**Machine Learning for Forecasting: Supervised Learning with Multivariate Time Series**

**An Introduction to the Auto-regressive Distributed Lags Model using Python.**

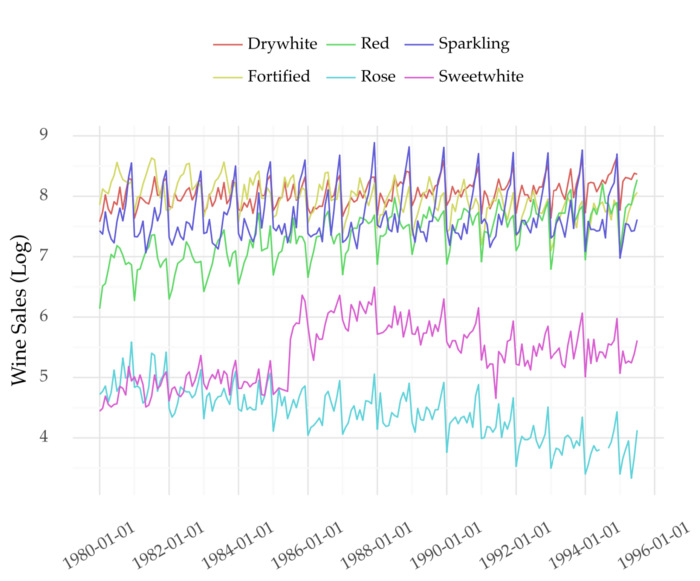


Figure 1: Multivariate time series about monthly wine sales. Link for the data source [here](https://pkg.yangzhuoranyang.com/tsdl/). Image by Author.

Here, you’ll learn how to create forecasting models with multivariate time series.

This is a follow up to my [previous post](https://towardsdatascience.com/machine-learning-for-forecasting-transformations-and-feature-extraction-bbbea9de0ac2). There, I describe the main steps for supervised learning with univariate time series.

**Introduction**

A time series is multivariate if it contains more than one variable.

See Figure 1 for an example. It shows a monthly multivariate time series about the sales of different types of wine. Each wine type is a variable in the time series.

Suppose you want to forecast one of the variables. Say, the sales of sparkling wine (personal favourite :-) ). How can you build a model to do that?

A common approach is to take that variable and view it as a univariate time series. There are plenty of methods designed to model these series. Examples include ARIMA, exponential smoothing, or Facebook’s Prophet. [Auto-regressive machine learning approaches are increasingly used](https://towardsdatascience.com/machine-learning-for-forecasting-transformations-and-feature-extraction-bbbea9de0ac2).

Yet, other variables may contain important clues about future sales of sparkling wine. Take a look at the correlation matrix below.

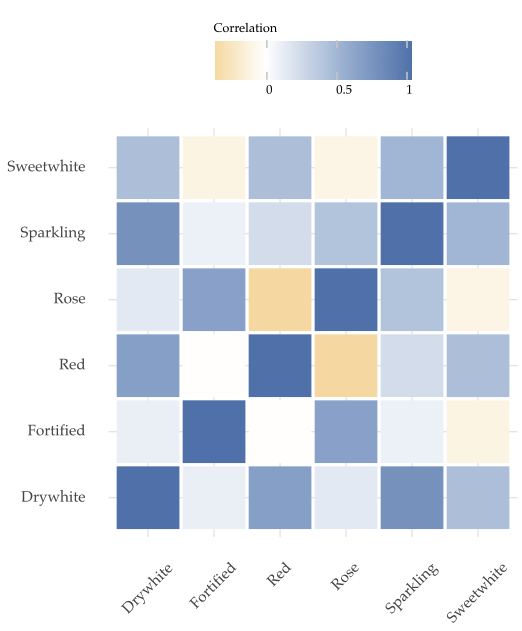


Figure 2: Correlation matrix between different types of wine. Image by Author.

The sales of sparkling wine (second row) show a decent correlation with the sales of other wines.

So, it might be a good idea to try and include these variables in the model.

We can do this with an approach called **Auto-Regressive Distributed Lag (ARDL)**.

**Auto-Regressive Distributed Lag**

**Auto-regression with univariate time series**

As the name implies, the ARDL model settles on auto-regression.

Auto-regression is the backbone of most univariate time series models. It works in two main steps.

First, we transform the (univariate) time series from a sequence of values to a matrix. We do this with the method time delay embedding. Despite the fancy name, this approach is quite simple. The idea is to model each value based on the past recent values before it. [Check my previous post for a detailed explanation and implementation](https://towardsdatascience.com/machine-learning-for-forecasting-transformations-and-feature-extraction-bbbea9de0ac2).

Then, we build a regression model. The future values represent the target variable. [The explanatory variables are the past recent values](https://towardsdatascience.com/machine-learning-for-forecasting-transformations-and-feature-extraction-bbbea9de0ac2).

**The multivariate case**

The idea is similar for multivariate time series. But, you also add the past values of other variables to the explanatory variables. This leads to the method called **Auto-Regressive Distributed Lags.** The *Distributed Lags* name refers to the use of the lags of extra variables.

Putting it all together. The future values of a variable in a time series depend on its own lags and the lags of other variables.

Let’s code this method to make it clear.

**Hands On**

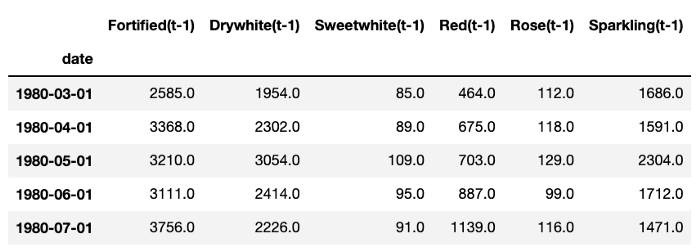
Multivariate time series often refer to sales data of many related products. We’ll use the wine sales time series as example. You can get it from [here](https://rdrr.io/cran/Rssa/man/AustralianWine.html) or [here](https://pkg.yangzhuoranyang.com/tsdl/). Yet, the ARDL approach is also applicable to other domains besides retail.

**Transforming the Time Series**

We start by transforming the time series using the script below.

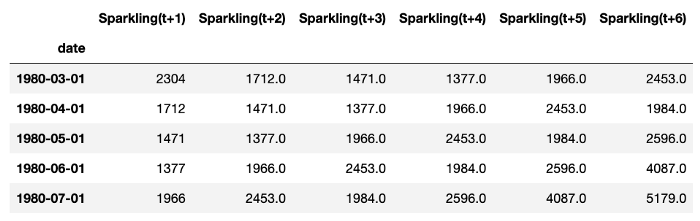
We apply the function [*time\_delay\_embedding*](https://towardsdatascience.com/machine-learning-for-forecasting-transformations-and-feature-extraction-bbbea9de0ac2)to each variable in the time series (lines 18–22). The results are concatenated into a single pandas data frame in line 23.

The explanatory variables (*X*) are the last 12 known values of each variable at each time step (line 29). Here’s how these look for the lag t-1 (other lags omitted for conciseness):



A sample of the explanatory variables at lag t-1. Image by Author.

The target variables are defined in line 30. These refer to the future 6 values of sparkling wine sales:



A sample of the target variables. Image by Author.

**Building a Model**

After preparing the data, you’re ready to build a model. Below, I apply a simple training and testing cycle using a Random Forest.

After fitting the model (line 11), we get the predictions in the test set (line 14). The model gets a mean absolute error of 288.13.

**Choosing the Number of Lags**



Photo by [Mikael Kristenson](https://unsplash.com/@mikael_k?utm_source=medium&utm_medium=referral) on [Unsplash](https://unsplash.com?utm_source=medium&utm_medium=referral)

We used 12 lags of each variable as explanatory variables. This was defined in the parameter *n\_lags* of the function *time\_delay\_embedding.*

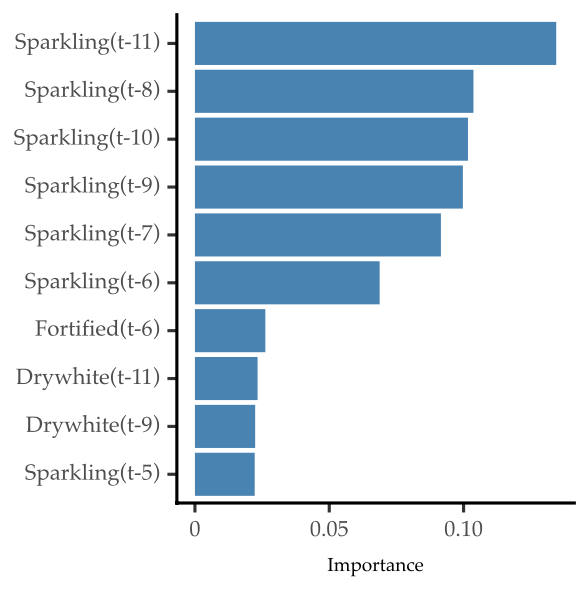
How should you set the value of this parameter?

It’s difficult to say apriori how many values should be included. That depends on the input data and the specific variable.

A simple way to approach this is to use feature selection. First, start with a fair amount of values. Then reduce this number according to importance scores or forecasting performance.

Here’s a simplified version of this process. The top 10 features are selected according to the Random Forests’ importance scores. Then, the training and testing cycle is repeated.

The top 10 features show better forecasting performance than all original predictors. Here’s the importance of these features:



Importance scores of the top 10 features. Image by Author

As expected, the lags of the target variable (Sparkling) are the most important. But, some lags of other variables are also relevant.

**Extensions to ARDL**

**Multiple Target Variables**

We aimed at forecasting a single variable (sparkling wine). What if we are interested in forecasting several ones?

This would lead to a method called Vector Auto-Regressive (VAR).

Like in ARDL, each variable is modelled based on its lags and the lags of other variables. VAR is used when you want to predict many variables, not just one.

**Relation to Global Forecasting Models**

It’s worth noting that ARDL is not the same as a [Global Forecasting Model](https://medium.com/towards-data-science/introduction-to-global-forecasting-models-3ca8e69a6524).

In the case of ARDL, the information of each variable is added in the explanatory variables. The number of variables is usually low and of the same size.

Global forecasting models pool the historical observations of many time series. A model is fit with these observations. So, each new series is added as new observations. Besides, global forecasting models usually involve up to thousands of time series. In a [previous post](https://medium.com/towards-data-science/introduction-to-global-forecasting-models-3ca8e69a6524), I describe how Global Forecasting Models operate. These are increasingly used approaches for forecasting.

**Take-Aways**

* A multivariate time series is contains two or more variables;
* The ARDL method can be used for supervised learning with multivariate time series;
* Optimize the number of lags using feature selection strategies.
* Use a VAR method if you want to predict more than one variable.

Thanks for reading, and see you in the next story!

**References**

[1] Rob Hyndman and Yangzhuoran Yang (2018). tsdl: Time Series Data Library. v0.1.0. <https://pkg.yangzhuoranyang.com/tsdl/>