Hierarchical Time Series Forecasting

By Aditi Shrivastava, Akshat Bhargava, and Deeksha Kamath

Motivation and Background

Forecasting plays an important role in making informed decisions in many areas of a company. Any industry with a product offering needs to develop a forecasting system that involves several approaches to predicting uncertain events. Such forecasting systems require the development of expertise in identifying problems, applying a range of forecasting methods, selecting appropriate methods for each problem, and evaluating and refining forecasting methods over time.

Among various industries, fashion is one of the most volatile industries. Trends in colours, prints, cuts, patterns, and materials change faster than ever, making retail forecasting a challenge for established brands and new ones alike. Not every retailer will want to risk scaling their business. It takes a lot of monetary resources to regroup to catch-up with evolving fast-fashion trends or prepare to meet seasonal demands. Retailers sometimes suffer huge losses from unsold inventory and liquidation costs.

Tech-savvy fashion retailers use big data to follow and predict trends, prepare for customer demand, segment customers, optimize pricing and promotions based on customer preferences, and monitor real-time analytics to track business outcomes. One such retailer approached FIND Innovation Labs, a start-up based in Vancouver, and thus this capstone project came to fruition. The retailer provided historical sales-transactions data to FIND, and on understanding that it was a case of Time-Series Forecasting problem, they required a way to generate forecasts that are consistent across dimensions.

We were motivated to investigate this problem and take a step in solving an industry-wide data challenge. Through this project, we would be able to apply our knowledge of the best data science techniques to help a retailer learn the insights present within the data they have and also predict their sales. This would, in turn, help the retailer be better prepared with proper inventory-management practices in place during peak sales periods and find innovative ways to rake in profits during slower periods.

Time series can often be naturally disaggregated by various attributes of interest. For example, the total number of clothing items sold by a fashion company can be disaggregated by product types such as shirts, trousers, athletic wear, loungewear, and hybrids like athleisure. Each of these can be disaggregated into finer categories. For example, shirts can be divided into business formal, business casual or casual, and so the collection of time series follows a hierarchical aggregation structure. We needed a way to produce forecasts that aggregate on all these levels in the same way as it is common to produce disaggregated forecasts based on disaggregated time series, and we usually require the forecasts to add up in the same way as the data. For example, forecasts of regional sales should add up to give forecasts of state sales, which should, in turn, add up to give a forecast for the national sales.

Problem Statement

In many applications, there are multiple time series that are hierarchically organized and can be aggregated at several different levels in groups based on time, geography or product type. The spatial dimension captures the geographic distribution of the retail stores at different levels like country, province and city. The temporal dimension defines the chunks of time for which data are lumped together; for instance yearly, monthly or weekly. The product hierarchy represents an administrative organization of products in some levels that suggests a degree of similarity: department, category and style.

These multiple time series that are hierarchically organized and can be aggregated at several different levels in groups based on products, geography or some other features are called "hierarchical time series". We were interested in understanding the data and the patterns and trends present within.

Our **first question** was, "What is the best approach to hierarchical forecasting to create coherent forecasts across various aggregation structures?" The approaches we highlight are bottom-up, top-down and middle-out. The resulting revised forecasts add up appropriately across the hierarchies, are unbiased and have minimum variance amongst all combination forecasts under some simple assumptions.

Our **second question** was "How to automate the choice of the appropriate model?" In particular, we sought answers to these questions such as how the best model could be fit on the data, what the parameters of this model looked like if the same model could be used across all hierarchies or a different model needs to be employed for each hierarchy. In case the same model was used, did the parameters change?

Data Science Pipeline

1. **Data Collection:**

The dataset was provided to us by FIND and consisted of 3-4 years of anonymized sales transaction data of a Fashion retailer. The data contained product hierarchy information represented in the following schema:

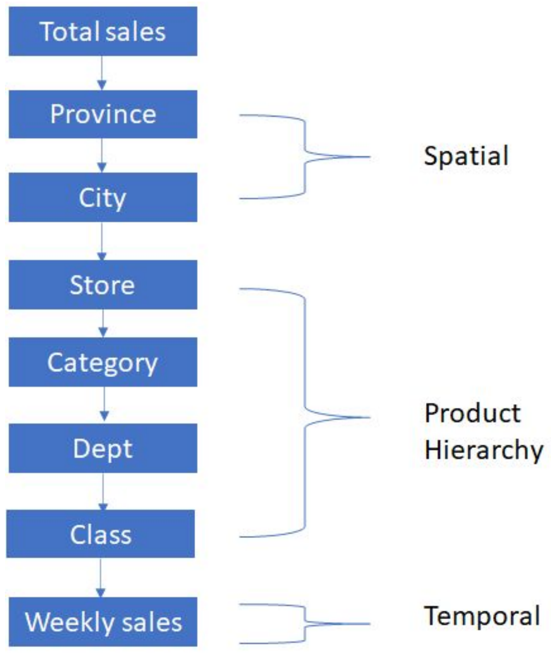
(time)
year - int number eg. 2012
month - int number eg. 1
week - int number eg. 4

(space)
province - string eg prov1
city - string eg city5
store - string eg store10

(product)
category - string eg cat44
department - string eg department33
class - string eg. class21
vendor - string eg. vendor 4
size - string eg. size11

(values)
totalQuantity - int number like 10
totalSales - float number like 45.3

The natural hierarchies present in the data across the different dimensions are:
Spatial: province -> city -> store
Product: category -> department -> class
Temporal: yearly -> monthly -> weekly



2. Data Preprocessing:

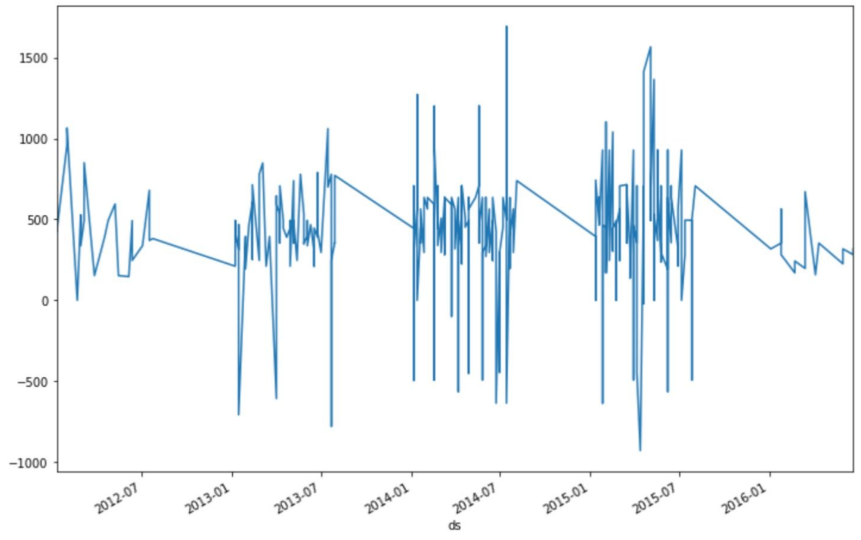
The data presented was saved as a csv file, and it had no missing values, hence imputations were not needed. On the other hand, the data also did not have a DateTime column, instead of having separate year, month and week columns.

The lowest temporal resolution of the data was weekly, therefore at a minimum, we could view total sales for a week, for a given month and year. The temporal data needed a better representation, for ease of feature access and to save storage space as well, therefore the three columns of the year, month and week were combined into a single **DateTime** type column. This is achieved by adding leading zeros to the month and week columns and storing it as a DateTime type object using an appropriate time format-specifier.

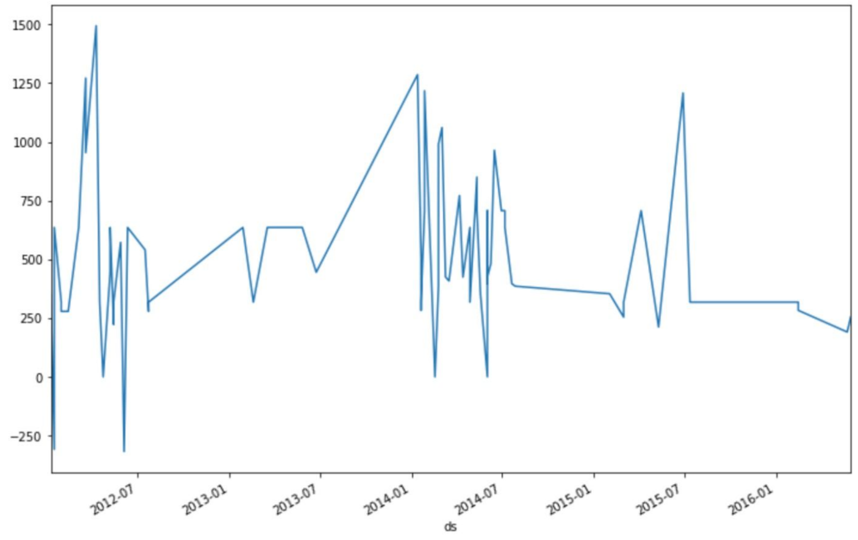
Once the main data was generated with the clean date format, we proceeded to create subsets for each hierarchy level. After discussions with FIND, we concluded that the features to be taken into consideration while creating these subsets are Province, City, Store, Department, Category and Class. Therefore six datasets were generated from the main data, wherein each file the sales were aggregated over each of these features, using GROUP BY functionality.

3. Data Exploration:

As the data is a time-series, we performed an Exploratory Data Analysis to gain an understanding of the data. We wanted to identify seasonality patterns in the sales and look for trends to figure out what kind of transformation we need to apply before generating a prediction model.



Lumpy



Intermittent

Since there are more than 2000 different time-series, we explored methodologies to segment time-series into categories. These categories share similar characteristics and can be modelled using the same machine learning models. The data that we obtained could be segmented into two different categories based on the demand variability inherently present in the data - **Lumpy** and **Intermittent**.

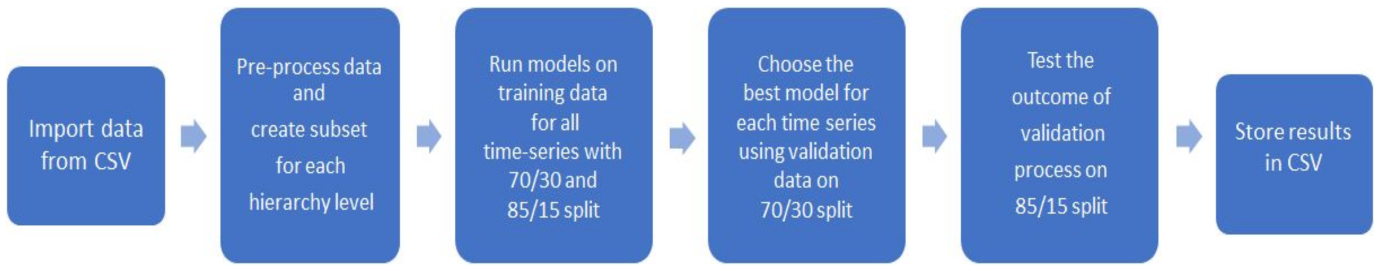
4. Data Modelling:

With the appropriate subsets created and characteristics identified, we proceeded with creating models for forecasting. We chose 2 widely used statistical models ARIMA and Exponential Smoothing, and 1 machine learning model Facebook Prophet. The training was done in two iterations, each time using a different train-test split of the data. For the first iteration, the data was split into 70/30 percentages and for the second, the data was split into 85/15. The metric of evaluation used was the Mean Absolute Percentage Error (MAPE).

We proceeded as follows:

- Fit each model over the 70/30 split data.
- Fit each model over the 85/15 split data.
- Choose the best model by training and validating the 70/30 split data.
- Apply the best model over the 85/15 split data using MAPE as the metric.
- Store the forecasted sales values into a CSV file.

Our data science pipeline can thus be summarized using the following diagram:


Import data from CSV
Pre-process data and create subset for each hierarchy level
Run models on training data for all time-series with 70/30 and 85/15 split
Choose the best model for each time series using validation data on 70/30 split
Test the outcome of validation process on 85/15 split
Store results in CSV

Methodology

- **Tools:**

We leveraged python to develop this automatic hierarchical time series forecasting system. Python has a vast collection of libraries like Statsmodel for exploratory data analysis along with creating and fitting various statistical models like ARIMA and Exponential Smoothing, machine learning models like FBProphet, and Matplotlib for plotting forecasts during training. We used Tableau to display the final forecasting results for the chosen best approach. As the computations scaled very quickly, we used Google Colab for running the kernels and generating results.

- <u>**Analysis Methods:**</u>

To get answers to our question, we took a deep dive into understanding the patterns present in the data. The accuracy of a forecast strongly depends on the product's forecastability. We can find evidence of it in the demand history characteristics. Understanding a product's forecastability is a must to set correct anticipations on what can be achieved in terms of forecast accuracy. In the end, it is not about reaching the highest accuracy, it is about being able to answer customer demand when it hits.

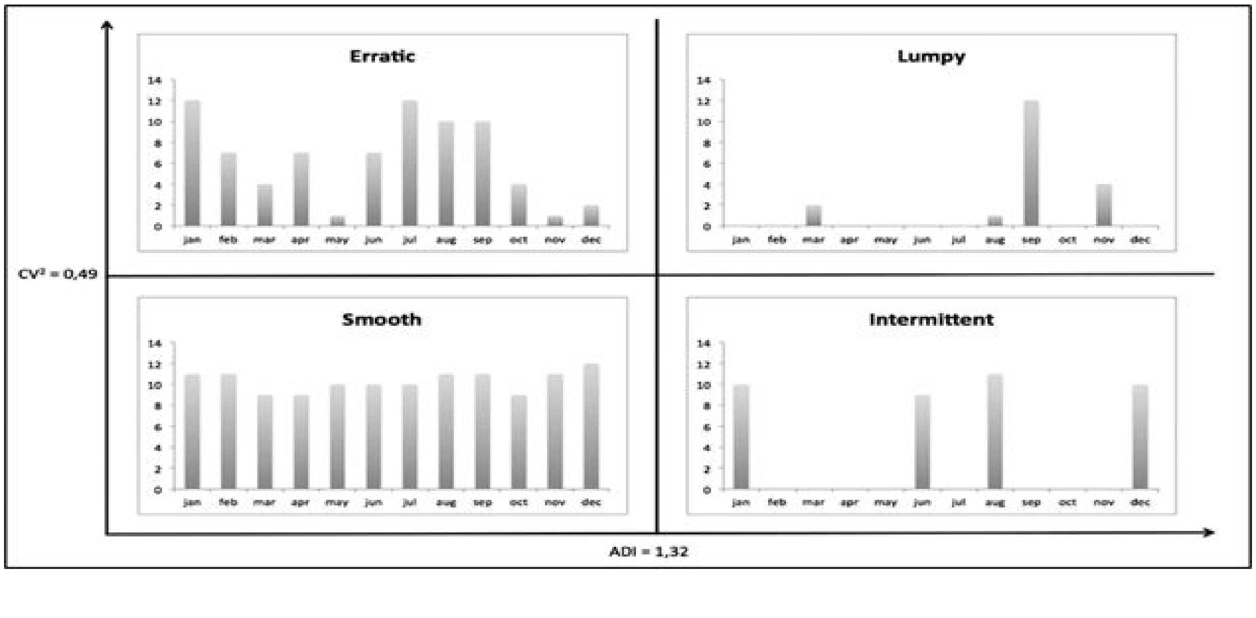
To determine a product forecastability, we apply two coefficients:

- ADI - Average demand interval - It measures the demand regularity in time by computing the average interval between two demands.
- CV^2 - square of the Coefficient of Variation. It measures the variation in quantities.

Based on these 2 coefficients, any time series can be classified into the following 4 segments:

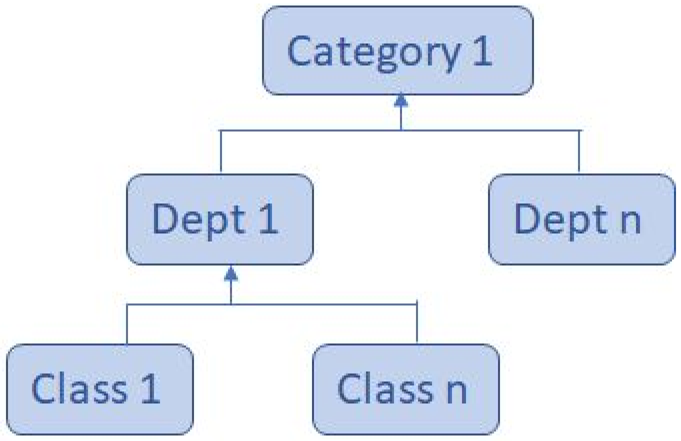
Demand	**Coefficient values**	**Characteristic**
Smooth	ADI < 1.32 and CV^2 < 0.49	The demand is very regular in time and in quantity.
Intermittent	ADI >= 1.32 and CV^2 < 0.49	The demand history shows very little variation in demand quantity but a high variation in the interval between two demands.
Erratic	ADI < 1.32 and CV^2 >= 0.49	The demand has regular occurrences in time with high quantity variations.
Lumpy	ADI >= 1.32 and CV^2 >= 0.49).	The demand is characterized by a large variation in quantity and in time. It is actually impossible to produce a reliable forecast, no matter which forecasting tools you use. This particular type of demand pattern is unforecastable.

Below are some of the examples of time-series belonging to each category-

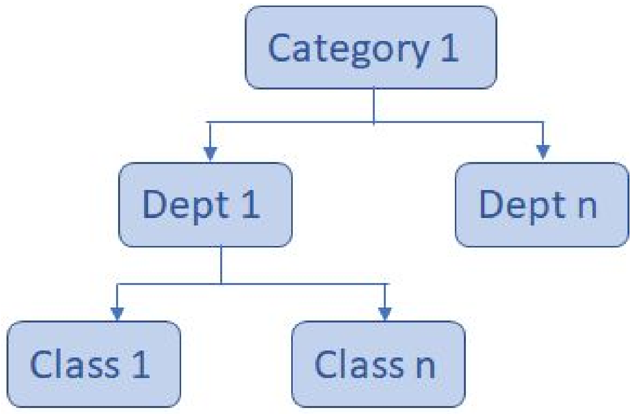


The other important methodology we applied to solve this problem is to figure out ways of generating coherent forecasts: In order to do this, we outline three methods of aggregating data.

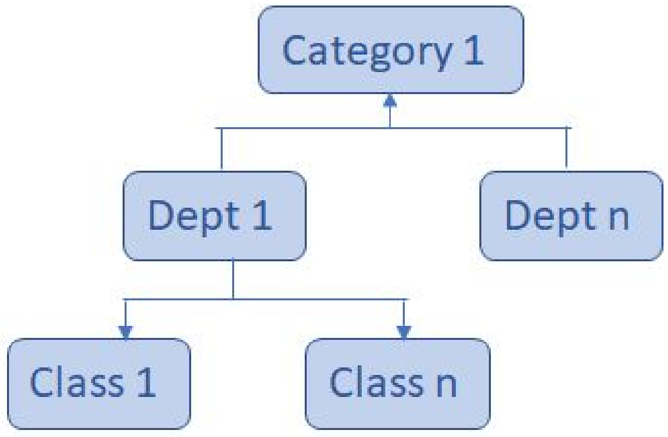
Bottom-up - This approach involves first generating forecasts for each series at the bottom level, and then summing these to produce forecasts for all the series in the structure.



Top-down - They involve first generating forecasts for the Total series, and then disaggregating these down the hierarchy.



Middle-out - The middle-out approach combines bottom-up and top-down approaches. First, a "middle level" is chosen and forecasts are generated for all the series at this level. For the series above the middle level, coherent forecasts are generated using the bottom-up approach by aggregating the "middle-level" forecasts upwards. For the series below the "middle level", coherent forecasts are generated using a top-down approach by disaggregating the "middle level" forecasts downwards.



- <u>**Models Used:**</u>

ARIMA: In an autoregression model, we forecast the variable of interest using a linear combination of past values of the variable. The term autoregression indicates that it is a regression of the variable against itself. An autoregressive model of order p can be written as

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \varepsilon_t$$

where ε_t is white noise. This is referred to as an $AR(p)$ model, an autoregressive model of Order p

Rather than using past values of the forecast variable in a regression, a moving average model uses past forecast errors in a regression-like model.

$$y_t = c + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q}$$

where ε_t is white noise. This is referred to as an $MA(q)$ model, a moving average model of order q.

If we combine differencing with autoregression and a moving average model, we obtain a non-seasonal ARIMA model. ARIMA is the acronym for AutoRegressive Integrated Moving Average,

$$y'_t = c + \phi_1 y'_{t-1} + \cdots + \phi_p y'_{t-p} + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t$$

y_t' is the differenced series. We call this an $ARIMA(p,d,q)$ model, where

p =order of the autoregressive part;
d =degree of first differencing involved;
q =order of the moving average part.

A seasonal ARIMA model is formed by including additional seasonal terms in the ARIMA models we have seen so far. It is written as follows:

$$\text{ARIMA} \quad \underbrace{(p,d,q)}_{\substack{\uparrow \\ \text{Non-seasonal part} \\ \text{of the model}}} \quad \underbrace{(P,D,Q)_m}_{\substack{\uparrow \\ \text{Seasonal part of} \\ \text{of the model}}}$$

where m=number of observations per year

As there is an obvious seasonality associated with fashion sales, we chose the **seasonal ARIMA** model for generating forecasts.

To automate the parameter selection process, we went ahead with an **Auto ARIMA** implementation, thereby automating the selection of both the seasonal and non-seasonal parameters for each time series that could be produced. As we wanted to produce forecasts for a period of a year, we set the value of m to 52 (52 weeks in a year)

Exponential Smoothing: Forecasts produced using Exponential Smoothing are weighted averages of past observations, with the weights decaying exponentially as the observations get older. It requires recognizing key components: seasonality and trend, and the way in which these enter the smoothing method: additive, multiplicative and damped.

$$\hat{y}_{T+1|T} = \alpha y_T + \alpha(1-\alpha)y_{T-1} + \alpha(1-\alpha)^2 y_{T-2} + \cdots,$$

We began with Simple Exponential Smoothing. It just requires a smoothing parameter, $0 \le \alpha \le 1$, which determines the rate in which the weights associated with each observation will decrease. Since this model does not take the trend and seasonal patterns into account, the model performance was not satisfactory.

We then moved to Holt's Winter method which generates forecasts using 3 smoothing equations: level, trend, and seasonal components. It also considers the damping parameter which dampens the trend to a flat line sometime in the future. This ensures that the model does not over-forecast for a longer time-series. The results obtained from this model were pretty close to the actual sales value.

Facebook Prophet: The Prophet uses a decomposable time series model with three main model components: trend, seasonality, and holidays. They are combined in the following equation:

$$y(t)= g(t) + s(t) + h(t) + \varepsilon t$$

g(t): piecewise linear or logistic growth curve for modeling non-periodic changes in time series
s(t): periodic changes (e.g. weekly/yearly seasonality)
h(t): effects of holidays (user-provided) with irregular schedules
ε_t : error term accounts for any unusual changes not accommodated by the model.

Evaluation

It is pretty easy to train-validate-test if you have a dataset for testing too. But for forecasting in a real-time scenario, we will not have the testing data for measuring our model's accuracy. To solve this issue, we took 15% of our dataset as an offset to train-validate our models.

We first train our models on the 70% data of the 70-15-15% split dataset. And record the best working models on the next 15% of the dataset. This provides us with the best model for each time-series. Then, we train the 85% of the data of 85-15% split dataset with these best models for the corresponding time-series to get the predictions on the next 15%.

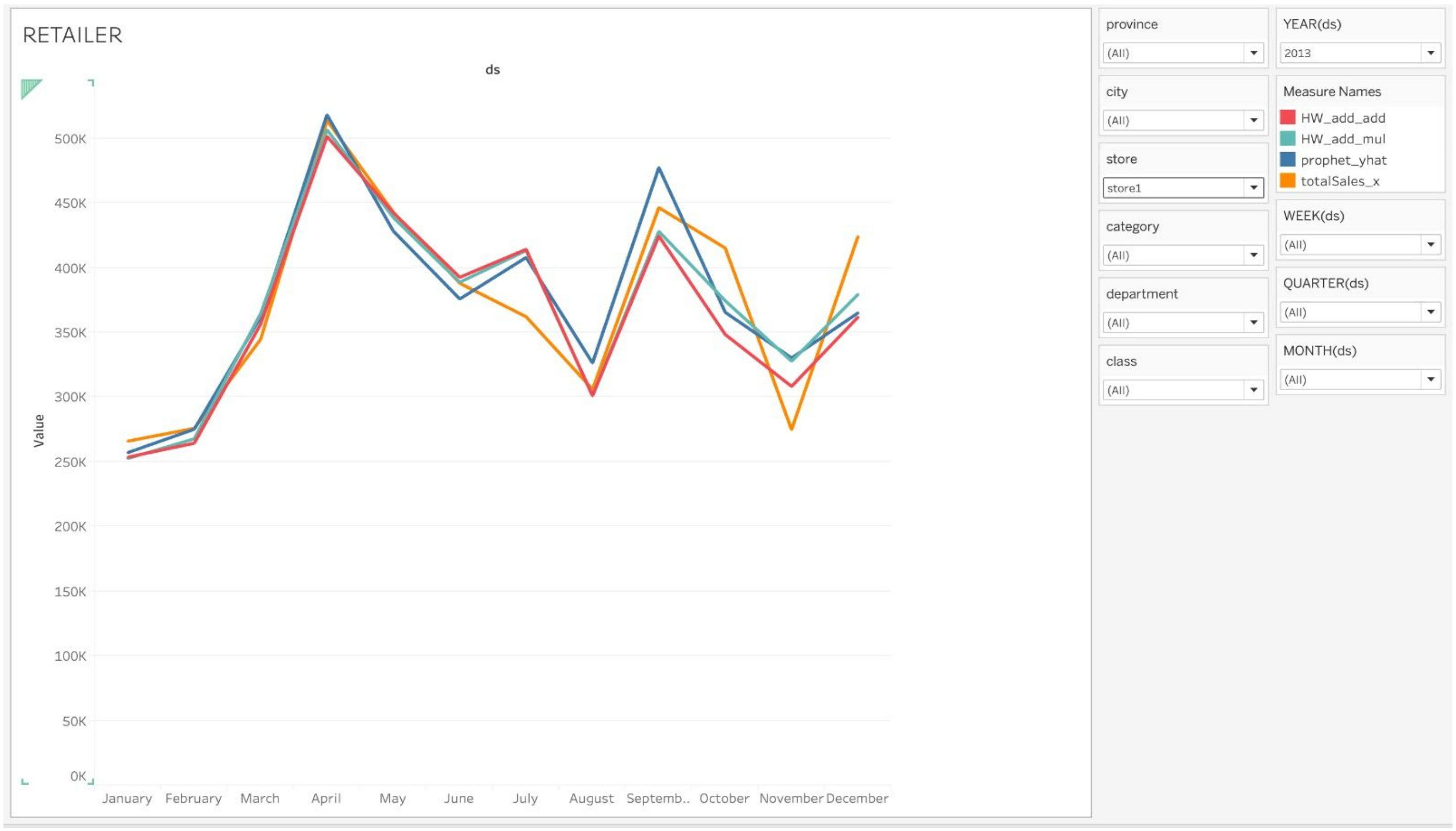
We recorded the following results -
We started with the bottom-up as it is pretty easy to compute. We recorded 1178 correct matches for the best working model out of 2217 time-series. This means that we were correct 1178 times out of 2217 times for the best model on that time-series.

Then we tried the top-down approach. We recorded 524 correct matches out of 2217. Because historical proportions used for disaggregation do not take account of how those proportions may change over time, the top-down approach tends to produce less accurate forecasts at lower levels of the hierarchy.

Then we tried, middle-out approach. We recorded 1439 correct matches out of 2217. Since it combines both the bottom-up and top-down approaches, it gives us an optimal sweet spot of aggregation for accurate forecasts.

Data Product

We have developed multiple python scripts to automate the whole data workflow from data ingestion to forecasting, everything is handled by the scripts. In the end, we get forecasts for all the time-series according to the best model for that time-series. We are using Tableau to present our forecasting results so that the user is able to filter the data based on any aggregation level like - province, city, store, department, category and class values in the window on the right pane. Thus we have successfully automated the selection of a viewing hierarchical forecast. Here is a snapshot of our Tableau Dashboard:



province: (All)
city: (All)
store: store1
category: (All)
department: (All)
class: (All)
YEAR(ds): 2013
WEEK(ds): (All)
QUARTER(ds): (All)
MONTH(ds): (All)

Key Learnings

This project helped us gain an in-depth knowledge of hierarchical time-series forecasting. Some of which are listed below:

- **Stationarity and Seasonality:** A stationary time series is one whose properties do not depend on the time at which the series is observed, i.e. it has constant mean and variance, and covariance is independent of time. Seasonality refers to patterns in the time series which get repeated within a fixed time period. We observed how the performance of our models improved taking these factors into consideration.

- **Segmentation:** We learned how the classification of all time-series in a hierarchy into 4 categories based on demand regularity and variation in quantities can improve the forecast accuracy.

- **Different forecasting models:** We applied several models to find the best suitable model for all time-series at different hierarchical levels. We learned about Prophet, ARIMA and Exponential Smoothing using Holt's Winter and how to optimize the results of these models by choosing the appropriate parameters.

- **Different approaches:** To generate consistent forecasts for different hierarchical levels, we tried 3 approaches namely Bottom-up, Top-down and Middle-out which are explained above in detail.

Summary

Time-series data is analyzed to determine the long term trend to forecast the future or perform some other kind of analysis. Moreover, hierarchical time-series requires preserving the relationships between different aggregation levels and also within the same hierarchy. We have developed an automated system that can generate consistent forecasts at different aggregation levels by choosing the model which generates the best forecasts for a particular time-series.

References

[1]https://arxiv.org/pdf/1907.01960.pdf
[2]https://otexts.com/fpp2/
[3]https://www.statsmodels.org/stable/examples/notebooks/generated/exponential_smoothing.html
[4]https://www.researchgate.net/publication/320089625_Spare_parts_management_for_irregular_demand_items