DIGITAL
TRANSGORMATION
MANAGEMENT

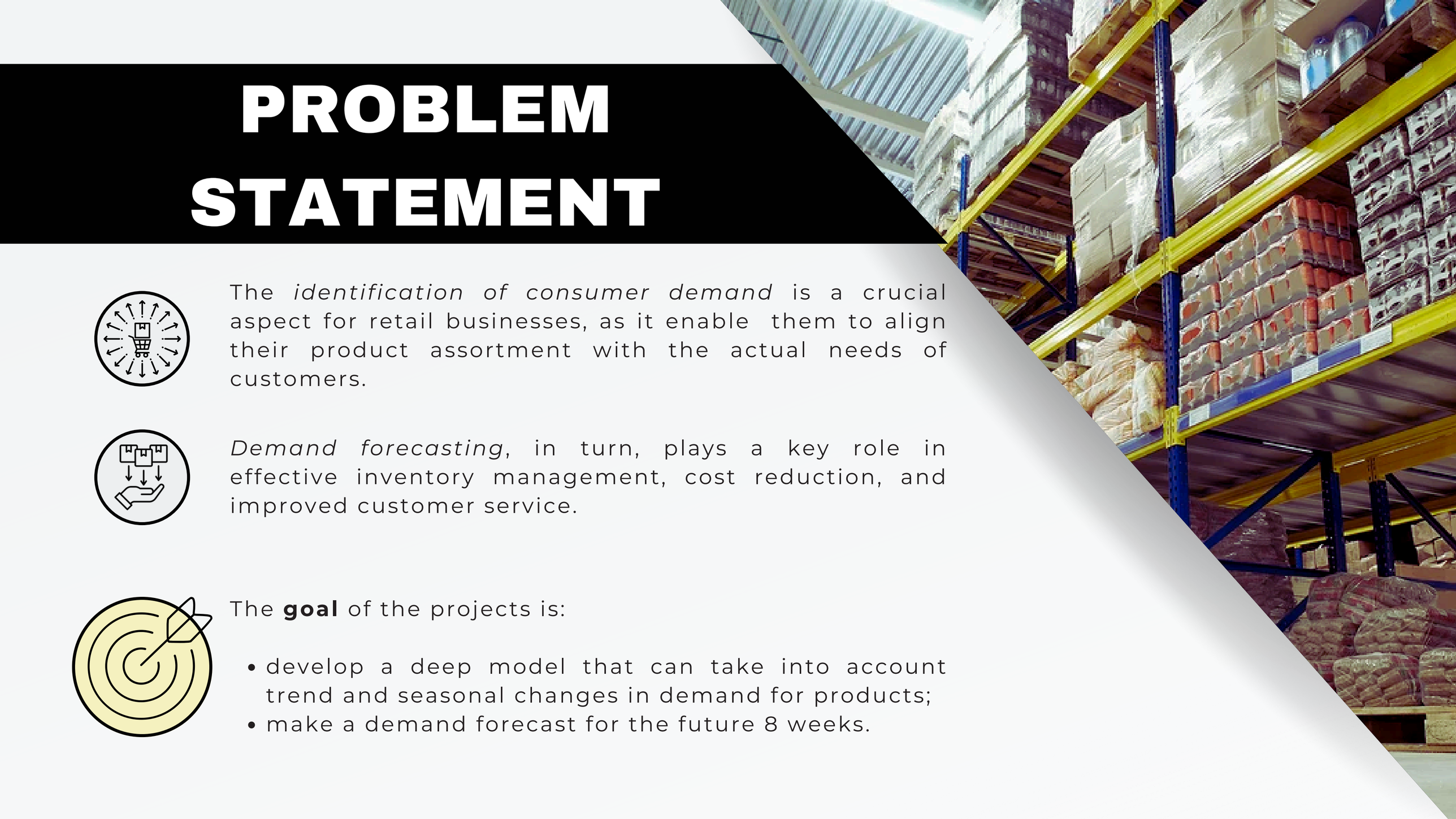
FOOD DEMAND
FORECAST

MACHINE LEARNING PROJECT

2025

CONTENT

PROBLEM STATEMENT



The *identification of consumer demand* is a crucial aspect for retail businesses, as it enable them to align their product assortment with the actual needs of customers.

Demand forecasting, in turn, plays a key role in effective inventory management, cost reduction, and improved customer service.

The **goal** of the projects is:

- develop a deep model that can take into account trend and seasonal changes in demand for products;
- make a demand forecast for the future 8 weeks.

DATA IMPORT

The initial stage involves importing an archive called **Food Demand Dataset** from Kaggle

The archive includes five files, three of which will be used:

ORDERS

- Information about all orders made over the past 3 years
- 456 548 rows × 9 columns
- Columns: id, week, center id, meal id, checkout price, base price, emailer for promotion, homepage featured, num orders

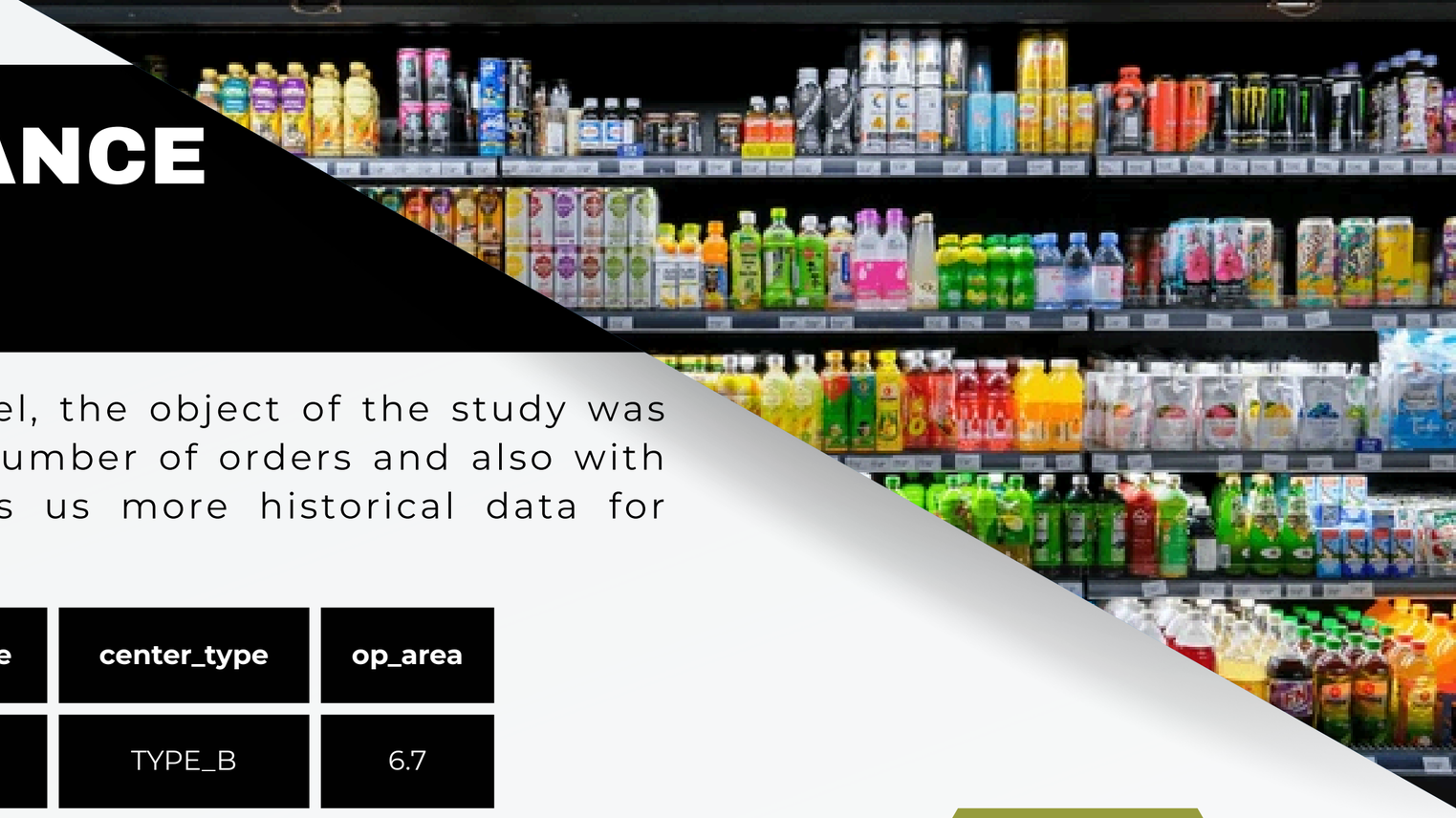
CENTERS

- Information about retail centers and their location
- 77 rows × 5 columns
- Columns: center id, city code, region code, center type, op area

MEALS

- Information about products sold
- 51 rows × 3 columns
- Columns: meal id, category, cuisine

FIRST ACQUAINTANCE WITH DATA



Since our goal is to predict the order level, the object of the study was chosen to be the store with the largest number of orders and also with the highest costs for them, which gives us more historical data for training models - **Center №13**

center_id	city_code	region_code	center_type	op_area
13	590	56	TYPE_B	6.7

To perform demand forecasting, it was first necessary to select a specific product as the subject of analysis. The selection was based on the total number of units ordered, leading us to choose product with **ID 1885**, which had the highest overall volume.

meal_id	category	cuisine
1885	Beverages	Thai

Total:
145 orders
334 334 units
145 weeks

DATA PREPROCESSING

Due to the retail-demand-analysis project we can note that:

01 There is no one column in the table that has NaN value;

02 No duplicate column / row values in the table;

03 Column 'id' has only unique values.

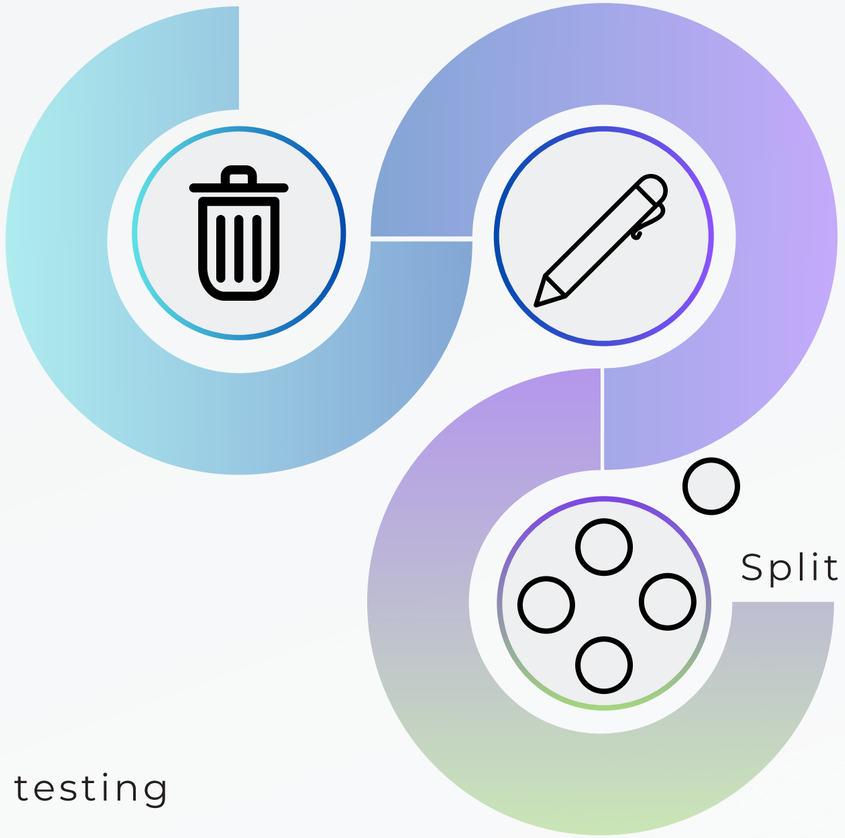
We need:

01 Delete unnecessary columns and rows;

02 Rename columns so they express clear meaning;

03 Outliers testing

04 Train-Test Splitting: splitting the dataset into training and testing in a ratio of 80 to 20



DATA PREPROCESSING: OUTLIERS DETECTION

Due to the Outliers Detection analysis we determined next 7 outliers:

Index 49 (week № 50): 3 week of December in the 1 year
Index 52 (week № 53): 1 week of January in the 2 year
Index 111 (week № 112): 4 week of February in the 3 year
Index 124 (week № 125): 4 week of May in the 3 year
Index 125 (week № 126): 1 week of June in the 3 year
Index 131 (week № 132): 2 week of July in the 3 year
Index 139 (week № 140): 2 week of September in the 3 year

To understand whether it is necessary to correct these emissions or to save them, we conduct an *analysis of the dependence* of demand on another variable - price.

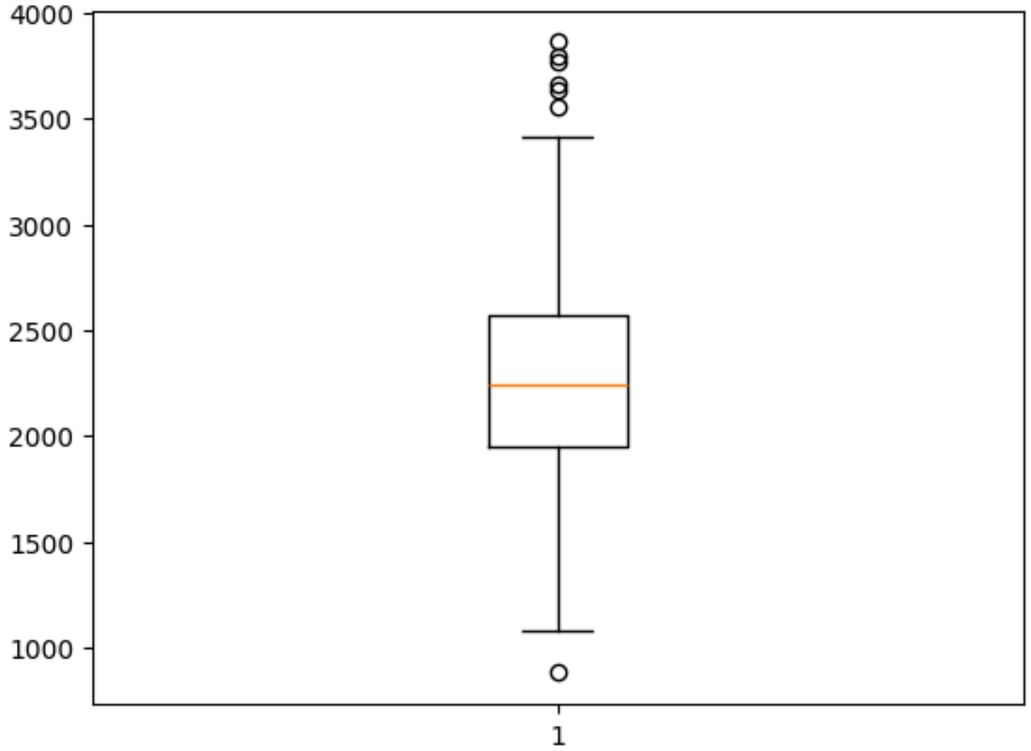


Figure 1. Boxplot of Outliers analysis

DATA ANALYSIS: CORRELATION

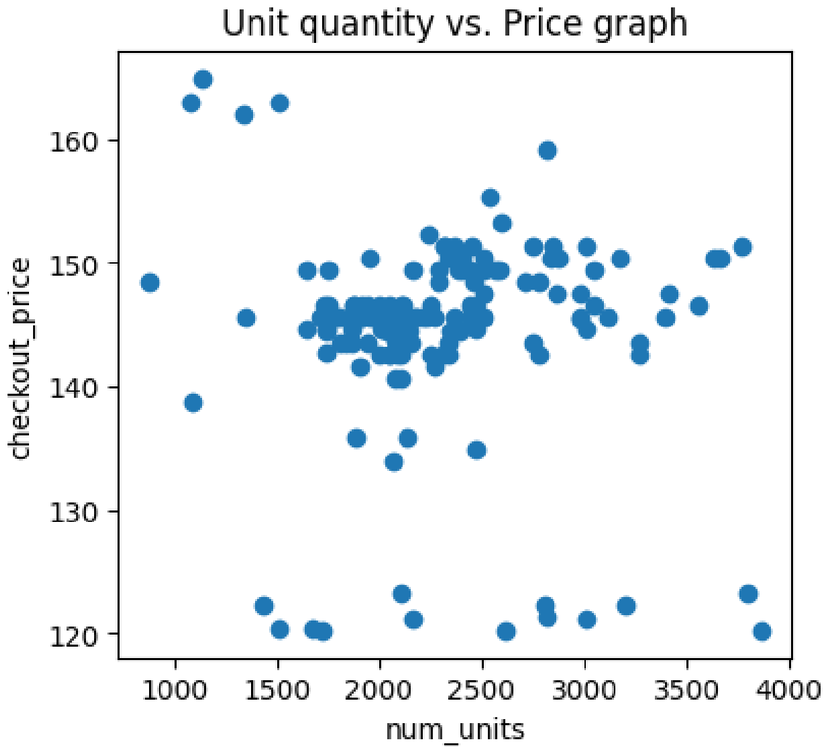


Figure 2. Relationship between Order Quantity and Checkout Price

Pearson correlation coefficient

Correlation value (r) = -0.081
Significance (p_value) = 0.335

- weak inverse linear link
- not statistically insignificant

Granger Causality Test

The SSR-based F-test and the other tests all yield p > 0.05

- the absence of any statistically significant predictive link

In summary, the variables, product price and order volume, appear to evolve ***independently***. As a result, price was excluded from the feature set used in subsequent demand forecasting models.

DATA PREPROCESSING: HANDLING OUTLIERS

Total after modifications: **145 rows**

We returned to the handling outliers without keeping in their original form:

- Index 124 (week № 125): 4 week of May in the 3 year
- Index 125 (week № 126): 1 week of June in the 3 year

For these two consecutive outliers, the values were replaced with the average of their immediate neighboring points.

- Index 49 (week № 50): 3 week of December in the 1 year
- Index 52 (week № 53): 1 week of January in the 2 year
- Index 111 (week № 112): 4 week of February in the 3 year
- Index 131 (week № 132): 2 week of July in the 3 year
- Index 139 (week № 140): 2 week of September in the 3 year

These five anomalies were adjusted using a threepoint median filter, computed over the previous, current, and subsequent observations.

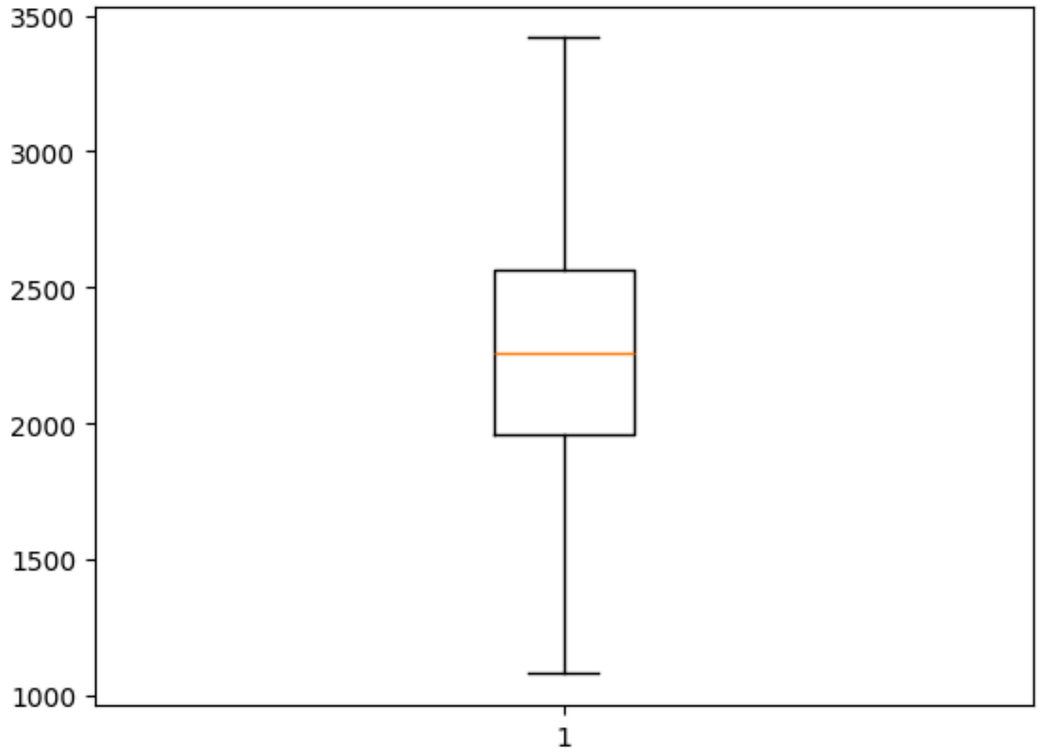


Figure 3. Updated boxplot of Outliers analysis

SEASONAL – TREND DECOMPOSITION

TREND

Hodrick-Prescott Filter

- The smoothing parameter λ was set to 129.600, following standard practice in time series research;
- The resulting trend captures the overall direction of demand change and appears relatively smooth.

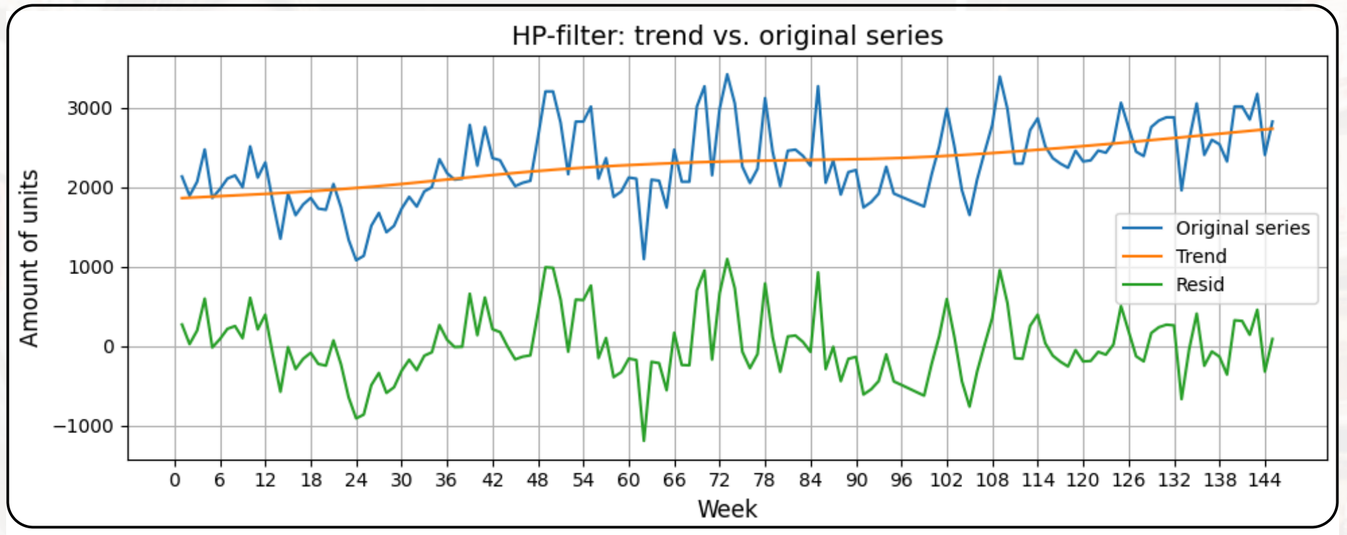


Figure 4. Hodrick-Prescott Filter

LOESS smoothing

- Trend is non-linear;
- The rate of growth and decline changes over time

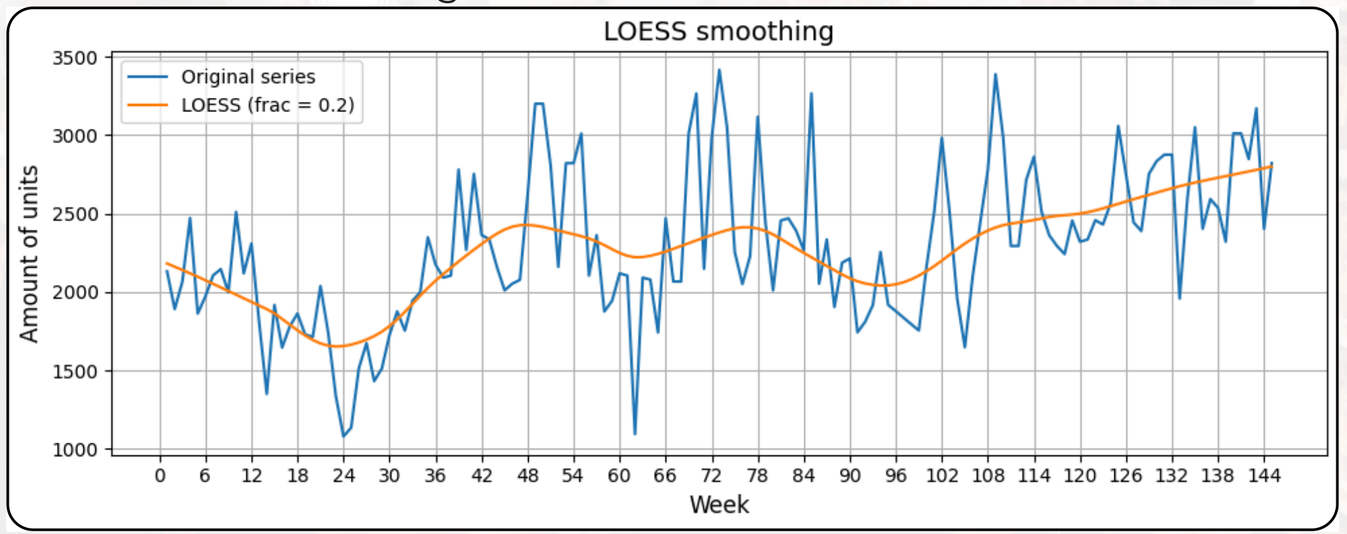


Figure 5. LOESS smoothing trend

SEASONAL – TREND DECOMPOSITION

SEASONALITY

Autocorrelation

- Rapid decay after lag 1–6: the series is highly correlated with itself at very short horizons, then correlations taper off toward **zero**;
- No prominent spikes at 'seasonal' lags (52 weeks for annual seasonality, or 12 weeks for quarterly).

Fast Fourier Transform

- Tool of Spectral analysis;
- The dominant spectral peak was found at a frequency corresponding to a period of **72 weeks** (the strongest repeating cycle).

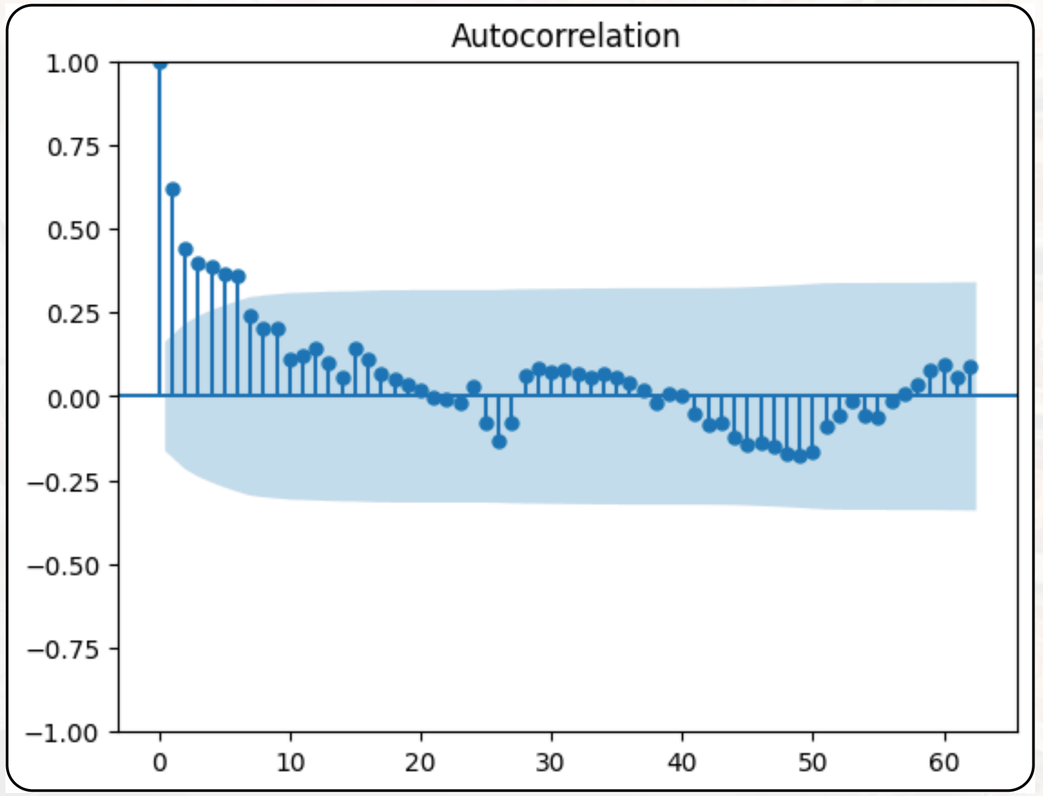


Figure 6. Autocorrelation graph

SEASONAL – TREND DECOMPOSITION

DECOMPOSITION

Zero seasonality *by HP Filter*

- **Trend**: the trend shape is almost linear, and catches long-term changes, without reacting to individual spikes;
- **Residuals**: strongly flutter around zero, without a noticeable slope or shift;
- The **decomposition** is additive: noise, rare peaks went into the residuals.

72-week seasonality *by seasonal_decompose from statsmodels*

- **Trend**: a smooth upward curve - orders really do climb over time;
- **Seasonal**: a jagged, noisy line that doesn't repeat clearly;
- **Residual**: almost identically zero everywhere—an artifact of over.

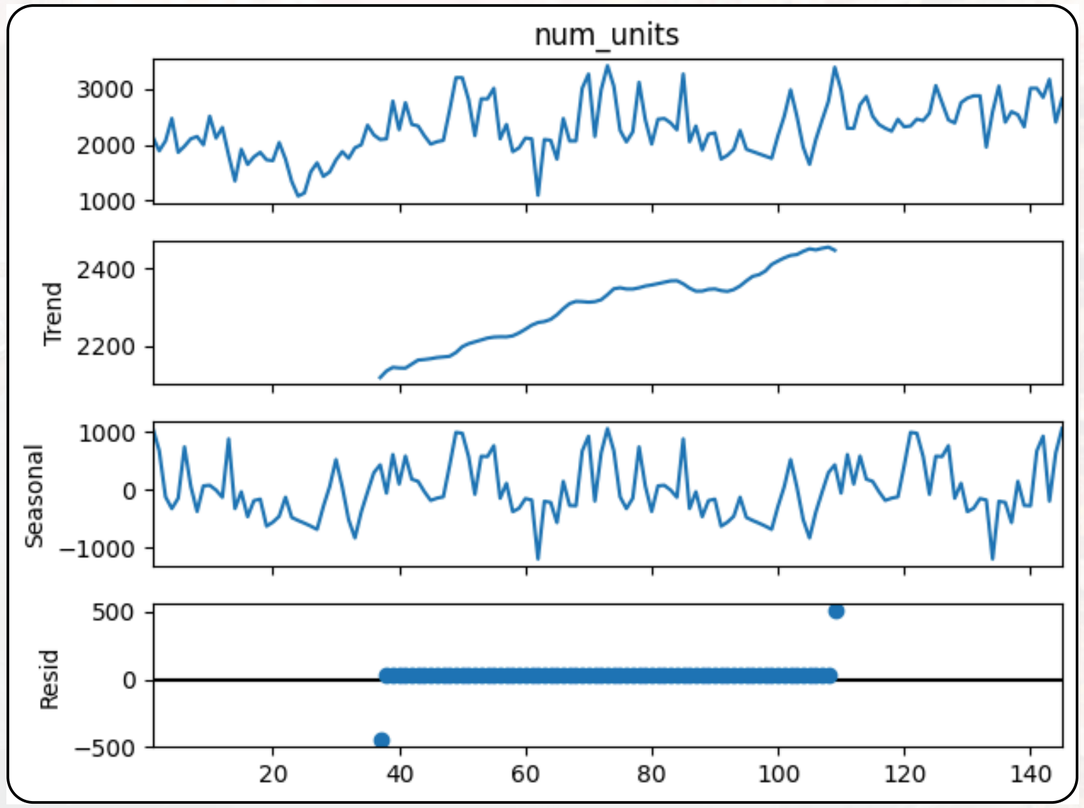


Figure 7. Decomposition for 72-week seasonality

MACHINE LEARNING MODELING

ARIMA

for the zero seasonality hypothesis

01 Searching for optimal coefficients for the model:

- For **manual selection,** we offer several options for coefficients based on ACF & PACF graphs
- For **automatic selection,** we use *AutoARIMA from StatsForecast* function, that automatically discovers the optimal order for the class of ARIMA models

02 *Comparison of models to identify the best one*

Model	AIC	BIC
ARIMA (0, 1, 5)	1708.67	1725.14
ARIMA (3, 1, 1)	1709.11	1722.83
ARIMA (1, 1, 1)	1705.4	1713.63

Table 1. Comparison of ARIMA models by AIC and BIC

03 *Prediction on test set*

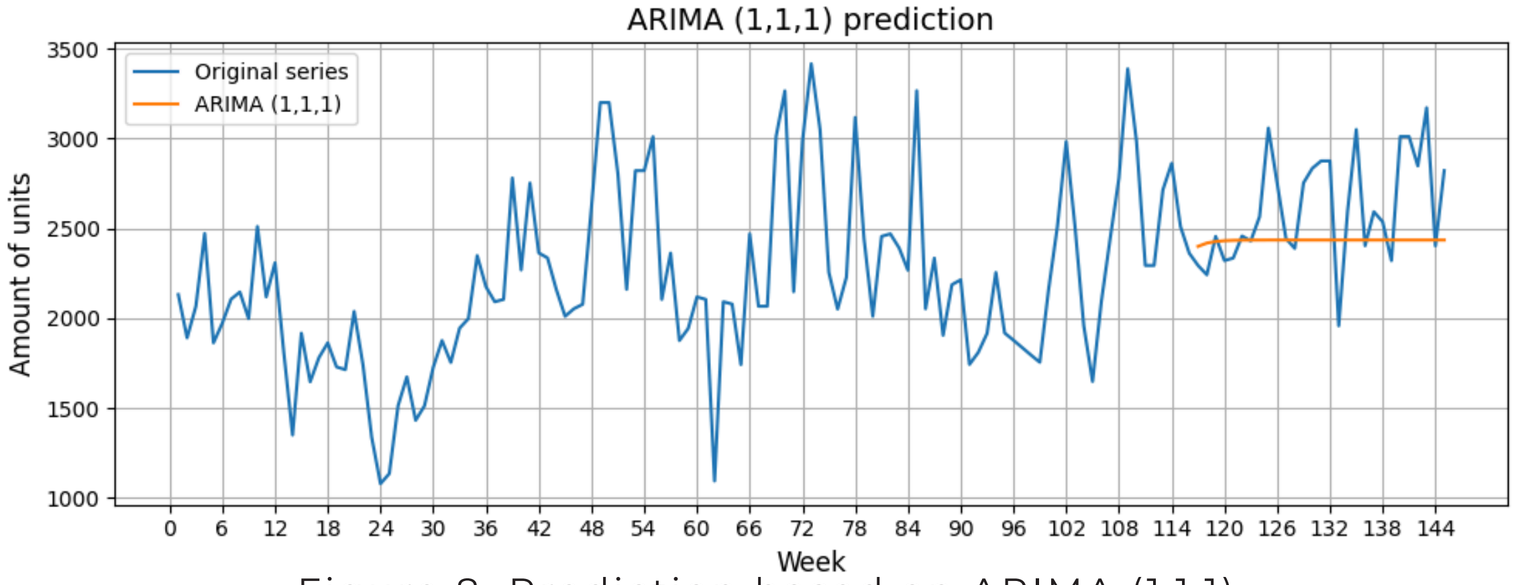


Figure 8. Prediction based on ARIMA (1,1,1)

MACHINE LEARNING MODELING

SARIMA

for the 72-week seasonality hypothesis

01 Searching for optimal coefficients for the model:

- For **automatic selection,** we use *AutoARIMA from StatsForecast* function, that automatically discovers the optimal order for the class of ARIMA models

02 *Comparison of models to identify the best one*

Model	AIC	BIC
ARIMA (3, 1, 1)	1709.11	1722.83
ARIMA (1, 1, 1)	1705.4	1713.63
SARIMA (1, 1, 1)	1705.4	1713.63

Table 2. Comparison of SARIMA with other models by AIC and BIC

03 *Prediction on test set*

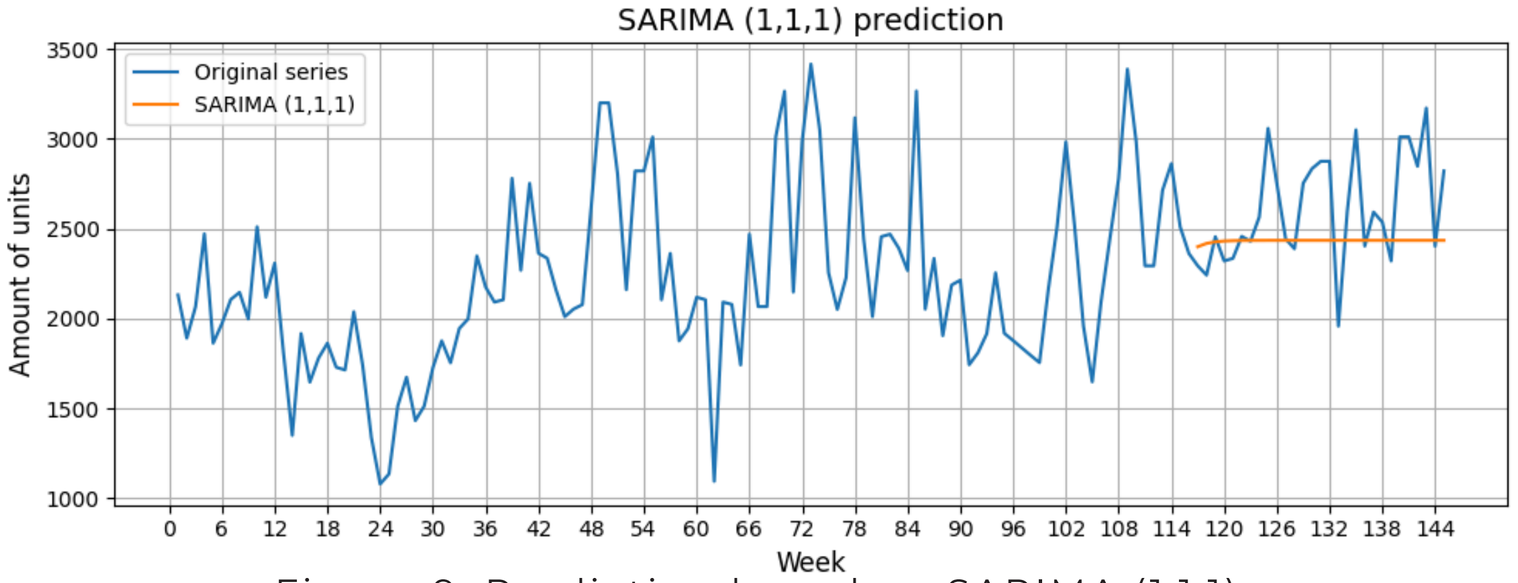


Figure 9. Prediction based on SARIMA (1,1,1)

MACHINE LEARNING MODELING

XGBOOST

for the zero seasonality hypothesis

Seasonal length equals **1**

01 Preparing the dataset:

- Trend - feature
- Normal lags 1–10
- Exponential smoothing
- Moving average
- Remove gaps

02 Hyperparameter tuning was performed using *Randomized SearchCV* with XGBRegressor.

Optimal configuration was determined:

- n_estimators - Set 500;
- learning_rate - Set 0.1;
- max_depth - Set 3;
- subsample - Set 0.6;
- colsample_bytree - Set 1.0

03 *Prediction on test set*

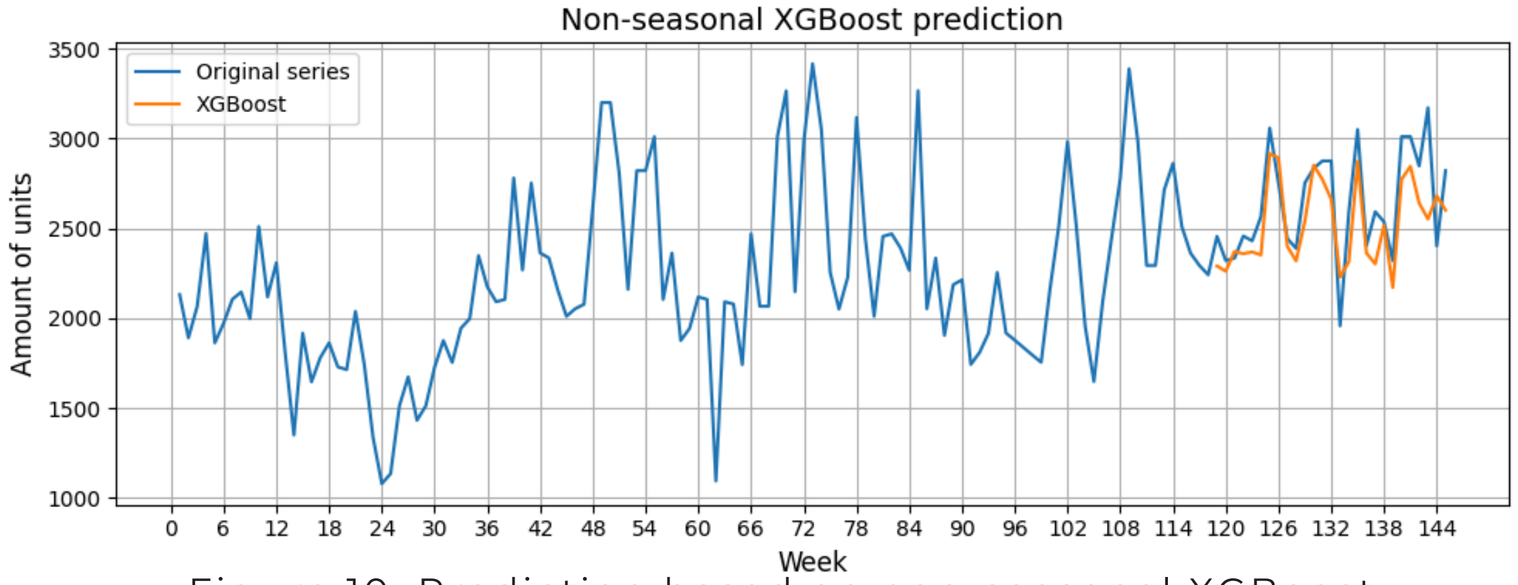
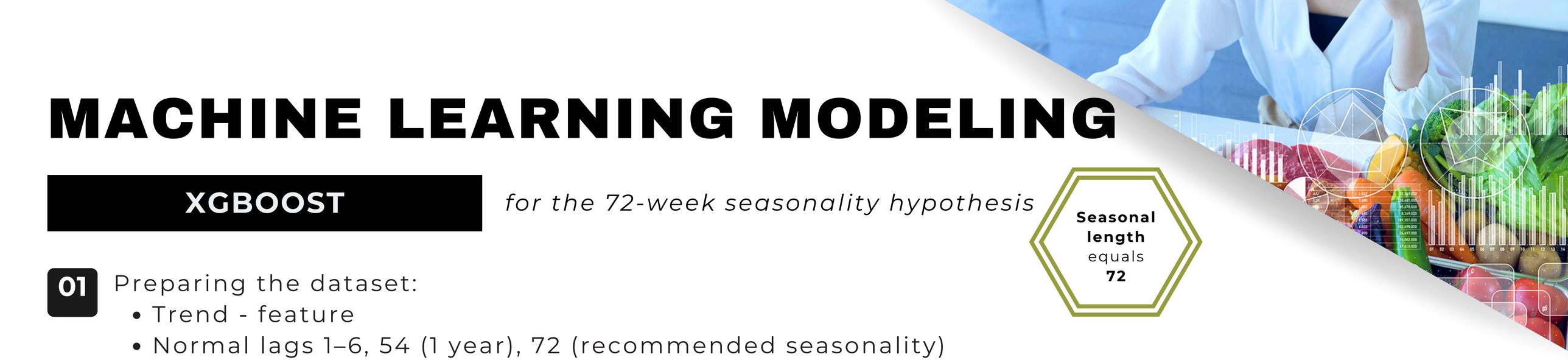


Figure 10. Prediction based on non-seasonal XGBoost

MACHINE LEARNING MODELING



XGBOOST

for the 72-week seasonality hypothesis

Seasonal length equals **72**

01 Preparing the dataset:

- Trend - feature
- Normal lags 1–6, 54 (1 year), 72 (recommended seasonality)
- Moving average for 72 points
- Remove gaps
- Sinusoidal and cosinusoidal terms for 72 point cycle

02 Hyperparameter tuning was performed using *Randomized SearchCV* with XGBRegressor.

Optimal configuration was determined:

- n_estimators - Set 300;
- learning_rate - Set 0.01;
- subsample - Set 0.8;
- colsample_bytree - Set 1.0
- reg_alpha - Set 0.5;
- reg_lambda - Set 0.5

03 *Prediction on test set*

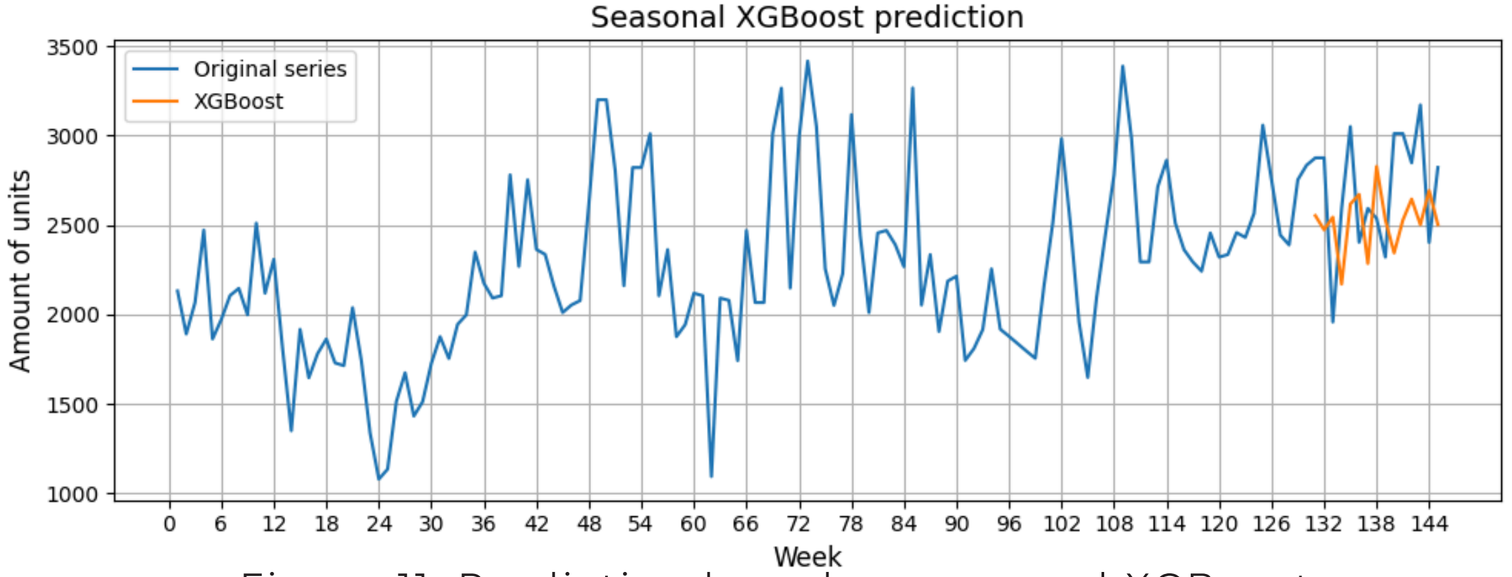


Figure 11. Prediction based on seasonal XGBoost

MACHINE LEARNING MODELING

RANDOM FOREST

for the zero seasonality hypothesis

Seasonal length equals **1**

01 Preparing the dataset:

- Normal lags 1–6, 10, 24;
- Rolling mean (window = 4);
- Rolling standard deviation (window = 8);
- Remove gaps

02 Hyperparameter tuning was performed using *Randomized SearchCV* with RandomForestRegressor.

Optimal configuration was determined:

- n_estimators - Set 200;
- max_features - Set 'log2';
- min_samples_split - Set 4;
- random_state - Set 42.

03 *Prediction on test set*

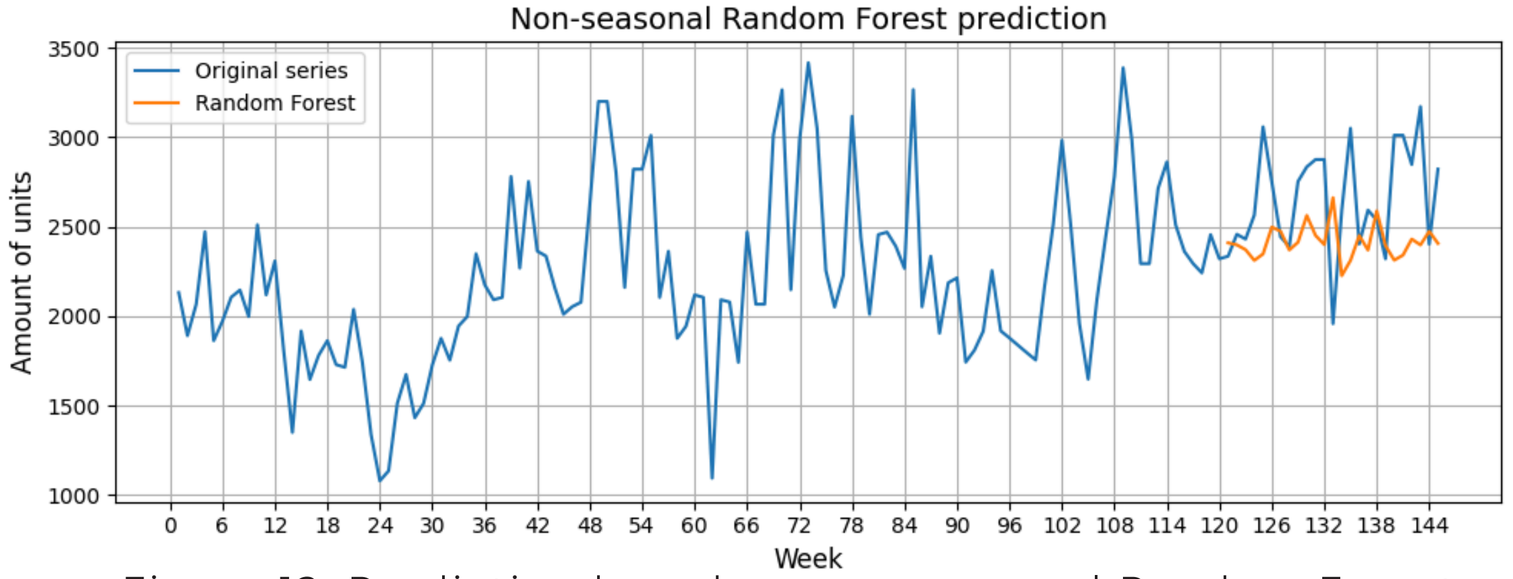


Figure 12. Prediction based on non-seasonal Random Forest

MACHINE LEARNING MODELING

RANDOM FOREST

for the 72-week seasonality hypothesis

Seasonal length equals **72**

01 Preparing the dataset:

- Normal lags 1–15
- Shift for 72 points
- Rolling mean (window = 3);
- Rolling standard deviation (window = 7);
- Remove gaps
- Sinusoidal and cosinusoidal terms for 72 point cycle

02 Hyperparameter tuning was performed using *Randomized SearchCV* with RandomForestRegressor.

Optimal configuration was determined:

- n_estimators - Set 200;
- max_features - Set 'log2';
- min_samples_split - Set 4;
- random_state - Set 42.

03 *Prediction on test set*

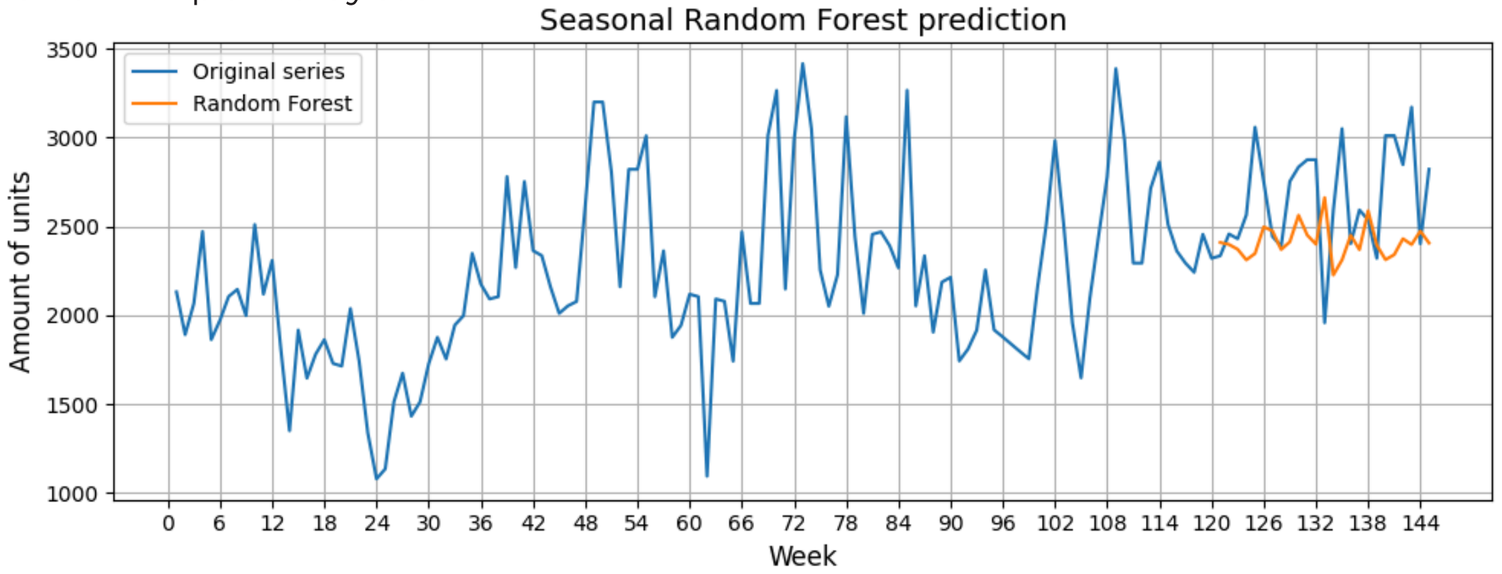


Figure 13. Prediction based on non-seasonal Random Forest

EVALUATION OF MODELS

The metrics used to evaluate the accurancy of models are:

- **MSE** - Mean Squared Error is computed as the average of the squared errors;
- **RMSE** - Root Mean Squared Error is the square root of the Mean Squared Error;
- **MAPE** - Mean Absolute Percent Error is computed by taking the error for each prediction, divided by the actual value

Model	MSE	RMSE	MAPE
Arima (1,1,1)	118 223.96	343.84	0.1
Sarima (1,1,1)	118 223.96	343.84	0.1
XGBoost non-seasonal	43 286.78	208.05	0.06
XGBoost seasonal	175 164.23	418.53	0.15
Random Forest non-seasonal	174 710.91	417.98	0.12
Random Forest seasonal	174 710.91	417.98	0.12

Table 3. Comparison of models based on accurancy

DEMAND FORECAST

01 Preparing the features:

- Trend - feature
- Normal lags 1–10
- Exponential smoothing
- Moving average

02 Using trained model, make forecast for 8 weeks:

For week 146: 2698 beverages
For week 147: 2598 beverages
For week 148: 2621 beverages
For week 149: 2718 beverages
For week 150: 2737 beverages
For week 151: 2682 beverages
For week 152: 2698 beverages
For week 153: 2738 beverages

03 *Plot a graph with a confidence interval*

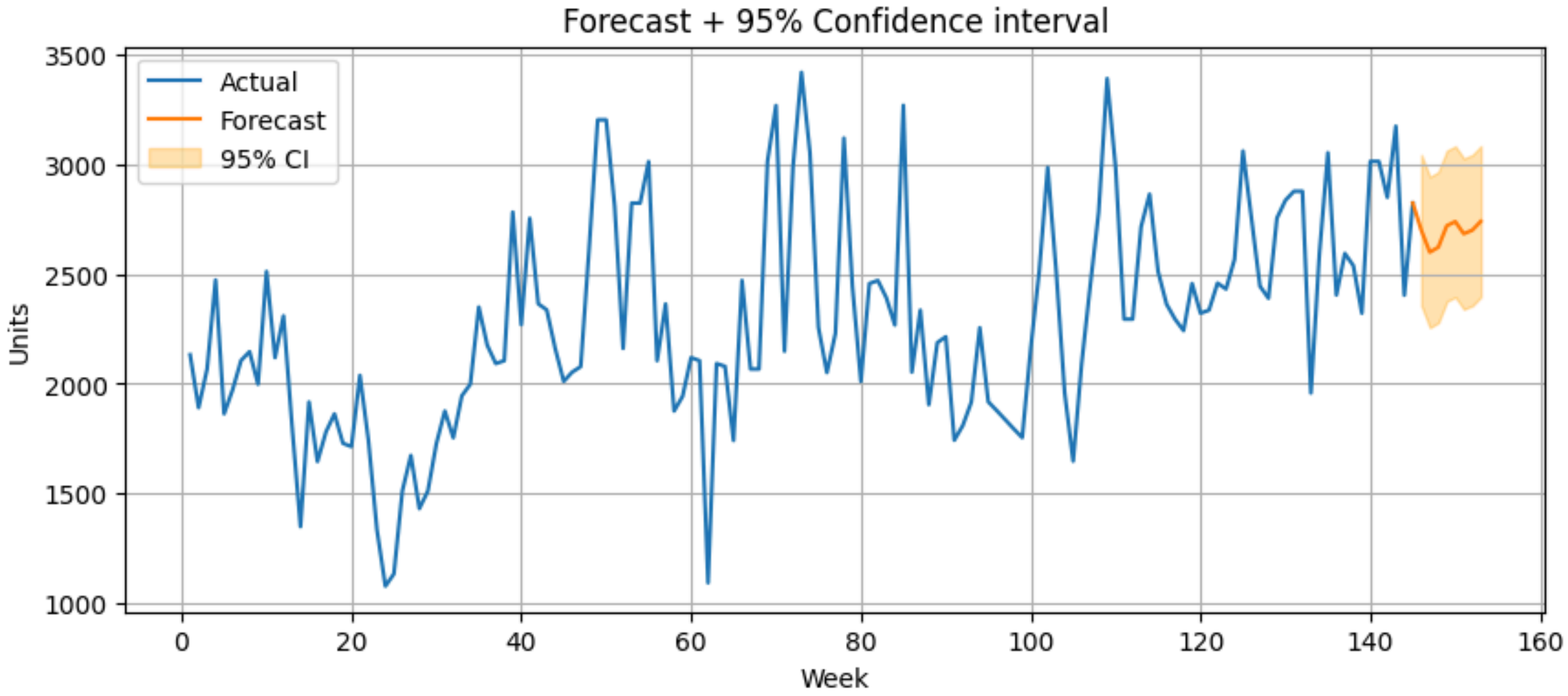


Figure 14. 8 week forecast with 95% confidence interval

CONCLUSION

XGBOOST

The resulting forecast does not appear realistic from a business perspective.

- Exponential smoothing is based on future values that were unknown;
- The underlying structure of the dataset may exhibit random walk – like behavior, which poses a significant challenge for model generalization.

ARIMA & SARIMA

They may have underperformed on the test set due to their inherent limitations - they are best suited for capturing linear trends, while the demand data likely contains more complex, nonlinear patterns

RANDOM FOREST

It failed to deliver robust predictions, possibly due to insufficient tuning or its limited capacity to handle temporal dependencies effectively.

IMPROVE FORECAST

- Applying **Grid Search** or **Bayesian Optimization** for hyperparameter tuning instead of randomized search;
- Use other time-series forecasting models such as **SVR**, **Prophet**, or deep learning architectures like **LSTM** and **Temporal Convolutional Networks** (TCNs).