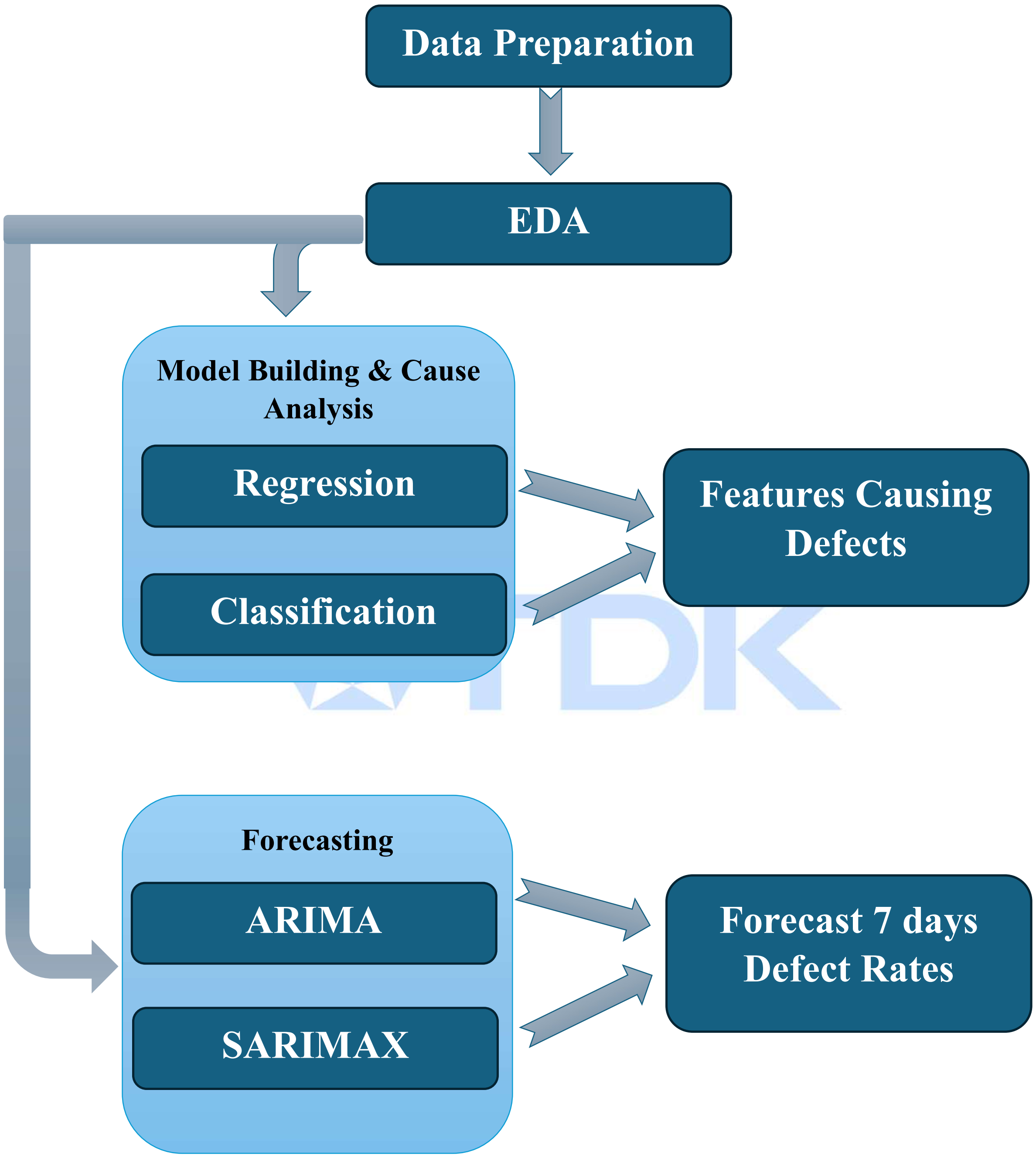
Attracting Tomorrow

Defect Rate Early Identification

Predictive Analysis

<u>**Project Flow**</u>



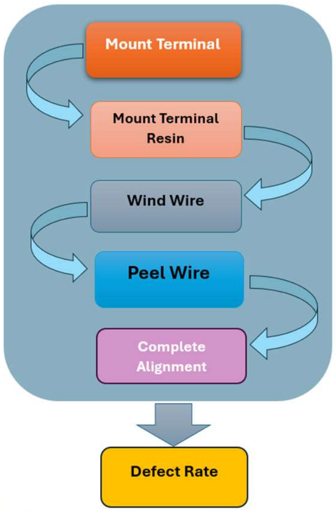
1. Data Importing & Preparation

The first crucial step in identifying defect rates early in the production process is to carefully import and prepare the datasets, ensuring they align correctly for analysis. The process involves multiple stages, starting from importing production data to preparing it for analysis and model building.

1. Data Importing & Preparation

1.1 Sequential Production Process:

The production process consists of several sequential steps, each contributing to the final product. The steps in the production process are as follows:



Each step in the production sequence must be performed in the exact order, as they form a linear workflow where the completion of one step triggers the beginning of the next.

1.2 Defect Rate:

The defect rate is calculated at the final stage of the production process. It measures the proportion of defective units in each sample, which provides an insight into the overall production quality. The defect rate is calculated using the following formula:

$$\text{Defect Rate} = (\text{Number of Defective Units} / \text{Total Sample Size}) \times 100$$

For example, if 100 units are sampled, and 5 units are found defective, the defect rate is:

$$\text{Defect Rate} = (5 / 100) \times 100 = 5\%$$

This defect rate is then used to assess the quality of the production process and predict potential issues in future production runs.

1.3 Importing All Datasets:

To begin the analysis, the data for each production step and the defect rates is imported from CSV files. The datasets include information on each production process, such as time, measurement count, and other key parameters for each of the production stages (Mount Terminals, Mount Terminals Resin, Wind Wire, Peel Wire, and Check Alignment), as well as the defect rate data.

- **Mount Terminals (mt)**
- **Mount Terminals Resin (mtr)**
- **Wind Wire (ww)**
- **Peel Wire (pw)**

- **Check Alignment (ca)**
- **Defect Rates (dr)**

1.4 Data Sorting & Adding Prefix:

Upon inspection of the datasets, it was observed that the Measurement Count column contains unique values across all the datasets. Moreover, there is a linear relationship between Measurement Count and Time, meaning that as Measurement Count increases, the Time also increases.

To ensure proper alignment of the datasets, all the datasets are sorted based on the Measurement Count and Time columns. This sorting ensures that the measurements across different production steps are chronologically aligned.

Additionally, it was noted that each production step dataset uses the same column names. To distinguish them and avoid confusion during analysis, a prefix is added to each column name corresponding to its respective production process step. This ensures clarity and helps in identifying the origin of each feature.

A user-defined function is used to add the prefix to selected columns, and this function is applied across all production process datasets.

1.5 Production Process Data Concatenation:

The production process involves several sequential steps, with defect rate inspection occurring only after the final production step, proper alignment of time is essential. However, the timing of different production steps does not align naturally across datasets. To resolve this, the minimum time for each production step is aligned with the minimum time of the subsequent process for the first four stages.

For the final step, **Check Alignment**, the time value from this stage is used as the reference, as the defect rate inspection occurs only after the completion of this step. The assumption is that the **Check Alignment** step serves as the last point in the process, marking the moment when defect rate evaluation is performed.

After sorting all the datasets based on Measurement Count and Time, the datasets for each production step are concatenated. This ensures that the production steps are properly aligned with the defect rate data, allowing for accurate analysis.

1.6 Defect Rates Data Sorting Based on Time:

The defect rate data, upon review, is found to be in reverse chronological order. To ensure that it aligns with the other production datasets, it is necessary to reorder the defect rate data to reflect the correct time sequence.

This is accomplished by reversing the order of the rows in the defect rate dataset, ensuring that the data is aligned with the production process datasets. The index column is also dropped during this step to clean up the dataset.

2. Exploratory Data Analysis - Data Cleaning

2.1 Dropping the Date & Time Columns (Starting 3 Steps):

As we prepare the data for analysis, we identify that the **Date** and **Time** columns present in the first four steps of the production process (Mount Terminals, Mount Terminals Resin, Wind Wire, and Peel Wire) are not essential for our defect rate prediction model. These columns are removed from the datasets, as they are redundant, with the final **Check Alignment** step providing the relevant time-based data.

By eliminating these columns, we simplify the data and ensure that only relevant time information from the last production stage is retained. This approach assumes that only the time from the **Check Alignment** step is significant in predicting defect rates, as the inspection of defects occurs at this final stage.

2.2 Handling Missing Hour Information in Check Alignment & Defect Rate Data:

In the **Check Alignment** data and **Defect Rate** data, the time information is incomplete—specifically, the **Time** column is missing the hour information. The minutes and seconds are recorded, but the hour field is absent. Given that the minute values follow a sequential order, a logical assumption is made to infer the hour for each time interval.

The following transformations are applied:

- The first interval in the **Check Alignment** data is assumed to represent the 0th hour.
- The second interval represents the 1st hour.
- The third interval corresponds to the 2nd hour.

This same method is applied to the **Defect Rate** dataset. The data is adjusted to include the appropriate hour information, allowing for correct datetime alignment across the datasets. After adding the hour information, the **Time** and **Date** columns are concatenated into a single **DateTime** column in both the **Check Alignment** and **Defect Rate** datasets.

Afterward, the individual **Time** and **Date** columns are dropped, leaving only the **DateTime** column for further analysis.

2.3 Removing Columns with No Variance:

Upon analyzing the datasets, it becomes clear that certain columns across all production steps contain no variance, meaning their values remain constant or have no meaningful fluctuation over time. These columns are identified and removed from the datasets to streamline the analysis and improve model efficiency. Some examples of such columns include:

- Columns such as MeasurementCount, OverallJudgment, and OutputBufferMargin from various production steps (e.g., Mount Terminals, Mount Terminals Resin, Wind Wire, Peel Wire, Check Alignment).
- Irrelevant columns that provide no variance in the data, such as Alarm_No, InspectionExecutionID, and various judgment-related columns.

Removing these columns reduces noise in the dataset, ensuring that only features with meaningful variance contribute to model training.

2.4 Outlier Detection and Treatment:

To better understand the distribution of the production data and detect outliers, summary statistics are generated, and box plots are visualized for each feature. This provides a clear overview of the spread and potential outliers across all columns.

Outliers are defined as values that fall outside the bounds of a specified threshold. In this case, outliers are clipped at the lower and upper bounds set to the 1st and 99th percentiles. Some columns, such as #go, #auto, LowerLeft_MoveDistanceX, and LowerRight_ProductCenter_IrradiationDistanceX, exhibit more significant outlier presence. These columns are treated by capping their values at the specified percentile bounds.

A user-defined function is applied to clip outliers for all float-type columns in the dataset, ensuring that extreme values do not unduly influence the model training process.

2.5 Data Rolling:

The production data is collected at a finer granularity (seconds), while the defect rate is recorded at the minute level. To align these datasets and create a consistent time-based relationship, the data is aggregated at the minute level.

Assumptions:

- Each record represents a unit in the production process.
- All units produced within a given minute contribute to the defect rate for that minute.
- The defect rate is calculated every minute based on the number of units produced in that same minute.

Both the production data and defect rate data are resampled to the minute level, using mean aggregation for numerical columns. This rolling aggregation ensures that the data is consistent and aligned across the datasets, making it suitable for time series analysis.

2.6 Adding Temporal Features:

To capture temporal patterns in the data, two new features, **Hour** and **Minute**, are extracted from the **DateTime** column. These features provide valuable information about the time of day, which may influence defect rates due to factors such as fatigue, production scheduling, or machine performance.

By resetting the index to manipulate the **DateTime** column, the **Hour** and **Minute** values are extracted and added as new columns. After this transformation, the **DateTime** column is set back as the index, ensuring proper alignment for time series forecasting.

2.7 Train-Test Split:

To prepare for model building, the dataset is split into training and testing sets. The training set consists of 80% of the data, while the testing set contains the remaining 20%. This split allows for the model to be trained on historical data and tested on unseen data to evaluate performance.

The target variable, **Defect Rate**, is separated from the feature variables, ensuring that the model can learn the relationship between production parameters and defect rates.

2.8 Scaling:

Feature scaling is applied to standardize the range of features in the dataset. StandardScaler from sklearn.preprocessing is used to normalize the features to have a mean of 0 and a standard deviation of 1. This ensures that no feature dominates the model training process due to its larger numerical range.

The scaler is fitted on the training data and then applied to the test data to maintain consistency in feature scaling across both datasets. This step is crucial for ensuring that all features are on the same scale, making them comparable for machine learning algorithms.

3. Model Building & Evaluation – Cause Analysis:

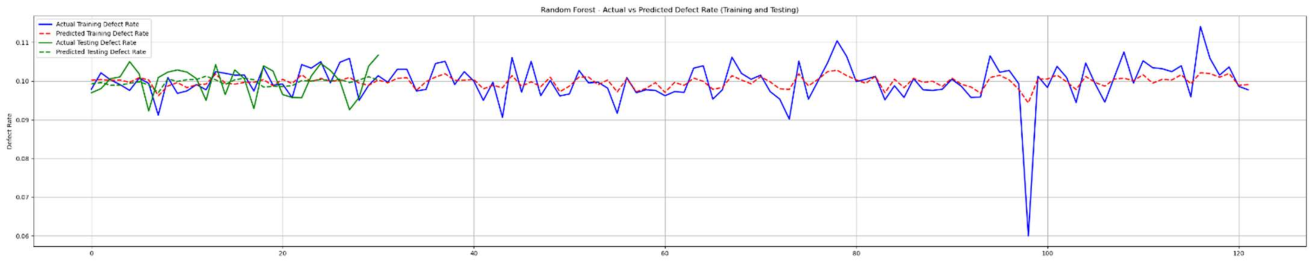
3.1 Regression approach:

Random Forest:

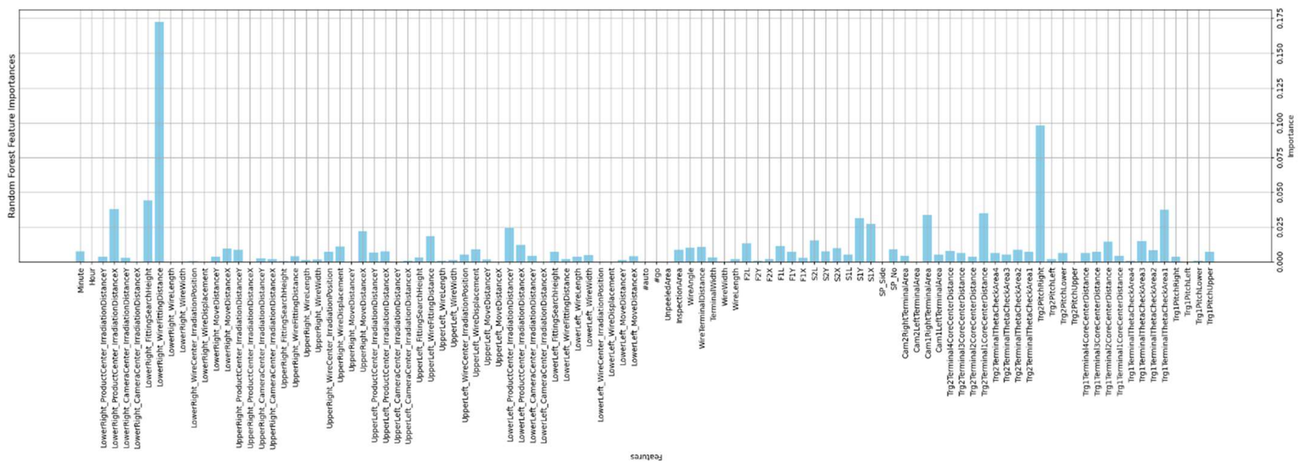
The first model we evaluated was **Random Forest**, a robust ensemble learning technique that typically performs well on structured data. By leveraging the **GridSearchCV** method, we tuned the hyperparameters, including the number of estimators (n_estimators), tree depth (max_depth), and the number of samples required to split or create leaves in each tree (min_samples_split, min_samples_leaf). After fitting the model to the training data, the optimal set of hyperparameters was identified.

The Random Forest model achieved an impressive **MAPE of 0.0339**, which immediately made it a strong contender in the race. The results indicated that Random Forest could capture the underlying patterns in the data effectively without significant overfitting, as evidenced by its ability to generalize well to the test data.

Random Forest Predictions:



Random Forest Feature Importance:



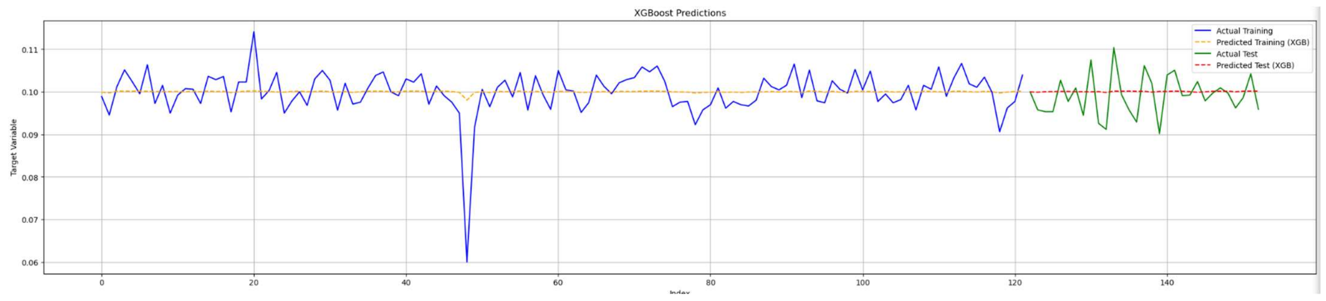
XGBoost:

The code outputs the Random Forest model as the final one, even though I created an XGBoost model and compared its performance. The following are my observations about the XGBoost.

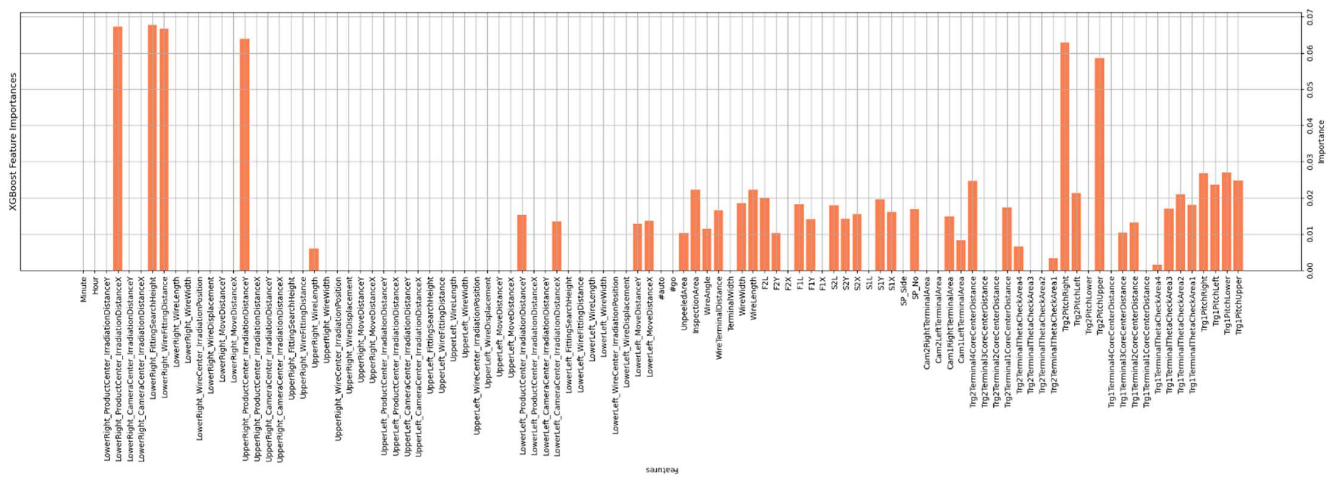
XGBoost is a more advanced gradient boosting technique. It performs exceptionally well on tabular data by using boosted decision trees and optimizing a set of parameters like the learning rate (learning_rate), number of estimators, tree depth (max_depth), and subsampling rate (subsample).

Despite the fine-tuning, the XGBoost model delivered a slightly higher MAPE of **0.0340** compared to Random Forest. While XGBoost performed well, it didn't outperform Random Forest, suggesting that its configuration may not have been optimal for this specific dataset. However, it still remained a strong alternative to Random Forest for future exploration. (

XG Boost Train & Test Predictions:

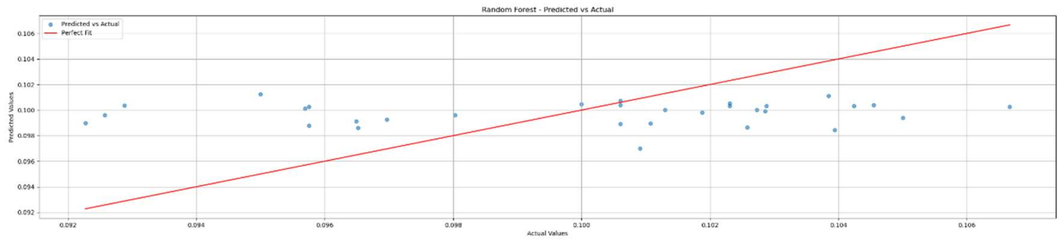


XG Boost Future Importance:



Best Model Selection:

After training and evaluating all models, **Random Forest** emerged as the best-performing model with the lowest MAPE of **0.0386**. This was slightly better than XGBoost. This outcome suggests that for this dataset and problem, Random Forest's ensemble approach with simple decision trees was more effective at capturing the key patterns in the data without the need for complex deep learning architectures.



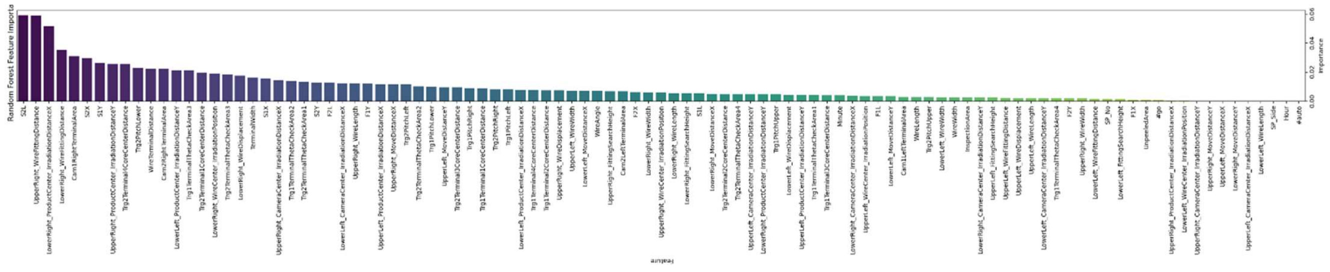
Model Visualization & Feature Importance:

To better understand the inner workings of the Random Forest model, we visualized its **feature importances**. This showed which features were driving the model's predictions. Visualizing these feature importances helps to gain insights into the data and identify which variables have the most impact on the target variable.

Similarly, we visualized the **predictions** of Random Forest, XGBoost. This step gave us a direct comparison of how well the models performed on both the training and test data. The plots clearly showed that Random Forest was able to predict the target variable with great accuracy, as the predicted values closely aligned with the actual values.

Cumulative Feature Importance:

To refine the model further, we conducted a **feature selection** step based on the feature importances derived from the Random Forest model. By identifying the features that contribute most to the predictions, we narrowed down the list of features to those that account for 99% of the importance. This reduced feature set was then used to re-train the model.



After filtering out the less important features, we observed that Random Forest's performance did not degrade, confirming that the most critical features were already selected. This selective approach can simplify the model and potentially reduce overfitting.

3.2 Classification Approach:

To achieve this, we combined several statistical and machine learning techniques, and classification models, to identify the production steps (e.g., Mount Terminals, Mount Terminal Resin, Wind Wire, Peel Wire, Check Alignment) that have the most significant impact on defect rates.

1. **Defining the Defect Rate Categories:**
   - To facilitate the analysis, we categorized the defect rate into two intervals based on its average value:
     - **Low Defect Rate (0)**: Values below the average defect rate.

        - **High Defect Rate (1)**: Values above the average defect rate.
    - This categorization allows us to focus on variations in defect rates and their relationship with production steps, simplifying the analysis.
2. **Feature Importance from Classification Models:**
    - We trained classification models (Random Forest and XGBoost) on the categorized defect rates to evaluate feature importance.
    - Both models were optimized using **GridSearchCV** for hyperparameter tuning to find the best configuration.
    - After training, we used the models to evaluate which features had the most influence on the defect rate classification.
3. **Classification Model Results:**
    - **Random Forest Model:**
        - The **Random Forest classifier** produced an F1-Score of **0.4318**, indicating moderate performance in classifying defect rates.
        - Key features identified by the Random Forest model as important include 'Trg2Terminal3CoreCenterDistance', 'Trg1Terminal4CoreCenterDistance', and others.
    - **XGBoost Model:**
        - The **XGBoost classifier** performed slightly worse than Random Forest, with a **weighted average F1-Score of 0.4283** on the test data.
        - Despite the lower performance, it also pointed to similar important features as Random Forest, reinforcing the insights about critical production steps.

Feature Importance & Insights

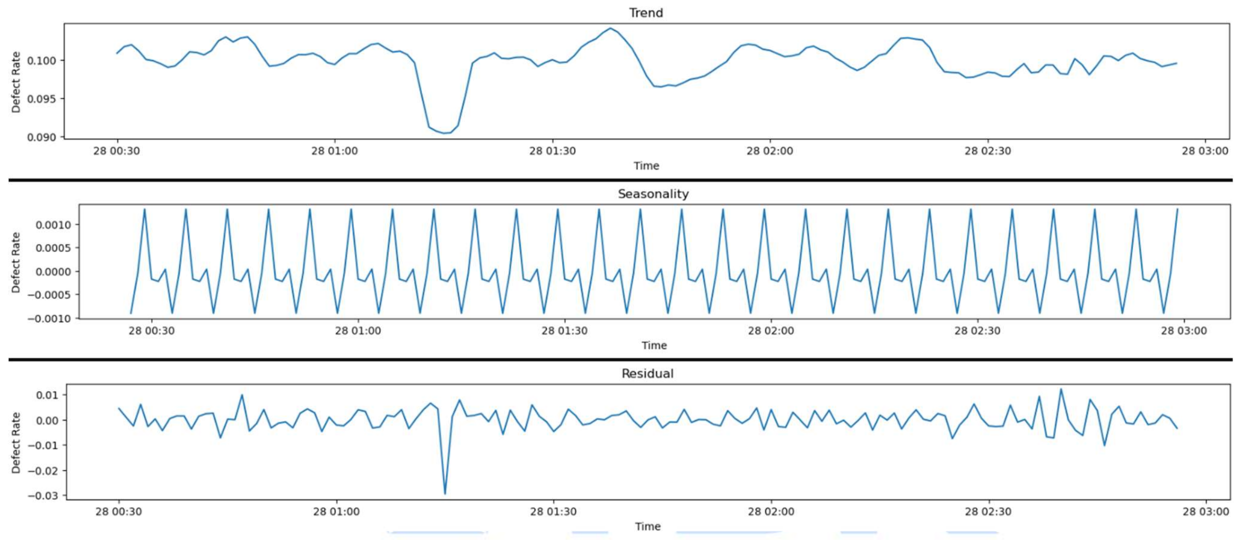
- **Top Features Impacting Defect Rates**:
    - Both models identified certain key features that influence the defect rate. These features were categorized by production stages:
        - **Mount Terminal (Stage 1)**: Features like 'Trg2Terminal3CoreCenterDistance', 'Trg1Terminal4CoreCenterDistance', 'Trg2PitchLeft', and 'Trg2PitchRight' were identified as important contributors to defect rates.
        - **Mount Terminal Resin (Stage 2)**: 'Cam1RightTerminalArea' and 'Cam2RightTerminalArea' were highly impactful.
        - **Wind Wire (Stage 3)**: 'S1L', 'S2L', and 'S2X' were crucial features influencing defect rates.
        - **Peel Wire (Stage 4)**: No significant features were identified in this stage in relation to defect rates, suggesting that the features from this stage do not significantly contribute to defects.
        - **Complete Alignments (Stage 5)**: Features like 'UpperRight_MoveDistanceX', 'LowerRight_MoveDistanceX', and 'UpperRight_CameraCenter_IrradiationDistanceX' were significant.

4. Defect Rate Prediction for the Next 7 Days

4.1. Time Series Decomposition:

The first step in understanding the behaviour of the defect rate (y) was to decompose the time series data into its underlying components. The time series decomposition revealed three key elements:

- **Trend:** This component captures the long-term movement or direction of the defect rate over time, though no significant trend was observed.
- **Seasonality:** This reflects periodic fluctuations in the defect rate, recurring every 6 minutes.
- **Residuals:** These represent the random fluctuations or noise that cannot be explained by the trend or seasonality and appeared to be well-behaved.

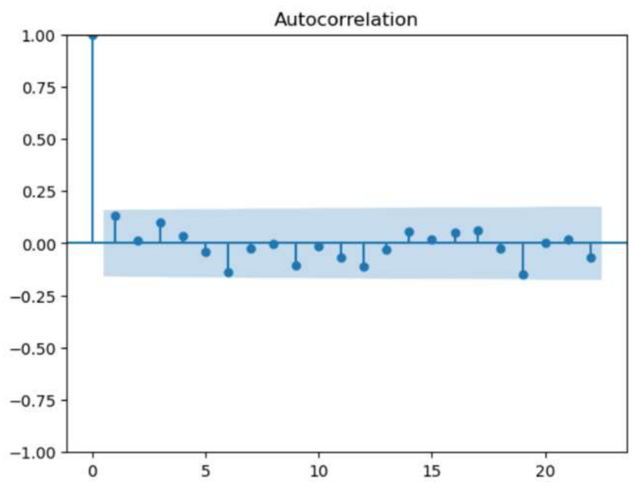


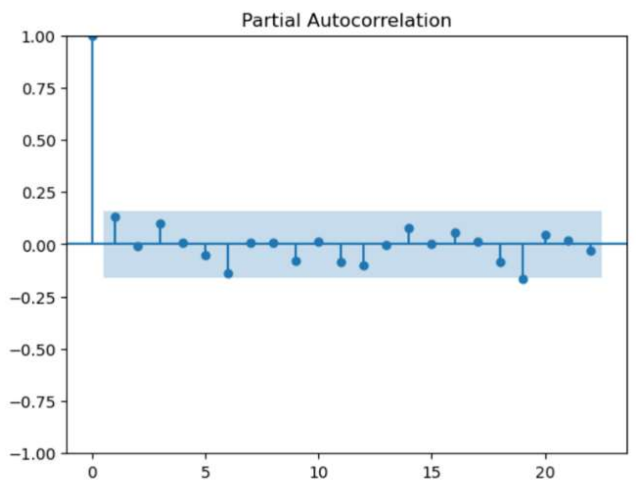
4.2. Stationarity Check with ADF Test

Next, the stationarity of the defect rate time series was tested using the Augmented Dickey-Fuller (ADF) test. The result showed a highly negative ADF statistic of approximately -10.67 with a p-value of 4.09×10^{-19}, indicating that the defect rate time series is stationary. This meant no differencing was required, which is a good sign for reliable forecasting.

4.3. ACF & PACF Plots

To identify the autocorrelations in the data and decide on the order of the ARIMA model, we plotted the Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF). These plots helped to visualize the relationship between the current value of the defect rate and its past values.



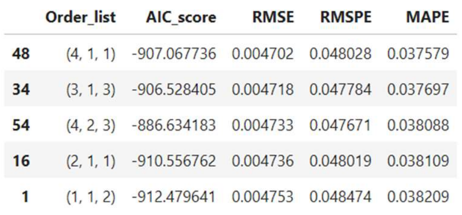


4.4. ARIMA Model Selection

The ARIMA (Auto-Regressive Integrated Moving Average) model was applied to the data to capture the time series characteristics and forecast future values. To determine the best parameters (p, d, q), we used an exhaustive search over various combinations and evaluated the models based on key performance metrics like AIC, RMSE, RMSPE, and MAPE.

The top-performing ARIMA models included combinations like (4, 1, 1) and others, with the lowest RMSE, RMSPE, and MAPE scores.

Output: The ARIMA model with order (4, 1, 1) had the best performance, achieving the lowest RMSE and MAPE.



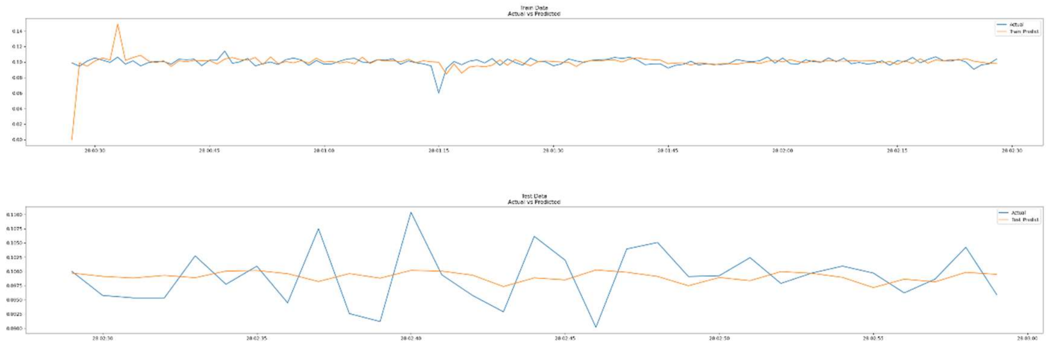
	Order_list	AIC_score	RMSE	RMSPE	MAPE
48	(4, 1, 1)	-907.067736	0.004702	0.048028	0.037579
34	(3, 1, 3)	-906.528405	0.004718	0.047784	0.037697
54	(4, 2, 3)	-886.634183	0.004733	0.047671	0.038088
16	(2, 1, 1)	-910.556762	0.004736	0.048019	0.038109
1	(1, 1, 2)	-912.479641	0.004753	0.048474	0.038209

4.5. Model Fitting and Prediction

Once the best ARIMA model was identified, it was fit to the training dataset. The model was then used to predict the training and test datasets. The actual versus predicted values were plotted to visualize how well the model performed on both datasets.

4.6. SARIMAX Model (Seasonal ARIMA)

Given the seasonality in the defect rate data, a Seasonal ARIMA (SARIMAX) model was explored. This model considers both seasonal and non-seasonal components of the time series data. The evaluation process was similar to ARIMA, but it also included seasonal orders. The best SARIMAX model was selected based on its AIC score and other metrics.



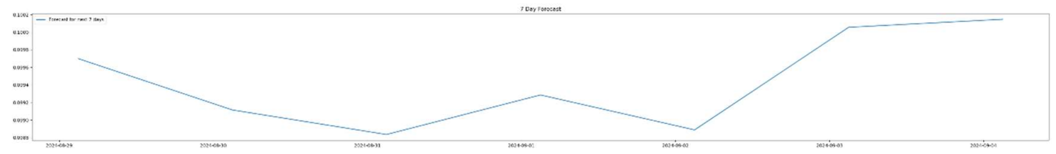
The final model chosen for forecasting was SARIMAX with the order (2, 1, 2) and seasonal order (1, 1, 3, 6). This model accounted for both trend and seasonal fluctuations in the defect rate.

4.7. Final Forecast for the Next 7 Days

After fitting the SARIMAX model, it was used to forecast defect rates for the next 7 days (or 1*24*60 points, depending on the resolution of the data). The model produced a forecast for future defect rates based on the past data.

Comparison Between ARIMA and SARIMAX Models:

Finally, to compare the performance of the ARIMA and SARIMAX models, the forecasts for the testing period were plotted. Both models' predictions were compared against the actual test data, and it was observed that the SARIMAX model, incorporating seasonality, provided a slightly better fit.



Conclusion:

This process provides a detailed and comprehensive approach to predicting defect rates. Through time series decomposition, stationarity checks, and model fitting (ARIMA and SARIMAX), we identified the best model for forecasting defect rates. The SARIMAX model emerged as the most accurate, incorporating both trend and seasonal components, and was used for the final 7-day forecast.

Disclaimer: The results, evaluation metrics, and visualizations presented in this analysis may not accurately reflect the true performance or underlying patterns due to potential inconsistencies and misalignments in the data. While the methodologies and models applied provide useful insights, the presence of data irregularities and noise could have affected the accuracy of the outcomes. The findings and interpretations should be considered preliminary, and further data refinement and validation are recommended to achieve more reliable results.

Thank You