Time series modeling, most of the time, uses past observations as predictor variables. But sometimes, we need external variables that affect the target variables. To include those variables, we have to use regression models. However, we are going to use **dynamic regression** to capture elaborated patterns; the difference from the orthodox regression models is that residuals are not white noise and are modeled by **ARIMA**.

The residuals we mentioned above, have autocorrelation, which means contain information. To indicate that, we will show ε_t as η_t. In this way, the residuals term η_t can be modeled by ARIMA. For instance, dynamic regression with ARIMA(1,1,1) as described:

$$y_t = \beta_0 + \beta_{1,t} + \ldots + \beta_k x_{k,t} + \eta_t$$
$$(1 - \phi_1 B)(1 - B)\eta_t = (1 + \theta_1 B)\varepsilon_t$$

ε_t denotes the white noise and **B**, the backshift notation. As we can see above equation, There two error terms: the one from regression model, η_t, and the other from ARIMA model, ε_t.

In the previous article, we have created the dataset variable, *df_xautry*. We will transform it into the multivariate time series and split it as a test and training set. Finally, we will model the training data.

```
library(dplyr)
library(forecast)

#Building the multivariate time series
df <- df_xautry[-1]
df_mts <- df %>% ts(start=c(2013,1),frequency=12)

#Split the dataset
train <- df_mts %>% window(end=c(2020,12))
test <- df_mts %>% window(start=2021)

#Modeling the training data
fit_dynamic <- auto.arima(train[,"xau_try_gram"], xreg =train[,c(1,2)])

#Series: train[, "xau_try_gram"]
#Regression with ARIMA(1,0,2) errors

#Coefficients:
#          ar1      ma1     ma2   intercept       xe   xau_usd_ounce
#       0.9598  -0.0481  0.4003  -150.8309   43.8402          0.1195
#s.e.   0.0390   0.0992  0.1091    23.7781    2.1198          0.0092

#sigma^2 estimated as 27.15:  log likelihood=-293.31
#AIC=600.62   AICc=601.89   BIC=618.57
```

Based on the above results, we have ARIMA(1,0,2) model as described below:

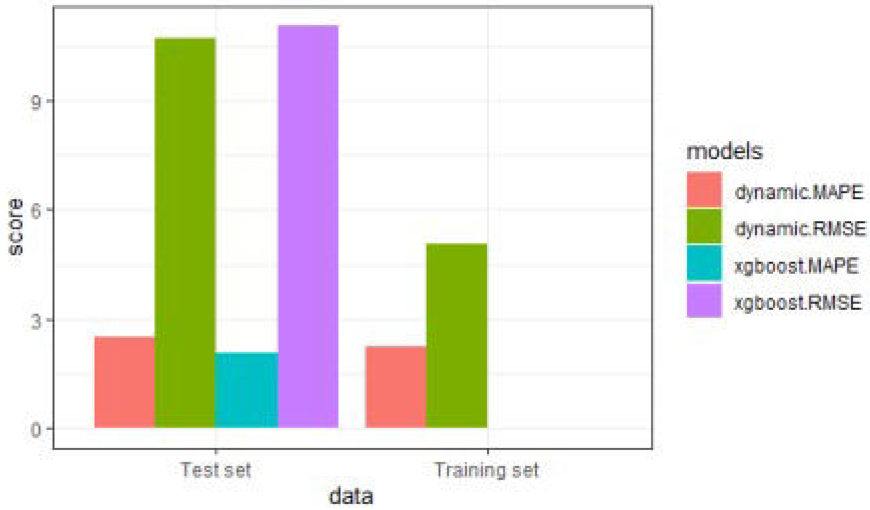
$$(1 - \phi B)y_t = c + (1 + \theta_1 B + \theta_2 B^2)\varepsilon_t$$

Now, we will do forecasting and then calculate accuracy. The accuracy for xgboost will be calculated from the *forecast_xautrygram* variable.

```
#Forecasting
fcast_dynamic <- forecast(fit_dynamic, xreg = test[,1:2])

#Accuracy
acc_dynamic <- fcast_dynamic %>%
  accuracy(test[,3]) %>%
  .[,c("RMSE","MAPE")]


acc_xgboost <- forecast_xautrygram %>%
  accuracy(test[,3]) %>%
  .[,c("RMSE","MAPE")]
```

In order to visualize the accuracy results, we're going to build the data frame and prepare it for a suitable bar chart.

```
#Tidying the dataframe
df_comparison <- data.frame(
  "dynamic"=acc_dynamic,
  "xgboost"=acc_xgboost
)

df_comparison
#             dynamic.RMSE dynamic.MAPE xgboost.RMSE xgboost.MAPE
#Training set     5.044961     2.251683  0.001594868  0.000805107
#Test set        10.695489     2.501123 11.038134819  2.060825426

library(tidyr)
df_comparison %>%
  rownames_to_column(var = "data") %>%
  gather(`dynamic.RMSE`, `dynamic.MAPE`,`xgboost.RMSE`, `xgboost.MAPE`,
         key = "models", value="score") -> acc_comparison

acc_comparison
#          data       models        score
#1 Training set dynamic.RMSE  5.044960948
#2     Test set dynamic.RMSE 10.695489161
#3 Training set dynamic.MAPE  2.251682989
#4     Test set dynamic.MAPE  2.501122965
#5 Training set xgboost.RMSE  0.001594868
#6     Test set xgboost.RMSE 11.038134819
#7 Training set xgboost.MAPE  0.000805107
#8     Test set xgboost.MAPE  2.060825426



#Plotting comparing models
ggplot(acc_comparison,aes(x=data,y=score,fill = models)) +
  geom_bar(stat = "identity",position = "dodge") +
  theme_bw()
```



Conclusion

When we examined the above results and the bar chart for unseen data, we are seeing some interesting results. For the training set, the xgboost model has near-zero accuracy rates which can lead to overfitting. The dynamic model looks slightly better for the RMSE but vice versa for the MAPE criteria.