M5 Forecasting – Accuracy

Final status report

Team: Economic Science Squad

Ismail Aigunov iaaygunov@edu.hse.ru
Xianghao Li xli@edu.hse.ru

Introduction

Repository: https://github.com/XHaoLi/Project-MLDM

The M5 Competition

Background: The Makridakis Open Forecasting Center (MOFC) at the University of Nicosia conducts cutting-edge forecasting research and provides business forecast training. The MOFC is well known for its Makridakis Competitions, the first of which ran in the 1980s. M5 competition is the fifth iteration.

The **objective** of M5 competition is using hierarchical sales data from Walmart to forecast daily sales for the next 28 days.

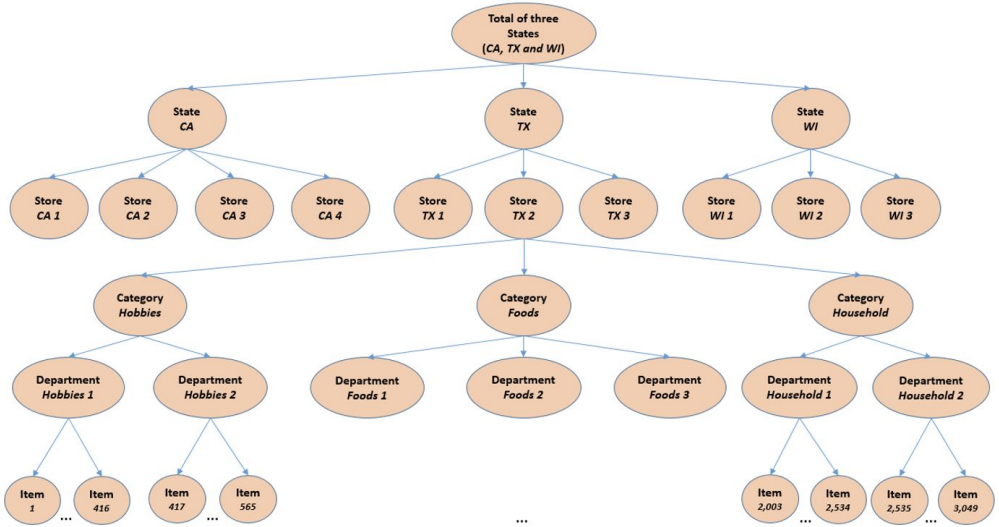
The **data**, covers stores in three US States (California, Texas, and Wisconsin) and includes item level, department, product categories, and store details.

M5 Forecasting - Accuracy: https://www.kaggle.com/competitions/m5-forecasting-accuracy/overview

Introduction

The M5 dataset made available by **Walmart**, involves the unit sales of various products sold in the USA.

Organized in the form of **grouped time series**, involves the unit sales of **3,049 products**, classified in **3 product categories** (Hobbies, Foods, and Household) and **7 product departments**. The products are sold across **ten stores**, located in **three States** (CA, TX, and WI).



Introduction

The historical data range from **2011-01-29** to **2016-06-19**. The products have a (maximum) selling history of 1,941 days / 5.4 years (**test data of h=28 days not included**).

The dataset consists three (3) files: ***"calendar.csv"***, ***"sell_prices.csv"*** and ***"sales_train.csv"***.

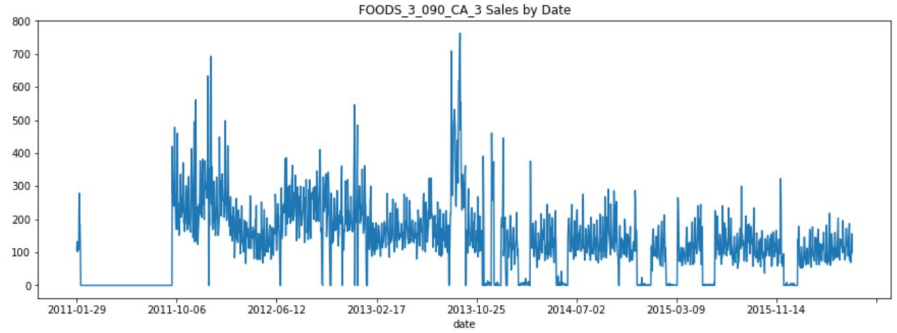
- ***"calendar.csv"*** - information about the dates the products are sold.
- ***"sell_prices.csv"*** - information about the price of the products sold per store and date.
- **"*sales_train.csv*"** - the historical daily unit sales data per product and store.

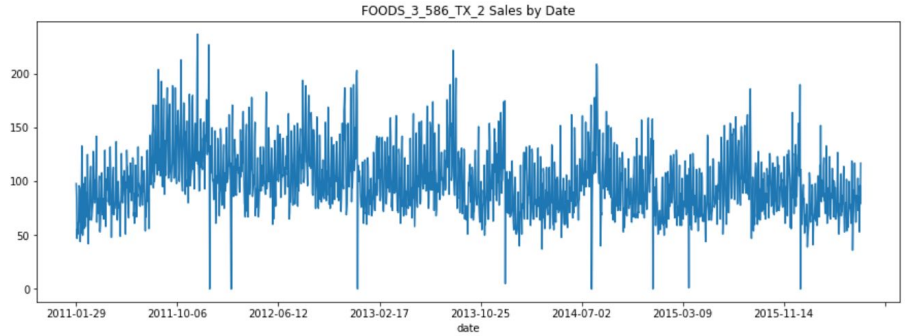
Details can be viewed in the file "Data.ipynb"

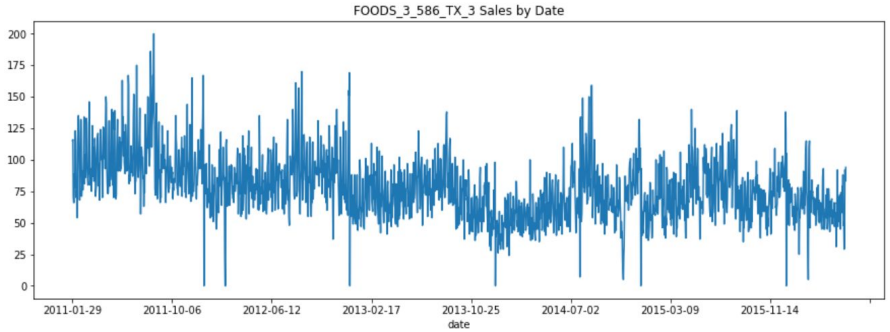
https://github.com/XHaoLi/Project-MLDM/blob/main/Data.ipynb

Exploratory and Descriptive Analysis

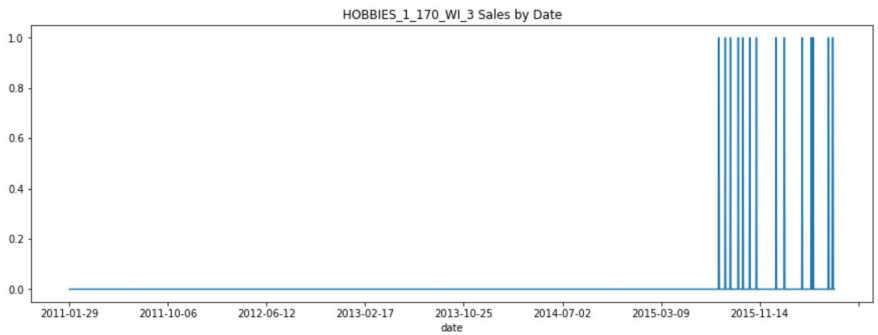
Top 3 units by Sales volume

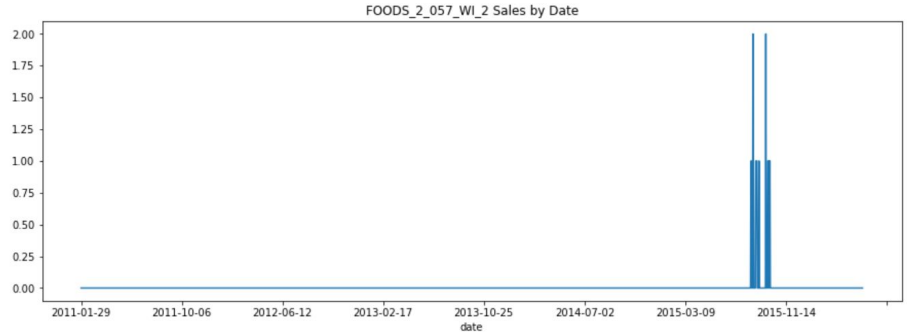


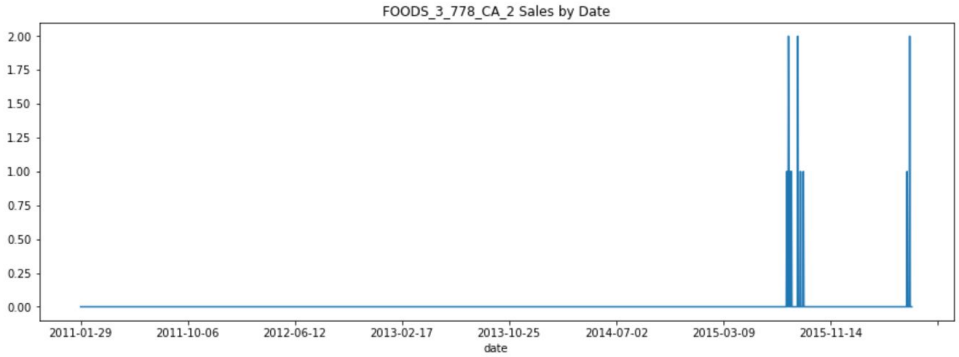




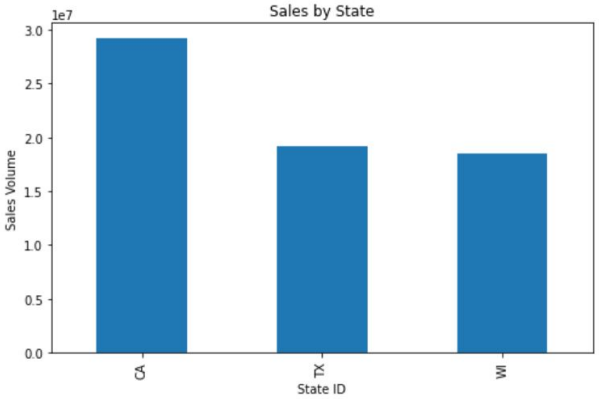
Last 3 Units by Sales Volume



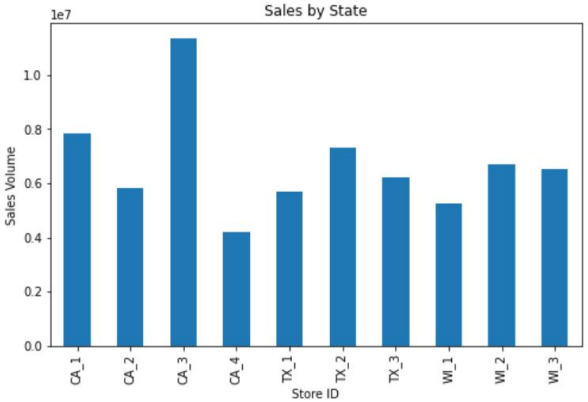


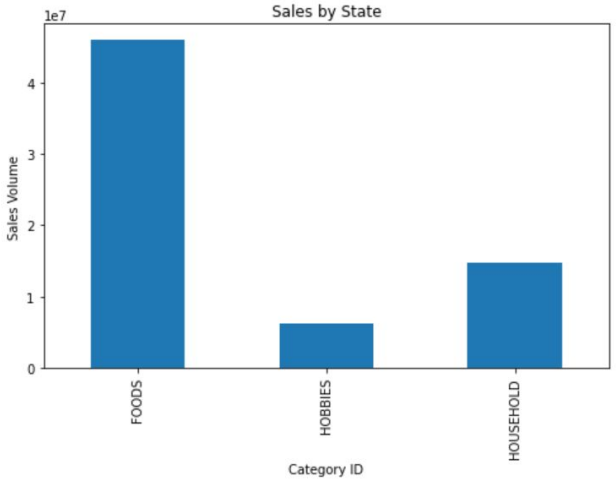


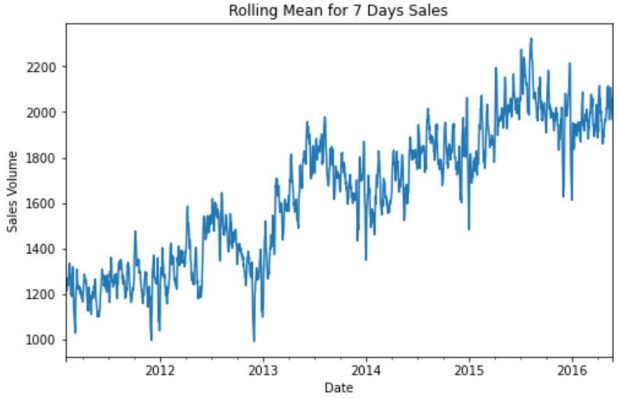
Sales Volume by State, Stores and Categories

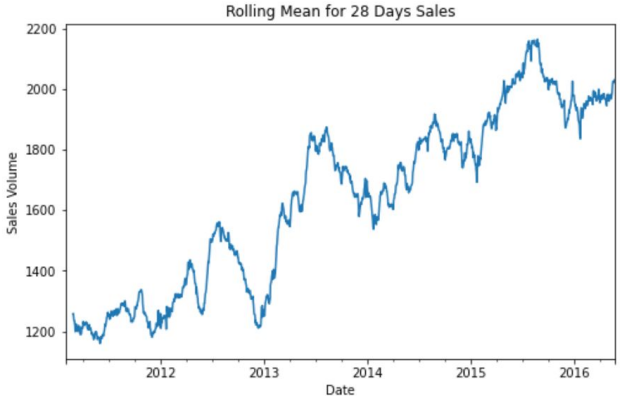


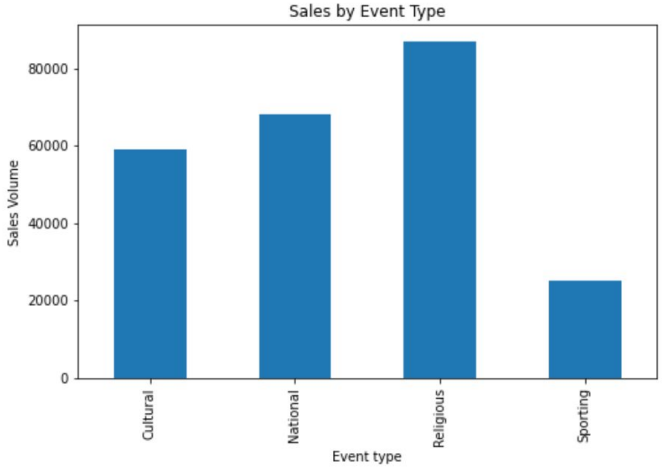
Total amount of sales account for 66 927 173 units



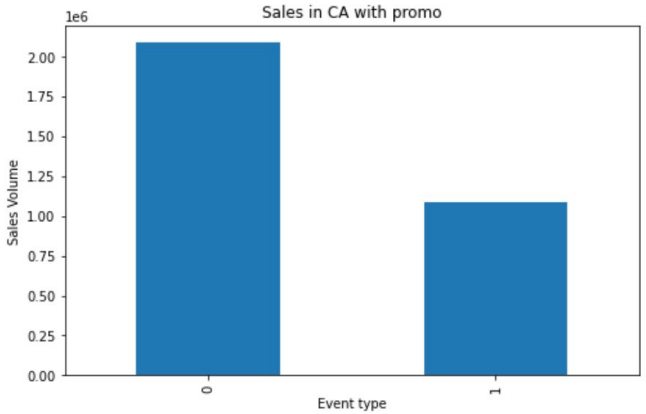


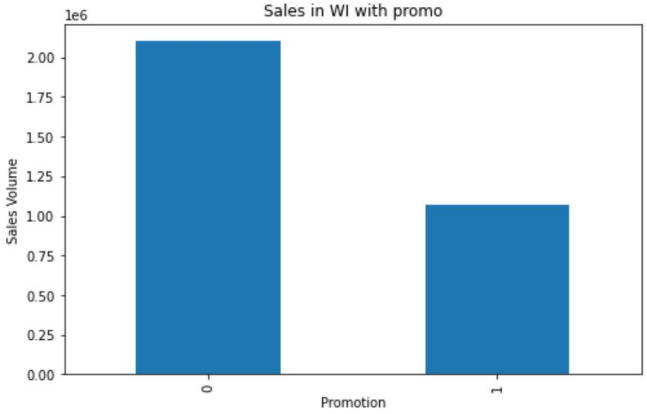

Rolling Mean for 7 Days Sales
Sales Volume
2200
2000
1800
1600
1400
1200
1000
2012
2013
2014
2015
2016
Date

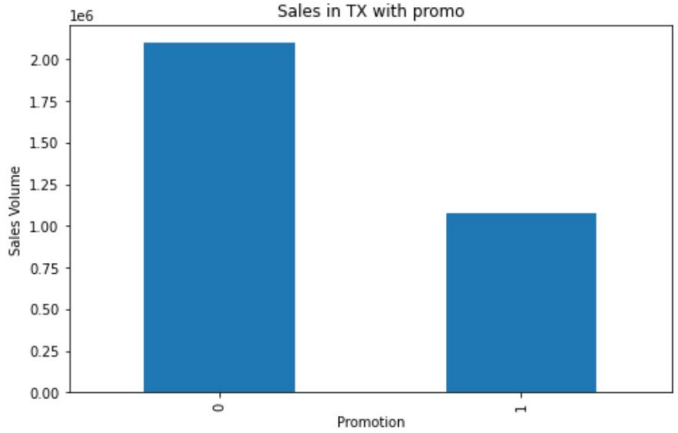

Rolling Mean for 28 Days Sales
Sales Volume
2200
2000
1800
1600
1400
1200
2012
2013
2014
2015
2016
Date


Sales by Event Type
Sales Volume
80000
60000
40000
20000
0
Cultural
National
Religious
Sporting
Event type

Promo Sales


Sales in CA with promo
1e6
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Sales Volume
0
1
Event type


Sales in WI with promo
1e6
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Sales Volume
0
1
Promotion


Sales in TX with promo
1e6
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Sales Volume
0
1
Promotion

Main insights about the data

- Series values substantially differ across units of stock range
- Food category is the most popular across all states and stores
- Rolling mean both for 7 and 28 days contains upward trend
- Events and holidays may affect sales, type of event matters
- Promo activities affect sales

XGBoost model

XGBoost Regressor Model

Motivation for implementation:

The method was proposed by

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. Paper presented at the *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, , 13-17-August-2016* 785-794.

Is actively used to forecast time-series:

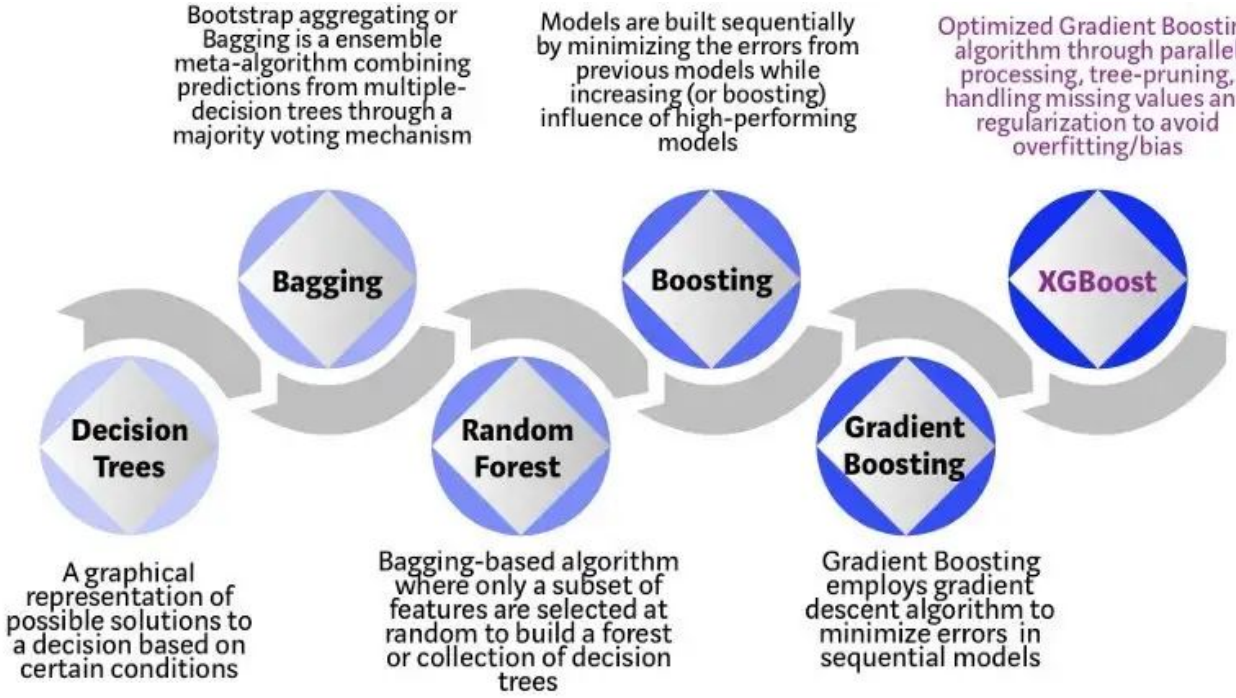
Wang, Y., & Guo, Y. (2020). Forecasting method of stock market volatility in time series data based on mixed model of ARIMA and XGBoost. *China Communications*, *17*(3), 205-221.

Alim, M., Ye, G. H., Guan, P., Huang, D. S., Zhou, B. S., & Wu, W. (2020). Comparison of ARIMA model and XGBoost model for prediction of human brucellosis in mainland China: a time-series study. *BMJ open*, *10*(12)

Fang, Z. G., Yang, S. Q., Lv, C. X., An, S. Y., & Wu, W. (2022). Application of a data-driven XGBoost model for the prediction of COVID-19 in the USA: a time-series study. *BMJ open*, *12*(7)

Lv, C. X., An, S. Y., Qiao, B. J., & Wu, W. (2021). Time series analysis of hemorrhagic fever with renal syndrome in mainland China by using an XGBoost forecasting model. *BMC infectious diseases*, *21*(1), 1-13

XGBoost Approach visualization



Feature Engineering

3 types of features:

- based on dates and calendar days, weeks and months
- dummy-variables to catch event- and promotion- specific information
- lag features to account for past consumption patterns

Feature Engineering

```python
# Processing of events calendar to get dummies on each type
events = cal[["date","event_type_1", "snap_CA", "snap_TX", "snap_WI"]]
events.set_index("date", inplace = True)
events.index = pd.to_datetime(events.index)
events = pd.get_dummies(events, prefix="event", dtype = int)
```

```python
def create_features(dataframe):

    """
    Creating time series features based on date to shift the problem to ML plane
    """
    dataframe = dataframe.copy()
    dataframe['dayofweek'] = dataframe.index.dayofweek
    dataframe['quarter'] = dataframe.index.quarter
    dataframe['month'] = dataframe.index.month
    dataframe['year'] = dataframe.index.year
    dataframe['dayofyear'] = dataframe.index.dayofyear
    dataframe['dayofmonth'] = dataframe.index.day
    dataframe['weekofyear'] = dataframe.index.isocalendar().week
    dataframe['weekofyear'] = dataframe['weekofyear'].astype(int)

    """
    Creating event-specific features
    """
    dataframe = dataframe.join(events)

    """
    Introducing lag features
    """
    dataframe["lag1"] = dataframe.iloc[:,0].shift(1).astype(float)
    dataframe["lag7"] = dataframe.iloc[:,0].shift(7).astype(float)
    dataframe["lag28"] = dataframe.iloc[:,0].shift(28).astype(float)
    dataframe["lag_1y"] = dataframe.iloc[:,0].shift(365).astype(float)
    dataframe["lag_2y"] = dataframe.iloc[:,0].shift(730).astype(float)
    dataframe["lag_3y"] = dataframe.iloc[:,0].shift(730).astype(float)


    return dataframe
```

Model evaluation

Number of estimators = 500

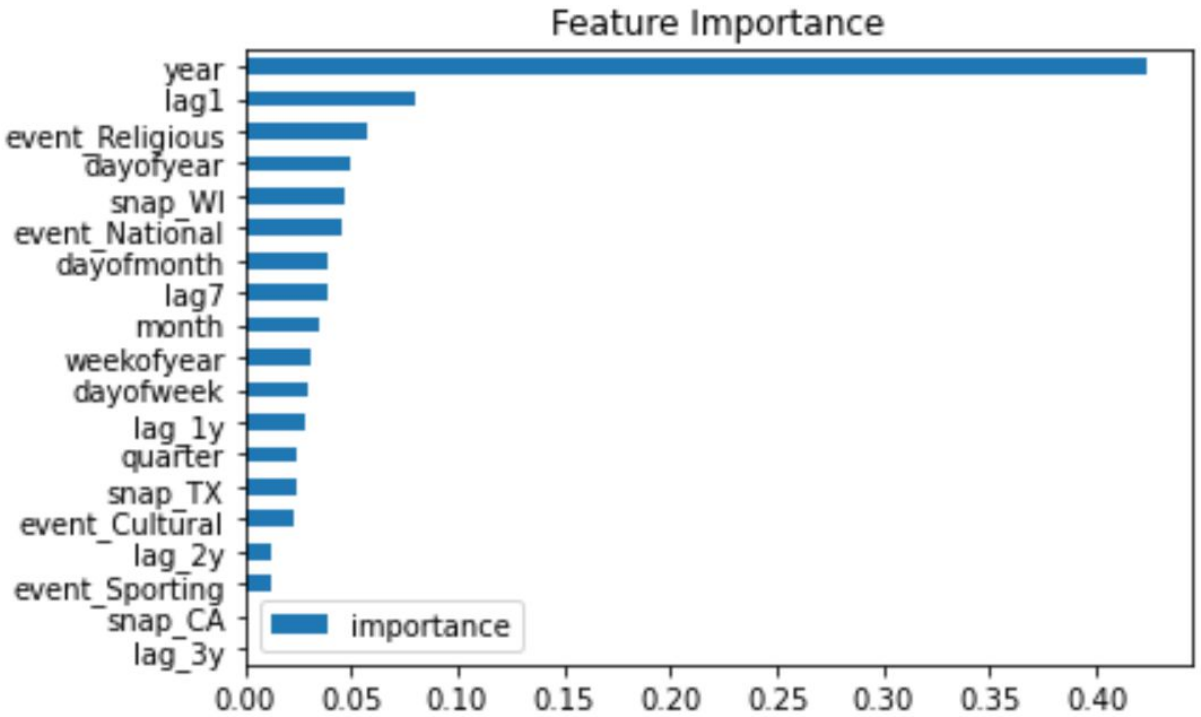
Learning rate = 0.01

```
X_train = unit_sales[:"2016-04-24"][FEATURES]
y_train = unit_sales[:"2016-04-24"][TARGET]

X_valid = unit_sales["2016-04-25":"2016-05-22"][FEATURES]
y_valid = unit_sales["2016-04-25":"2016-05-22"][TARGET]

X_test = unit_sales["2016-05-23":][FEATURES]
```

```
100%|██████████| 30490/30490 [5:09:35<00:00,  1.64it/s]
```

Prediction vs. Actual data for FOODS_3_090_CA_3_evaluation is 25.17

Feature Importance



ARIMA model

Autoregressive integrated moving average model (ARIMA)

Developed to model time series processes

AR = $$Y_t = \delta + \theta Y_{t-1} + \varepsilon_t,$$

Yt depends linearly upon its previous value Yt−1, where εt denotes a serially uncorrelated innovation with a mean of zero and a constant variance.

MA = $$Y_t = \mu + \varepsilon_t + \alpha \varepsilon_{t-1}.$$

Y1 is a weighted average of ε1 and ε0. where μ is the mean.

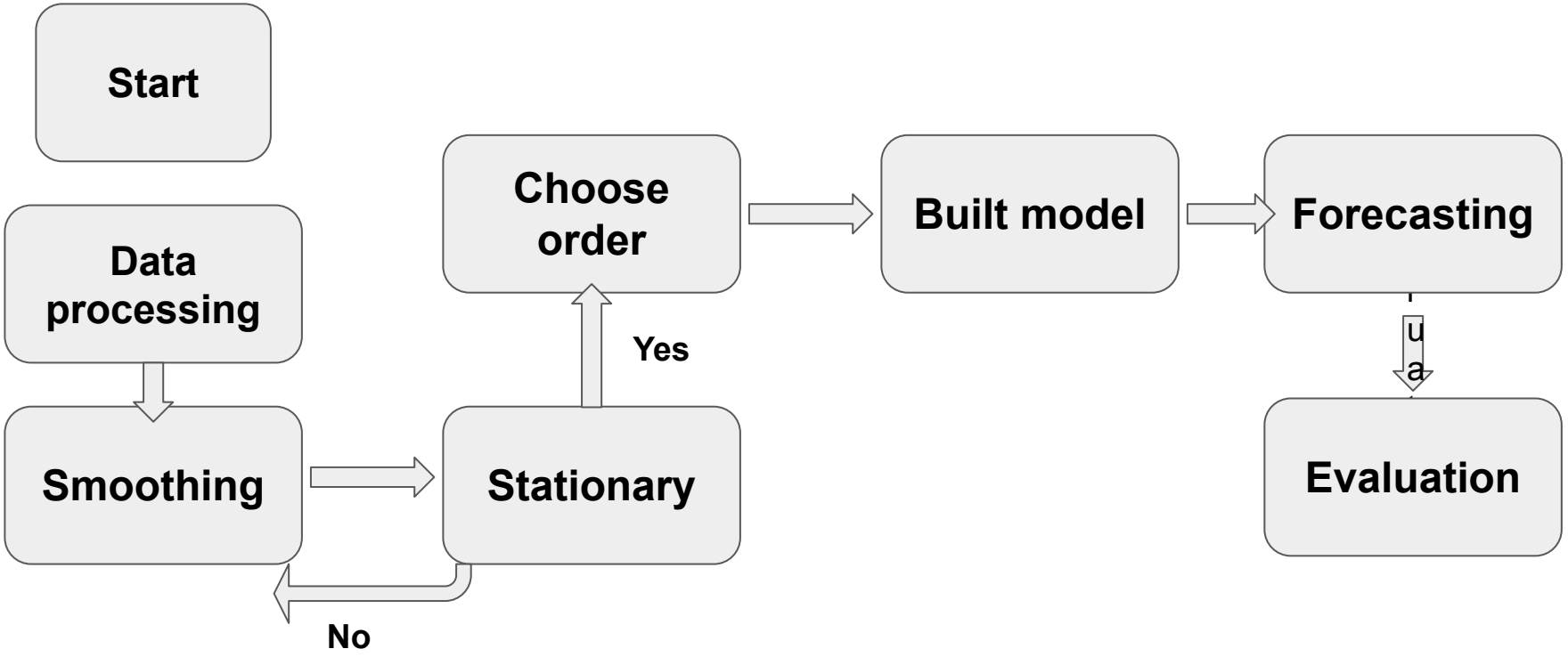
Autoregressive integrated moving average model (ARIMA)

Strictly stationary: unaffected by a change of time origin

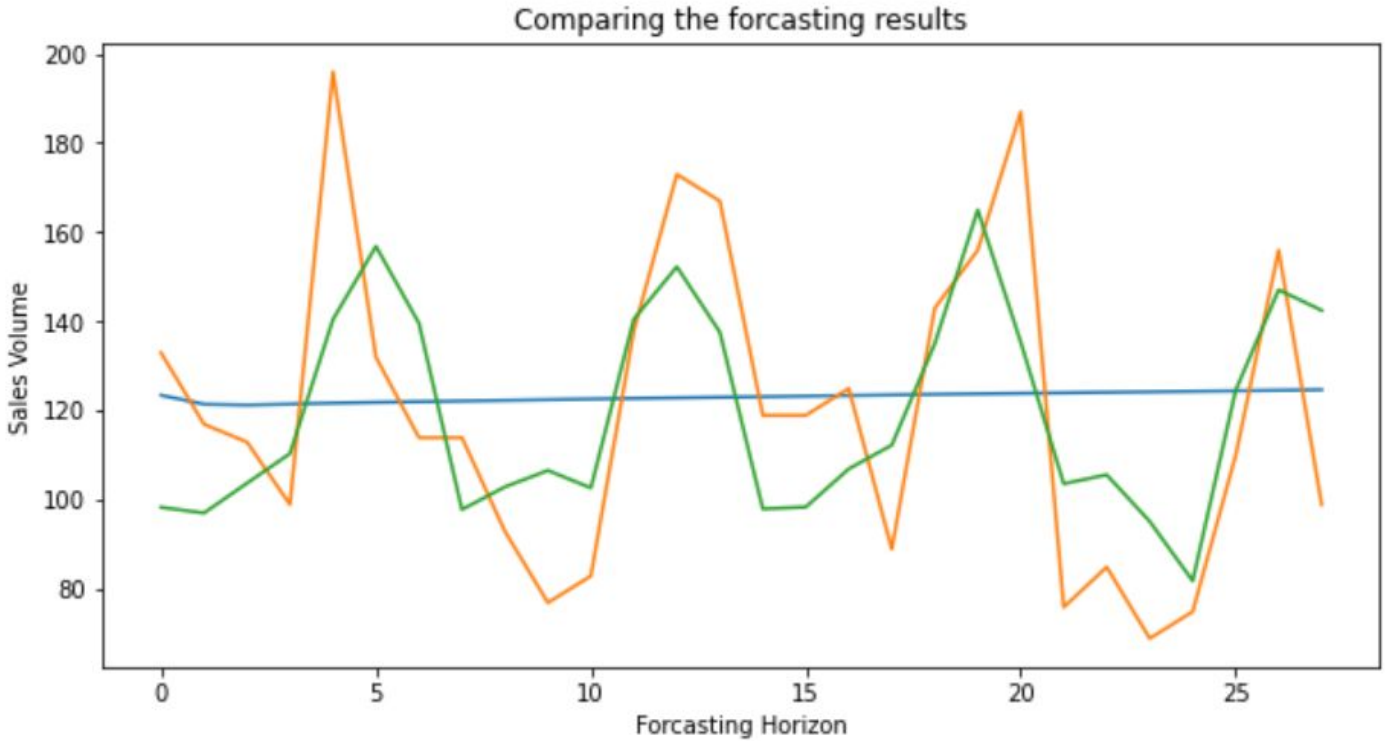
Weakly stationary: if for all t it holds that:

$$E\{Y_t\} = \mu < \infty$$
$$V\{Y_t\} = E\{(Y_t - \mu)^2\} = \gamma_0 < \infty$$
$$\text{cov}\{Y_t, Y_{t-k}\} = E\{(Y_t - \mu)(Y_{t-k} - \mu)\} = \gamma_k, \quad k = 1, 2, 3, \ldots$$

Autoregressive integrated moving average model (ARIMA)



Comparing the results for FOODS_3_090_CA_3_evaluation



Orange - actual data, Blue - ARIMA, Green - XGBRegressor