Time Series Forecasting for Sales Order

Cuili Huang

November 2019

Contents

The Problem

For many businesses, accurately predicting sales quantities is an important task: based on the predictions, a company will be able to plan its raw material inventories, place purchasing orders, schedule production, and also to deploy its resources such as sales forces, transportation fleets, and warehouses manpower. In other words, reliable sales forecasting enables a company to allocate its resource efficiently and effectively.

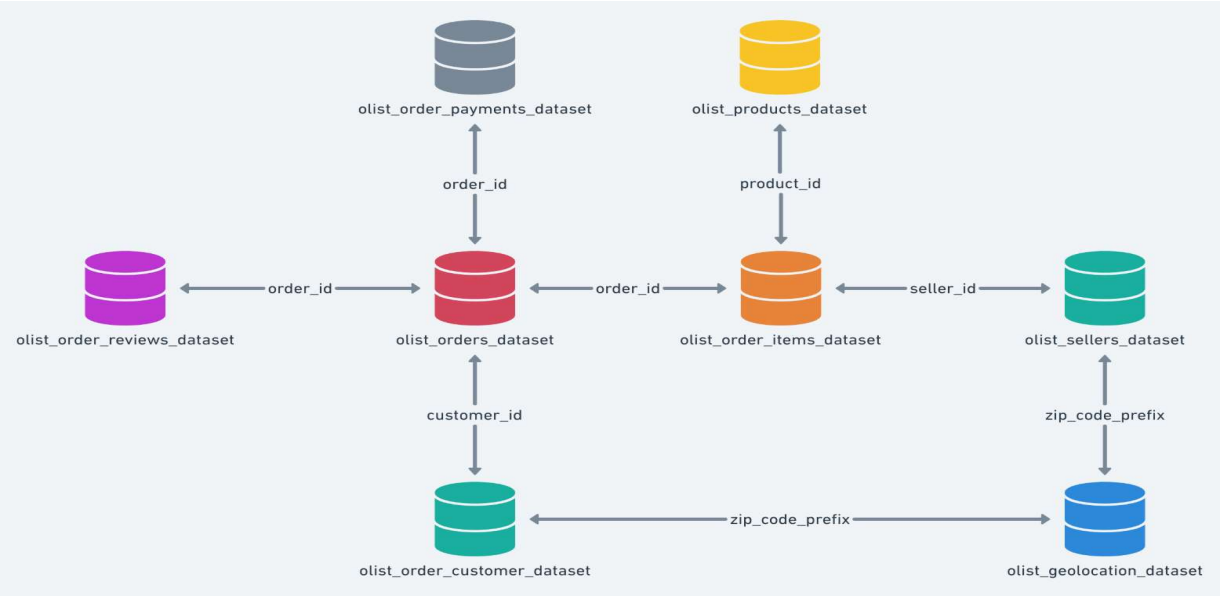
In this project, I tried to develop a machine learning model that can be used to predict the quantity of sales order of a Brazilian e-commerce business (whose business model is similar to eBay's). For such a business, having reliable information of its future sales would help it estimate its sales revenues, manage costs, and plan marketing activities.

The Data

Data acquisition

The datasets are acquired from Kaggle. They include nine csv files: order_payments, products, order_review, orders, order_items, sellers, order_customer, geolocation, product_category_name_translation. The data schema is as following (Exhibit 1).

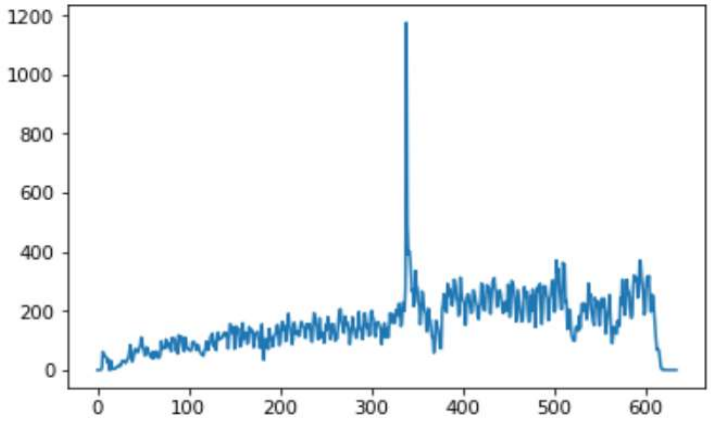
Exhibit 1



Data Wrangling

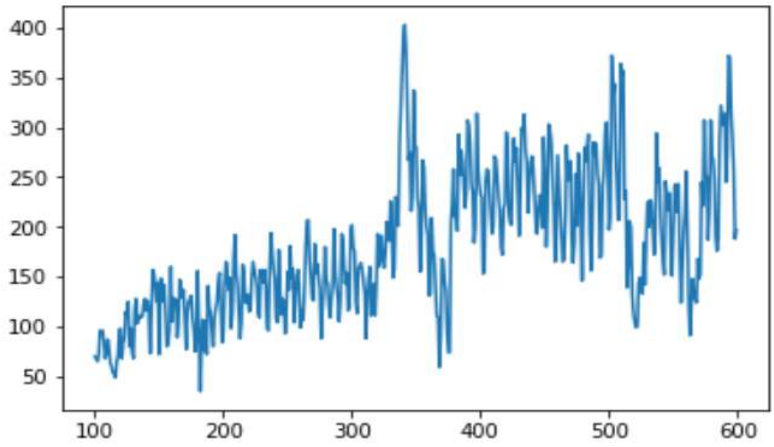
Since there is no sales quantity per day or per month available in the original datasets, I derived this information mainly from the “order” csv file. This dataset includes the column of order_id and order_purchase_timestamp (i.e. at what time the order was placed). First of all, I converted the datetime format 'YYYY-MM-DD HH-MM-SS' to 'YYYY-MM-DD' so that I can count the quantity of order by day. Secondly, I created a new dataframe called new_orders which only has two columns: order_purchase_timestamp_day and order_count. The plot the column of order_count shows that the observations between 0 to 100 and that after 600 are too small to be meaningful. Therefore, I sliced out the indexes between 100 to 600 (see Exhibit 2) for model training and testing.

Exhibit 2



Moreover, it is obvious that there are outliers in the plot (values of which are extremely high). After performing statistical analysis, I removed two outliers, values of which are 1176 and 499. Exhibit 3 is the plot after removing the outliers.

Exhibit 3



At last, I converted the dataframe to a time series for the use of next stage. (A time series is a series of data points indexed in time order. In Python, time series has its own attributes like dataframe.)

Time Series Analysis

Before I trained the machine learning model, I did a thorough analysis to discover the autocorrelation between lags, the impact of white noises, and the trend and seasonality of the data.

Autocorrelation Function (ACF) Plot

Just as correlation measures the extent of a linear relationship between two variables, autocorrelation measures the linear relationship between lagged values of a time series.

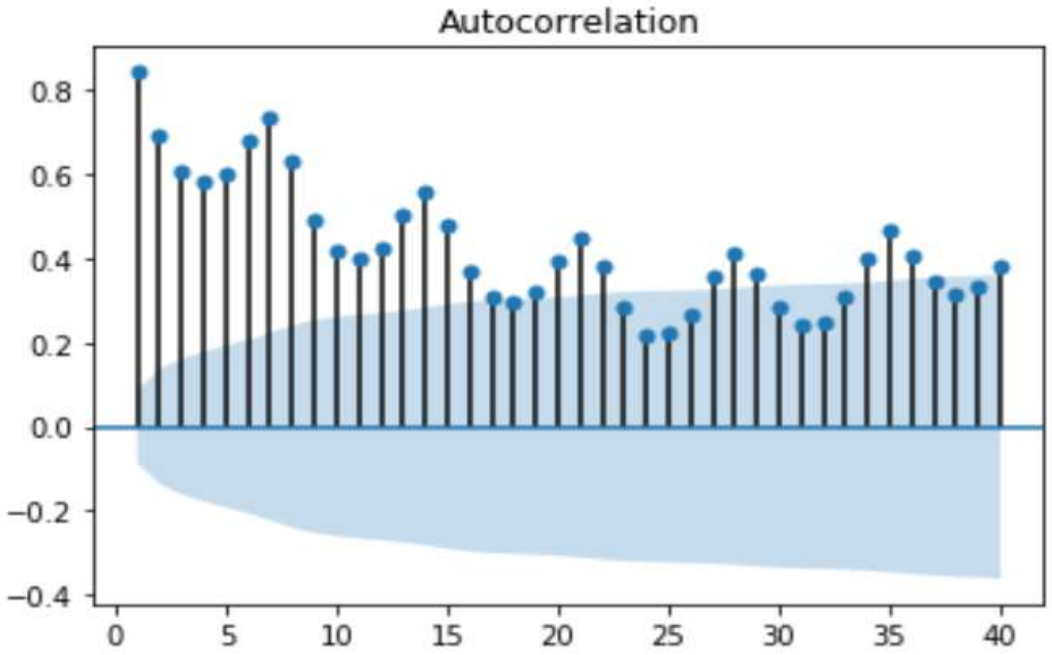
For example, r_1 measures the relationship between y_t and y_{t-1}, r_2 measures the relationship between y_t and y_{t-2}, and so on. The value of r_k can be written as

$$r_k = \frac{\sum_{t=k+1}^{T} (y_t - \bar{y})(y_{t-k} - \bar{y})}{\sum_{t=1}^{T} (y_t - \bar{y})^2},$$

where T is the length of the time series.

The following plot (Exhibit 4) shows the correlation of y_t and its previous 40 lags. If use r=0.5 as cut off, y_t and its eight previous lags are positively correlated. Moreover, the ACF plot suggests that there is a strong seasonality of the data: from the peak, sales will decrease gradually until it reaches the trough at the fourth day; then it gradually increases and reaches the peak again at the seventh day. In other words, there is a seven days seasonal pattern. It looks like a weekly pattern because people may have more time to shop on-line at the weekend and doing less so during the week.

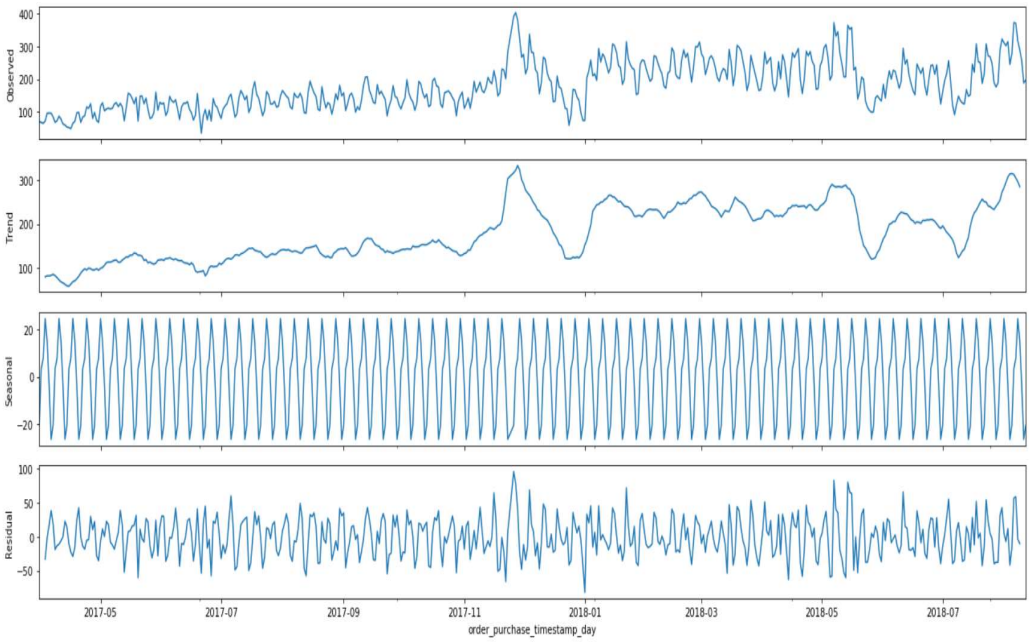
Exhibit 4



Decomposition

In the following I used seasonal_decompose from statstmodels API to decompose the time series into trend, seasonality, and residual components. By observing the plot of the time series, I assumed that this is an additive decomposition (timeseries=trend+seasonality+residual). The trend plot indicates that there is a slight upward trend in the data and the seasonal plot shows a strong cyclical pattern which is also proved in the ACF plot.

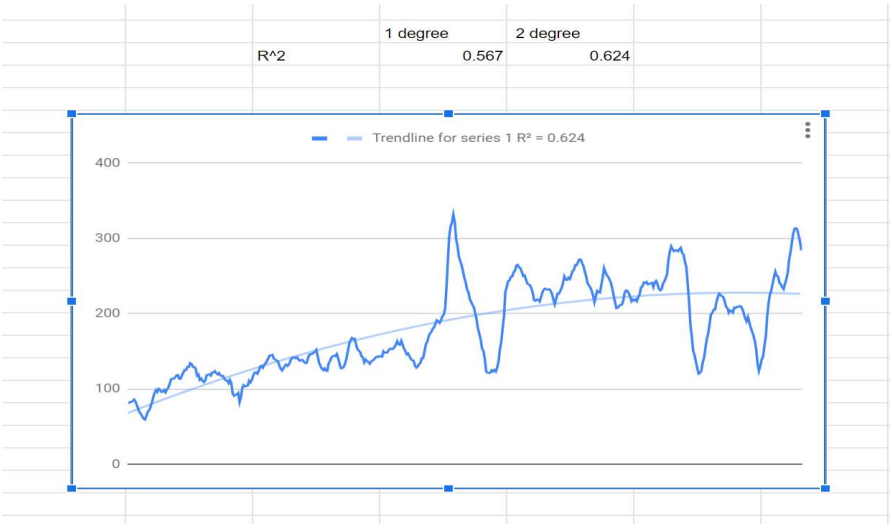
Exhibit 5



Trendline research for trend data

I extracted the trend data from the decomposition and exported to excel. By applying trendline research on google excel, I discovered that the trendline looks like a 2-degree function.

Exhibit 5



Time Series Forecasting

Base Model

Unlike other machine learning models, typically time series uses first 80% of the dataset as training dataset and the latter 20% as testing dataset. In this project, as there is seasonality in the data, I choose to use SARIMAX (Seasonal-ARIMA) model. For SARIMAX model, the most important parameters are order=(p, d, q) and seasonal_order=(P, D, Q, s). Order will be used if the data is not cyclical, and seasonal_order will be used if the data is cyclical. Both order and seasonal_order will be used if part of the data is cyclical and part is not. For this time series, seasonal_order will be used since it is all cyclical.

According to the ACF plot, y_t and its eight previous lags are correlated. Therefore, P equals to 8. According to the trendline research, it is a 2-degree function. Hence, D equals to 2. "s" represents the seasonality, which equals to 7. Regarding Q, the previous white noise that affect the y_t, we leave it as 1 at this moment (the hyperparameter to be tuned later). The evaluation metric to be used is Symmetric Mean Absolute Percentage Error (SMAPE).

$$\mathrm{SMAPE} = \frac{100\%}{n} \sum_{t=1}^{n} \frac{|F_t - A_t|}{|A_t| + |F_t|}$$

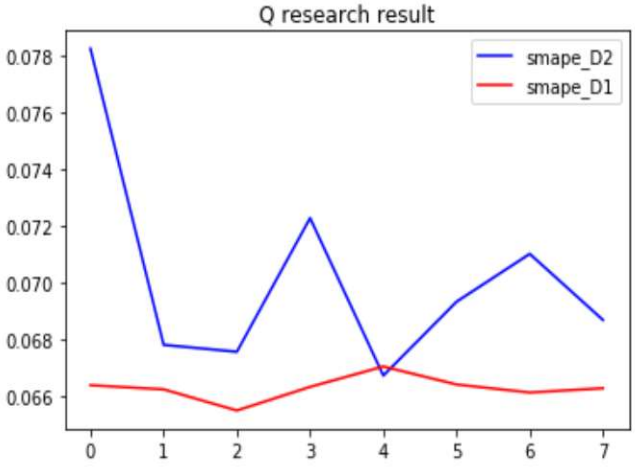
When I started the train the model, as there is a very strong collinearity, I changed P from 8 to 7 to reduce this effect. As a result, SMAPE for this base model is 6.78%.

Tune Hyperparameter

Based on the base model, I used "for loop" to research for the optimal value of Q (ranging from 0 to 7). The result shows that when Q equals to 4, SMAPE yields the lowest score: 6.67%. Furthermore, although the trendline research suggests that the trend data is a 2-degree function, I continued to do a research of D equals to 1 (as the

R^2 are pretty close between D=1 and D=2). According to the following diagram (Exhibit 6), SMAPE yield the lowest score when P=7, D=1, Q=2, S=7. Consequentially, these are the optimal parameters to use for the model.

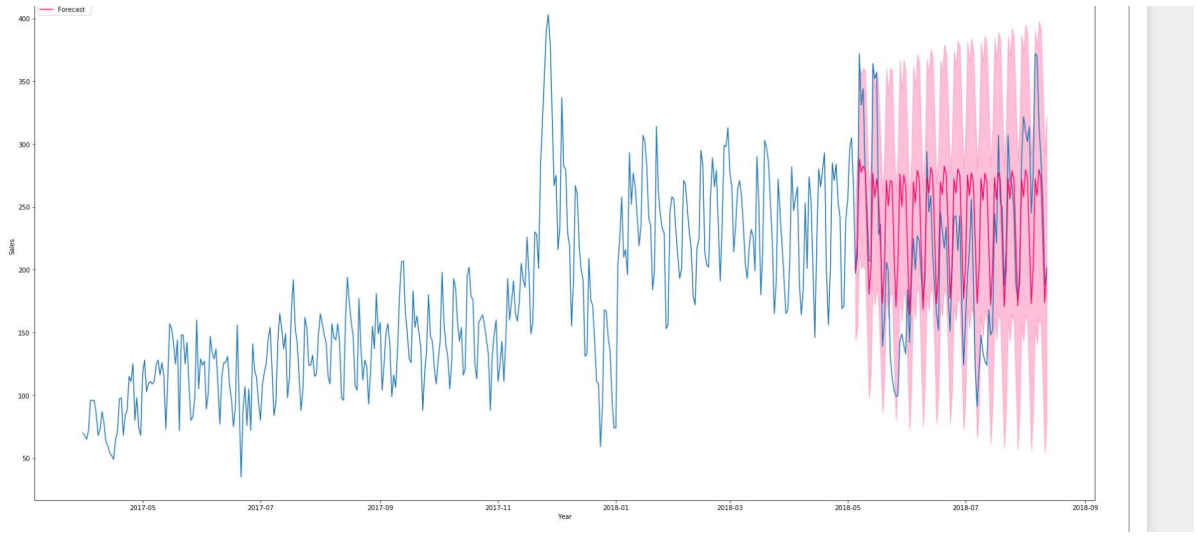
Exhibit 6



Prediction

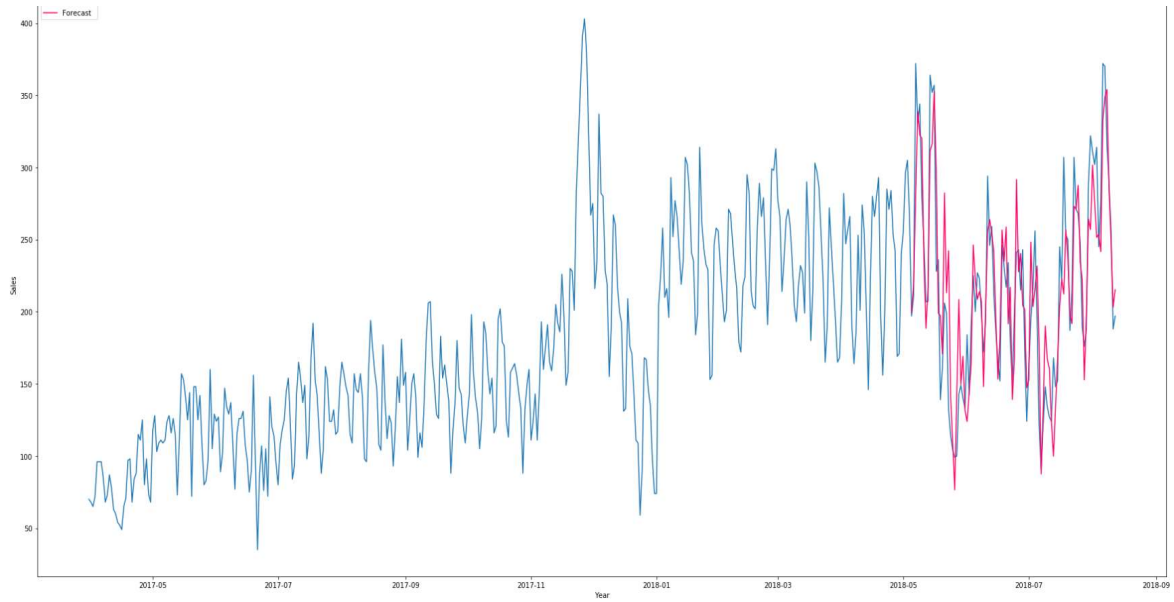
Based on the optimal hyperparameters derived from above research. I use the SARIMAX model to predict the quantity of sales order for the following 100 days. Surprisingly, the prediction is not accurate compared to the actual data. It looks like the prediction repeat itself after the first few data points (Exhibit 7).

Exhibit 7



To solve this problem, I use the “one-step-ahead” approach: always use the actual data to predict the next value. The result (Exhibit 8) indicates that this method is more accurate at predicting future values.

Exhibit 8



Conclusion

Unlike other types of machine learning modeling, time series forecasting does not require comprehensive feature analysis or engineering, as there are only two features in the dataset: time series indexes and correspondent values. Time series is a Python object like dataframe, which has many attributes. One can do grouping, sorting, or calculation with it. Moreover, unlike other machine learning modeling randomly assign data points into either training or testing dataset, time series forecasting requires that data points for training and testing are temporally continuous.

The most important part of time series forecasting is time series analysis, which includes autocorrelation function plot, time series decomposition and trendline research. The results of these researches will derive the most important parameters for SARIMAX model. In addition, unlike other machine learning models use grid search or random search, SARIMAX model uses “for loop” to tune hyperparameters.

Finally, as the prediction will repeat itself after the first few intervals, performing forecast for a long period of time will yield unreliable information. In practice, performing one-step-ahead forecasting will derive more reliable result that can be used for business purpose.