Time-Series Forecasting

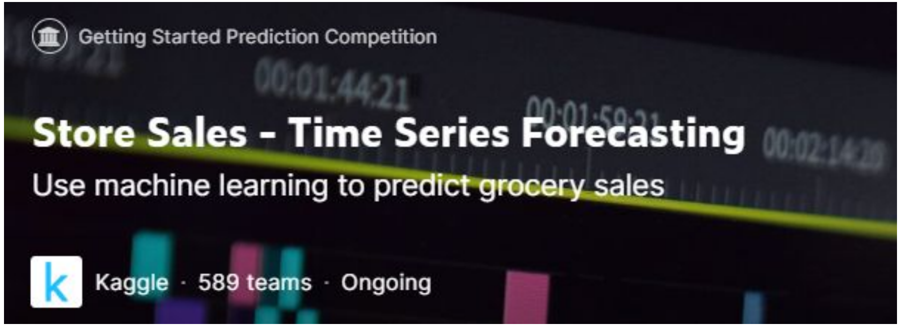
using Machine Learning to predict grocery store sales

Project Work on Data Mining M

Pasit Khantigul

Contents

Goal of the project



Problem: time-series prediction problem

presented as a Kaggle competition.

Time-series: a set of observations recorded over time.

Goal: use machine learning to forecast *store sales* on data from Corporación Favorita:

➔ forecast for the **next 15 days** (from the last day of the training data)
➔ get lowest score from submissions

$$\text{RMSLE} : [\sqrt{\tfrac{1}{n} \textstyle\sum_{i=1}^{n} \left(\log(1+\hat{y}_i) - \log(1+y_i)\right)^2}]$$

by **modelling** both:

- time dependence features
- serial dependence features

hybrid model

Datasets

1. training.csv
2. transaction.csv
3. stores.csv
4. holiday_events.csv
5. oil.csv
6. test.csv
7. sample_submission.csv

store_nbr	family	date	sales
1	AUTOMOTIVE	2013-01-01	0.000000
		2013-01-02	2.000000
		2013-01-03	3.000000
		2013-01-04	3.000000
		2013-01-05	5.000000
...	...	...	...
9	SEAFOOD	2017-08-11	23.830999
		2017-08-12	16.859001
		2017-08-13	20.000000
		2017-08-14	17.000000
		2017-08-15	16.000000

.to_period('D')

store_nbr	family	date	id	onpromotion
1	AUTOMOTIVE	2017-08-16	3000888	0
		2017-08-17	3002670	0
		2017-08-18	3004452	0
		2017-08-19	3006234	0
		2017-08-20	3008016	0

date	type	locale	locale_name	description	transferred
2012-03-02	Holiday	Local	Manta	Fundacion de Manta	False
2012-04-01	Holiday	Regional	Cotopaxi	Provincializacion de Cotopaxi	False
2012-04-12	Holiday	Local	Cuenca	Fundacion de Cuenca	False
2012-04-14	Holiday	Local	Libertad	Cantonizacion de Libertad	False
2012-04-21	Holiday	Local	Riobamba	Cantonizacion de Riobamba	False

EDA results

- Total duration of data:

  Training data: 2013-01-01 --> **2017-08-15**

  forecast origin

  Test data: **2017-08-16** --> 2017-08-31

  forecast horizon

  valid test size = 15

- Total number of families (of products): **33**
  - some family does not have any sales for some stores -> forecast easily to 0 sales.

- Total number of stores: **54**

- The top 5 most sold are Grocery, beverages, cleaning, dairy, and produce. Grocery and beverage account for more than 50% of total sales.

- Correlation between oil prices and (avg) sales and (avg) transaction suggests that the country's economic status and everyday grocery consumption do not have a particular relationship.

	dcoilwtico	sales	transactions
dcoilwtico	1.000000	-0.500805	0.04319
sales	-0.500805	1.000000	0.37651
transactions	0.043190	0.376510	1.00000

Modelling time dependence

A series is *time dependent* if its values can be predicted from the time they occured - from **time-step features**, which can be modelled from:

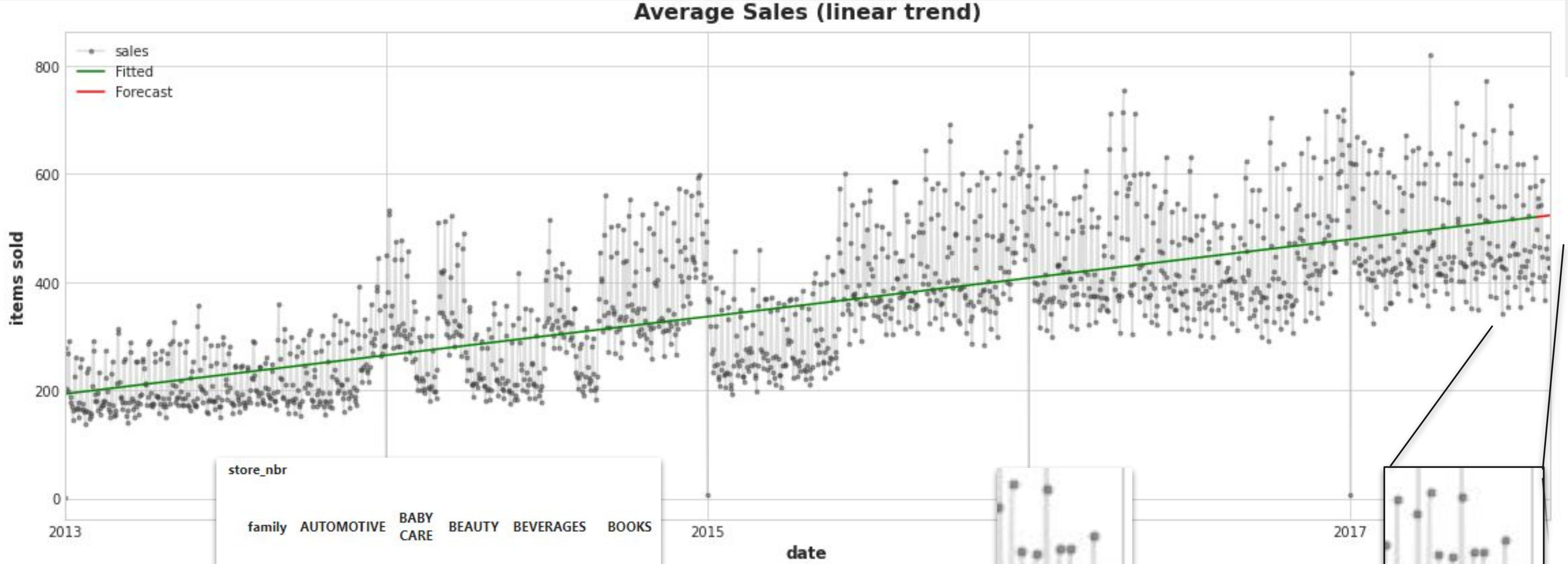
➔ **Trend :** a persistent, long-term change in the mean of the series, estimated from the Moving Average Plot.

```
trend = avg_sales.rolling(
    window=365, #smooth over
    center=True,
    min_periods=183,
).mean()
```

```
y = avg_sales.copy()
dp = DeterministicProcess(
    index=y.index,
    constant=True,
    order=1,
    drop=True,
)

X = dp.in_sample()
X.head()
```

```
X_train, X_valid, y_train, y_valid = train_test_split(X, y, test_size=15, shuffle=False)

model = LinearRegression(fit_intercept=False).fit(X_train, y_train)

y_fit = pd.DataFrame(model.predict(X_train), index=X_train.index, columns=y_train.columns).clip(0.0)
y_pred = pd.DataFrame(model.predict(X_valid), index=X_valid.index, columns=y_valid.columns).clip(0.0)

rmsle_train = mean_squared_log_error(y_train, y_fit) ** 0.5
rmsle_valid = mean_squared_log_error(y_valid, y_pred) ** 0.5
```

date	const	trend	trend_squared	trend_cubed
2013-01-01	1.0	1.0	1.0	1.0
2013-01-02	1.0	2.0	4.0	8.0
2013-01-03	1.0	3.0	9.0	27.0
2013-01-04	1.0	4.0	16.0	64.0
2013-01-05	1.0	5.0	25.0	125.0

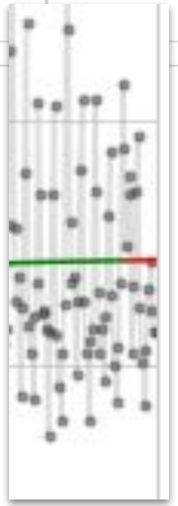


store_nbr					
family	AUTOMOTIVE	BABY CARE	BEAUTY	BEVERAGES	BOOKS
date					
2017-08-01	3.989250	0.0	3.531821	2196.628603	0.716912
2017-08-02	3.987273	0.0	3.534058	2196.550189	0.718472
2017-08-03	3.985285	0.0	3.536296	2196.469286	0.720033
2017-08-04	3.983288	0.0	3.538537	2196.385892	0.721597
2017-08-05	3.981282	0.0	3.540779	2196.300004	0.723162

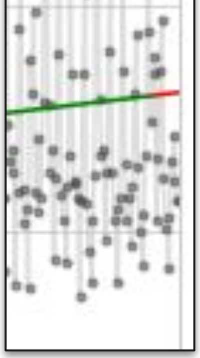
5 rows × 1782 columns

Trend for each family: y_pred =

polynomial order 3 =



polynomial order 1 =



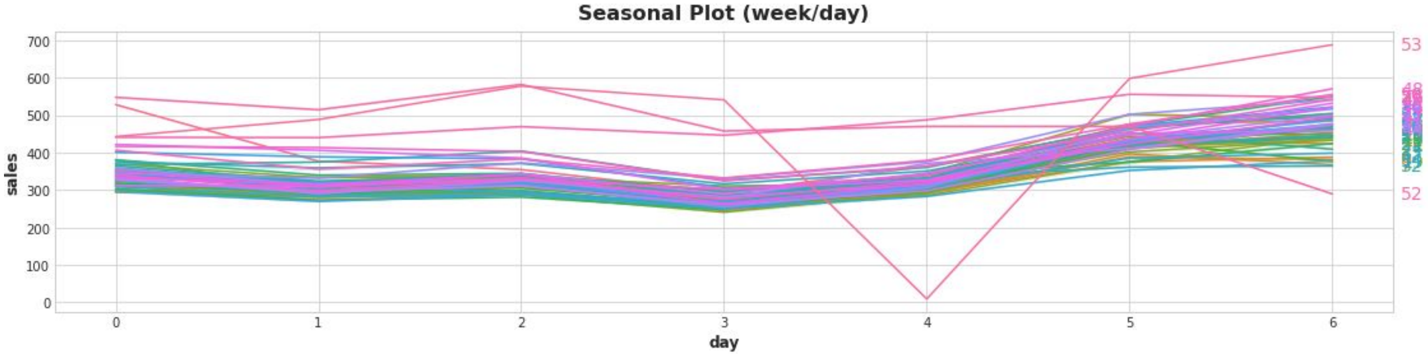
Seasonality

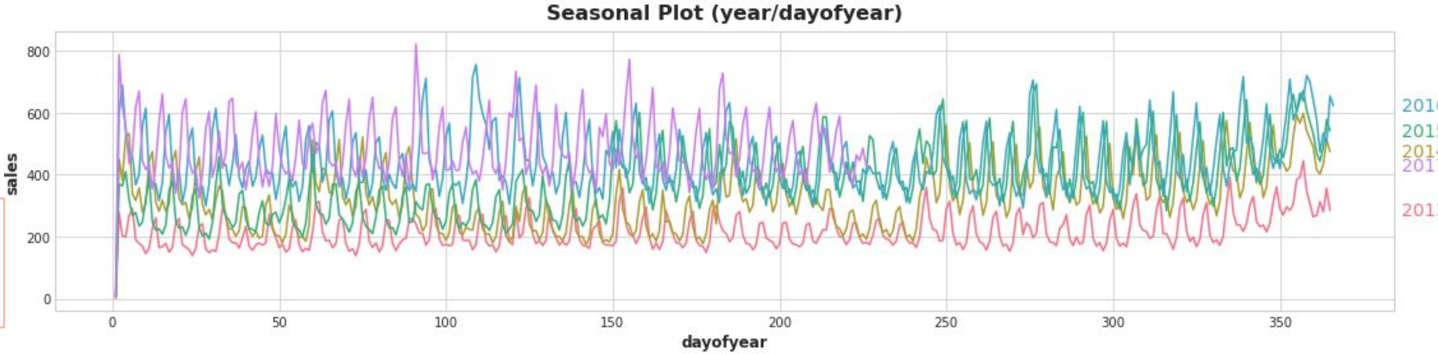
= regular, periodic changes in the mean of the series caused by weekly, monthly, seasonal or yearly patterns.

➔ **Seasonal indicators:**

For a season with few observations represented through *one-hot-encoded categorical features.*

weekly seasonality
= > added 6 features

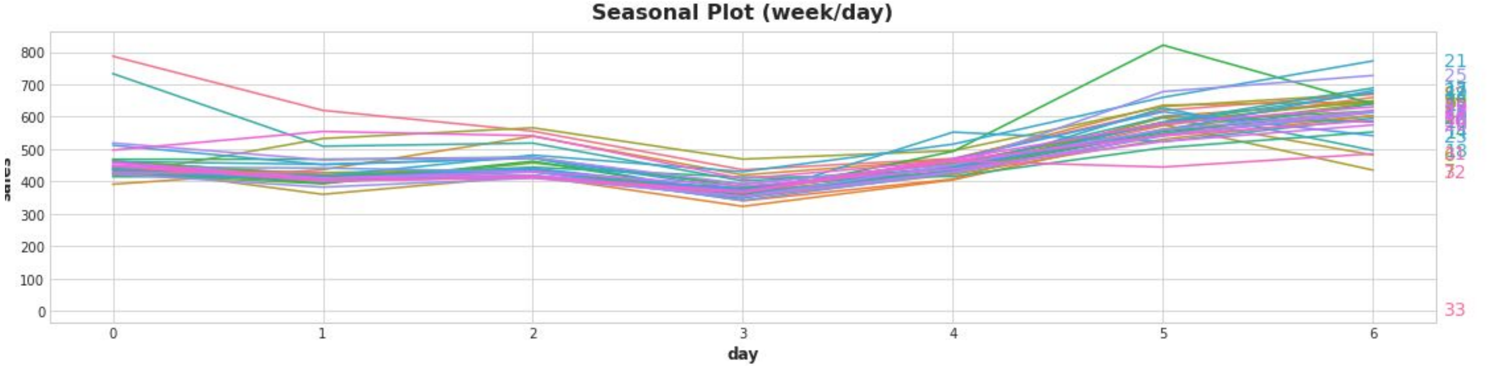


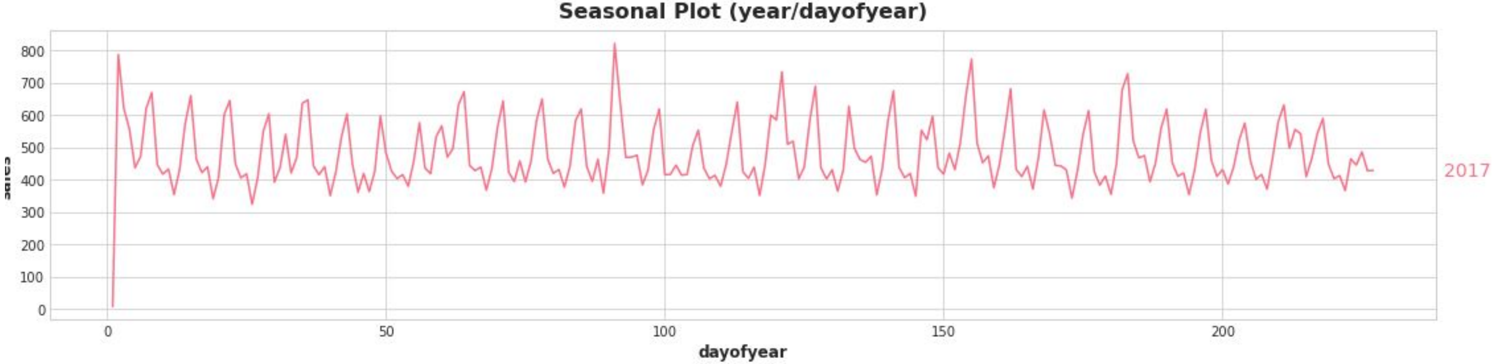


Period restriction of 2017

```
Training RMSLE: 1.16594
Validation RMSLE: 0.58326
```

-> Since the previous model does much better in the validation set than in the training set.





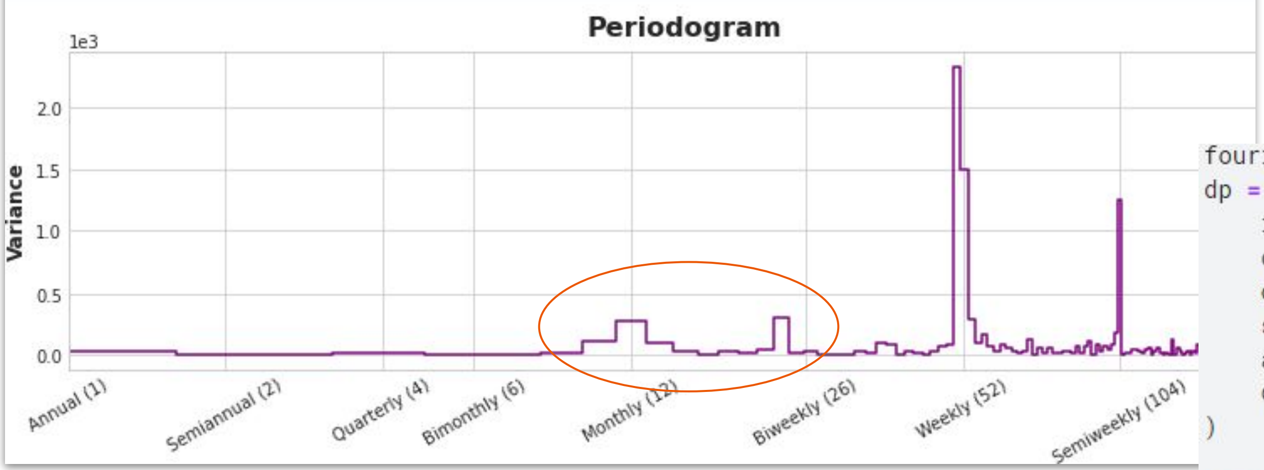
… but for long seasons (with frequent observations)?

➔ **Fourier Feature:** a pair of sine and cosine curves for each potential frequency in the season starting with the longest, to capture the overall shape of the seasonal curve.

How many pairs?

monthly/biweekly seasonality
= > added 2 Fourier features

```
plot_periodogram(avg_sales.loc['2017']);
```




```
fourier_M = CalendarFourier(freq='M', order=4)
dp = DeterministicProcess(
    index=y.index,
    constant=True,                   # dummy featu
    order=1,                         # linear tren
    seasonal=True,                   # weekly seas
    additional_terms=[fourier_M],    # annual se
    drop=True,                       # drop terms
)

X = dp.in_sample()
```

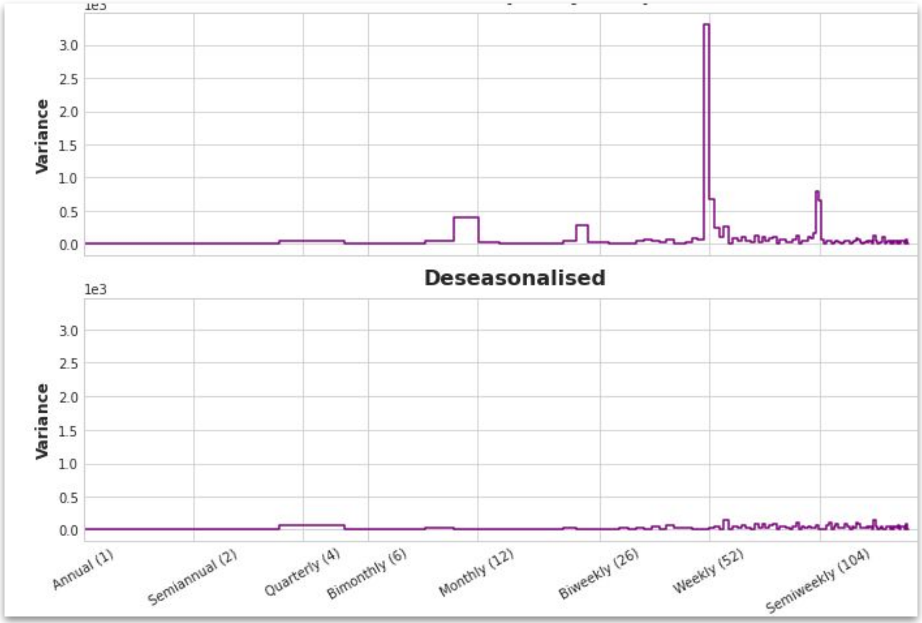
Residual Analysis

A residual of a model are differences between **target** the model was trained on and the **predictions** the model makes, representing what the model failed to learn about the target from the features.

- **Deseasonalising**: to check the effectiveness of seasonality modelling:

  plot the residuals against a feature to get the deseasonalised and detrended plot.

- Hybrid forecasting
  - by considering the additive model :
  - `series = trend + seasons + cycles + error`



Holiday Feature

Selecting only national and regional, from 2017, created as seasonal indicator : `X_holidays = pd.get_dummies(holidays)`

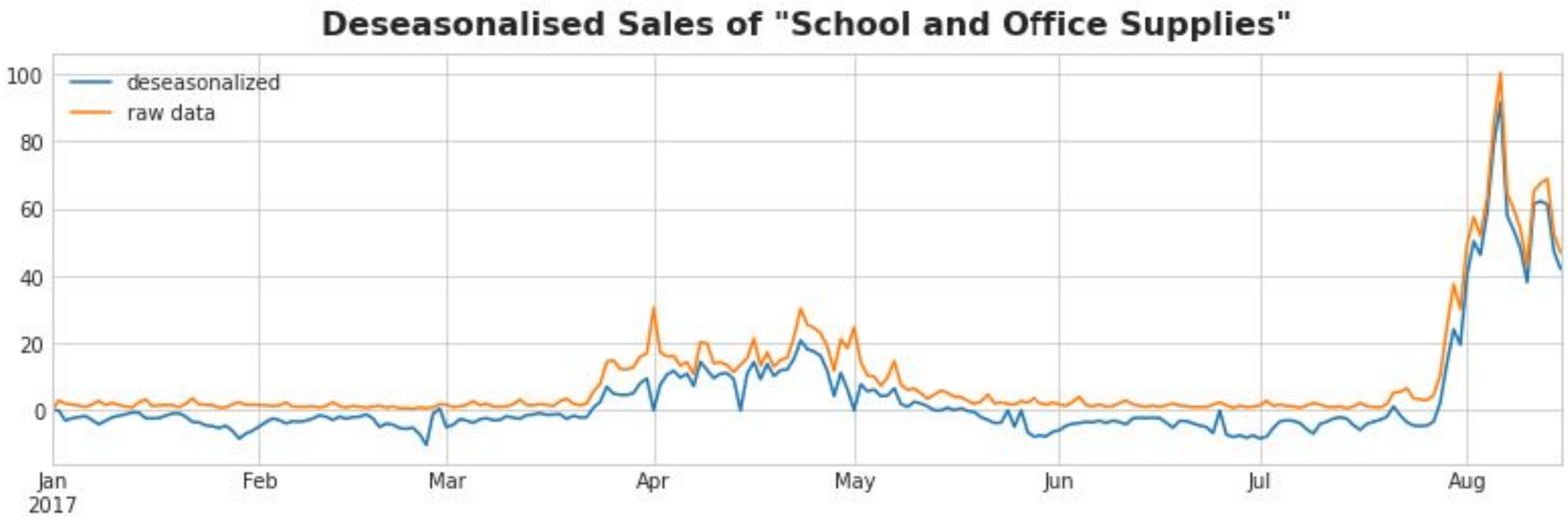
X_1 :

date	const	trend	s(2,7)	s(3,7)	s(4,7)	s(5,7)	s(6,7)	s(7,7)	sin(1,freq=M)	cos(1,freq=M)	...	description_Dia de la Madre-1	description_Dia del Trabajo
2017-01-01	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	1.000000	...	0.0	0.0
2017-01-02	1.0	2.0	1.0	0.0	0.0	0.0	0.0	0.0	0.201299	0.979530	...	0.0	0.0

Modelling serial dependence

A time series has serial dependence when an observation can be predicted from previous observations, what happened in the **recent past** is relevant. Can be linear or non-linear.

➔ **Cyclic behaviour**: pattern of changes associated with how the value at one time depends on values at previous time and not necessarily on time-steps.



Lagged Series and Plots

Lag features are values at prior timesteps obtained by shifting the observations of the target series .

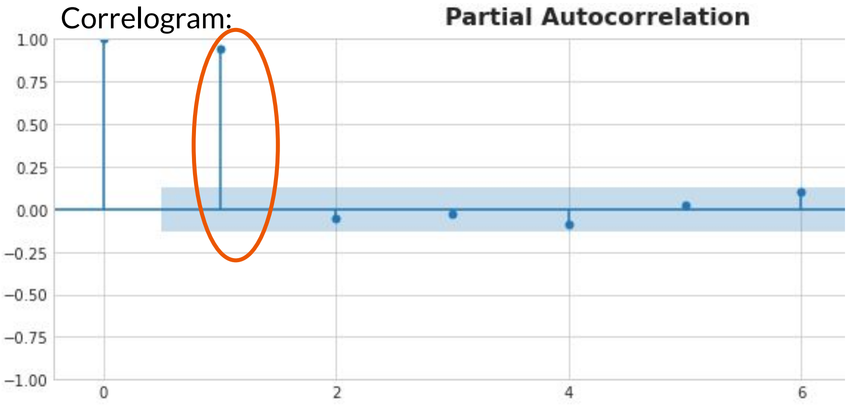
Choosing lags through **Partial Autocorrelation**: shows the amount of new correlation that the lag contributes (Only for linear serial dependency).

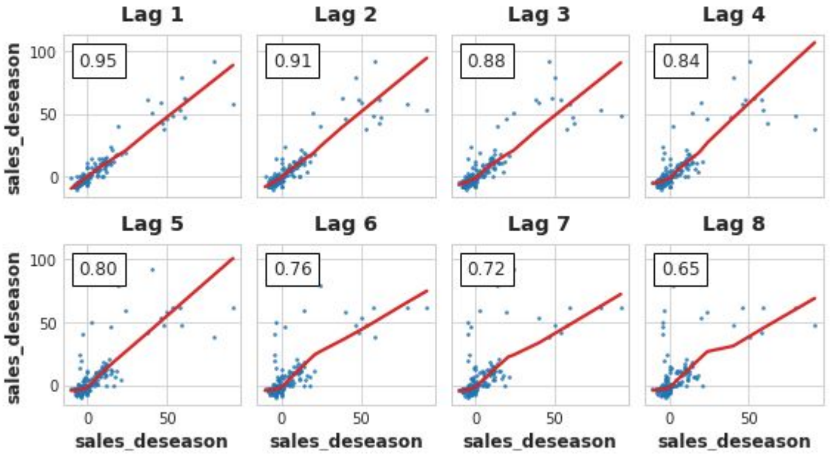
Add Lag 1 feature of Supply Sales

residual supply sales

	sales_deseason	y_lag_1
date		
2017-01-01	-9.436896e-15	NaN
2017-01-02	-1.257173e-14	-9.436896e-15
2017-01-03	-3.050247e+00	-1.257173e-14
2017-01-04	-2.342834e+00	-3.050247e+00
2017-01-05	-1.975145e+00	-2.342834e+00

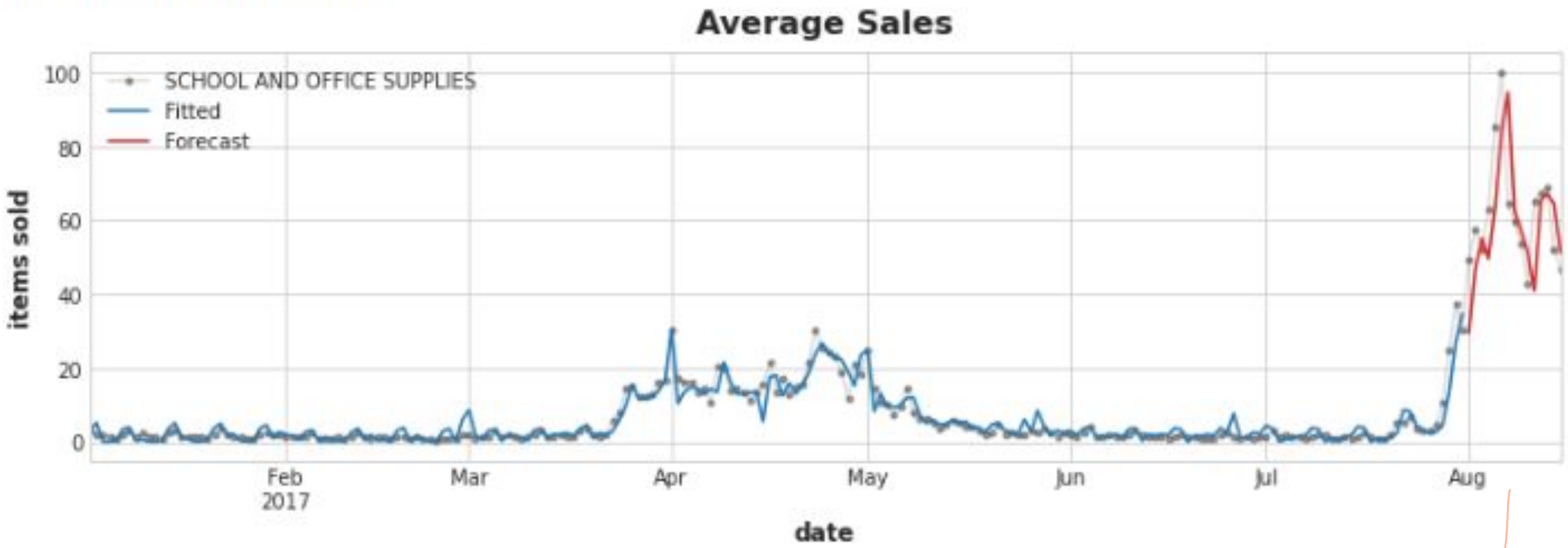
Correlogram:





Training and Forecasting on residual:

```
Training RMSLE: 0.43198
Validation RMSLE: 0.24564
```



Looking for a **leading indicator** providing "early notice" of changes in the target.

Add '**onpromotion' feature**

time-step delay

Hybrid Model: forecaster that combine complementary learning algorithms.

Additive model: series = $\overbrace{\text{trend + seasons}}^{\text{mod\_1}}$ + $\overbrace{\text{cycles}}^{\text{mod\_2}}$ + error

Motivation: Target-transforming algorithm cannot extrapolate trends.

Phases:

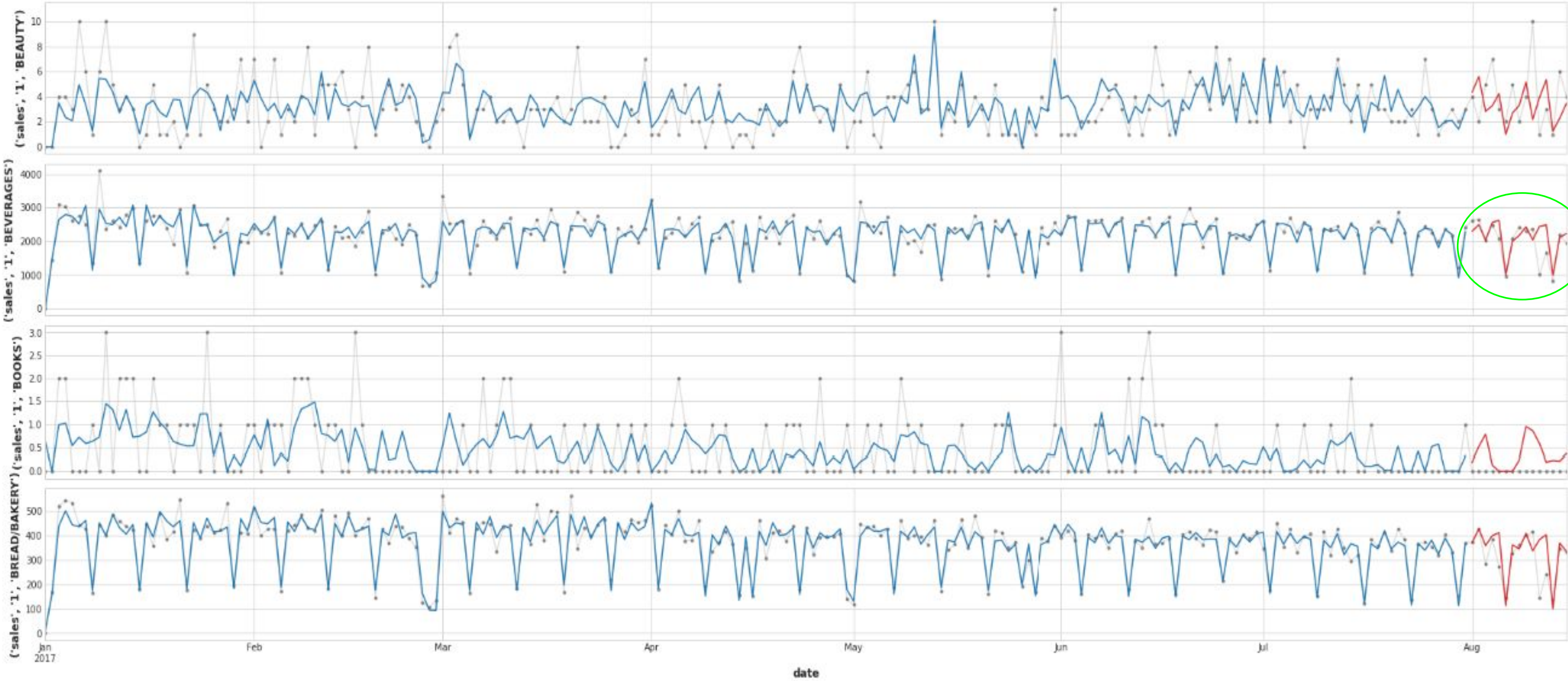
1. Train and predict with first model
2. Train and predict with second model on residuals
3. Add to get overall predictions

```
mod_1 = LinearRegression(fit_intercept=False) # for time
mod_2 = KNeighborsRegressor() # for serial-dependence

model = BoostedHybrid(model_1=mod_1, model_2=mod_2)

model.fit(X_1_train, X_2_train, y_train)
y_fit = model.predict(X_1_train, X_2_train).clip(0.0)
y_pred = model.predict(X_1_valid, X_2_valid).clip(0.0)
```

```
def fit(self, X_1, X_2, y):
    # Train model_1
1   self.model_1.fit(X_1, y)
    # Make predictions
    y_fit = pd.DataFrame(
        self.model_1.predict(X_1),
        index=X_1.index, columns=y.columns,
    )
    # Compute residuals
    y_resid = y - y_fit

    # Train model_2 on residuals
2   self.model_2.fit(X_2, y_resid)
```

```
def predict(self, X_1, X_2):
    # Predict with model_1
    y_pred = pd.DataFrame(
        self.model_1.predict(X_1),
        index=X_1.index, columns=self.y_columns,
    )
    # Add model_2 predictions to model_1 predictions
3   y_pred += self.model_2.predict(X_2)
    return y_pred
```

Forecasting Results (best submission score 0.58129)



Conclusions

- The forecasting results are good, but not for every families.

- The most accurate predictions are for top-sales families of product (e.g. Beverages, Grocery, Cleaning, etc...).

- There are several possible variation of hybrid model design:
  - One model for each family (33 models): to identify different features for each family.
  - One model for each store (54 models): identify feature such as local holidays, analysing ‘store.csv’ dataset.

- The use of different kind of plots are essential to the workflow of Time-Series Forecasting.