Abstract

This project is about timeseries and forecasting. We implement exponential smoothing on real TV viewership data to predict the future values of viewers. We have the data from January 2014 to December 2017 which we separate into test and training set. We train our model from January to 2014 to December 2016 and test our model on 2017 data. We then check the accuracy of our model using mean absolute error between real and predicted values.

Keywords: [Timeseries, forecast, exponential smoothing, prediction, TV viewership, MAE]

[Title Here, up to 12 Words, on One to Two Lines]

We have the Tv viewership data from January 2014 to December 2017. We first convert the data into timeseries then analyze it using STL, ACF, PACF plots. There are 5 network categorical factors and 11 daypart time periods and hence for each combination there are 55 time series in this dataset. We make use of exponential smoothing using HoltzWinters function in R. We then make forecast using the forecast function and finally compare our forecast with the actual values by using mean absolute error.

**Methodology and Analysis**

Part 1) Data cleaning and creating all the 55-time series.

We have the dates in format YYYYMM indicating the year and the month. We first convert it into YYYY-MM-DD format for better analysis of data.

Let us see out data using plot.ts

A screenshot of a cell phone

Description generated with high confidence

It does not make much sense.

Based on daytime period and network we split the dataframe into a list using the following code.

out <- split(tv,list(tv$network,tv$daypart))

Then by using the following code we generate the required 55 timeseries dataframes

for (i in 1:length(out)) {

assign(paste0("tv\_ts", i), as.data.frame(out[[i]]))

}

Part 2: Analysis on time series data

We create a function which will have all the required operations that are performed on the time series data. Then we will run the function for all the 55 timeseries dataframes.

We first split the data into test and training set. We train our model from January to 2014 to December 2016 and test our model on 2017 data.

Let us analyze the results on a single data frame first:

First, we convert our dataframe into a time series object using

timeseries <- ts(train$viewers, start = 2014-01-01 ,frequency = 12)

We plot this using plot.ts

A close up of a map

Description generated with high confidence

We see the data from 2012 to 2015 but is actually from 2014 to 2017.

We then analyze it using STL decomposition.

STL stands for “Seasonal and Trend decomposition using Loess”.

A close up of a map

Description generated with very high confidence

The four graphs are the original data, seasonal component, trend component and the remainder.

We can see the periodic seasonal pattern extracted from the original data and the trend which moves from 118 to 126.

Let us use ACF and PACF on it.

A screenshot of a cell phone

Description generated with very high confidence

The function Acf computes and estimate of the autocorrelation function of a time series. It refers to the way observations in a time series are related to each other and is measured by a simple correlation between current observation and observation p(number of lags) periods from the current one.

There is no gradual decrease in this and we can see a cutoff at 1. As it is ACF hence it is mostly a MA(1) process.

A screenshot of a cell phone

Description generated with very high confidence

Function Pacf computes an estimate of the partial autocorrelation function of a time series. Partial autocorrelation is used to measure the degree of association between yt and yt-p when the effects of other lags 1,2,3,… are removed.

Now we perform exponential smoothing using HoltWinters:

ts\_Hw <- HoltWinters(timeseries, beta=FALSE, gamma=FALSE)

As we have a time series that can be described using an additive model we can use it to make a short-term forecast.

We then make use of forecast function from the forecast library to generate a prediction on our training data.

ts\_forecast <- forecast(timeseries, h =12)

plot(ts\_forecast)

lines(timeseries1, col = 'red')

We can see the forecast with 80 and 95 levels with the forecast indicated by the blue line and the real values of the test data is the red line.

Let us look at the forecast of a different data frame.

A close up of a map

Description generated with high confidence

We then calculate how close these forecasts are with the actuals by creating plots and calculating the mean absolute error (MAE).

We can calculate the MAE using:

mean(abs(test$viewers - ts\_forecast$fitted ))

which we get as 10.81 and is the lowest among all the other time series. Most of the values range around 30 to 40 for the other time series.

**Conclusion**

We have learned the implementation of time series and calculated the future values using the exponential smoothing forecasting. We can see how close our forecast is to the actual data. We have learned about ACF, PACF, STL and developed a more understanding about trends seasonality, lags, trends, noise.

The best forecast with the least mean absolute error was 10.81 and the worst was 215.25 while most of the values were between 10 to 15.

A screenshot of a cell phone

Description generated with very high confidence

We have mostly 0 viewers for this combination of network and daypart hence we cannot generate a forecast for this.

A close up of a screen

Description generated with high confidence

Also, with this combination of network and daypart we cannot generate a timeseries forecast.

We have low values of MAE with networks B ,C ,E compared to networks A and E.

We have highest number of viewers for network C and the lowest number of viewers for network D.

For network A

We have the highest number of viewers with

|  |  |
| --- | --- |
| M,T,W,R,F,S,Su 8:00 PM - 11:00 PM with 26712 viewers for network A |  |

M,T,W,R,F,S,Su 8:00 PM - 11:00 PM viewers 20256 for network D

M,T,W,R,F,S,Su 8:00 PM - 11:00 PM viewers 35544 for network E

We can see that M,T,W,R,F,S,Su 8:00 PM - 11:00 PM is the prime time and has many viewers for all networks and S,Su 6:00 AM - 8:00 AM has the least number of viewers for all users.

Hence, we have successfully understood and implemented all the 55 timeseries using R.

References

1) Srivastava, T., Dar, P., & NSS. (2018, April 20). A Complete Tutorial on Time Series Modeling in R. Retrieved from https://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/

2) Using R for Time Series Analysis¶. (n.d.). Retrieved from http://a-little-book-of-r-for-time-series.readthedocs.io/en/latest/src/timeseries.html#holt-s-exponential-smoothing

3) (n.d.). Retrieved from https://stat.ethz.ch/R-manual/R-devel/library/stats/html/acf.html