Capstone Project:

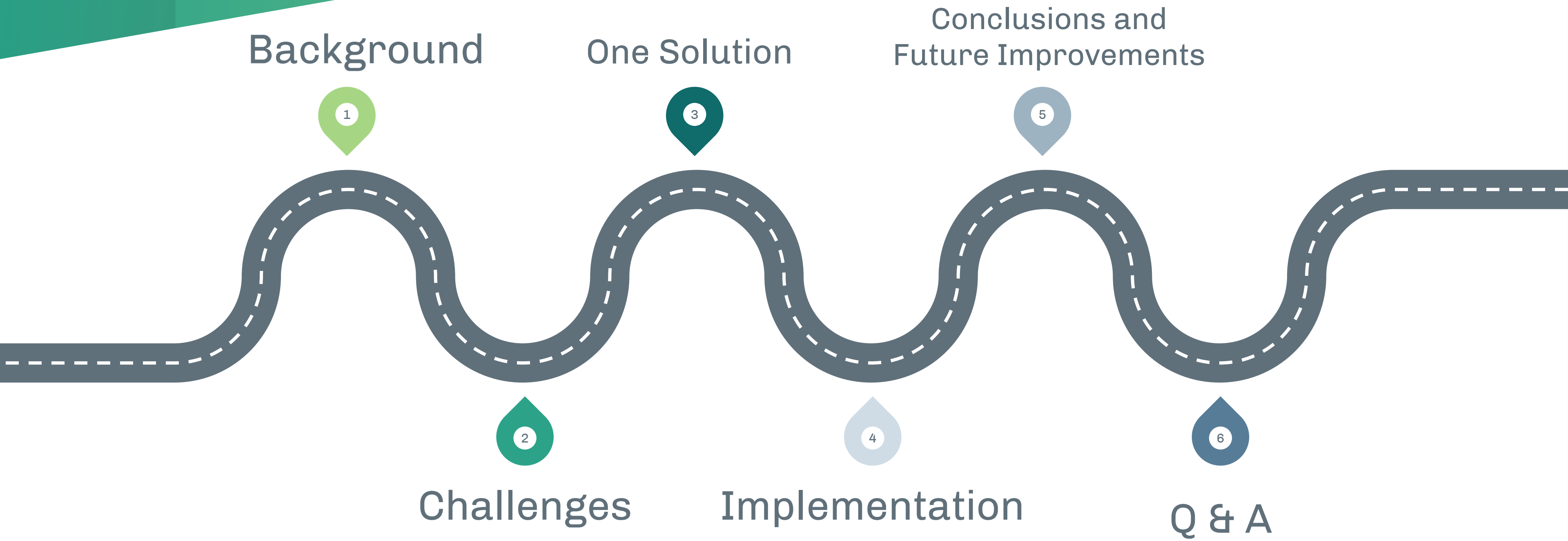
Predict Future Sales

- **Hierarchical Forecasting & Reconciliation**

Problem Statement

To build robust models to forecast monthly sales for each product in each retail store of 1C company, provided that most historical sales data are intermittent and sparse

Roadmap



1 Background
2 Challenges
3 One Solution
4 Implementation
5 Conclusions and Future Improvements
6 Q & A

Background: 1C company

1C company

A Russian software developer, distributor and publisher based in Moscow.

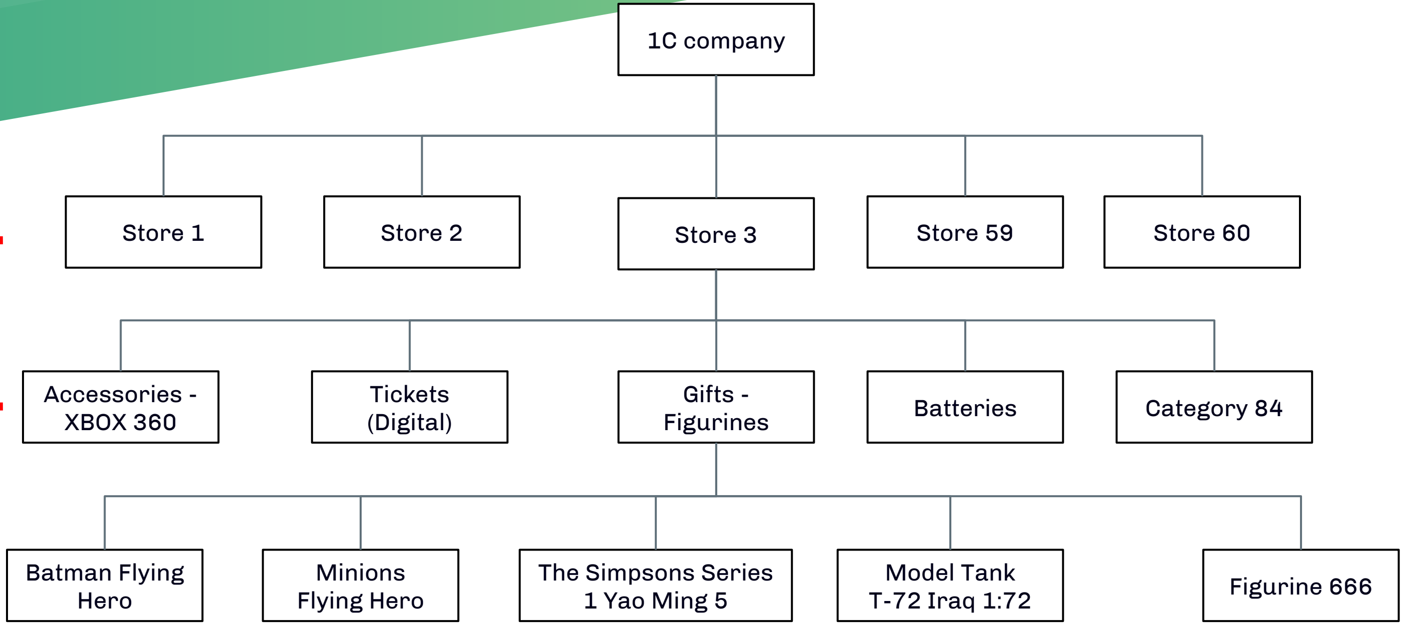
It is also running a retail chain selling computer software, related services, game consoles, video games, etc.

Retail Chain of 1C company

Total 60 stores

Total 84 product categories

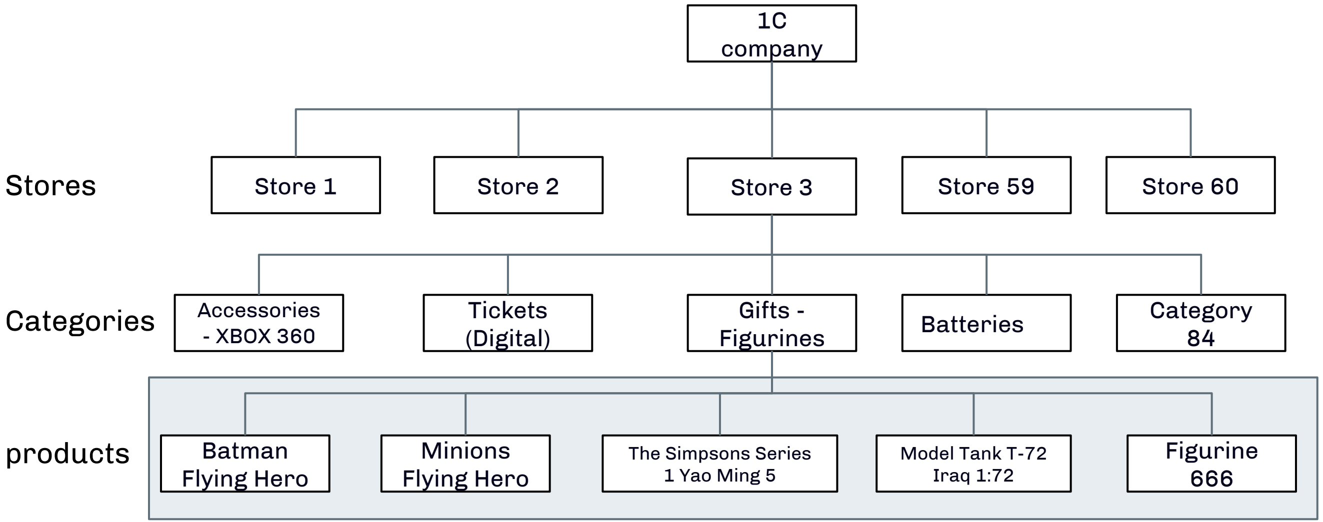
Total 22,170 products



Each store sell different products

Hello, I am Yao Ming

Retail Chain of 1C company



- ❑ We are given daily historical sales data at this level
- ❑ Our task is to predict monthly sales of each product at this level too

Challenges I

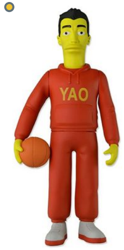
- Massive datasets
  - 2.9 millions records
  - From Jan 2013 to Oct 2015
- Messy data
  - the list of shops and products slightly changed every month
  - Some stores have been shut down while some stores just opened for one month
  - Some stores have very similar name, and seems to be same store
    - Zhukovsky st. Chkalova 39m?
    - Zhukovsky st. Chkalova 39m^2
  - Quite a lot new arrival products without any historical sales data

Challenges II

I had 25 transactions in all shops in 2015!

- Intermittent and sparse historical sales data
    - Most products only have occasional transactions
    - The sales time series data contains many embedded zeros
        - Have no clearly defined trend
        - Does NOT exhibit any seasonal behavior
        - Very difficult for a conventional time series model such as ARIMA to forecast.

I only had 1 transaction in 2015



One solution:

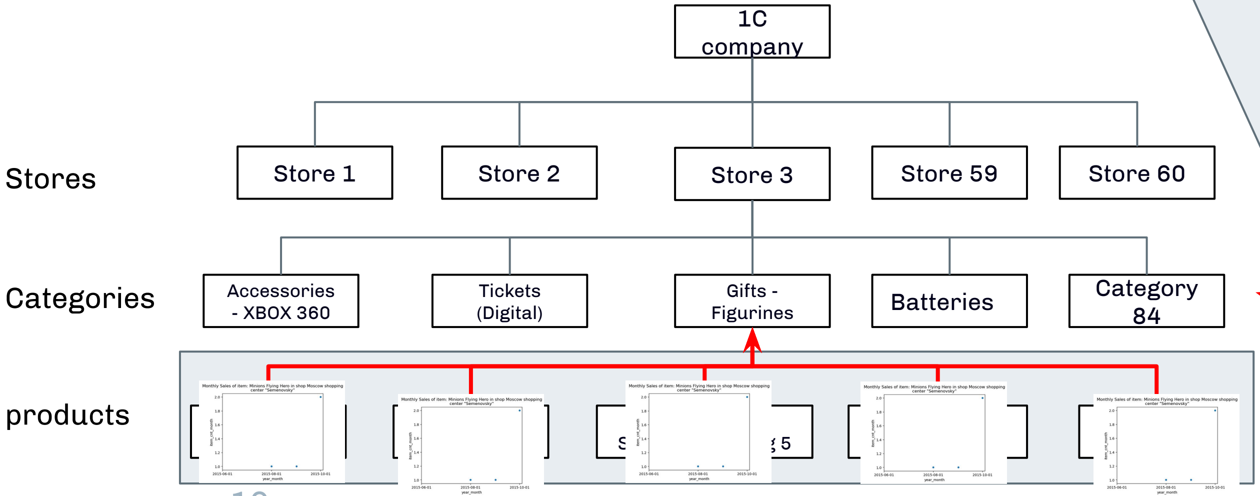
Hierarchical Time Series Forecasting & Reconciliation

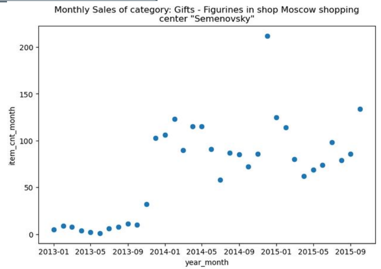
- Step 1: Building Hierarchical Time Series:
- Step 2: Hierarchical Forecasting:
- Step 3: Forecast reconciliation:

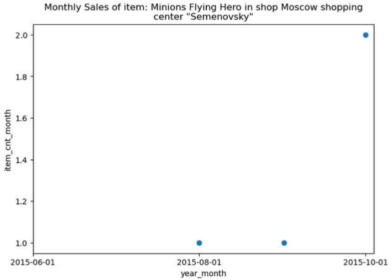
Building Hierarchical Time Series:

Step 1: Aggregation From Bottom Level

Sales data at bottom level can be aggregated up to middle level.



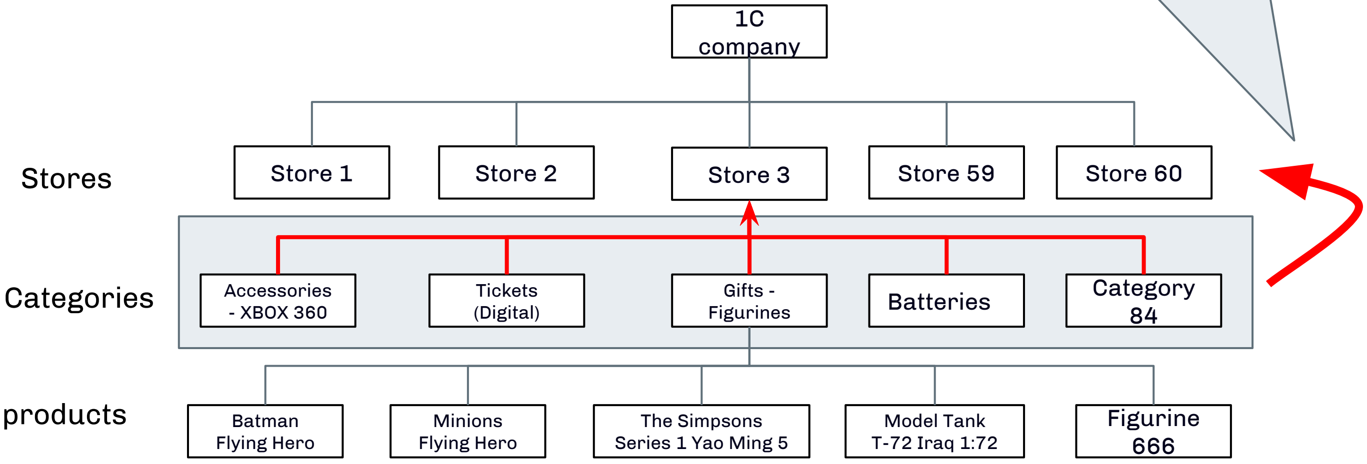


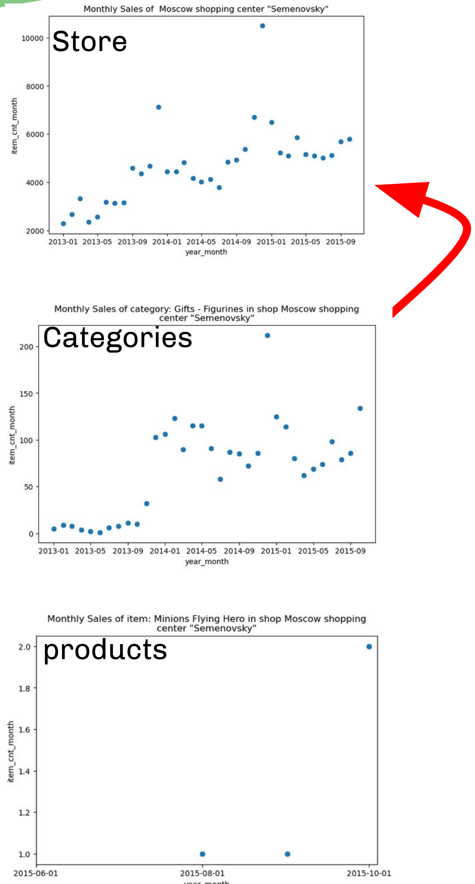


Building Hierarchical Time Series:

Step 2: Aggregation From Middle Level

Sales data at middle level can be aggregated up to store level.





Building Hierarchical Time Series:

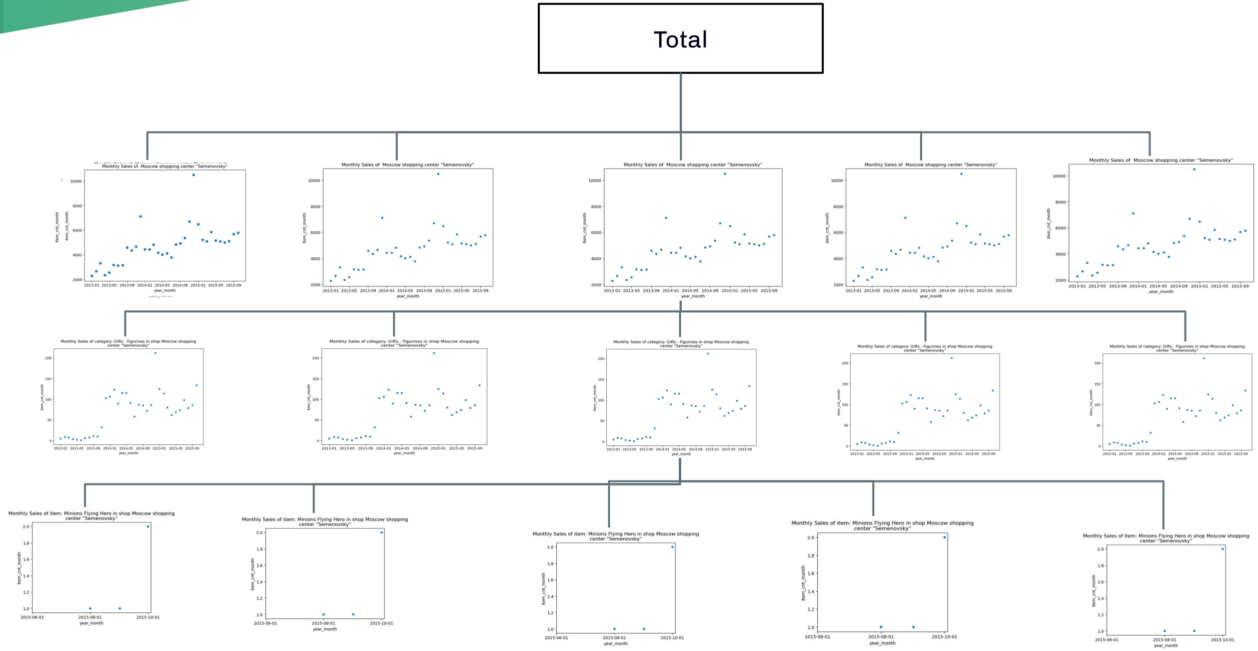
Done!

Hierarchical Time Series

Stores

Categories

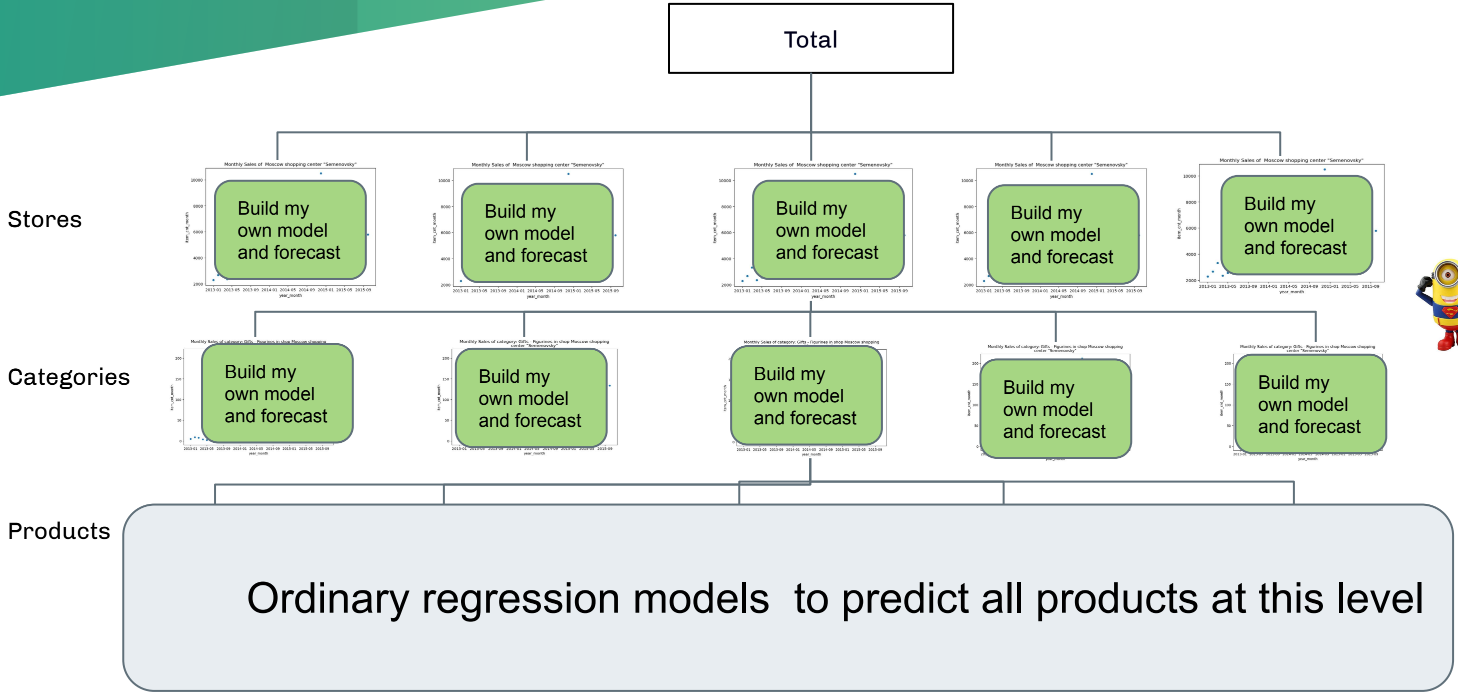
Products



Aggregate

Hierarchical Forecasting:

Hierarchical Forecasting



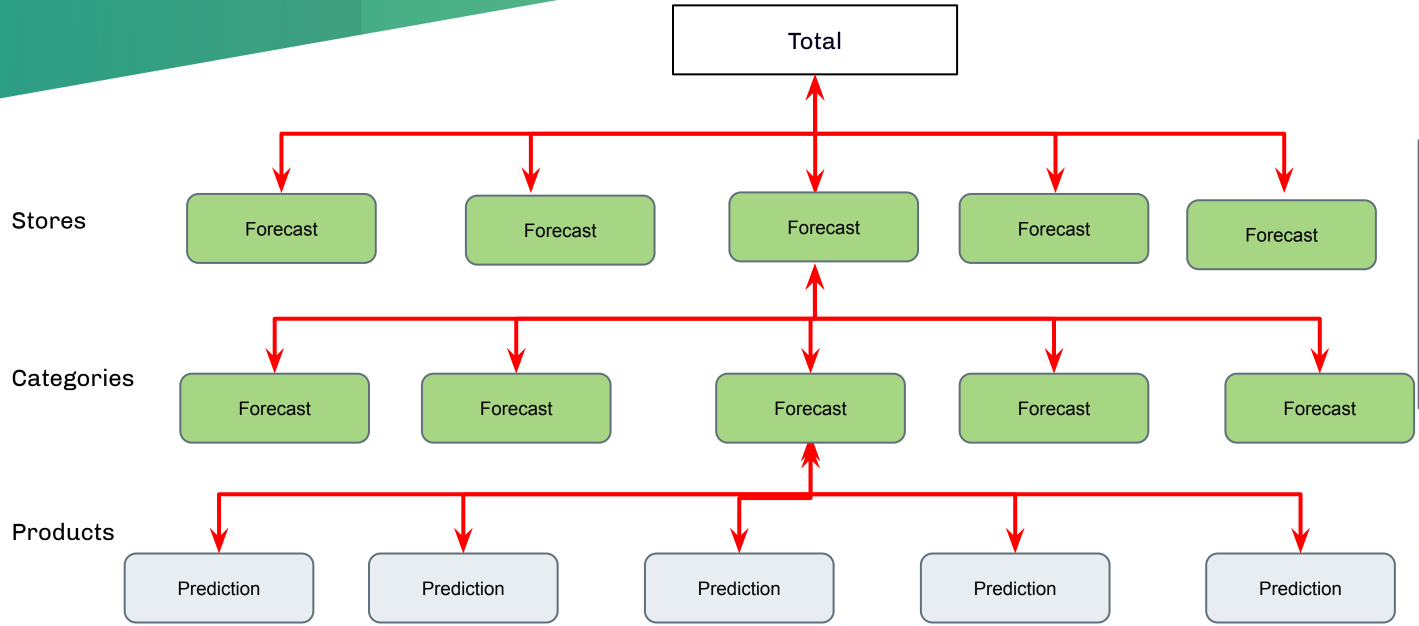
Time Series at category level and store level have noticeable trends and seasonality. We can build time series models on each time series

Without noticeable trends and seasonality, we need to use ordinary regression algorithms such as LightBGM to predict all product sales.
exogenous variable:

- Product name
- Product price
- Lagged data

Forecast Reconciliation:

Forecasting Reconciliation

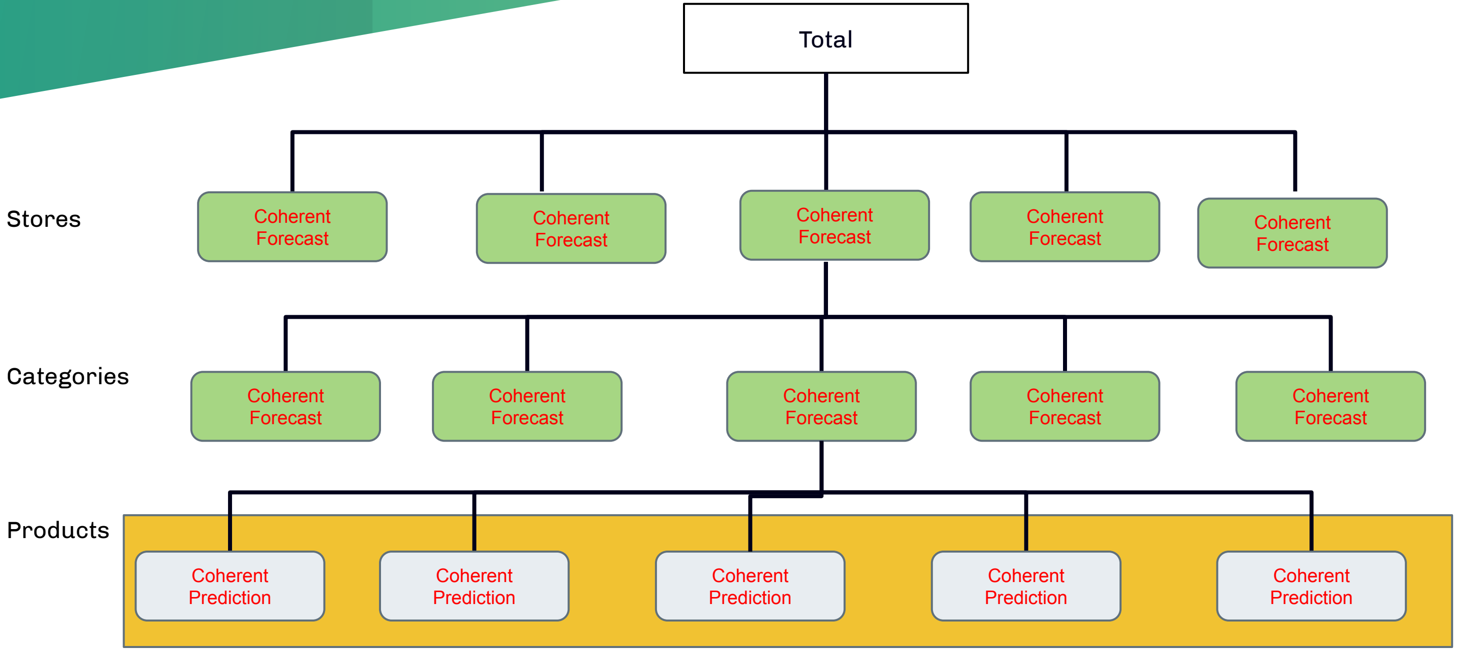


Reconciliation:

- the process of adjusting forecasts to make them coherent and sum up correctly in the hierarchy

Forecast Reconciliation:

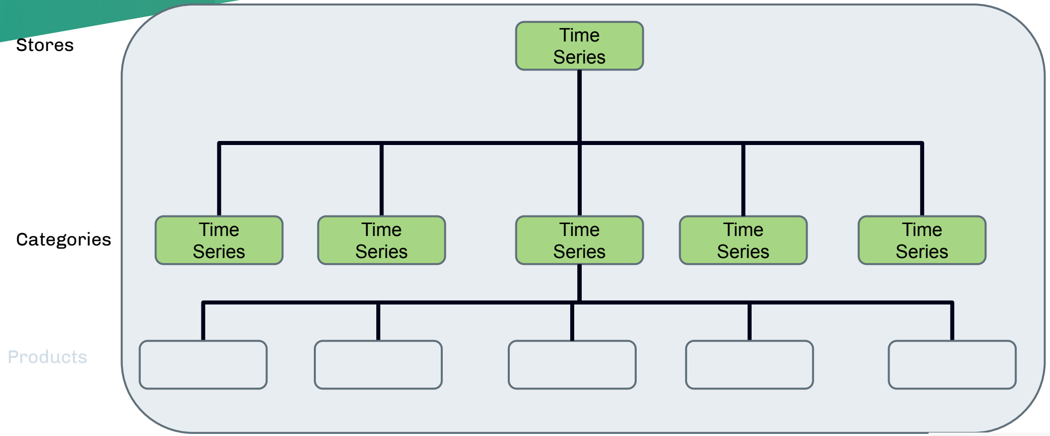
Done!



Implementation

- Build Hierarchical Time Series
- Top Levels Time Series Modeling and Forecasting
- Bottom level modeling and performance
- Forecasting Reconciliation

Implementation of Building Hierarchical Time Series

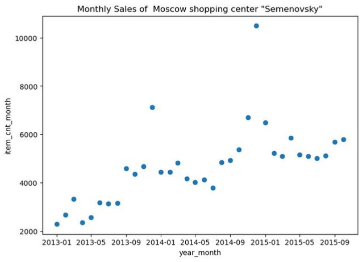


- Due to memory constraints, we can only build hierarchical time series up to individual store level
- Python Package: scikit-hts
- Sample Code:

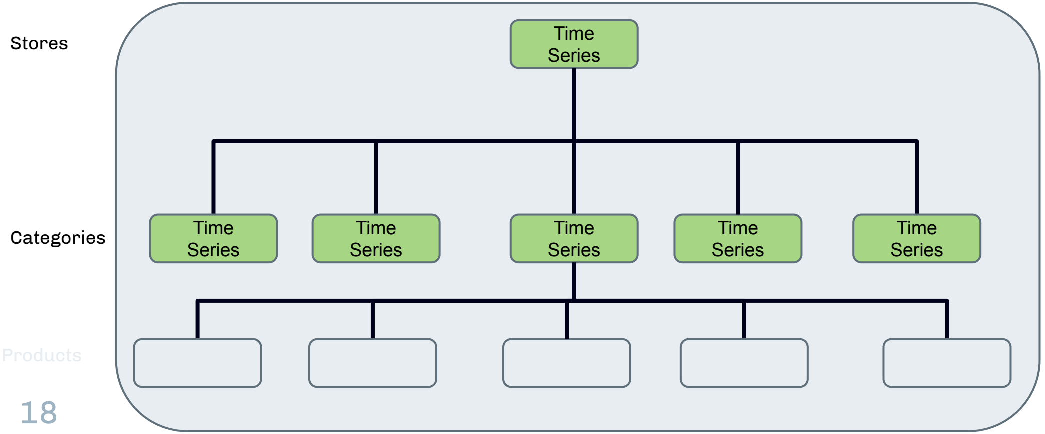
```
################ create hierachical time series for the shop
##################################################################
level_names = ['item_category_id','item_id']
hierarchy = [['item_category_id']]
df_hier, sum_mat, sum_mat_labels = hts.functions.get_hierarchichal_df(df_oneshop,
                                                                      level_names=level_names,
                                                                      hierarchy=hierarchy,
                                                                      date_colname='year_month',
                                                                      val_colname='item_cnt_month')
```

We have built 41 such hierarchical trees for 41 stores. There are around 3,444 time series created in total

Implementation of Top Levels Modeling



Time Series at top levels have noticeable trends and seasonality. We can build time series models on each time series



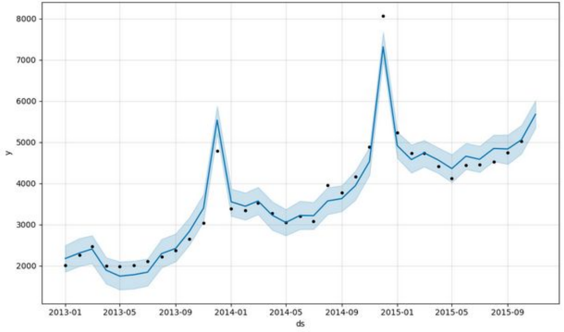
- Python Package: Facebook Prophet
- Took into consideration:
  - yearly-Seasonality
  - Holidays
  - Hyperparameter tuning
    - Changepoint_prior_scale
    - Seasonality_prior_scale
    -
- Sample Codes:

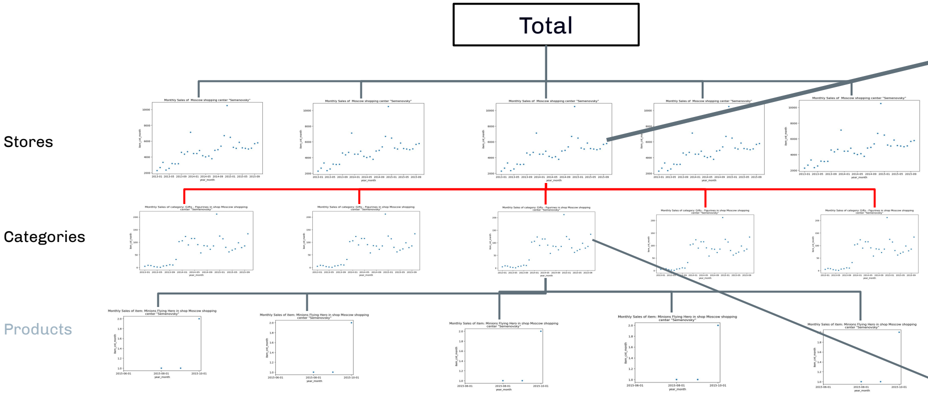
```
#create model based on best_params
m = Prophet(changepoint_prior_scale=best_params['changepoint_prior_scale'],
            seasonality_prior_scale=best_params['seasonality_prior_scale'],
            yearly_seasonality=True)
m.add_country_holidays(country_name='RU')
m.fit(df)
```

Top levels modeling and performance

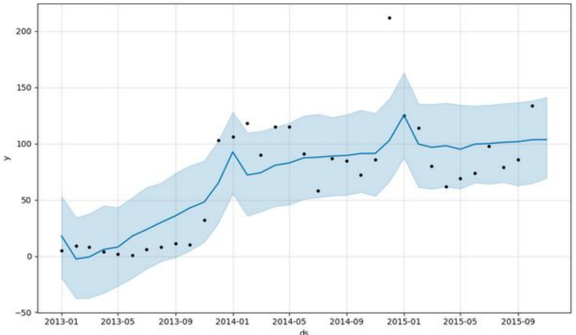
```
forecasting shop: s31

getting best preds for forecast_total_sales
{'changepoint_prior_scale': 0.01, 'seasonality_prior_scale': 0.5}
           ds      yhat  yhat_lower  yhat_upper
32 2015-09-01 4,835.13    4,465.94    5,178.16
33 2015-10-01 5,057.05    4,718.74    5,407.77
34 2015-11-01 5,676.31    5,345.01    6,008.91
```

Facebook Prophet Forecasting
at store level





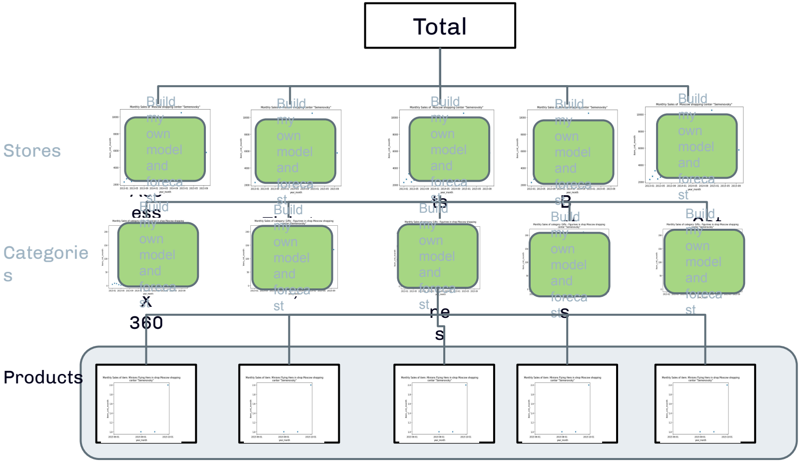

```
getting best preds for s31_ c72
{'changepoint_prior_scale': 0.2, 'seasonality_prior_scale': 0.01}
           ds    yhat  yhat_lower  yhat_upper
32 2015-09-01  101.75       62.76      136.62
33 2015-10-01  103.49       64.70      138.14
34 2015-11-01  103.61       69.35      141.08
```



We have built 3,444 models for every time series in top levels of every stores

Facebook Prophet Forecasting
at category level

Implementation of Bottom level modeling and performance



- Intermittent and sparse data:
  - Can not use time series models same as top levels
- Used ordinary regression models:
  - LightGBM
  - XGBRegressor
  - StackingRegressor
- Made use of different exogenous variables to improve prediction:
  - Product name (TfidfVectorizer)
  - Product price
  - Lagged data
  - etc

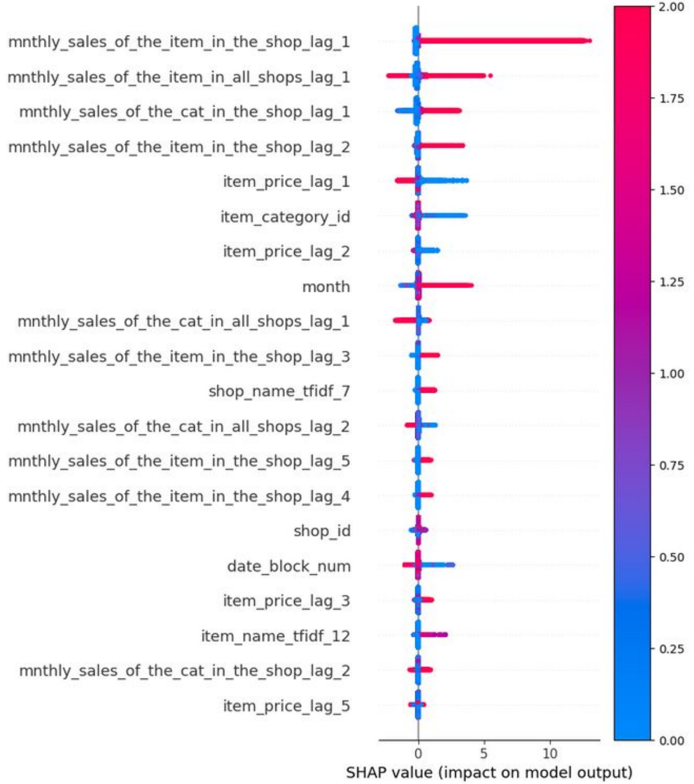
Bottom level modeling and performance

	Training Set (RMSE)	Testing Set (RMSE)	Generalization %	Kaggle Score (RMSE)
LightGBM	1.08	0.93	14.38	0.94407
XGBRegressor	0.97	0.94	3.10	0.96352
StackingRegressor	0.99	0.92	7.17	0.94510

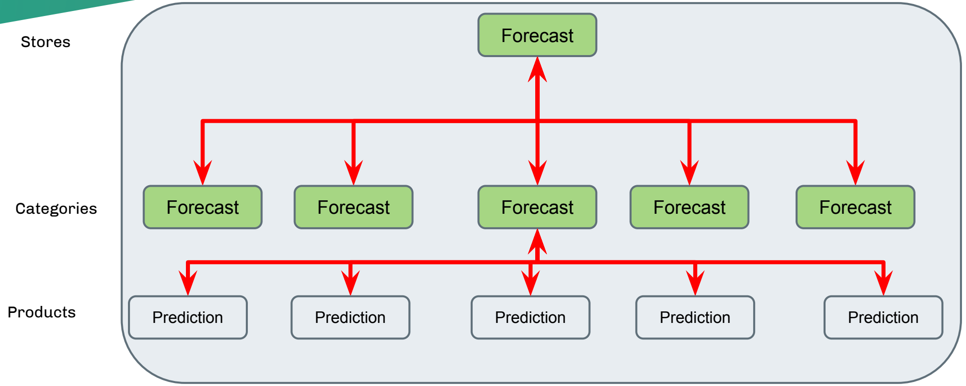
Shap values of model “LightGBM”

- **Shapley value:**
  - **the average of the marginal contributions of a feature across all permutations**

- **Most important features:**
  - **Previous monthly sales of the product in the shop**
  - **Previous monthly sales of the product in all shops**
  - **Previous monthly sales of the product category in the shop**



Implementation of Forecast Reconciliation




```
################## reconciliation
####################################################
pred_dict = collections.OrderedDict()
for label in sum_mat_labels:
        pred_dict[label] = pd.DataFrame(data=forecasts[label].values, columns=['yhat'])
revised = hts.functions.optimal_combination(pred_dict, sum_mat, method='WLSS', mse={})
revised_forecasts = pd.DataFrame(data=revised[0:,0:],
                                 index=forecasts.index,
                                 columns=sum_mat_labels)
```

- Reconciliation was carried out on store by store
- Python package:
  - scikit-hts
- reconciliation strategies:
  - Ordinary least squares (OLS) :
    - minimises the total OLS within all coherent forecasts in the hierarchy
  - Structurally weighted least squares (WLSS):
    - minimises the total WLSS within all coherent forecasts in the hierarchy
  - Variance-weighted least squares (WLSV):
    - minimises the total WLSV within all coherent forecasts in the hierarchy

Forecast Reconciliation and Performance

	Before Reconciliation	After Reconciliation					
		Reconciliation Strategy: OLS		Reconciliation Strategy: WLSS		Reconciliation Strategy: WLSV	
	Kaggle Score (RMSE)	Kaggle Score (RMSE)	Improvement %	Kaggle Score (RMSE)	Improvement %	Kaggle Score (RMSE)	Improvement %
LightGBM	0.94407	1.21477	-28.67	0.93976	0.47	-	-
XGBRegressor	0.96352	1.22888	-27.54	0.95546	0.84	-	-
StackingRegressor	0.94510	1.21736	-28.81	0.94063	0.47	-	-

Bug

OLS strategy produced many negative results. It made the coherent prediction worse

WLSS strategy produced better results in overall

We encountered some errors when running WLSV strategy

Analysis of Forecast Reconciliation Performance

	Before Reconciliation	After Reconciliation					
	Kaggle Score (RMSE)	Reconciliation Strategy: OLS		Reconciliation Strategy: WLSS		Reconciliation Strategy: WLSV	
		Kaggle Score (RMSE)	Improvement %	Kaggle Score (RMSE)	Improvement %	Kaggle Score (RMSE)	Improvement %
LightGBM	0.94407	1.21477	-28.67	0.93976	0.47	-	-
XGBRegressor	0.96352	1.22888	-27.54	0.95546	0.84	-	-
StackingRegressor	0.94510	1.21736	-28.81	0.94063	0.47	-	-

Bug

- The Improvement of reconciliation is not encouraging
- Possible reasons:
  - Due to memory constraints, we just built hierarchical time series up to one store, which may not able to reflect macro trends or seasonalities
  - Limited hyperparameter tuning
  - Many products/categories have too limited sales transactions
  - Many state-of-the-art reconciliation strategies have not supported by Scikit-hts
    - MinTSample
    - MinTShrink

Conclusions

- Hierarchical Forecasting & Reconciliation did improve the overall prediction of intermittent time series with correct reconciliation strategy
- Hierarchical Forecasting & Reconciliation can apply to any time series with a hierarchy structure
- Scikit-hts offers a lot of flexibilities
  - Assist to build hierarchical time series from bottom level data
  - We can choose different ML algorithms for different levels of hierarchical time series
  - 3 reconciliation strategies are available at this moment.

Future Improvements

- To build a bigger hierarchy to contain all data
- To try different reconciliation strategies:
  - Variance Scaling
  - MinTSample
  - MinTShrink
- To try different time series models:
  - LSTM
  - N-BEATS
- More feature engineering for bottom level modelling

Q & A

Thank you!!