

Chapter 3

RESEARCH METHODS

3.1 Framework Used Description

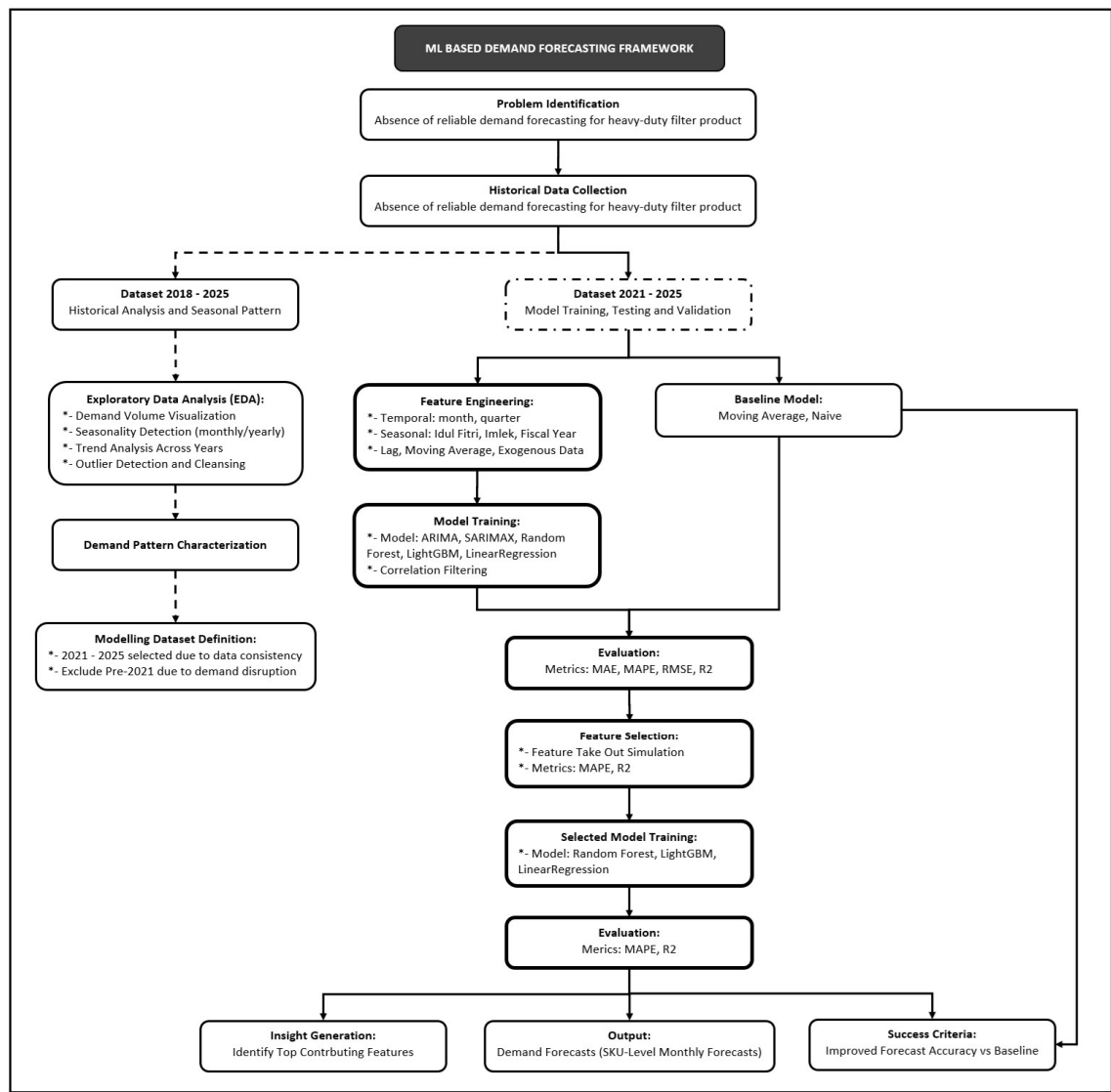


Figure 3.1: Overall forecasting framework combining exploratory analysis (2018–2025), model development (2021–2025), and evaluation stages to predict incoming customer orders at SKU level.

Figure 3.1 illustrates the end-to-end forecasting framework developed in this study to generate monthly demand forecasts at the SKU level for heavy-duty filter products. The framework comprises five main stages: (1) Problem Identification and Data Collection, (2) Exploratory Analysis and Demand Pattern Characterization, (3) Model Development, (4) Forecast Evaluation, and (5) Insight Generation.

3.1.1 Problem Identification and Data Collection

The study begins by addressing the lack of a systematic demand forecasting system for heavy-duty filter products, which has historically disrupted inventory and production planning. As a response, historical customer order data from January 2018 to December 2025 is collected. The data set contains monthly order quantities per SKU, covering various item types such as Full Flow (FF), Hydraulic Filter (HF), Oil Filter (OF), and Special Design Filter (SX).

3.1.2 Exploratory Data Analysis (EDA) and Demand Pattern Characterization

An exploratory analysis is conducted for the first time using the entire dataset (2018-2025) to identify the main patterns and demand behaviors. This includes:

- **Visualization of demand volume** in years and types of items.
- **Detection of seasonality** on both monthly and yearly scales.
- **Trend analysis** to capture stable or declining product cycles.
- **Outlier detection and cleansing**, especially around known calendar-driven demand peaks such as Idul Fitri and Chinese New Year.

This stage enables *demand pattern characterization*, where SKUs are profiled according to their regularity, seasonality, and sensitivity to external events. Based on these insights, the period 2021 to 2025 is selected as the modeling window because of its more consistent demand signals. Data prior to 2021 are retained for historical reference but excluded from model training due to pandemic-driven anomalies and irregularities.

3.1.3 Model Development Framework (2021–2025)

The modeling phase is executed in progressive stages:

3.1.3.1 Baseline Modeling

Baseline models are constructed using simple statistical methods (e.g. Naïve, Moving Average) to serve as benchmarks for assessing the added value of machine learning models. These models assume either the last observed value or the average past demand as the forecast.

3.1.3.2 Naïve Baseline Model Parameters

The Naïve baseline model serves as the simplest forecasting benchmark. It assumes that the forecast for the next period is equal to the demand observed in the most recent period. Formally:

$$\hat{y}_{t+1} = y_t$$

where $\hat{y}_{t+1}$ is the forecast for the next period and y_t is the actual demand at time t . This model does not involve any tunable parameters; its only mechanism is a one-step lag of the actual demand series. Despite its simplicity, the Naïve model provides a useful reference point: if advanced models such as Random Forest and LightGBM cannot outperform this benchmark, their added complexity would not be justified.

3.1.3.3 Moving Average (MA) Baseline Parameters

The Moving Average (MA) baseline model estimates future demand by averaging a fixed number of past observations. The only tunable parameter of this model is the *window size* k , which determines how many past periods are considered. Formally:

$$\hat{y}_{t+1} = \frac{1}{k} \sum_{i=0}^{k-1} y_{t-i}$$

where $\hat{y}_{t+1}$ is the forecast for the next period, y_{t-i} is the actual demand observed in previous periods, and k is the window size.

In this study, the MA baseline was tested using three window sizes:

- $k = 1$ (MA(1)) — equivalent to the Naïve model, using only the last observed value.
- $k = 3$ (MA(3)) — averages the last three months, balancing responsiveness and smoothing.

- $k = 6$ (MA(6)) — averages the last six months, providing greater smoothing but less sensitivity to sudden changes.

The choice of k influences the forecast characteristics: Smaller values of k make the model more reactive to demand fluctuations but may capture noise, while larger values of k smooth short-term variations at the expense of responsiveness. These parameter settings establish a simple yet interpretable benchmark against which the performance of machine learning models can be evaluated.

3.1.3.4 Feature Engineering

To improve prediction quality, additional features are generated from the raw data, including:

- **Temporal variables:** Month, quarter, and year identifiers.
- **Calendar-based seasonal indicators:** Dummy variables for Idul Fitri, Imlek (Chinese New Year), and fiscal cycles.
- **Statistical variables:** Lag features, moving averages (MA-1 to MA-6), and external data where available.

3.1.3.5 Initial Model Training (All Features)

Using the engineered features, multiple models are trained and evaluated.

- **Random Forest**, a tree-based ensemble model that combines multiple decision trees by bagging. Parameters used in this study included:

Table 3.1: Random Forest Parameters Used in This Study

Parameter	Value
n_estimators	100 (default number of trees)
criterion	squared_error (minimizes mean squared error)
max_depth	None (nodes expanded until all leaves are pure)
min_samples_split	2 (minimum samples required to split a node)
min_samples_leaf	1 (minimum samples required at a leaf node)
max_features	sqrt (square root of number of features)
bootstrap	True (samples drawn with replacement)
oob_score	False (out-of-bag evaluation not used)
n_jobs	None (single-threaded execution)
random_state	42 (fixed seed for reproducibility)

The parameters listed in Table 3.1 correspond to the default configuration of the RandomForestRegressor in scikit-learn, with the exception of the random_state, which was fixed at 42 to ensure the reproducibility of the results. At this stage of the study, no hyperparameter tuning was applied; instead, the default settings were retained to establish the baseline performance of the Random Forest model. This approach allows for a fair comparison against other models (e.g., Linear Regression and LightGBM) and provides a consistent reference point before more advanced optimization techniques such as grid search or randomized search are considered in subsequent analyses.

- **LightGBM**, a gradient boosting framework optimized for speed and efficiency. Parameters included:

Table 3.2: LightGBM Parameters Used in This Study

Parameter	Value
n_estimators	100 (default number of boosting iterations)
learning_rate	0.1 (default step size shrinkage)
max_depth	-1 (no limit on tree depth)
num_leaves	31 (maximum leaves per tree)
min_child_samples	20 (minimum data per leaf)
subsample	1.0 (no row subsampling)
colsample_bytree	1.0 (no feature subsampling)
reg_alpha	0.0 (L1 regularization term)
reg_lambda	0.0 (L2 regularization term)
n_jobs	-1 (use all available CPU cores)
random_state	42 (fixed seed for reproducibility)

Table 3.2 represents the default configuration of LGBMRegressor in scikit-learn, with the exception of random_state, which was fixed at 42 to ensure reproducibility. At this stage of the analysis, no hyperparameter tuning was conducted; instead, the default LightGBM settings were applied to provide a baseline evaluation of its forecast performance. This approach enables a fair and consistent comparison with Random Forest and Linear Regression models, while also establishing a reference point for subsequent experiments that may incorporate parameter optimization to further enhance predictive accuracy.

- **Linear Regression** A parametric model that estimates the relationship between independent variables and the target demand using the ordinary least squares (OLS) method. This model has no hyperparameter; its coefficients are derived analytically to minimize the sum of squared residuals.

Table 3.3: Linear Regression Parameters Used in This Study

Parameter	Value
fit_intercept	True (model learns an intercept term)
copy_X	True (copies input matrix X)
n_jobs	None (single-threaded execution)
positive	False (coefficients can be positive or negative)

Table 3.3 indicate that the Linear Regression model was implemented using the default

configuration of the `LinearRegression` class in scikit-learn. This corresponds to an ordinary least squares (OLS) regression with an intercept term and without any coefficient constraints. Linear Regression was included in this study primarily as a baseline model, offering a simple and interpretable benchmark against which the performance of more advanced machine learning approaches such as Random Forest and LightGBM could be evaluated. Although less flexible in capturing nonlinear patterns, the model provides valuable insights into the linear relationships between explanatory features and demand.

These parameter settings were chosen on the basis of commonly recommended defaults and iterative tuning to achieve a stable benchmark performance. They serve as the initial comparison point for evaluating the effect of feature engineering and feature selection on forecast accuracy.

3.1.3.6 Feature Selection as the Feature Reduction Method

Feature selection in this study was performed using a *Feature Reduction Method*, also referred to as a *Feature Take-Out Simulation*. The aim of this method is to identify and retain only those features that provide meaningful contributions to the predictive performance of the forecasting model, while discarding irrelevant or redundant variables. The overall procedure is illustrated in Figure 3.2.

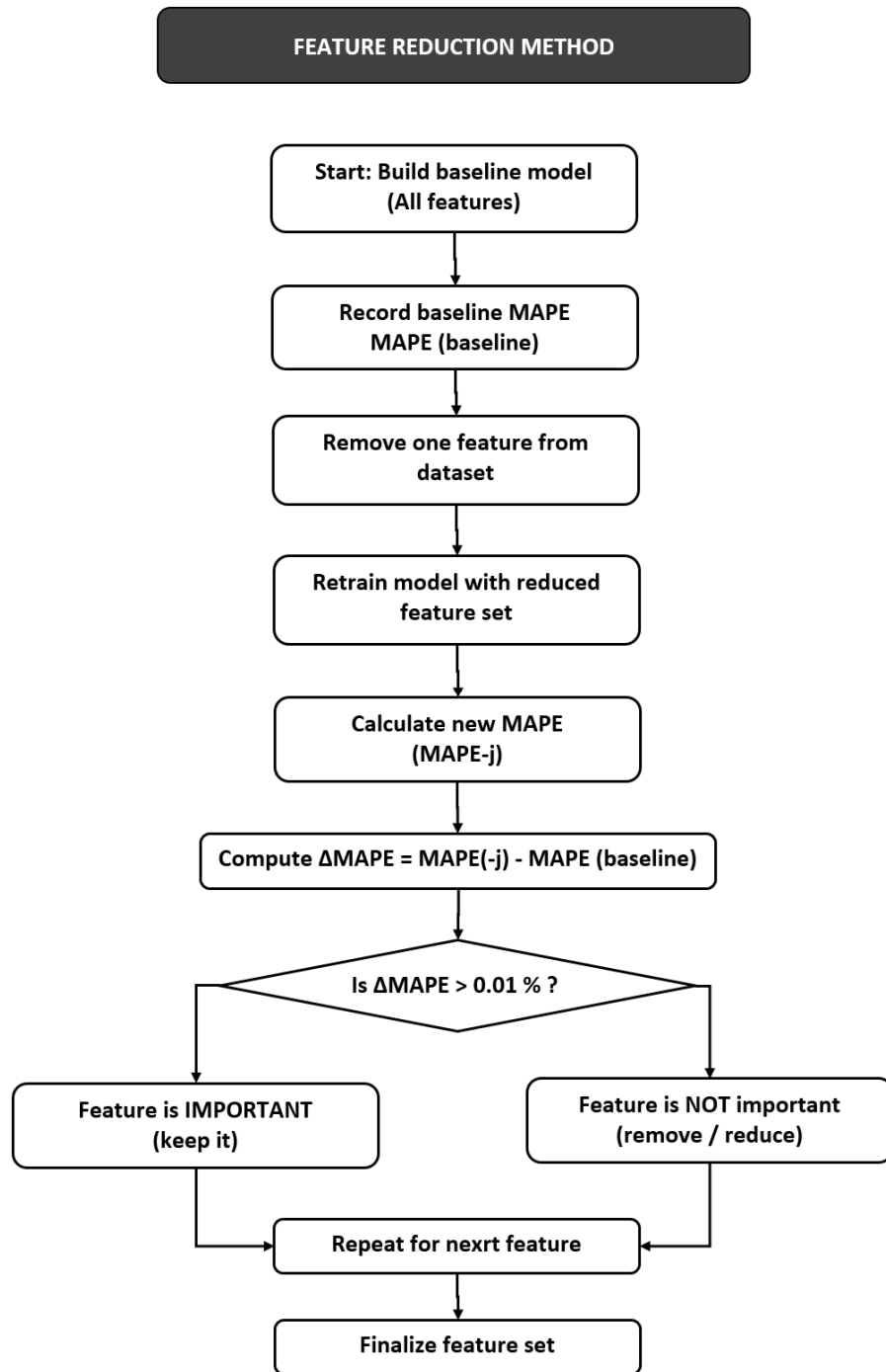


Figure 3.2: Feature Reduction Method (Feature Take-Out Simulation) Flowchart

The process can be described in the following steps:

1. **Build the Baseline Model (All Features).** A baseline forecasting model is constructed first using the complete set of candidate features. The Mean Absolute Percentage

Error (MAPE) of this model, indicated as $MAPE_{baseline}$, is recorded as the reference performance.

2. **Remove One Feature at a Time.** Each feature is individually excluded from the dataset, while all others are retained, creating a reduced feature set.
3. **Retrain the Model with the Reduced Feature Set.** The forecasting model is re-trained using the reduced dataset that excludes the chosen feature.
4. **Calculate the New MAPE.** The predictive performance of the reduced model is evaluated by computing a new MAPE, denoted as $MAPE_{-j}$, where j represents the excluded feature.
5. **Compute the Change in Accuracy ($\Delta MAPE$).** The impact of removing each feature is quantified as:

$$\Delta MAPE = MAPE_{-j} - MAPE_{baseline}$$

6. **Decision Rule for Feature Importance.**

- If $\Delta MAPE > 0.01\%$, the excluded feature is considered **important**, since its removal degrades the predictive accuracy.
- If $\Delta MAPE \leq 0.01\%$, the feature is considered **not important** and may be discarded or reduced.

7. **Iterative Evaluation Across All Features.** Steps 2 to 6 are repeated sequentially for each characteristic, ensuring that the contribution of every variable is independently assessed.
8. **Finalize the Feature Set.** After all features have been evaluated, the final set of important features is established and subsequently used to train the optimized forecasting models.

This method provides a model-based empirical criterion for feature selection. Unlike purely statistical correlation measures, the Feature Take-Out Simulation directly evaluates the impact of each feature on predictive accuracy, thus aligning the selection of features with the main forecasting objective of this study.

3.1.3.7 Comparison with Other Feature Selection Methods

It is worth noting that there are several alternative feature selection approaches.

- **Filter Methods** (e.g., correlation analysis, mutual information) evaluate the features independently of the model. Although computationally efficient, these methods do not capture interactions between variables and may overlook features that are weak in isolation but strong in combination.
- **Wrapper Methods** such as recursive feature elimination (RFE) iteratively build models by adding or removing subsets of features. These methods are more accurate than filter methods, but are computationally more expensive.
- **Embedded Methods** (e.g., tree-based feature importance, LASSO regression) integrate feature selection within model training. These approaches are efficient, but are often biased toward certain types of variables, such as categorical splits in tree-based models.
- **Permutation Importance** evaluates the increase in prediction error when the values of a feature are randomly permuted. Although model-agnostic, this method may be unstable when features are highly correlated.

Table 3.4: Comparison of Feature Selection Methods

Method	Description	Advantages	Limitations
Filter	Selects features based on statistical measures (e.g., correlation, mutual information) independent of the model.	• Simple and computationally fast • Works as a preprocessing step	• Ignores feature interactions • May discard weak but complementary features
Wrapper	Iteratively evaluates subsets of features by training models (e.g., Recursive Feature Elimination).	• Captures feature interactions • Produces feature sets tailored to model performance	• Computationally expensive • Risk of overfitting on small datasets
Embedded	Performs feature selection during model training (e.g., LASSO, decision tree importance).	• Efficient (built into learning process) • Scales well to high-dimensional data	• Bias toward certain feature types • Dependent on model assumptions
Feature Take-Out Simulation (This Study)	Builds a baseline model, removes one feature at a time, retrains, and measures change in predictive accuracy (Δ MAPE).	• Directly measures impact on forecasting accuracy • Model-specific and interpretable • Aligns with research objective	• Computationally more demanding • Requires repeated retraining

As shown in Table 3.4, the simulation of the elimination of features was chosen because it directly measures the change in predictive accuracy when a feature is excluded. This provides a transparent and task-specific criterion for importance that is both interpretable and closely related to the forecasting objective. Although computationally more demanding than filter or embedded methods, it ensures that only features with proven contribution to demand forecast accuracy are retained in the final model.

3.1.3.8 Selected Model Training

After selecting the most relevant features, the final model training is performed using the best-performing algorithms (e.g., Random Forest and LightGBM) and evaluated on a holdout dataset representing unseen periods.

3.1.4 Forecast Evaluation

All models - baseline and machine learning - are evaluated using standardized performance metrics over the 2021–2025 period:

- **Mean Absolute Error (MAE)**
- **Mean Absolute Percentage Error (MAPE)**
- **Root Mean Squared Error (RMSE)**
- **Coefficient of Determination (R^2)**

This multimetric evaluation ensures that models are assessed not only in terms of average magnitude of error, but also in terms of scale independence and variance explanation.

3.1.5 Insight Generation and Success Criteria

Model interpretation is performed to identify the most influential features. This allows for practical insight into which temporal or behavioral drivers (e.g., seasonality, lags) most affect future demand.

The project is considered successful if the selected ML models demonstrate statistically significant improvements in forecast accuracy over baseline models for all types of items. Forecasts are generated at the SKU-month level and are designed to support monthly planning processes in demand, supply, and inventory management.

Having established the methodological framework, including the preparation of the dataset, the engineering of features, and the application of the Feature Take-Out Simulation for feature selection, the next step is to evaluate the forecasting models. The next chapter presents the empirical results obtained from training and testing the Random Forest, and Linear Regression models on the selected feature sets. The results are reported in terms of standard evaluation metrics, namely Mean Absolute Percentage Error (MAPE), and the coefficient of determination (R^2). This chapter also examines the comparative performance of models before and after feature engineering and feature selection, highlighting the contribution of these methodological choices to the overall accuracy of the forecast.

