**Time Series Forecasting for Sales Order**

**Milestone Report**

**Cuili Huang**

**November 2019**

**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.

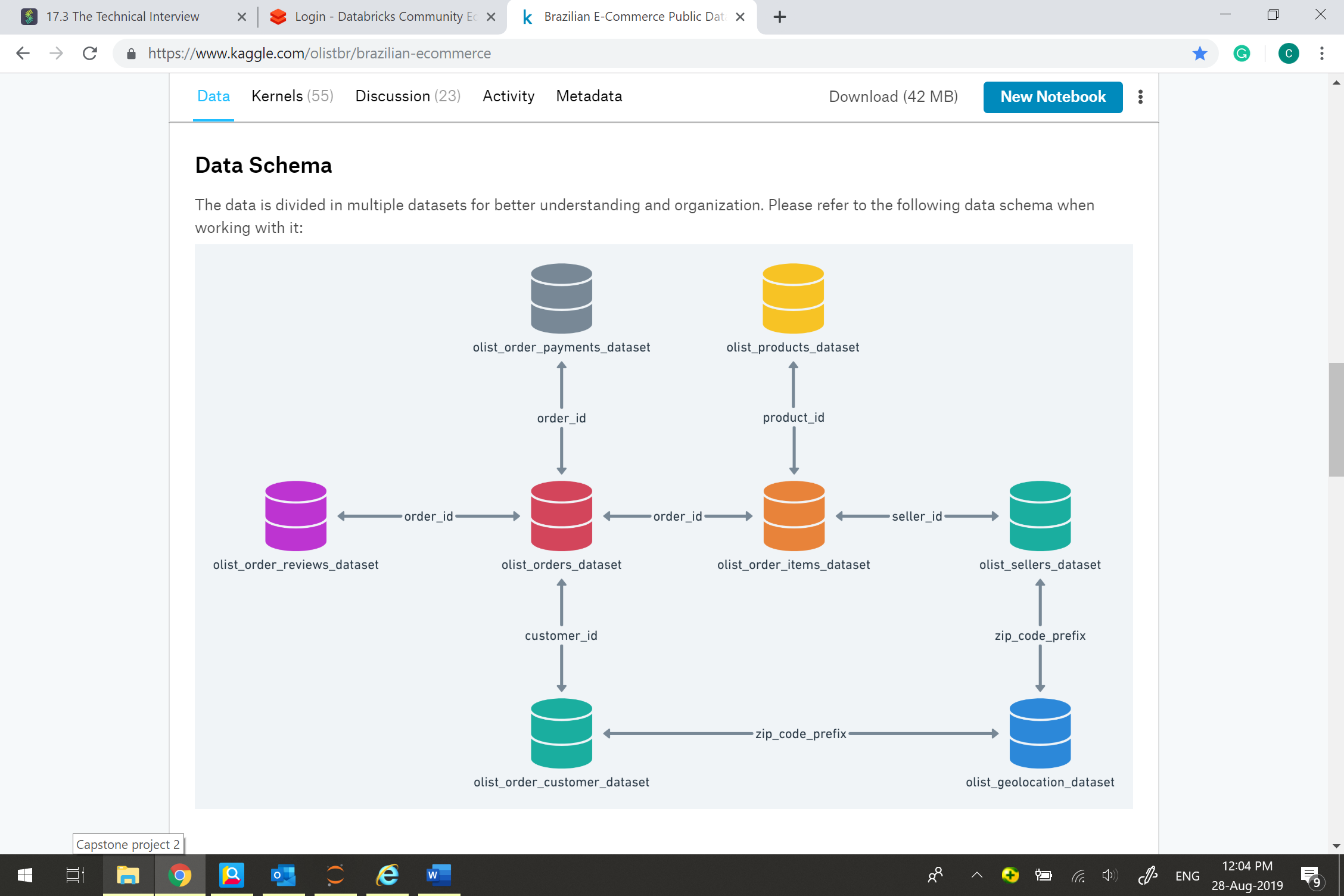
In this project, I tried to develop a machine learning model that can be used to predict the quantity of sales order of a Brazilian e-commerce business (whose business model is similar to eBay’s). For such a business, having reliable information of its future sales would help it estimate its sales revenues,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

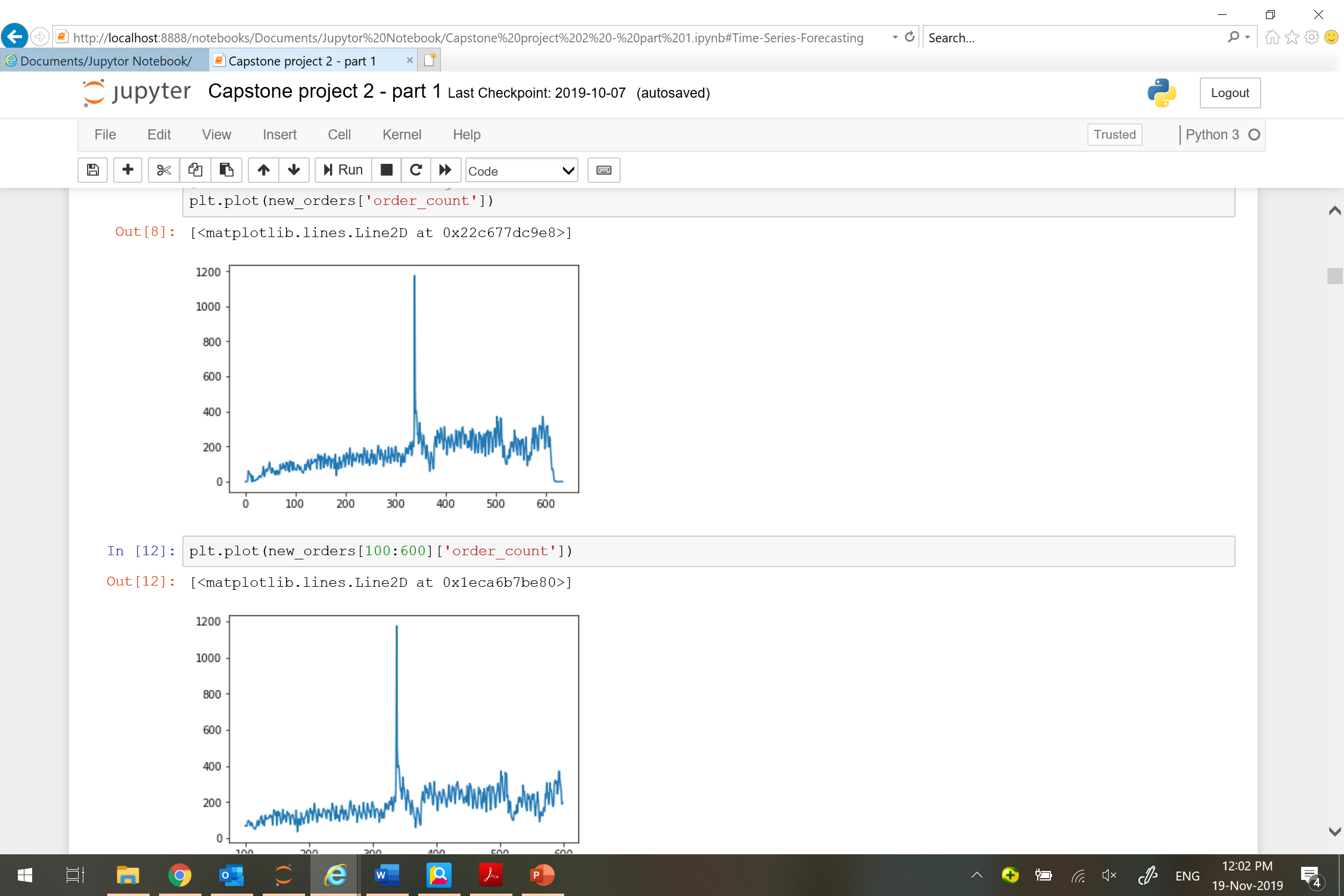
Exhibit 1



*Data Wrangling*

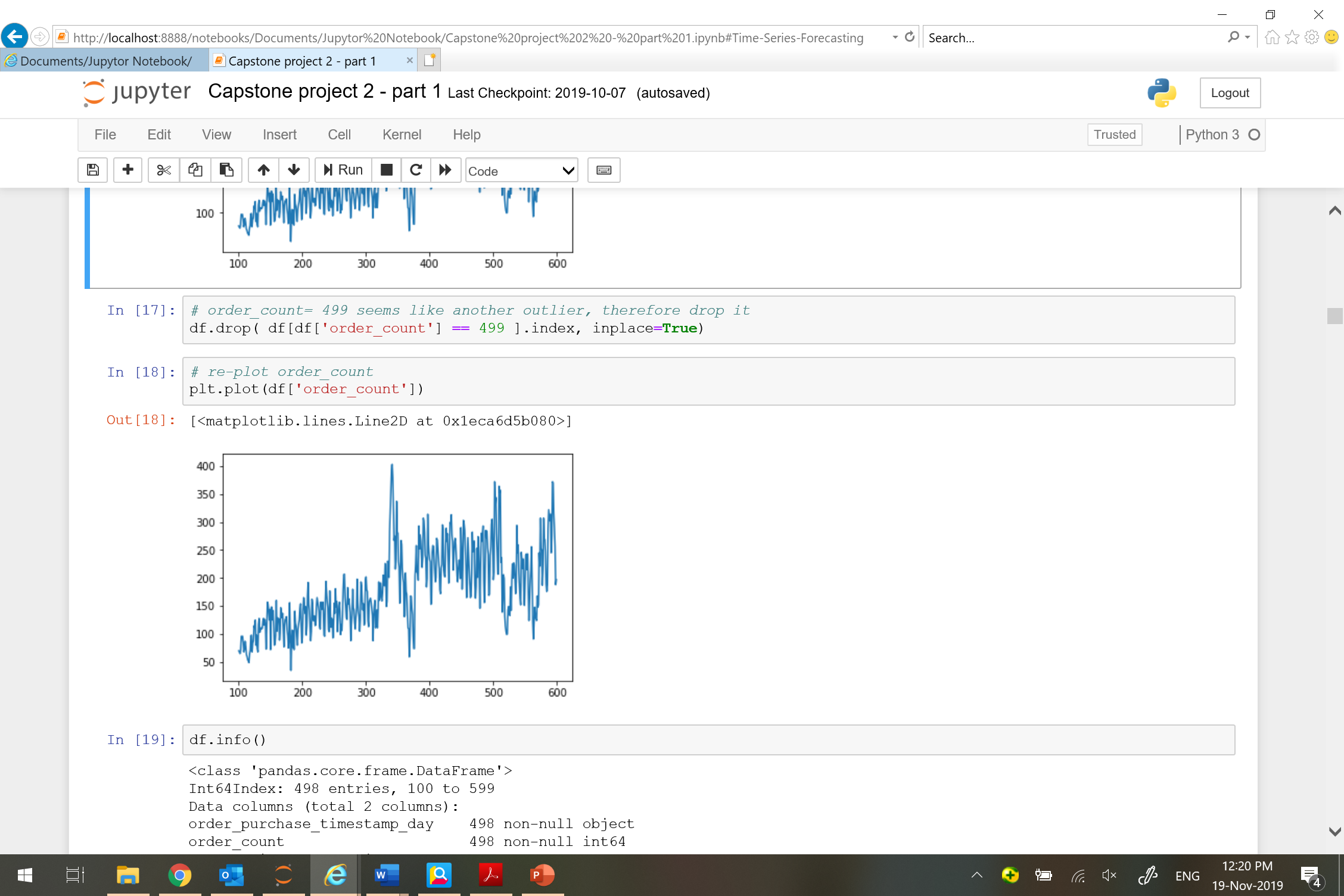
Since there is no sales quantity per day or per month available in the original datasets, I derived this information mainly from the “order” csv file. This dataset includes the column of order\_id and order\_purchase\_timestamp (i.g. at what time the order was placed). First of all, I converted the datetime 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.)

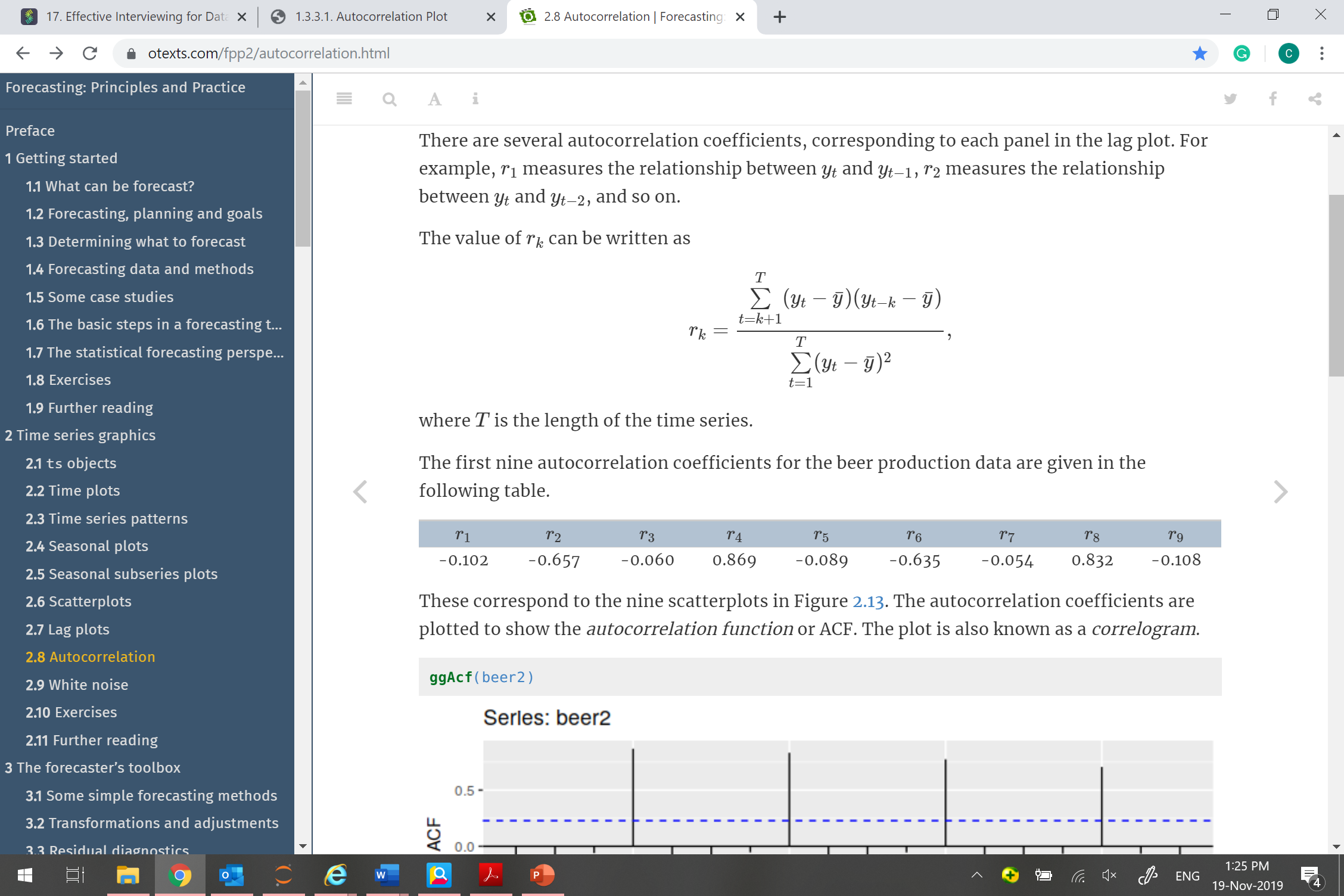
**Time Series Analysis**

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, r1 measures the relationship between yt and yt−1, r2 measures the relationship between yt and yt−2, and so on. The value of rk can be written as



where T is the length of the time series.

The following plot (Exhibit 4) shows the correlation of yt and its previous 40 lags. If use r=0.5 as cut off, yt 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

