error

Rolling_MAE_dataset_ETS_MNA<- mean(abs(Rolling_errors_dataset_ETS_MNA)) #Mean
absolute error
```

Rolling_MAPE_dataset_ETS_MNA<- 100 * mean(abs(Rolling_errors_dataset_ETS_MNA)/dataset_rolling_holdout_ETS_MNA)

Rolling_RMSE_dataset_ETS_MNA<- sqrt(mean(Rolling_errors_dataset_ETS_MNA^2)) #Root mean squared error

Rolling_MAPE_dataset_ETS_MNA

Rolling_RMSE_dataset_ETS_MNA

Create summary table for error measure of ETA MNA Model

Summary<- rbind(Summary,c(ETS_MNA_ME_dataset,ETS_MNA_MSE_dataset,ETS_MNA_MAE_dataset,ETS_MNA_MAPE_dataset,ETS_MNA_RMSE_dataset))

Summary<- rbind(Summary,c(Rolling_ME_dataset_ETS_MNA,Rolling_MSE_dataset_ETS_MNA,Rolling_MAE_dataset_ETS_MNA,Rolling_MAPE_dataset_ETS_MNA,Rolling_RMSE_dataset_ETS_MNA))

colnames(Summary) <- c('ME','MSE','MAE','MAPE','RMSE')

rownames(Summary) <- c('Naive Model Error Measures','Naive Model Error Measures with Rolling origin', 'Seasonal Naive Model Error Measures', 'Seasonal Naive Model Error Measures with Rolling origin', ' ETS ANA Model Error Measures', ' ETS ANA Model Error Measures with Rolling origin',

' ETS AAA Model Error Measures', ' ETS AAA Model Error Measures with Rolling origin',

' ETS MAM Model Error Measures', ' ETS MAM Model Error Measures with Rolling origin',

' ETS MNA Model Error Measures', ' ETS MNA Model Error Measures with Rolling origin')

Summary <- as.table(Summary)

Summary

names(Summary) <- c("Naive Model Error Measures","Naive Model Error Measures with Rolling origin","Seasonal Naive Model Error Measures","Seasonal Naive Model Error Measures with Rolling origin"

,"ETS ANA Model Error Measures","ETS ANA Model Error Measures with Rolling origin"

```
              ,"ETS AAA Model Error Measures","ETS AAA Model Error Measures with Rolling origin",
              ,"ETS MAM Model Error Measures","ETS MAM Model Error Measures with Rolling origin",
              ,"ETS MNA Model Error Measures","ETS MNA Model Error Measures with Rolling origin")
knitr::kable(Summary)

# ARIMA Prep ---------------------------------------------------------------

# Perform KPSS and ADF test on time series
kpss.test(dataset_train)
adf.test(dataset_train)

# Plot the ACF and PACF plots
tsdisplay(dataset)

#Differencing order
nsdiffs(dataset)  # Seasonal diff
ndiffs(dataset) # Normal diff

# First order Differencing
diff_dataset_train <- diff(dataset_train)

# Plot ACF and PACF of first differences
tsdisplay(diff_dataset_train)

# KPSS/ADF test of 1st order diff series
kpss.test(diff_dataset_train)
```

```
adf.test(diff_dataset_train)

# Plot ACF and PACF of first and seasonal differences
tsdisplay(diff(diff((dataset_train),lag=7)))

# Plot ACF and PACF of seasonal differences
tsdisplay(diff(dataset_train, lag=7))

# Second order diff
diff2_data <- diff(dataset_train, differences=2)
tsdisplay(diff2_data)

## Fitting the ARIMA/SARIMA Models
tsdisplay(dataset_train)



## Seasonal ARIMA Model (7,1,1)(0,1,3) ------------------------------------

# Model implementation
Arima711_Seasoanl013_dataset<- Arima(dataset_train, order=c(7,1,1),  seasonal=c(0,1,3))

# Check the coeff of SARIMA Models
Arima711_Seasoanl013_dataset

# Check residuals of SARIMA model
checkresiduals(Arima711_Seasoanl013_dataset)

# Check ACF/PACF plot of residuals
```

```
tsdisplay(residuals(Arima711_Seasoanl013_dataset))

# Forecasting

FC_Arima711_Seasoanl013_dataset <- forecast(Arima711_Seasoanl013_dataset, h=h)

# Error measures

Arima711_Seasoanl013_dataset_errors <- dataset_test -
(FC_Arima711_Seasoanl013_dataset)$mean

Arima711_Seasoanl013_dataset_ME <- mean(Arima711_Seasoanl013_dataset_errors)
#Mean error

Arima711_Seasoanl013_dataset_MSE <- mean(Arima711_Seasoanl013_dataset_errors^2)
#Mean squared error

Arima711_Seasoanl013_dataset_MAE <-
mean(abs(Arima711_Seasoanl013_dataset_errors)) #Mean absolute error

Arima711_Seasoanl013_dataset_MAPE <- 100 *
mean(abs(Arima711_Seasoanl013_dataset_errors)/dataset_test)

Arima711_Seasoanl013_dataset_RMSE <-
sqrt(mean(Arima711_Seasoanl013_dataset_errors^2)) # Root mean squared error

Arima711_Seasoanl013_dataset_errors

Arima711_Seasoanl013_dataset_MAPE

Arima711_Seasoanl013_dataset_RMSE

## Rolling origin for Arima711_Seasoanl013

dataset_rolling_forecasts_Arima711_Seasoanl013 <- matrix(NA, nrow=origins, ncol=H)

dataset_rolling_holdout_Arima711_Seasoanl013 <- matrix(NA, nrow=origins, ncol=H)

colnames(dataset_rolling_forecasts_Arima711_Seasoanl013) <- paste0("horizon",c(1:H))

rownames(dataset_rolling_forecasts_Arima711_Seasoanl013) <-
paste0("origin",c(1:origins))

dimnames(dataset_rolling_holdout_Arima711_Seasoanl013) <-
dimnames(dataset_rolling_forecasts_Arima711_Seasoanl013)
```

```
for(i in 1:origins)

{

  # Create a ts object out of the dataset data

  dataset_rolling_train_set <- ts(dataset[1:(dataset_rolling_train_length+i-1)],

                               frequency=frequency(dataset),

                               start=start(dataset))


  # Write down the holdout values from the test set

  dataset_rolling_holdout_Arima711_Seasoanl013[i,] <- dataset_rolling_test[i-1+(1:H)]


  # Produce forecasts and write them down

  dataset_rolling_forecasts_Arima711_Seasoanl013[i,] <-
forecast(arima(dataset_rolling_train_set, order=c(7,1,1),  seasonal=c(0,1,3)),h=H)$mean

}




## MAPE for Rolling origin of Arima711_Seasoanl013

Rolling_errors_dataset_Arima711_Seasoanl013<-
dataset_rolling_holdout_Arima711_Seasoanl013 -
dataset_rolling_forecasts_Arima711_Seasoanl013

Rolling_ME_dataset_Arima711_Seasoanl013<-
mean(Rolling_errors_dataset_Arima711_Seasoanl013) #Mean error

Rolling_MSE_dataset_Arima711_Seasoanl013<-
mean(Rolling_errors_dataset_Arima711_Seasoanl013^2) #Mean squared error

Rolling_MAE_dataset_Arima711_Seasoanl013<-
mean(abs(Rolling_errors_dataset_Arima711_Seasoanl013)) #Mean absolute error

Rolling_MAPE_dataset_Arima711_Seasoanl013<- 100 *
mean(abs(Rolling_errors_dataset_Arima711_Seasoanl013)/dataset_rolling_holdout_Arima
711_Seasoanl013)

Rolling_RMSE_dataset_Arima711_Seasoanl013<-
sqrt(mean(Rolling_errors_dataset_Arima711_Seasoanl013^2)) #Root mean squared error
```

```
Rolling_MAPE_dataset_Arima711_Seasoanl013

Rolling_RMSE_dataset_Arima711_Seasoanl013



# Create summary table for error measure of Seasonal Naive Model


Summary_table<-rbind(Summary_table,c(Arima711_Seasoanl013_dataset_ME,Arima711_Seasoanl013_dataset_MSE,Arima711_Seasoanl013_dataset_MAE,Arima711_Seasoanl013_dataset_MAPE,Arima711_Seasoanl013_dataset_RMSE))

Summary_table<-rbind(Summary_table,c(Rolling_ME_dataset_Arima711_Seasoanl013,Rolling_MSE_dataset_Arima711_Seasoanl013,Rolling_MAE_dataset_Arima711_Seasoanl013,Rolling_MAPE_dataset_Arima711_Seasoanl013,Rolling_RMSE_dataset_Arima711_Seasoanl013))

colnames(Summary_table) <- c('ME','MSE','MAE','MAPE','RMSE')

rownames(Summary_table) <- c('Naive Model Error Measures','Naive Model Error Measures with Rolling origin', 'Seasonal Naive Model Error Measures', 'Seasonal Naive Model Error Measures with Rolling origin',

                        'Arima 711 Seasonal 013 Error Measures', 'Arima 711 Seasonal 013 Error Measures with Rolling origin')

Summary_table <- as.table(Summary_table)

Summary_table

names(Summary_table) <- c("Naive Model Error Measures","Naive Model Error Measures with Rolling origin","Seasonal Naive Model Error Measures","Seasonal Naive Model Error Measures with Rolling origin",

                      "Arima 711 Seasonal 013 Error Measures","Arima 711 Seasonal 013 Error Measures with Rolling origin")

knitr::kable(Summary_table)




## Seasonal ARIMA Model (2,0,2)(0,1,2) ------------------------------------
```

```
Arima202_Seasoanl012_dataset<- Arima(dataset_train, order=c(2,0,2),  seasonal=c(0,1,2))

Arima202_Seasoanl012_dataset # 2946.29   4385.27


checkresiduals(Arima202_Seasoanl012_dataset)


tsdisplay(residuals(Arima202_Seasoanl012_dataset)) # Failed in Residuals plot


FC_Arima202_Seasoanl012_dataset <- forecast(Arima202_Seasoanl012_dataset, h=h)

FC_Arima202_Seasoanl012_dataset


Arima202_Seasoanl012_dataset_errors <- dataset_test -
FC_Arima202_Seasoanl012_dataset$mean

Arima202_Seasoanl012_dataset_ME <- mean(Arima202_Seasoanl012_dataset_errors)
#Mean error

Arima202_Seasoanl012_dataset_MSE <- mean(Arima202_Seasoanl012_dataset_errors^2)
#Mean squared error

Arima202_Seasoanl012_dataset_MAE <-
mean(abs(Arima202_Seasoanl012_dataset_errors)) #Mean absolute error

Arima202_Seasoanl012_dataset_MAPE <- 100 *
mean(abs(Arima202_Seasoanl012_dataset_errors)/dataset_test)

Arima202_Seasoanl012_dataset_RMSE <-
sqrt(mean(Arima202_Seasoanl012_dataset_errors^2)) # Root mean squared error


Arima202_Seasoanl012_dataset_MAPE

Arima202_Seasoanl012_dataset_RMSE

## Rolling origin for Arima202_Seasoanl012


dataset_rolling_forecasts_Arima202_Seasoanl012 <- matrix(NA, nrow=origins, ncol=H)

dataset_rolling_holdout_Arima202_Seasoanl012 <- matrix(NA, nrow=origins, ncol=H)
```

```
colnames(dataset_rolling_forecasts_Arima202_Seasoanl012) <- paste0("horizon",c(1:H))

rownames(dataset_rolling_forecasts_Arima202_Seasoanl012) <-
paste0("origin",c(1:origins))

dimnames(dataset_rolling_holdout_Arima202_Seasoanl012) <-
dimnames(dataset_rolling_forecasts_Arima202_Seasoanl012)

for(i in 1:origins)

{

 # Create a ts object out of the dataset data

 dataset_rolling_train_set <- ts(dataset[1:(dataset_rolling_train_length+i-1)],

                          frequency=frequency(dataset),

                          start=start(dataset))


 # Write down the holdout values from the test set

 dataset_rolling_holdout_Arima202_Seasoanl012[i,] <- dataset_rolling_test[i-1+(1:H)]


 # Produce forecasts and write them down

 dataset_rolling_forecasts_Arima202_Seasoanl012[i,] <-
forecast(arima(dataset_rolling_train_set, order=c(2,0,2),  seasonal=c(0,1,2)),h=H)$mean

}


## MAPE for Rolling origin of Arima202_Seasoanl012

Rolling_errors_dataset_Arima202_Seasoanl012<-
dataset_rolling_holdout_Arima202_Seasoanl012 -
dataset_rolling_forecasts_Arima202_Seasoanl012

Rolling_ME_dataset_Arima202_Seasoanl012<-
mean(Rolling_errors_dataset_Arima202_Seasoanl012) #Mean error

Rolling_MSE_dataset_Arima202_Seasoanl012<-
mean(Rolling_errors_dataset_Arima202_Seasoanl012^2) #Mean squared error

Rolling_MAE_dataset_Arima202_Seasoanl012<-
mean(abs(Rolling_errors_dataset_Arima202_Seasoanl012)) #Mean absolute error
```

```
Rolling_MAPE_dataset_Arima202_Seasoanl012<- 100 *
mean(abs(Rolling_errors_dataset_Arima202_Seasoanl012)/dataset_rolling_holdout_Arima
202_Seasoanl012)

Rolling_RMSE_dataset_Arima202_Seasoanl012<-
sqrt(mean(Rolling_errors_dataset_Arima202_Seasoanl012^2)) #Root mean squared error


Rolling_MAPE_dataset_Arima202_Seasoanl012

Rolling_RMSE_dataset_Arima202_Seasoanl012

## Seasonal ARIMA Model (2,0,2)(1,1,1) ------------------------------------


Arima202_Seasoanl111_dataset<- Arima(dataset_train, order=c(2,0,2),  seasonal=c(1,1,1))


Arima202_Seasoanl111_dataset # 2945.85   4385.02


checkresiduals(Arima202_Seasoanl111_dataset)


tsdisplay(residuals(Arima202_Seasoanl111_dataset)) # Failed in Residuals plot


FC_Arima202_Seasoanl111_dataset <- forecast(Arima202_Seasoanl111_dataset,
h=h)$mean


Arima202_Seasoanl111_dataset_errors <- dataset_test - FC_Arima202_Seasoanl111_dataset

Arima202_Seasoanl111_dataset_ME <- mean(Arima202_Seasoanl111_dataset_errors)
#Mean error

Arima202_Seasoanl111_dataset_MSE <- mean(Arima202_Seasoanl111_dataset_errors^2)
#Mean squared error

Arima202_Seasoanl111_dataset_MAE <-
mean(abs(Arima202_Seasoanl111_dataset_errors)) #Mean absolute error

Arima202_Seasoanl111_dataset_MAPE <- 100 *
mean(abs(Arima202_Seasoanl111_dataset_errors)/dataset_test)
```

```
Arima202_Seasoanl111_dataset_RMSE <-
sqrt(mean(Arima202_Seasoanl111_dataset_errors^2)) # Root mean squared error


Arima202_Seasoanl111_dataset_MAPE

Arima202_Seasoanl111_dataset_RMSE

## Rolling origin for Arima202_Seasoanl111


dataset_rolling_forecasts_Arima202_Seasoanl111 <- matrix(NA, nrow=origins, ncol=H)

dataset_rolling_holdout_Arima202_Seasoanl111 <- matrix(NA, nrow=origins, ncol=H)

colnames(dataset_rolling_forecasts_Arima202_Seasoanl111) <- paste0("horizon",c(1:H))

rownames(dataset_rolling_forecasts_Arima202_Seasoanl111) <-
paste0("origin",c(1:origins))

dimnames(dataset_rolling_holdout_Arima202_Seasoanl111) <-
dimnames(dataset_rolling_forecasts_Arima202_Seasoanl111)

for(i in 1:origins)

{

  # Create a ts object out of the dataset data

  dataset_rolling_train_set <- ts(dataset[1:(dataset_rolling_train_length+i-1)],

                          frequency=frequency(dataset),

                          start=start(dataset))


  # Write down the holdout values from the test set

  dataset_rolling_holdout_Arima202_Seasoanl111[i,] <- dataset_rolling_test[i-1+(1:H)]


  # Produce forecasts and write them down

  dataset_rolling_forecasts_Arima202_Seasoanl111[i,] <-
forecast(arima(dataset_rolling_train_set, order=c(2,0,2),  seasonal=c(1,1,1)),h=H)$mean

}


## MAPE for Rolling origin of Arima202_Seasoanl111
```

```
Rolling_errors_dataset_Arima202_Seasoanl111<-
dataset_rolling_holdout_Arima202_Seasoanl111 -
dataset_rolling_forecasts_Arima202_Seasoanl111

Rolling_ME_dataset_Arima202_Seasoanl111<-
mean(Rolling_errors_dataset_Arima202_Seasoanl111) #Mean error

Rolling_MSE_dataset_Arima202_Seasoanl111<-
mean(Rolling_errors_dataset_Arima202_Seasoanl111^2) #Mean squared error

Rolling_MAE_dataset_Arima202_Seasoanl111<-
mean(abs(Rolling_errors_dataset_Arima202_Seasoanl111)) #Mean absolute error

Rolling_MAPE_dataset_Arima202_Seasoanl111<- 100 *
mean(abs(Rolling_errors_dataset_Arima202_Seasoanl111)/dataset_rolling_holdout_Arima
202_Seasoanl111)

Rolling_RMSE_dataset_Arima202_Seasoanl111<-
sqrt(mean(Rolling_errors_dataset_Arima202_Seasoanl111^2)) #Root mean squared error


Rolling_MAPE_dataset_Arima202_Seasoanl111




## Seasonal ARIMA Model (1,1,3)(2,1,2) ------------------------------------


Arima113_Seasoanl212_dataset<- Arima(dataset_train, order=c(1,1,3),  seasonal=c(2,1,2))


Arima113_Seasoanl212_dataset # 2953.3    4395.27


checkresiduals(Arima113_Seasoanl212_dataset)


tsdisplay(residuals(Arima113_Seasoanl212_dataset))


FC_Arima113_Seasoanl212_dataset <- forecast(Arima113_Seasoanl212_dataset,
h=h)$mean
```

```
Arima113_Seasoanl212_dataset_errors <- dataset_test - FC_Arima113_Seasoanl212_dataset

Arima113_Seasoanl212_dataset_ME <- mean(Arima113_Seasoanl212_dataset_errors)
#Mean error

Arima113_Seasoanl212_dataset_MSE <- mean(Arima113_Seasoanl212_dataset_errors^2)
#Mean squared error

Arima113_Seasoanl212_dataset_MAE <-
mean(abs(Arima113_Seasoanl212_dataset_errors)) #Mean absolute error

Arima113_Seasoanl212_dataset_MAPE <- 100 *
mean(abs(Arima113_Seasoanl212_dataset_errors)/dataset_test)

Arima113_Seasoanl212_dataset_RMSE <-
sqrt(mean(Arima113_Seasoanl212_dataset_errors^2)) # Root mean squared error


Arima113_Seasoanl212_dataset_MAPE

Arima113_Seasoanl212_dataset_RMSE


## Rolling origin for Arima113_Seasoanl212


dataset_rolling_forecasts_Arima113_Seasoanl212 <- matrix(NA, nrow=origins, ncol=H)

dataset_rolling_holdout_Arima113_Seasoanl212 <- matrix(NA, nrow=origins, ncol=H)

colnames(dataset_rolling_forecasts_Arima113_Seasoanl212) <- paste0("horizon",c(1:H))

rownames(dataset_rolling_forecasts_Arima113_Seasoanl212) <-
paste0("origin",c(1:origins))

dimnames(dataset_rolling_holdout_Arima113_Seasoanl212) <-
dimnames(dataset_rolling_forecasts_Arima113_Seasoanl212)

for(i in 1:origins)

{

 # Create a ts object out of the dataset data

 dataset_rolling_train_set <- ts(dataset[1:(dataset_rolling_train_length+i-1)],

                   frequency=frequency(dataset),

                   start=start(dataset))
```

```
  # Write down the holdout values from the test set

  dataset_rolling_holdout_Arima113_Seasoanl212[i,] <- dataset_rolling_test[i-1+(1:H)]


  # Produce forecasts and write them down

  dataset_rolling_forecasts_Arima113_Seasoanl212[i,] <-
forecast(arima(dataset_rolling_train_set, order=c(1,1,3),  seasonal=c(2,1,2)),h=H)$mean


}


## MAPE for Rolling origin of Arima113_Seasoanl212

Rolling_errors_dataset_Arima113_Seasoanl212<-
dataset_rolling_holdout_Arima113_Seasoanl212 -
dataset_rolling_forecasts_Arima113_Seasoanl212

Rolling_ME_dataset_Arima113_Seasoanl212<-
mean(Rolling_errors_dataset_Arima113_Seasoanl212) #Mean error

Rolling_MSE_dataset_Arima113_Seasoanl212<-
mean(Rolling_errors_dataset_Arima113_Seasoanl212^2) #Mean squared error

Rolling_MAE_dataset_Arima113_Seasoanl212<-
mean(abs(Rolling_errors_dataset_Arima113_Seasoanl212)) #Mean absolute error

Rolling_MAPE_dataset_Arima113_Seasoanl212<- 100 *
mean(abs(Rolling_errors_dataset_Arima113_Seasoanl212)/dataset_rolling_holdout_Arima
113_Seasoanl212)

Rolling_RMSE_dataset_Arima113_Seasoanl212<-
sqrt(mean(Rolling_errors_dataset_Arima113_Seasoanl212^2)) #Root mean squared error


Rolling_MAPE_dataset_Arima113_Seasoanl212

Rolling_RMSE_dataset_Arima113_Seasoanl212

## Seasonal ARIMA Model (3,0,2)(1,1,2) -------------------------------------


Arima302_Seasoanl112_dataset<- Arima(dataset_train, order=c(3,0,2),  seasonal=c(1,1,2))
```

```
Arima302_Seasoanl112_dataset # 2947.62   4382.65

checkresiduals(Arima302_Seasoanl112_dataset)

tsdisplay(residuals(Arima302_Seasoanl112_dataset)) # Failed in Residuals plot

FC_Arima302_Seasoanl112_dataset <- forecast(Arima302_Seasoanl112_dataset, h=h)

plot(FC_Arima302_Seasoanl112_dataset)

Arima302_Seasoanl112_dataset_errors <- dataset_test - FC_Arima302_Seasoanl112_dataset

Arima302_Seasoanl112_dataset_ME <- mean(Arima302_Seasoanl112_dataset_errors)
#Mean error

Arima302_Seasoanl112_dataset_MSE <- mean(Arima302_Seasoanl112_dataset_errors^2)
#Mean squared error

Arima302_Seasoanl112_dataset_MAE <-
mean(abs(Arima302_Seasoanl112_dataset_errors)) #Mean absolute error

Arima302_Seasoanl112_dataset_MAPE <- 100 *
mean(abs(Arima302_Seasoanl112_dataset_errors)/dataset_test)

Arima302_Seasoanl112_dataset_RMSE <-
sqrt(mean(Arima302_Seasoanl112_dataset_errors^2)) # Root mean squared error

Arima302_Seasoanl112_dataset_MAPE

Arima302_Seasoanl112_dataset_RMSE

## Rolling origin for Arima302_Seasoanl112

dataset_rolling_forecasts_Arima302_Seasoanl112 <- matrix(NA, nrow=origins, ncol=H)

dataset_rolling_holdout_Arima302_Seasoanl112 <- matrix(NA, nrow=origins, ncol=H)

colnames(dataset_rolling_forecasts_Arima302_Seasoanl112) <- paste0("horizon",c(1:H))
```

```
rownames(dataset_rolling_forecasts_Arima302_Seasoanl112) <-
paste0("origin",c(1:origins))

dimnames(dataset_rolling_holdout_Arima302_Seasoanl112) <-
dimnames(dataset_rolling_forecasts_Arima302_Seasoanl112)

for(i in 1:origins)

{

 # Create a ts object out of the dataset data

 dataset_rolling_train_set <- ts(dataset[1:(dataset_rolling_train_length+i-1)],

                   frequency=frequency(dataset),

                   start=start(dataset))


 # Write down the holdout values from the test set

 dataset_rolling_holdout_Arima302_Seasoanl112[i,] <- dataset_rolling_test[i-1+(1:H)]


 # Produce forecasts and write them down

 dataset_rolling_forecasts_Arima302_Seasoanl112[i,] <-
forecast(arima(dataset_rolling_train_set, order=c(3,0,2),  seasonal=c(1,1,2)),h=H)$mean

}


## MAPE for Rolling origin of Arima302_Seasoanl112

Rolling_errors_dataset_Arima302_Seasoanl112<-
dataset_rolling_holdout_Arima302_Seasoanl112 -
dataset_rolling_forecasts_Arima302_Seasoanl112

Rolling_ME_dataset_Arima302_Seasoanl112<-
mean(Rolling_errors_dataset_Arima302_Seasoanl112) #Mean error

Rolling_MSE_dataset_Arima302_Seasoanl112<-
mean(Rolling_errors_dataset_Arima302_Seasoanl112^2) #Mean squared error

Rolling_MAE_dataset_Arima302_Seasoanl112<-
mean(abs(Rolling_errors_dataset_Arima302_Seasoanl112)) #Mean absolute error

Rolling_MAPE_dataset_Arima302_Seasoanl112<- 100 *
mean(abs(Rolling_errors_dataset_Arima302_Seasoanl112)/dataset_rolling_holdout_Arima
302_Seasoanl112)
```

```
Rolling_RMSE_dataset_Arima302_Seasoanl112<-
sqrt(mean(Rolling_errors_dataset_Arima302_Seasoanl112^2)) #Root mean squared error


Rolling_MAPE_dataset_Arima302_Seasoanl112

Rolling_RMSE_dataset_Arima302_Seasoanl112



# Forecast using Best ARIMA Model ----------------------------------------

Arima302_Seasoanl112_dataset<- Arima(dataset, order=c(3,0,2),  seasonal=c(1,1,2))


FC_Arima302_Seasoanl112_dataset <- forecast(Arima302_Seasoanl112_dataset,
h=h)$mean


FC_Arima302_Seasoanl112_dataset


## Seasonal ARIMA Model (4,0,2)(3,1,1) ------------------------------------


Arima402_Seasoanl311_dataset<- Arima(dataset_train, order=c(4,0,2),  seasonal=c(3,1,1))


Arima402_Seasoanl311_dataset


checkresiduals(Arima402_Seasoanl311_dataset)


tsdisplay(residuals(Arima402_Seasoanl311_dataset))



FC_Arima402_Seasoanl311_dataset <- forecast(Arima402_Seasoanl311_dataset,
h=h)$mean
```

```
Arima402_Seasoanl311_dataset_errors <- dataset_test - FC_Arima402_Seasoanl311_dataset

Arima402_Seasoanl311_dataset_ME <- mean(Arima402_Seasoanl311_dataset_errors)
#Mean error

Arima402_Seasoanl311_dataset_MSE <- mean(Arima402_Seasoanl311_dataset_errors^2)
#Mean squared error

Arima402_Seasoanl311_dataset_MAE <-
mean(abs(Arima402_Seasoanl311_dataset_errors)) #Mean absolute error

Arima402_Seasoanl311_dataset_MAPE <- 100 *
mean(abs(Arima402_Seasoanl311_dataset_errors)/dataset_test)

Arima402_Seasoanl311_dataset_RMSE <-
sqrt(mean(Arima402_Seasoanl311_dataset_errors^2)) # Root mean squared error


Arima402_Seasoanl311_dataset_MAPE

Arima402_Seasoanl311_dataset_RMSE

## Rolling origin for Arima402_Seasoanl311


dataset_rolling_forecasts_Arima402_Seasoanl311 <- matrix(NA, nrow=origins, ncol=H)

dataset_rolling_holdout_Arima402_Seasoanl311 <- matrix(NA, nrow=origins, ncol=H)

colnames(dataset_rolling_forecasts_Arima402_Seasoanl311) <- paste0("horizon",c(1:H))

rownames(dataset_rolling_forecasts_Arima402_Seasoanl311) <-
paste0("origin",c(1:origins))

dimnames(dataset_rolling_holdout_Arima402_Seasoanl311) <-
dimnames(dataset_rolling_forecasts_Arima402_Seasoanl311)

for(i in 1:origins)

{

  # Create a ts object out of the dataset data

  dataset_rolling_train_set <- ts(dataset[1:(dataset_rolling_train_length+i-1)],

                          frequency=frequency(dataset),

                          start=start(dataset))
```

```
  # Write down the holdout values from the test set

  dataset_rolling_holdout_Arima402_Seasoanl311[i,] <- dataset_rolling_test[i-1+(1:H)]


  # Produce forecasts and write them down

  dataset_rolling_forecasts_Arima402_Seasoanl311[i,] <-
forecast(arima(dataset_rolling_train_set, order=c(4,0,2),  seasonal=c(3,1,1)),h=H)$mean


}


## MAPE for Rolling origin of Arima402_Seasoanl311

Rolling_errors_dataset_Arima402_Seasoanl311<-
dataset_rolling_holdout_Arima402_Seasoanl311 -
dataset_rolling_forecasts_Arima402_Seasoanl311

Rolling_ME_dataset_Arima402_Seasoanl311<-
mean(Rolling_errors_dataset_Arima402_Seasoanl311) #Mean error

Rolling_MSE_dataset_Arima402_Seasoanl311<-
mean(Rolling_errors_dataset_Arima402_Seasoanl311^2) #Mean squared error

Rolling_MAE_dataset_Arima402_Seasoanl311<-
mean(abs(Rolling_errors_dataset_Arima402_Seasoanl311)) #Mean absolute error

Rolling_MAPE_dataset_Arima402_Seasoanl311<- 100 *
mean(abs(Rolling_errors_dataset_Arima402_Seasoanl311)/dataset_rolling_holdout_Arima
402_Seasoanl311)

Rolling_RMSE_dataset_Arima402_Seasoanl311<-
sqrt(mean(Rolling_errors_dataset_Arima402_Seasoanl311^2)) #Root mean squared error


Rolling_MAPE_dataset_Arima402_Seasoanl311

Rolling_RMSE_dataset_Arima402_Seasoanl311


#  Final recommendation of ARIMA Model ------------------------------------
```

```
# Best ARIMA Model

#Fitting the SARIMA model

Sarima_dataset <- Arima(dataset_train, order=c(3,0,2), seasonal=c(1,1,2))

tsdisplay(residuals(Sarima_dataset)) #The spikes become insignificant

Sarima_dataset

checkresiduals(Sarima_dataset)

Forecast_Sarima_dataset <-forecast(Sarima_dataset, h = h)

Forecast_Sarima_dataset

plot(Forecast_Sarima_dataset)

#Error Measures

Sarima_errors_dataset = dataset_test - forecast(Sarima_dataset, h = h)$mean

Sarima_ME_dataset = mean(Sarima_errors_dataset) #Mean error

Sarima_MSE_dataset<- mean(Sarima_errors_dataset^2) #Mean squared error

Sarima_MAE_dataset<- mean(abs(Sarima_errors_dataset)) #Mean absolute error

Sarima_MAPE_dataset<- 100 * mean(abs(Sarima_errors_dataset)/dataset_test) #Mean
absolute percentage error

Sarima_RMSE_dataset<- sqrt(mean(Sarima_errors_dataset^2)) #Root mean squared error



## Rolling origin for Sarima


dataset_rolling_forecasts_Sarima <- matrix(NA, nrow=origins, ncol=H)

dataset_rolling_holdout_Sarima <- matrix(NA, nrow=origins, ncol=H)

colnames(dataset_rolling_forecasts_Sarima) <- paste0("horizon",c(1:H))

rownames(dataset_rolling_forecasts_Sarima) <- paste0("origin",c(1:origins))

dimnames(dataset_rolling_holdout_Sarima) <-
dimnames(dataset_rolling_forecasts_Sarima)

for(i in 1:origins)

{
```

```
 # Create a ts object out of the dataset data
 dataset_rolling_train_set <- ts(dataset[1:(dataset_rolling_train_length+i-1)],
                        frequency=frequency(dataset),
                        start=start(dataset))

 # Write down the holdout values from the test set
 #dataset_rolling_holdout_ES_ANA[i,] <- dataset_rolling_test[i-1+(1:h)]
 dataset_rolling_holdout_Sarima[i,] <- dataset_rolling_test[i-1+(1:H)]

 # Produce forecasts and write them down
 dataset_rolling_forecasts_Sarima[i,] <- forecast(arima(dataset_rolling_train_set,
order=c(0,1,2), seasonal=c(0,1,1)),h=H)$mean
}

## MAPE for Rolling origin of Sarima
Rolling_errors_dataset_Sarima <- dataset_rolling_holdout_Sarima -
dataset_rolling_forecasts_Sarima
Rolling_ME_dataset_Sarima<- mean(Rolling_errors_dataset_Sarima) #Mean error
Rolling_MSE_dataset_Sarima<- mean(Rolling_errors_dataset_Sarima^2) #Mean squared
error
Rolling_MAE_dataset_Sarima<- mean(abs(Rolling_errors_dataset_Sarima)) #Mean absolute
error
Rolling_MAPE_dataset_Sarima<- 100 *
mean(abs(Rolling_errors_dataset_Sarima)/dataset_rolling_holdout_Sarima)
Rolling_RMSE_dataset_Sarima<- sqrt(mean(Rolling_errors_dataset_Sarima^2)) #Root mean
squared error


# Auto ARIMA Model -------------------------------------------------------
```

```
# Try Auto ARIMA Model

auto_fit_dataset <- auto.arima(dataset_train,  trace = TRUE)

auto_fit_dataset

Forecast_auto_fit_dataset <- forecast(auto_fit_dataset, h = h)

plot(Forecast_auto_fit_dataset)

checkresiduals(auto_fit_dataset)

#Error measures for auto arima

auto_errors_dataset = dataset_test - Forecast_auto_fit_dataset

auto_ME_dataset <- mean(auto_errors_dataset) #Mean error

auto_MSE_dataset <- mean(auto_errors_dataset^2) #Mean squared error

auto_MAE_dataset <- mean(abs(auto_errors_dataset)) #Mean absolute error

auto_MAPE_dataset <- 100 * mean(abs(auto_errors_dataset)/dataset_test) #Mean absolute
percentage error

auto_RMSE_dataset <- sqrt(mean(auto_errors_dataset^2)) #Root mean squared error


auto_MAPE_dataset

auto_RMSE_dataset


## Rolling origin for Auto Arima


dataset_rolling_forecasts_AutoArima <- matrix(NA, nrow=origins, ncol=H)

dataset_rolling_holdout_AutoArima <- matrix(NA, nrow=origins, ncol=H)

colnames(dataset_rolling_forecasts_AutoArima) <- paste0("horizon",c(1:H))

rownames(dataset_rolling_forecasts_AutoArima) <- paste0("origin",c(1:origins))

dimnames(dataset_rolling_holdout_AutoArima) <-
dimnames(dataset_rolling_forecasts_AutoArima)

for(i in 1:origins)

{
```

```
  # Create a ts object out of the dataset data
  dataset_rolling_train_set <- ts(dataset[1:(dataset_rolling_train_length+i-1)],
                                  frequency=frequency(dataset),
                                  start=start(dataset))

  # Write down the holdout values from the test set
  #dataset_rolling_holdout_ES_ANA[i,] <- dataset_rolling_test[i-1+(1:h)]
  dataset_rolling_holdout_AutoArima[i,] <- dataset_rolling_test[i-1+(1:H)]

  # Produce forecasts and write them down
  dataset_rolling_forecasts_AutoArima[i,] <-
forecast(auto.arima(dataset_rolling_train_set),h=H)$mean
}

## MAPE for Rolling origin of Auto Arima
Rolling_errors_dataset_AutoArima<- dataset_rolling_holdout_AutoArima -
dataset_rolling_forecasts_AutoArima
Rolling_ME_dataset_AutoArima<- mean(Rolling_errors_dataset_AutoArima) #Mean error
Rolling_MSE_dataset_AutoArima<- mean(Rolling_errors_dataset_AutoArima^2) #Mean
squared error
Rolling_MAE_dataset_AutoArima<- mean(abs(Rolling_errors_dataset_AutoArima)) #Mean
absolute error
Rolling_MAPE_dataset_AutoArima<- 100 *
mean(abs(Rolling_errors_dataset_AutoArima)/dataset_rolling_holdout_AutoArima)
Rolling_RMSE_dataset_AutoArima<- sqrt(mean(Rolling_errors_dataset_AutoArima^2))
#Root mean squared error

Rolling_MAPE_dataset_AutoArima
Rolling_RMSE_dataset_AutoArima
# Auto Seasonal ARIMA Model ----------------------------------------------
```

```
library("smooth")


# Try Auto Sarima Model

autosarima_fit_dataset <- auto.ssarima(dataset_train,  stepwise=FALSE ,
approximation=FALSE)

autosarima_fit_dataset

Forecast_autosarima_fit_dataset <- forecast(autosarima_fit_dataset, h = h)$mean

plot(Forecast_autosarima_fit_dataset)

#Error measures for auto arima

autosarima_errors_dataset = dataset_test - Forecast_autosarima_fit_dataset

autosarima_ME_dataset <- mean(autosarima_errors_dataset) #Mean error

autosarima_MSE_dataset <- mean(autosarima_errors_dataset^2) #Mean squared error

autosarima_MAE_dataset <- mean(abs(autosarima_errors_dataset)) #Mean absolute error

autosarima_MAPE_dataset <- 100 * mean(abs(autosarima_errors_dataset)/dataset_test)
#Mean absolute percentage error

autosarima_RMSE_dataset <- sqrt(mean(autosarima_errors_dataset^2)) #Root mean
squared error


#detach("package:smooth", unload = TRUE)


## Rolling origin for Auto Seasonal Arima


dataset_rolling_forecasts_AutoSArima <- matrix(NA, nrow=origins, ncol=H)

dataset_rolling_holdout_AutoSSarima <- matrix(NA, nrow=origins, ncol=H)

colnames(dataset_rolling_forecasts_AutoSArima) <- paste0("horizon",c(1:H))

rownames(dataset_rolling_forecasts_AutoSArima) <- paste0("origin",c(1:origins))

dimnames(dataset_rolling_holdout_AutoSSarima) <-
dimnames(dataset_rolling_forecasts_AutoSArima)

for(i in 1:origins)
```

```
{
  # Create a ts object out of the dataset data
  dataset_rolling_train_set <- ts(dataset[1:(dataset_rolling_train_length+i-1)],
                               frequency=frequency(dataset),
                               start=start(dataset))

  # Write down the holdout values from the test set
  dataset_rolling_holdout_AutoSSarima[i,] <- dataset_rolling_test[i-1+(1:H)]

  # Produce forecasts and write them down
  dataset_rolling_forecasts_AutoSArima[i,] <-
forecast(auto.ssarima(dataset_rolling_train_set),h=H)$mean

}

## MAPE for Rolling origin of Auto Seasonal Arima
Rolling_errors_dataset_AutoSSarima<- dataset_rolling_holdout_AutoSSarima -
dataset_rolling_forecasts_AutoSArima
Rolling_ME_dataset_AutoSSarima<- mean(Rolling_errors_dataset_AutoSSarima) #Mean
error
Rolling_MSE_dataset_AutoSSarima<- mean(Rolling_errors_dataset_AutoSSarima^2) #Mean
squared error
Rolling_MAE_dataset_AutoSSarima<- mean(abs(Rolling_errors_dataset_AutoSSarima))
#Mean absolute error
Rolling_MAPE_dataset_AutoSSarima<- 100 *
mean(abs(Rolling_errors_dataset_AutoSSarima)/dataset_rolling_holdout_AutoSSarima)
Rolling_RMSE_dataset_AutoSSarima<- sqrt(mean(Rolling_errors_dataset_AutoSSarima^2))
#Root mean squared error

# Neural Network -----------------------------------------------------------
```

```
#Neural Network Auto regression
ann1 <-nnetar(dataset_train)

# Check the summary
summary(ann1)

# Check the ACF/PACF plot of residuals
tsdisplay(residuals(ann1))

# Forecast using NN
accnfcst<-forecast(ann1,h=h)
accnfcst
# Plot the forecast
autoplot(accnfcst)

#We create a simulation matrix to support 9 different outputs.
sim <- ts(matrix(0, nrow=12L, ncol=9L),start = end(dataset_train)[1L]+1L, frequency = 7)
#Simulate 9 possible future sample paths using bootstrapping. You will get a warning
related to the
#column names, just ignore it:
for(i in seq(9))
  sim[,i] <- simulate(ann1, nsim=12)

autoplot(dataset_train) + autolayer(sim)

fcast <- forecast(ann1, PI=TRUE, h=14)
autoplot(fcast)
```

```
#Error measures for NN

NN_errors_dataset = dataset_test - (accnfcst$mean)

NN_ME_dataset <- mean(NN_errors_dataset) #Mean error

NN_MSE_dataset <- mean(NN_errors_dataset^2) #Mean squared error

NN_MAE_dataset <- mean(abs(NN_errors_dataset)) #Mean absolute error

NN_MAPE_dataset <- 100 * mean(abs(NN_errors_dataset)/dataset_test) #Mean absolute
percentage error

NN_RMSE_dataset <- sqrt(mean(NN_errors_dataset^2)) #Root mean square


# Simple Regression ------------------------------------------------------------

#Import sdata

dataset <- read_excel("Assignment 1 Data.xls")


#colnames(data) <- c("Date","Transactions")

colnames(dataset) <- c("Transactions")


#Converting data to time series

dataset <- ts(dataset, frequency = 365, start = c(1996,77))


# Approach 1 -Replace with corresponding day of week data

which_na(dataset)


# Approach 2 - Replace with interpolation method

dataset <- na_interpolation(dataset)



# Find the outliers in the series

out <- boxplot.stats(dataset)$out
```

```
out

out_ind <- which(dataset %in% c(out))

out_ind

dataset[c(163,167,196,235,265,283,538,557,558,559,561,562,563,565,586,587,592,593,594,600
        ,621,628,649,670,684,691,698,711,712,726)]


#Replacing the outliers with median values

series_median = median(dataset)

series_median

dataset[c(163,167,196,235,265,283,538,557,558,559,561,562,563,565,586,587,592,593,594,600
        ,621,628,649,670,684,691,698,711,712,726)] = series_median


# Split the data into train and test sets

dataset_train <- window(dataset, start(dataset), (1998+66/365))

dataset_train

# Split the data into train and test sets

dataset_test <- window(dataset, (1998+67/365), end(dataset))

dataset_test


# Fit Simple Regression

Simple_Regression <- lm(Transactions ~ 1  , data=dataset_train)

# Summary

summary(Simple_Regression)

#Extract Residuals

Simple_Regression_residuals <- residuals(Simple_Regression)

#We will also need fitted values for our analysis, which can be extracted using fitted():

#Extract Residuals
```

```
Simple_Regression_fitted <- fitted(Simple_Regression)
#Plot Histogram
hist(Simple_Regression_residuals)
#QQ-Plot
qqnorm(Simple_Regression_residuals)
qqline(Simple_Regression_residuals)
#Jarque-Bera test
jarque.bera.test(Simple_Regression_residuals)
#Shapiro-Wilk test
shapiro.test(Simple_Regression_residuals)
#Kolmogorov-Smirnov test
ks.test(Simple_Regression_residuals,y="rnorm")
#Plot Residuals against Fitted Values
plot(Simple_Regression_fitted, Simple_Regression_residuals)
#Plot Residuals against Fitted Values
plot(Simple_Regression_fitted, Simple_Regression_residuals^2)
#ACF and PACF of the residuals
tsdisplay(Simple_Regression_residuals)

# Create Studentised Residuals
Simple_Regression_st <- rstandard(Simple_Regression)
# Plot the Residuals
plot(Simple_Regression_st)
# Draw two horizontal lines at 2 and -2 in red
abline(h=c(-2,2),col="red")

#Forecast from Simple Regression
Forecast_Simple_Regression <- predict(Simple_Regression, (as.data.frame(dataset_test)))
```

Forecast_Simple_Regression

#Error measures for Simple Regression

Simple_Regression_errors_dataset = dataset_test - Forecast_Simple_Regression

Simple_Regression_ME_dataset <- mean(Simple_Regression_errors_dataset) #Mean error

Simple_Regression_MSE_dataset <- mean(Simple_Regression_errors_dataset^2) #Mean squared error

Simple_Regression_MAE_dataset <- mean(abs(Simple_Regression_errors_dataset)) #Mean absolute error

Simple_Regression_MAPE_dataset <- 100 * mean(abs(Simple_Regression_errors_dataset)/dataset_test) #Mean absolute percentage error

Simple_Regression_RMSE_dataset <- sqrt(mean(Simple_Regression_errors_dataset^2)) #Root mean squared error

Simple_Regression_errors_dataset

Simple_Regression_MAPE_dataset

Multiple Regression Model Parameters---

library("greybox")

Add dummy variables for lag and seasonal variables

Auto regressive model with only lag - seasonal variable

The second one assumes that the

#seasonality has a stochastic structure (implying that it may change over time) and uses lagged

```
#variables.

#Lags of dataset

L1_dataset <- lag((as.vector(dataset)),k=1)
L1_dataset
L2_dataset <- lag((as.vector(L1_dataset)),k=1)
L2_dataset
L3_dataset <- lag((as.vector(L2_dataset)),k=1)
L3_dataset
L4_dataset <- lag((as.vector(L3_dataset)),k=1)
L4_dataset
L5_dataset <- lag((as.vector(L4_dataset)),k=1)
L5_dataset
L6_dataset <- lag((as.vector(L5_dataset)),k=1)
L6_dataset
L7_dataset <- lag((as.vector(L6_dataset)),k=1)
L7_dataset
L8_dataset <- lag((as.vector(L7_dataset)),k=1)
L8_dataset
L9_dataset <- lag((as.vector(L8_dataset)),k=1)
L9_dataset
L10_dataset <- lag((as.vector(L9_dataset)),k=1)
L10_dataset
L11_dataset <- lag((as.vector(L10_dataset)),k=1)
L11_dataset
L12_dataset <- lag((as.vector(L11_dataset)),k=1)
L12_dataset
```

```
L13_dataset <- lag((as.vector(L12_dataset)),k=1)

L13_dataset

L14_dataset <- lag((as.vector(L13_dataset)),k=1)

L14_dataset



#Add all Lags to the Data

dataset_colnames <- colnames(dataset)

dataset <- cbind(dataset, L1_dataset, L2_dataset, L3_dataset, L4_dataset, L5_dataset,
L6_dataset, L7_dataset, L8_dataset, L9_dataset, L10_dataset, L11_dataset, L12_dataset,
L13_dataset, L14_dataset)


# Change the column names

colnames(dataset) <- c("Transactions", "L1_dataset", "L2_dataset", "L3_dataset",
"L4_dataset", "L5_dataset", "L6_dataset", "L7_dataset", "L8_dataset", "L9_dataset",
"L10_dataset", "L11_dataset", "L12_dataset", "L13_dataset", "L14_dataset")


# Add the seasonal dummies


#Create Seasonal Dummies

Mon <- rep(c(1,0,0,0,0,0,0),105)

Tue <- rep(c(0,1,0,0,0,0,0),105)

Wed <- rep(c(0,0,1,0,0,0,0),105)

Thu <- rep(c(0,0,0,1,0,0,0),105)

Fri <- rep(c(0,0,0,0,1,0,0),105)

Sat <- rep(c(0,0,0,0,0,1,0),105)

Sun <- rep(c(0,0,0,0,0,0,1),105)


# Add the seasonal dummies to the dataset

dataset <- cbind(dataset,Mon,Tue,Wed,Thu,Fri, Sat, Sun)
```

```
# Change the column names

colnames(dataset) <- c("Transactions", "L1_dataset", "L2_dataset", "L3_dataset",
"L4_dataset", "L5_dataset", "L6_dataset", "L7_dataset", "L8_dataset", "L9_dataset",
"L10_dataset", "L11_dataset", "L12_dataset", "L13_dataset",
"L14_dataset","Mon","Tue","Wed","Thu", "Fri", "Sat", "Sun")


# Add the trend dummy


## Create Trend

dataset_trend <- c(1:735)


#Add trebd to the Data

data_colnames <- colnames(dataset)

dataset <- cbind(dataset, dataset_trend)


# Change the column names

colnames(dataset) <- c("Transactions", "L1_dataset", "L2_dataset", "L3_dataset",
"L4_dataset", "L5_dataset", "L6_dataset", "L7_dataset", "L8_dataset", "L9_dataset",
"L10_dataset", "L11_dataset", "L12_dataset", "L13_dataset",
"L14_dataset","Mon","Tue","Wed","Thu", "Fri", "Sat", "Sun", "dataset_trend")


# Split the data into train and test sets

dataset_train <- window(dataset, start(dataset), (1998+66/365))

dataset_train

# Split the data into train and test sets

dataset_test <- window(dataset, (1998+67/365), end(dataset))

dataset_test


# Regression model with lags Model -------------------------------------------------------
```

```
# Model

lags_model <- lm(Transactions ~   L1_dataset  + L2_dataset + L3_dataset+ L4_dataset +
L5_dataset + L6_dataset +  L7_dataset + L8_dataset + L9_dataset + L10_dataset +
L11_dataset + L12_dataset + L13_dataset, data=dataset_train)


# L1_dataset + L2_dataset + L4_dataset + L7_dataset

summary(lags_model)


tsdisplay(residuals(lags_model))


# Check for multi collinearity

#VIF for fit4

VIF(lags_model)


#Extract Residuals

lags_model_residuals <- residuals(lags_model)

#We will also need fitted values for our analysis, which can be extracted using fitted():

#Extract Residuals

lags_model_fitted <- fitted(lags_model)

#Plot Histogram

hist(lags_model_residuals)

#QQ-Plot

qqnorm(lags_model_residuals)

qqline(lags_model_residuals)

#Jarque-Bera test

jarque.bera.test(lags_model_residuals)

#Shapiro-Wilk test

shapiro.test(lags_model_residuals)
```

```
#Kolmogorov-Smirnov test
ks.test(lags_model_residuals,y="rnorm")
#Plot Residuals against Fitted Values
plot(lags_model_fitted, lags_model_residuals)
#Plot Residuals against Fitted Values
plot(lags_model_fitted, lags_model_residuals^2)
#ACF and PACF of the residuals
tsdisplay(lags_model_residuals)

#Forecast from lags_model
Forecast_lags_model <- predict(lags_model, (as.data.frame(dataset_test)))
plot(Forecast_lags_model)
Forecast_lags_model

#Error measures for lags_model
lags_model_errors_dataset = dataset_test - Forecast_lags_model
lags_model_ME_dataset <- mean(lags_model_errors_dataset) #Mean error
lags_model_MSE_dataset <- mean(lags_model_errors_dataset^2) #Mean squared error
lags_model_MAE_dataset <- mean(abs(lags_model_errors_dataset)) #Mean absolute error
lags_model_MAPE_dataset <- 100 * mean(abs(lags_model_errors_dataset)/dataset_test)
#Mean absolute percentage error
lags_model_RMSE_dataset <- sqrt(mean(lags_model_errors_dataset^2)) #Root mean
squared error


# Regression model with seasonal dummies ---------------------------------
```

when you consider the whole dataset in below lm function as at first index and you apply seasonal dummies to whole dataset, data here is dataset_train.after adding this you need to split the dataset into train and test and then you will get the result.

```
#Use the Seasonal Dummies

seasonaldummies <- lm(Transactions ~  Mon  + Tue +  Wed + Thu + Fri +  Sat ,
data=dataset_train)

summary(seasonaldummies)

dataset_train


#Extract Residuals

seasonaldummies_residuals <- residuals(seasonaldummies)

#We will also need fitted values for our analysis, which can be extracted using fitted():

#Extract Residuals

seasonaldummies_fitted <- fitted(seasonaldummies)

#Plot Histogram

hist(seasonaldummies_residuals)

#QQ-Plot

qqnorm(seasonaldummies_residuals)

qqline(seasonaldummies_residuals)

#Jarque-Bera test

jarque.bera.test(seasonaldummies_residuals)

#Shapiro-Wilk test

shapiro.test(seasonaldummies_residuals)

#Kolmogorov-Smirnov test

ks.test(seasonaldummies_residuals,y="rnorm")

#Plot Residuals against Fitted Values

plot(seasonaldummies_fitted, seasonaldummies_residuals)

#Plot Residuals against Fitted Values
```

```
plot(seasonaldummies_fitted, seasonaldummies_residuals^2)

#ACF and PACF of the residuals

tsdisplay(seasonaldummies_residuals)


#Forecast from seasonal dummies

Forecast_seasonaldummies <- predict(seasonaldummies, dataset_test)

plot(Forecast_seasonaldummies)

Forecast_seasonaldummies



#Error measures for seasonal dummies

seasonaldummies_errors_dataset = dataset_test - Forecast_seasonaldummies$mean

seasonaldummies_ME_dataset <- mean(seasonaldummies_errors_dataset) #Mean error

seasonaldummies_MSE_dataset <- mean(seasonaldummies_errors_dataset^2) #Mean
squared error

seasonaldummies_MAE_dataset <- mean(abs(seasonaldummies_errors_dataset)) #Mean
absolute error

seasonaldummies_MAPE_dataset <- 100 *
mean(abs(seasonaldummies_errors_dataset)/dataset_test) #Mean absolute percentage
error

seasonaldummies_RMSE_dataset <- sqrt(mean(seasonaldummies_errors_dataset^2))
#Root mean squared error


seasonaldummies_errors_dataset

seasonaldummies_MAPE_dataset


# Auto regressive model with lag , seasonal dummies variable --------------


Lag_seasonal <- lm(Transactions ~ L1_dataset + L2_dataset  + L4_dataset  + L7_dataset +
Mon + Tue + Wed  + Thu + Fri + Sat , data=dataset_train)
```

```
summary(Lag_seasonal)

tsdisplay(residuals(Lag_seasonal))

# Check for multi collinearity
#VIF for fit4

VIF(Lag_seasonal)

#Extract Residuals
Lag_seasonal_residuals <- residuals(Lag_seasonal)
#We will also need fitted values for our analysis, which can be extracted using fitted():
#Extract Residuals
Lag_seasonal_fitted <- fitted(Lag_seasonal)
#Plot Histogram
hist(Lag_seasonal_residuals)
#QQ-Plot
qqnorm(Lag_seasonal_residuals)
qqline(Lag_seasonal_residuals)
#Jarque-Bera test
jarque.bera.test(Lag_seasonal_residuals)
#Shapiro-Wilk test
shapiro.test(Lag_seasonal_residuals)
#Kolmogorov-Smirnov test
ks.test(Lag_seasonal_residuals,y="rnorm")
#Plot Residuals against Fitted Values
plot(Lag_seasonal_fitted, Lag_seasonal_residuals)
```

```
#Plot Residuals against Fitted Values
plot(Lag_seasonal_fitted, Lag_seasonal_residuals^2)
#ACF and PACF of the residuals
tsdisplay(Lag_seasonal_residuals)


#Forecast from Lag Seasonal
Forecast_Lag_seasonal <- predict(Lag_seasonal, (as.data.frame(dataset_test)))
plot(Forecast_Lag_seasonal)
Forecast_Lag_seasonal


#Error measures for Lag Seasonal
Lag_seasonal_errors_dataset = dataset_test - Forecast_Lag_seasonal
Lag_seasonal_ME_dataset <- mean(Lag_seasonal_errors_dataset) #Mean error
Lag_seasonal_MSE_dataset <- mean(Lag_seasonal_errors_dataset^2) #Mean squared error
Lag_seasonal_MAE_dataset <- mean(abs(Lag_seasonal_errors_dataset)) #Mean absolute
error
Lag_seasonal_MAPE_dataset <- 100 * mean(abs(Lag_seasonal_errors_dataset)/dataset_test)
#Mean absolute percentage error
Lag_seasonal_RMSE_dataset <- sqrt(mean(Lag_seasonal_errors_dataset^2)) #Root mean
squared error


Lag_seasonal_errors_dataset
Lag_seasonal_MAPE_dataset


# Auto regressive model with lag , seasonal dummies and trend variables --------


Lag_seasonal_trend <- lm(Transactions ~ L1_dataset + L2_dataset + L4_dataset +
L7_dataset +   Wed  + Thu + Fri + Sat + dataset_trend , data=dataset_train)


summary(Lag_seasonal_trend)
```

```
tsdisplay(residuals(Lag_seasonal_trend))

# Check for multi collinearity
#VIF for fit4

VIF(Lag_seasonal_trend)

#Extract Residuals
Lag_seasonal_trend_residuals <- residuals(Lag_seasonal_trend)
#We will also need fitted values for our analysis, which can be extracted using fitted():
#Extract Residuals
Lag_seasonal_trend_fitted <- fitted(Lag_seasonal_trend)
#Plot Histogram
hist(Lag_seasonal_trend_residuals)
#QQ-Plot
qqnorm(Lag_seasonal_trend_residuals)
qqline(Lag_seasonal_trend_residuals)
#Jarque-Bera test
jarque.bera.test(Lag_seasonal_trend_residuals)
#Shapiro-Wilk test
shapiro.test(Lag_seasonal_trend_residuals)
#Kolmogorov-Smirnov test
ks.test(Lag_seasonal_trend_residuals,y="rnorm")
#Plot Residuals against Fitted Values
plot(Lag_seasonal_trend_fitted, Lag_seasonal_trend_residuals)
#Plot Residuals against Fitted Values
plot(Lag_seasonal_trend_fitted, Lag_seasonal_trend_residuals^2)
```

```
#ACF and PACF of the residuals

tsdisplay(Lag_seasonal_trend_residuals)




#Forecast from Lag Seasonal Trend

Forecast_Lag_seasonal_trend <- predict(Lag_seasonal_trend, (as.data.frame(dataset_test)))


plot(Forecast_Lag_seasonal_trend)

Forecast_Lag_seasonal_trend


#Error measures for Lag Seasonal Trend

Lag_seasonal_trend_errors_dataset = dataset_test - Forecast_Lag_seasonal_trend

Lag_seasonal_trend_ME_dataset <- mean(Lag_seasonal_trend_errors_dataset) #Mean error

Lag_seasonal_trend_MSE_dataset <- mean(Lag_seasonal_trend_errors_dataset^2) #Mean
squared error

Lag_seasonal_trend_MAE_dataset <- mean(abs(Lag_seasonal_trend_errors_dataset)) #Mean
absolute error

Lag_seasonal_trend_MAPE_dataset <- 100 *
mean(abs(Lag_seasonal_trend_errors_dataset)/dataset_test) #Mean absolute percentage
error

Lag_seasonal_trend_RMSE_dataset <- sqrt(mean(Lag_seasonal_trend_errors_dataset^2))
#Root mean squared error


Lag_seasonal_trend_errors_dataset

Lag_seasonal_trend_MAPE_dataset

Lag_seasonal_trend_RMSE_dataset

Lag_seasonal_trend_RMSE_dataset


# Forecast using best regression -----------------------------------------
```

```
Lag_seasonal_trend <- lm(Transactions ~ L1_dataset + L2_dataset + L4_dataset +
L7_dataset +   Wed  + Thu + Fri + Sat + dataset_trend , data=dataset)

summary(Lag_seasonal_trend)

#Forecast from Lag Seasonal Trend
Forecast_Lag_seasonal_trend <- predict(Lag_seasonal_trend, (as.data.frame(dataset_test)))
plot(Forecast_Lag_seasonal_trend)
Forecast_Lag_seasonal_trend

# Model with AIC both selection -------------------------------------------

all_variable <- lm(Transactions ~ . , data=dataset_train)

AICSelection_directionmodel <- step (all_variable, direction = "both")

summary(AICSelection_directionmodel)

tsdisplay(residuals(AICSelection_directionmodel))

# Check for multi collinearity
#VIF for fit4

VIF(AICSelection_directionmodel)

#Extract Residuals
AICSelection_directionmodel_residuals <- residuals(AICSelection_directionmodel)
```

#We will also need fitted values for our analysis, which can be extracted using fitted():

#Extract Residuals

AICSelection_directionmodel_fitted <- fitted(AICSelection_directionmodel)

#Plot Histogram

hist(AICSelection_directionmodel_residuals)

#QQ-Plot

qqnorm(AICSelection_directionmodel_residuals)

qqline(AICSelection_directionmodel_residuals)

#Jarque-Bera test

jarque.bera.test(AICSelection_directionmodel_residuals)

#Shapiro-Wilk test

shapiro.test(AICSelection_directionmodel_residuals)

#Kolmogorov-Smirnov test

ks.test(AICSelection_directionmodel_residuals,y="rnorm")

#Plot Residuals against Fitted Values

plot(AICSelection_directionmodel_fitted, AICSelection_directionmodel_residuals)

#Plot Residuals against Fitted Values

plot(AICSelection_directionmodel_fitted, AICSelection_directionmodel_residuals^2)

#ACF and PACF of the residuals

tsdisplay(AICSelection_directionmodel_residuals)

Forecast using AIC selection model

Forecast_AICSelection_directionmodel <- predict(AICSelection_directionmodel, dataset_test)

plot(Forecast_AICSelection_directionmodel)

Forecast_AICSelection_directionmodel

```
#Error measures for Lag Seasonal Trend

AICSelection_directionmodel_errors_dataset = dataset_test -
Forecast_AICSelection_directionmodel

AICSelection_directionmodel_ME_dataset <-
mean(AICSelection_directionmodel_errors_dataset) #Mean error

AICSelection_directionmodel_MSE_dataset <-
mean(AICSelection_directionmodel_errors_dataset^2) #Mean squared error

AICSelection_directionmodel_MAE_dataset <-
mean(abs(AICSelection_directionmodel_errors_dataset)) #Mean absolute error

AICSelection_directionmodel_MAPE_dataset <- 100 *
mean(abs(AICSelection_directionmodel_errors_dataset)/dataset_test) #Mean absolute
percentage error

AICSelection_directionmodel_RMSE_dataset <-
sqrt(mean(AICSelection_directionmodel_errors_dataset^2)) #Root mean squared error


AICSelection_directionmodel_errors_dataset

AICSelection_directionmodel_MAPE_dataset

AICSelection_directionmodel_RMSE_dataset

## Regression validation



# Checking Accuracy -----------------------------------------------------


#Arithmetic mean method

accuracy(Forecast_Arithmetic_mean_dataset,dataset_test)

#Check the accuracy for simple moving average

accuracy(Forecast_SMA_dataset,dataset_test)

#Naive method

accuracy(Naive_method_dataset,dataset_test)

#ES method Forecast_ES_ANA_opt_dataset

accuracy(Forecast_ES_ANA_opt_dataset,dataset_test)
```

```
#ES method Forecast_ES_ANA_opt_dataset
accuracy(Forecast_ES_ANA_opt_dataset,dataset_test)
#ES method Forecast_ES_AAA_opt_dataset
accuracy(Forecast_ES_AAA_opt_dataset,dataset_test)

# Accuracy of all ARIMA Models

accuracy(FC_Arima711_Seasoanl013_dataset, dataset_test)
accuracy(FC_Arima202_Seasoanl012_dataset, dataset_test)
accuracy(FC_Arima202_Seasoanl111_dataset, dataset_test)
accuracy(FC_Arima113_Seasoanl212_dataset, dataset_test)
#accuracy(FC_Arima212_Seasoanl102_dataset, dataset_test)
accuracy(FC_Arima302_Seasoanl112_dataset, dataset_test)
accuracy(FC_Arima402_Seasoanl311_dataset, dataset_test)

#Check the accuracy for Sarima model
accuracy(Forecast_Sarima_dataset,dataset_test)
#Check the accuracy for auto arima model
accuracy(Forecast_auto_fit_dataset,dataset_test)
#Check the accuracy for auto arima model
accuracy(Forecast_autosarima_fit_dataset,dataset_test)
#Check the accuracy for NN model
accuracy(accnfcst,dataset_test)

# Creating a summary table for comparison of error measures ---------------

Summary_stats <-
matrix(c(Arithmetic_ME_dataset,Arithmetic_MSE_dataset,Arithmetic_MAE_dataset,Arithmetic_MAPE_dataset,Arithmetic_RMSE_dataset), ncol=5, byrow=TRUE)
```

```
Summary_stats <-
rbind(Summary_stats,c(SMA_ME_dataset,SMA_MSE_dataset,SMA_MAE_dataset,SMA_MAPE_
dataset,SMA_RMSE_dataaset3))

Summary_stats<-
rbind(Summary_stats,c(Naive_ME_dataset,Naive_MSE_dataset,Naive_MAE_dataset,Naive_M
APE_dataset,Naive_RMSE_dataset))

Summary_stats<-
rbind(Summary_stats,c(ES_ANA_ME_dataset,ES_ANA_MSE_dataset,ES_ANA_MAE_dataset,ES
_ANA_RMSE_dataset,ES_ANA_MAPE_dataset))

Summary_stats<-
rbind(Summary_stats,c(ES_AAA_ME_dataset,ES_AAA_MSE_dataset,ES_AAA_MAE_dataset,ES
_AAA_RMSE_dataset,ES_AAA_MAPE_dataset))

Summary_stats<-
rbind(Summary_stats,c(auto_ME_dataset,auto_MSE_dataset,auto_MAE_dataset,auto_MAPE_
dataset,auto_RMSE_dataset))

Summary_stats<-
rbind(Summary_stats,c(Sarima_ME_dataset,Sarima_MSE_dataset,Sarima_MAE_dataset,Sari
ma_MAPE_dataset,Sarima_RMSE_dataset))

colnames(Summary_stats) <- c('ME','MSE','MAE','MAPE','RMSE')

rownames(Summary_stats) <- c('Arithmetic Mean','SMA','Naive approach','ES ANA','ES
AAA','Auto Arima','SARIMA')

Summary_stats <- as.table(Summary_stats)

Summary_stats

View(Summary_stats)
```