Capstone Project: SP25 Sales Forecasting

Ban-693-01

Spring 2025

Ishraque Shahriar

iv7151

Introduction

Effective demand forecasting is critical for steel manufacturers to balance production, inventory, and logistics while minimizing costs. Overproduction leads to excess inventory expenses, while shortages can result in lost sales and operational disruptions. To address this challenge, a predictive model has been developed to forecast quarterly sales for each customer using company-specific data and economic indicators. By providing accurate sales projections, this model enables sales and operations teams to optimize production scheduling, streamline supply chain management, refine pricing strategies, and enhance financial planning. Ultimately, it supports data-driven decision-making, improving efficiency and overall business performance.

The model's accuracy will be assessed using Mean Squared Error (MSE) and Mean Absolute Error (MAE). MSE measures the average squared differences between predicted and actual sales, emphasizing larger errors, while MAE calculates the average absolute differences, providing a straightforward accuracy measure. A low MSE ensures minimal large deviations, and a low MAE indicates consistent predictions. Other metrics, such as RMSE and MAPE, could be explored, but MSE and MAE will be the primary focus for this project.

Overview of best solution

The best-performing approach (Attempt 10) leveraged a structured pipeline that combined temporal, company-level, and macroeconomic insights. Key elements included engineered sales lag features to capture recent performance trends, rolling economic indicators like consumer sentiment and interest rates, and company-specific statistics such as average sales and volatility. Categorical variables were effectively handled using label and one-hot encoding to prepare the data for modeling.

The model was trained using XGBoost, with hyperparameters fine-tuned via GridSearchCV, ensuring an exhaustive and targeted search across the most relevant parameter combinations. This configuration delivered the lowest validation error across all attempts, demonstrating strong predictive accuracy and model stability for forecasting Q8 and Q9 sales.

Summary of report structure

The capstone report is structured to guide the reader through the full lifecycle of the sales forecasting project. It begins with an introduction outlining the business problem, objectives, and evaluation metrics. It then presents the best-performing solution (Attempt 10), followed by detailed descriptions of both the company-level and economic datasets used. The methods section explains the step-by-step modeling process, including EDA, preprocessing, feature engineering, and hyperparameter tuning. A comprehensive results table compares all modeling attempts, highlighting improvements and insights at each stage. The report also reflects on lessons learned, documents the role of GenAI in various tasks, and discusses the future impact of AI on analytics roles. Finally, it offers a team-oriented project execution plan and concludes with an appendix containing the final model code.

Data

Train and Test Dataset

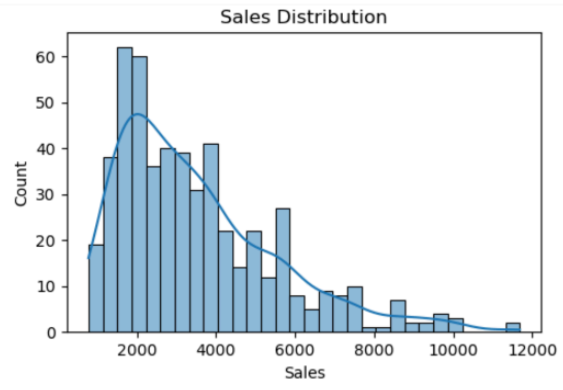
Feature	Description	Relationship to Sales
Quarter	The time period in which sales occurred (Q1, Q2, etc.).	Sales may exhibit seasonality.
Company	The company/customer buying steel.	Different customers may have different purchasing patterns.
Inventory Ratio	Inventory levels relative to sales.	High inventory might indicate lower future demand.
RevenueGrowth	Quarterly revenue growth of the company.	Higher revenue growth could lead to higher steel purchases.
Marketshare	Company's share in its industry.	A larger market share might indicate more consistent sales.
Bond rating	Creditworthiness of the company (AAA, BBB, etc.).	Stronger financial health may correlate with higher purchases.
Stock rating	Investment rating (Strong Buy, Buy, Hold, Sell, Strong Sell).	A "Buy" rating could indicate expected business growth.
Region	Geographical location of the company.	Demand may vary across regions based on economic activity.
Industry	The sector in which the company operates.	Some industries (e.g., construction) consume more steel.
QuickRatio	Liquidity measure (assets/liabilities).	Higher liquidity may enable more frequent purchases.
Sales (Target Variable)	The steel purchased by the company (in units).	The goal is to predict this variable.

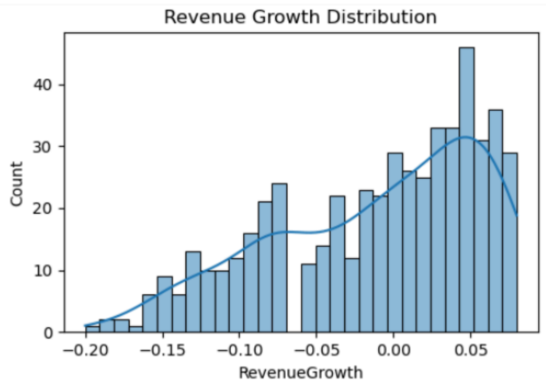
Economic Indicator Dataset

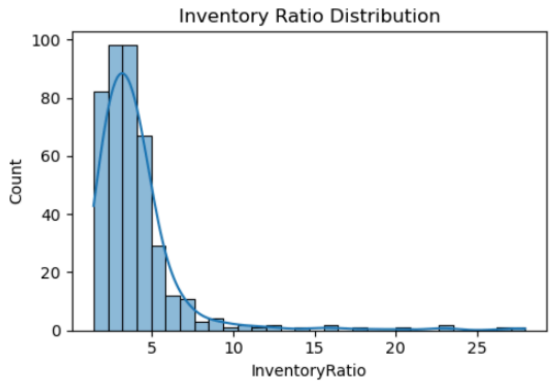
Feature	Description	Relationship to Sales
Month	Month of observation.	Helps align economic conditions with sales periods.
Consumer Sentiment	Confidence of consumers in the economy.	Higher sentiment could indicate stronger demand.
Interest Rate	Cost of borrowing money.	Higher rates may reduce investments and demand.
PMI	Purchasing Managers' Index (business activity).	A higher PMI suggests stronger industrial demand.
Money	Total money available in the economy.	More liquidity may drive business growth and

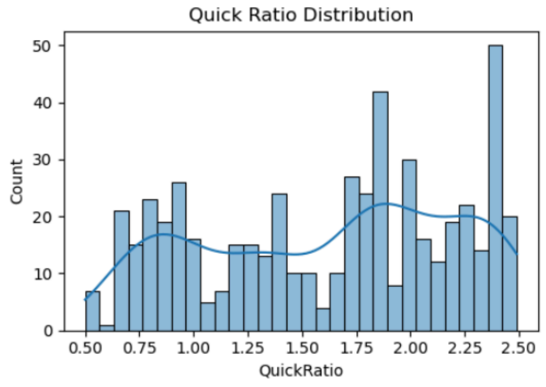
Supply		steel demand.
NationalEAI	National Economic Activity Index.	A higher index suggests a growing economy, potentially increasing steel demand.
Regional EAI	Economic activity index for East, West, South, North.	Regional demand variations based on economic conditions.

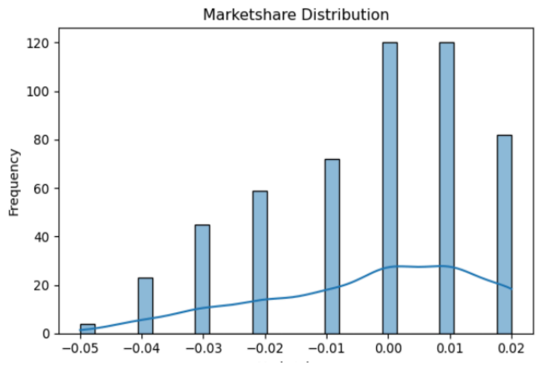
Data Distribution in Train dataset







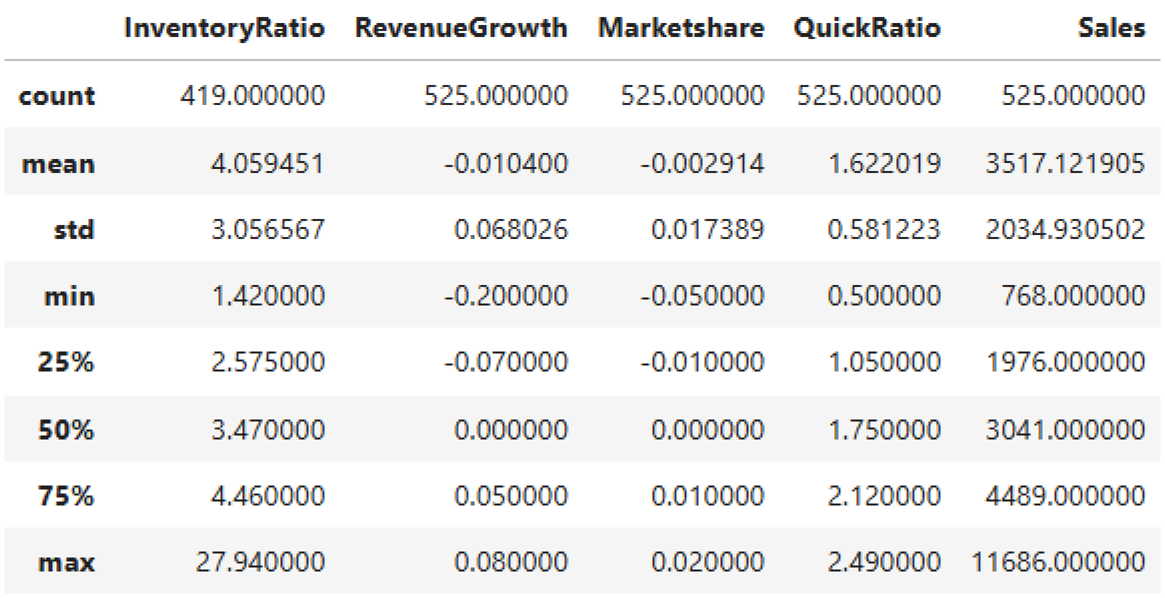




Graph Interpretations

- Sales Distribution: Skewed to the right, with most companies ordering 1,500–3,000 units, while a few exceed 10,000 units. A small group of customers accounts for the bulk of sales.
- Revenue Growth: Largely stable (-10% to +5%), though some companies show significant drops (below -15%) or strong growth (above +6%), potentially influencing demand.
- Inventory Ratio: Strongly skewed, with most values between 1.5 and 5, but some companies hold exceptionally high inventory (above 25), which may reduce future purchases.
- Quick Ratio: Displays a bimodal pattern, clustering around 1.0 and 2.3. Companies with very low quick ratios (~0.5) might struggle with liquidity, impacting purchasing ability.
- Market Share Growth: Slightly skewed, with most values between -0.02 and 0.02. A few company quarters show notable declines (-0.05) or gains (0.02), indicating minor fluctuations in market positioning. Frequency represents company quarters, not individual companies.

Descriptive Statistics



	InventoryRatio	RevenueGrowth	Marketshare	QuickRatio	Sales
count	419.000000	525.000000	525.000000	525.000000	525.000000
mean	4.059451	-0.010400	-0.002914	1.622019	3517.121905
std	3.056567	0.068026	0.017389	0.581223	2034.930502
min	1.420000	-0.200000	-0.050000	0.500000	768.000000
25%	2.575000	-0.070000	-0.010000	1.050000	1976.000000
50%	3.470000	0.000000	0.000000	1.750000	3041.000000
75%	4.460000	0.050000	0.010000	2.120000	4489.000000
max	27.940000	0.080000	0.020000	2.490000	11686.000000

Methods

Throughout the project, I explored multiple modeling strategies, starting with basic implementations and gradually enhancing the pipeline through advanced preprocessing, feature engineering, and model optimization. I experimented with baseline regression models, gradient boosting techniques, and various configurations of training-validation splits and economic feature integrations from given datasets.

The most effective approaches were attempts 9 and 10, with attempt 10 delivering the best overall performance. In this attempt, I built a forecasting model using XGBoost to predict quarterly sales for Q8 and Q9 based on historical company performance and economic indicators. I began with exploratory data analysis (EDA) to understand the structure and distribution of the datasets. I used visualizations such as boxplots, histograms, and time series plots to examine quarterly sales trends across companies and identify potential outliers, seasonal patterns, and correlations with economic indicators. This helped shape my feature engineering strategy. I then moved to preprocessing the data, converting quarter labels to numeric, imputing missing values, and merging rolling means of macroeconomic indicators into both train and test datasets. I engineered features such as sales lags, average past sales, sales trends, and company-level statistics to capture both temporal and business-specific patterns. Categorical variables were encoded, and the dataset was prepared for modeling. I used GridSearchCV to perform exhaustive hyperparameter tuning with cross-validation, which significantly improved model accuracy. Predictions were made for Q8 first, and the output was then used as a lag input for predicting Q9, following a sequential forecasting strategy. The final predictions were saved in the required submission format.

Results

No	Submission description (What was different from previous submission)	Train Error	Val Error	Test Error	Rank at Sub	Insights (Why did the approach work or not work)
1	The training dataset train.csv is loaded, cleaned, and preprocessed by handling missing values through imputation and ensuring data consistency through categorization and mapping of some columns. Features are selected by removing the target variable Sales. A Gradient	396.693546072248 47			1	This was my first submission. The entire train dataset was trained without being split so no validation error. No actual sales data in the test dataset so no test error.

	Boosting Regressor with 200 estimators, a learning rate of 0.1, and a random state of 42 is trained. The economic indicator dataset has not been merged.					
2	Tried to create a baseline by simply running multiple linear regression. Handled null values with median imputation. Most numeric columns were high to moderately skewed so median imputation was considered. Only included default numerical columns for training.	1513.6059	1750.7584	634.887	1	The result was not better than the previous submission. This might be because of not doing the additional cleaning work that was done in the previous submission and also skipping use of the gradient boosting regressor model.
2	The training dataset train.csv is loaded, cleaned, and preprocessed by handling missing values through imputation and ensuring data consistency through categorization and mapping of some columns(ordinal	1453.3514	1660.9411	961.6206	8	The result was not better than the previous submission. Using further data cleaning measures like Conversion of company names into numerical tags, Extraction of numeric values from the Quarter column, ordinal encoding to Bond rating and Stock rating and one-hot encoding for Industry and Region categories and then using the cleaned dataset to train multiple linear regression model did not generate better result. The original dataset had a simpler feature set before encoding and also fewer features

	encoding and one hot encoding). Features are selected by removing the target variable Sales.					since it only considered numeric features for training.
3	Worked on the preprocessed dataset with gradient boosting regressor model but this time followed a train-validation-test workflow instead of just training with the entire train dataset	321.322	903.064		7	The MAE in this model was almost 75 less than the first submission of the same model which is an indication of better performance by the model. The train-validation-test workflow was better as it ensures the model is evaluated only on unseen validation data before making final predictions on the test set. The validation set is used for performance evaluation before the test set is ever used.This prevents data leakage and avoids misleading performance metrics that arise from incorrectly evaluating test predictions against themselves. By keeping the test set truly unseen until the final step, the model better generalizes to real-world data.
4	Worked on the preprocessed dataset with gradient boosting regressor model but this time ran feature importance from gradient boosting to identify top 7 features with most impact on sales and used those only for training the model.	332.153	912.04		9	Cutting down the number of less important features usually helps avoid overfitting. Even though the selection of top 7 important features increased the MAE by around 11 then the last submission, the score in submission was better 779 to 728 after dropping features.

5	Worked on the preprocessed dataset with the XGB model this time. The dataset was merged with the economic indicator dataset and also introduced sales lags 1 and 2 in both the train and test dataset to capture sales trends.	143.61	689.82		7	Merging economic indicators provided external business context, improving macro trend detection, while sales lag features allowed XGBoost to model sales trends over time, capturing short-term momentum and seasonal patterns. Together, these enhancements helped the model identify relationships between past sales, economic conditions, and future sales trends, ultimately reducing Mean Absolute Error (MAE) and improving forecast accuracy.
6	Added Hyperparameter Tuning with RandomizedSearchCV to search across a range of important hyperparameters, used cross validation within hyper parameter tuning.	17.15	697.19		9	The first step allows the model to automatically find a more optimal set of parameters, which usually reduces overfitting and improves generalization to unseen data (Q8 & Q9). Even though the training MAE is significantly low and generates better predictions, the gap in between represents signs of overfitting.
7	Added Hyperparameter Tuning with GridSearchCV to search across a range of important hyperparameters, used cross validation within hyper parameter tuning.	135.70	671.33		8	In this experiment, I used GridSearchCV instead of RandomizedSearchCV to fully explore a small parameter space, resulting in better model tuning. I also applied sequential prediction, using Q8 forecasts as lag inputs for Q9, which improved accuracy by better reflecting real-world sales dependencies. training MAE is higher (135.70) but validation MAE is lower (671.33), than the previous attempt with random search, which is the real goal.

8	Used basic parameters for grid search without any tuning. Manual 80/20 train/test split, applied sequential modeling with individual training for q8 and q9.	163.24	893.99	624.85	13	Even Though the MAE scores are higher and wider apart, they help generate more accurate predictions of q8 and q9 sales. I have trained separate models for Q8 and Q9, allowing it to better capture quarter-specific patterns and dependencies. Additionally, it explicitly incorporates predicted Q8 sales as a lag feature for Q9, enhancing temporal accuracy.
9	This time I have incorporated 3 sales lags into the pipeline, their avg as well as sales trend from difference between the sales lags. I enhanced the feature set by merging rolling economic indicators, calculating company-level sales statistics like avg sales and sales standard deviation. Used randomized search, which sampled a specified number of parameter combinations from defined distributions. This allowed me to explore a wider hyperparameter	117.90	541.27	511.005	5	These steps enhanced the model's ability to capture both temporal and contextual patterns in the data. Additional Sales lag features provided valuable insights into recent performance trends for each company, while economic indicators and company-level statistics added macroeconomic and business-specific context. To improve efficiency, I transitioned to randomized search, which samples a specified number of parameter combinations from defined distributions. This allowed me to explore a wider hyperparameter space, including regularization terms like `reg_alpha` and `reg_lambda`, while significantly reducing training time and maintaining strong model performance.

	space, including regularization terms like `reg_alpha` and `reg_lambda`					
10	The data preprocessing part remained the same as before. I switched to GridSearchCV, which provided a more structured and exhaustive search. Although randomized search offered speed and flexibility, grid search ultimately delivered better model performance and more accurate sales predictions. This outcome suggests that the optimal hyperparameters for my model were within the defined grid and were more effectively captured through exhaustive tuning rather than random sampling.	60.85	69.29	497.45	2	I started with RandomizedSearchCV for hyperparameter tuning to quickly explore a broad set of parameter values, including regularization terms. However, I later opted for GridSearchCV, which performs a systematic and comprehensive search across a predefined parameter grid. While the randomized approach was faster and more flexible, GridSearchCV produced better model performance and more accurate sales forecasts. This indicates that the best-performing parameters were likely within the grid, and the exhaustive nature of grid search was more effective in identifying them than random sampling.

Analysis

Over the course of the semester, I went through a hands-on learning journey, experimenting with different modeling strategies and gradually improving my approach to both data preparation and model tuning. In the beginning, I overlooked the importance of proper validation, which led to overestimated performance and poor generalization. I quickly realized that simply adding more preprocessing or complex models doesn't always lead to better results. In some cases, it actually made things worse by introducing noise or redundancy.

As I refined my approach, I found that blending economic indicators with carefully engineered sales lag features made a big difference. These additions helped the model capture both broader market conditions and company-specific patterns. One of the biggest turning points was switching from RandomizedSearchCV to GridSearchCV, which gave me more control and ultimately better performance. Using sequential predictions, where Q8 results informed Q9 along with training quarter-specific models, brought a noticeable improvement in accuracy. This project taught me that the key to strong forecasting lies not just in technical implementation, but in thoughtful feature design, disciplined validation, and strategic model tuning.

AI Contributions

Task	**Gen AI Prompt**	**Results and Insights**
Data Preparation	ChatGPT: Create Python notebook to read the attached CSV files and explore the structure of train, test, and economic datasets.	ChatGPT generated reusable boilerplate code to load and inspect all data files. It helped identify column types and formatting issues early in the project.
Exploratory Data Analysis	ChatGPT: Suggest EDA steps and Python code to visualize company-level sales trends and economic indicator distributions.	GenAI provided EDA templates including boxplots, scatter plots, and correlation heatmaps. This improved understanding of quarterly patterns and macroeconomic relationships.

Data Preprocessing	ChatGPT: How to handle missing values and standardize quarter formatting in Pandas?	ChatGPT recommended using median imputation grouped by company and transforming Quarter strings to integers, ensuring the time series structure was preserved accurately.
Feature engineering	ChatGPT: What features can I generate for quarterly sales prediction using lag and macroeconomic data?	It suggested creating lag-based features, average of past quarters, and sales trend variables, which significantly improved model learning by adding temporal context.
Model Training	ChatGPT: How to build an XGBoost regression pipeline and tune hyperparameters using GridSearchCV?	AI guided the full modeling pipeline from data split to training and tuning. It helped implement both RandomizedSearchCV and GridSearchCV, with the latter yielding better MAE.
Model Evaluation	ChatGPT: How to interpret MAE and visualize actual vs predicted values?	It provided detailed plots (scatter plot of predicted vs actual) and MAE explanations, helping assess both overfitting and generalization performance.
Prediction Strategy	ChatGPT: How to use Q8 predictions as lag input for Q9 forecast in time series modeling?	Helped develop sequential forecasting logic, feeding Q8 outputs into Q9 lag features, boosting accuracy by aligning predictions with real-world dependencies.

Documentation & Reflection	ChatGPT: Summarize insights and lessons learned from texts in model experiments.	Assisted in articulating key takeaways and generating polished summaries for model evaluation and experimentation insights.

Reflection on Role of GenAI and future of Business Analytics

Generative AI (GenAI) is proving to be a powerful tool in business analytics, offering support across data exploration, preprocessing, model prototyping, and even natural language querying of databases and dashboards. It accelerates tasks like writing code, generating insights, and drafting reports, freeing up valuable time for analysts and data scientists to focus on strategic decision-making. In the near future, many routine analytics tasks such as data cleaning, report generation, dashboard building, and simple forecasting will likely be automated through AI-driven tools, making the analytics process faster and more accessible to non-technical users.

As AI automates more of the technical workload, the role of data analysts and data scientists will shift toward interpreting results, designing experiments, and solving business problems through creative thinking and domain expertise. Skills like SQL syntax memorization or manually writing boilerplate code are becoming less critical. Instead, soft skills such as communication, critical thinking, and business acumen, along with the ability to ask the right questions and translate insights into action, are becoming increasingly valuable. In this evolving landscape, those who can partner with AI rather than compete with it will be best positioned to lead in the future of business analytics.

Project Execution Plan for Analytics Team

Define Objectives & Roles: Align on the business goal: forecast Q8 and Q9 sales with minimal MAE. Assign roles across the team for e.g., data wrangling, modeling, evaluation, and documentation.

Collaborative Data Exploration: Begin with joint EDA sessions to uncover key sales trends, outliers, and seasonal patterns. Share early visual insights to shape feature engineering efforts.

Coordinate Data Cleaning & Feature Prep: Standardize preprocessing steps: quarter conversion, imputation, and encoding. Assign one subgroup to enrich datasets with economic indicators; another to build lag-based and statistical features.

Modeling & Tuning Strategy: Agree on modeling approach (e.g., XGBoost) and split tuning responsibilities. Use RandomizedSearchCV for initial trials and GridSearchCV for final fine-tuning.

Evaluate and Iterate as a Team: Regularly review validation metrics and MAE scores. Visualize actual vs predicted trends. Use team feedback to refine features and adjust model assumptions.

Prediction Pipeline & Handoff: Implement a shared, reproducible workflow for sequential Q8 to Q9 forecasting. Ensure outputs match submission format. Assign documentation and final review before handoff.

Appendix

Code for best performing model

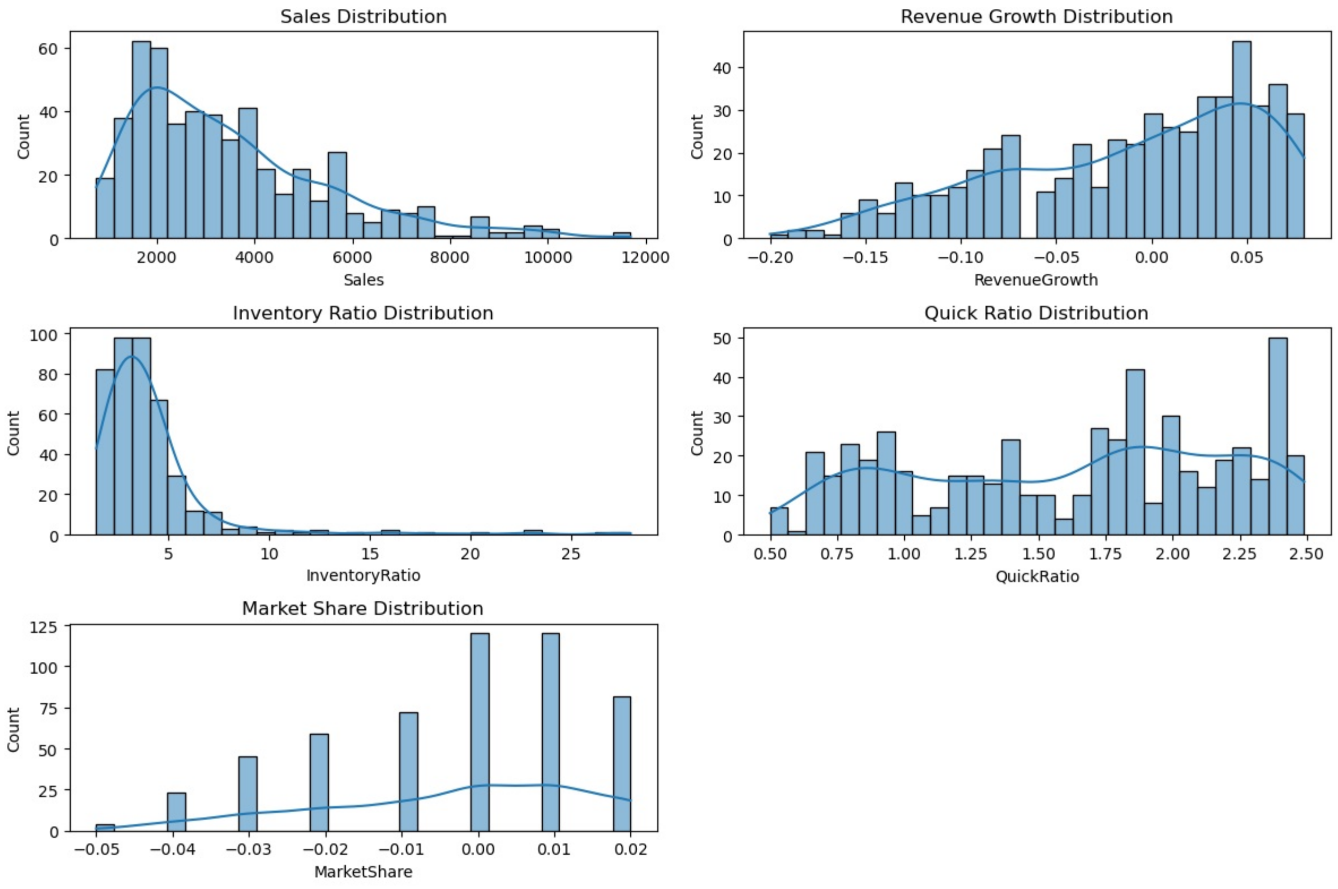
In [3]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from xgboost import XGBRegressor
from sklearn.metrics import mean_absolute_error
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.preprocessing import LabelEncoder
import numpy as np
```

In [4]:
```
## Load datasets
train_df = pd.read_csv("train.csv")
test_df = pd.read_csv("test.csv")
economic_df = pd.read_csv("EconomicIndicators.csv")
sample_submission_df = pd.read_csv("sample_submission.csv")
```

In [5]:
```
# Exploratory Data Analysis (EDA)
print("Train Data Summary:\n", train_df.info())
print("Test Data Summary:\n", test_df.info())

# Check for missing values
print("Missing values in Train Data:\n", train_df.isnull().sum())
print("Missing values in Test Data:\n", test_df.isnull().sum())

# Summary statistics
print("Train Data Description:\n", train_df.describe())
print("Test Data Description:\n", test_df.describe())

# Plot distributions of key numerical features
fig, axes = plt.subplots(3, 2, figsize=(12, 8))

sns.histplot(train_df['Sales'], bins=30, kde=True, ax=axes[0, 0])
axes[0, 0].set_title("Sales Distribution")
axes[0, 0].set_xlabel("Sales")
axes[0, 0].set_ylabel("Count")

sns.histplot(train_df['RevenueGrowth'], bins=30, kde=True, ax=axes[0, 1])
axes[0, 1].set_title("Revenue Growth Distribution")
axes[0, 1].set_xlabel("RevenueGrowth")
axes[0, 1].set_ylabel("Count")

sns.histplot(train_df['InventoryRatio'], bins=30, kde=True, ax=axes[1, 0])
axes[1, 0].set_title("Inventory Ratio Distribution")
axes[1, 0].set_xlabel("InventoryRatio")
axes[1, 0].set_ylabel("Count")

sns.histplot(train_df['QuickRatio'], bins=30, kde=True, ax=axes[1, 1])
axes[1, 1].set_title("Quick Ratio Distribution")
axes[1, 1].set_xlabel("QuickRatio")
axes[1, 1].set_ylabel("Count")

sns.histplot(train_df['Marketshare'], bins=30, kde=True, ax=axes[2, 0])
axes[2, 0].set_title("Market Share Distribution")
axes[2, 0].set_xlabel("MarketShare")
axes[2, 0].set_ylabel("Count")

axes[2, 1].axis("off")  # Hide the empty subplot

plt.tight_layout()
plt.show()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 525 entries, 0 to 524
Data columns (total 11 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Quarter         525 non-null    object 
 1   Company         525 non-null    object 
 2   InventoryRatio  419 non-null    float64
 3   RevenueGrowth   525 non-null    float64
 4   Marketshare     525 non-null    float64
 5   Bond rating     525 non-null    object 
 6   Stock rating    525 non-null    object 
 7   Region          525 non-null    object 
 8   Industry        525 non-null    object 
 9   QuickRatio      525 non-null    float64
 10  Sales           525 non-null    int64  
dtypes: float64(4), int64(1), object(6)
memory usage: 45.2+ KB
Train Data Summary:
 None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 11 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   RowID           150 non-null    int64  
 1   Quarter         150 non-null    object 
 2   Company         150 non-null    object 
 3   InventoryRatio  119 non-null    float64
 4   RevenueGrowth   150 non-null    float64
 5   Marketshare     150 non-null    float64
 6   Bond rating     150 non-null    object 
 7   Stock rating    150 non-null    object 
 8   Region          150 non-null    object 
 9   Industry        150 non-null    object 
 10  QuickRatio      150 non-null    float64
dtypes: float64(4), int64(1), object(6)
memory usage: 13.0+ KB
Test Data Summary:
 None
Missing values in Train Data:
 Quarter             0
Company             0
InventoryRatio    106
RevenueGrowth       0
Marketshare         0
Bond rating         0
Stock rating        0
Region              0
Industry            0
QuickRatio          0
Sales               0
dtype: int64
Missing values in Test Data:
 RowID              0
Quarter            0
Company            0
InventoryRatio    31
RevenueGrowth      0
Marketshare        0
Bond rating        0
Stock rating       0
Region             0
Industry           0
QuickRatio         0
dtype: int64
Train Data Description:
        InventoryRatio  RevenueGrowth  Marketshare  QuickRatio         Sales
count      419.000000     525.000000   525.000000  525.000000    525.000000
mean         4.059451      -0.010400    -0.002914    1.622019   3517.121905
std          3.056567       0.068026     0.017389    0.581223   2034.930502
min          1.420000      -0.200000    -0.050000    0.500000    768.000000
25%          2.575000      -0.070000    -0.010000    1.050000   1976.000000
50%          3.470000       0.000000     0.000000    1.750000   3041.000000
75%          4.460000       0.050000     0.010000    2.120000   4489.000000
max         27.940000       0.080000     0.020000    2.490000  11686.000000
Test Data Description:
             RowID  InventoryRatio  RevenueGrowth  Marketshare  QuickRatio
count  150.000000      119.000000     150.000000   150.000000  150.000000
mean    75.500000        4.243109      -0.001267    -0.002667    1.602867
std     43.445368        3.698519       0.059712     0.016330    0.585409
min      1.000000        1.510000      -0.180000    -0.050000    0.520000
25%     38.250000        2.730000      -0.040000    -0.010000    1.075000
50%     75.500000        3.520000       0.010000     0.000000    1.745000
75%    112.750000        4.525000       0.040000     0.010000    2.090000
max    150.000000       26.160000       0.080000     0.020000    2.480000
```

```
C:\Users\ishra\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is dep
recated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\ishra\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is dep
recated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\ishra\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is dep
recated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\ishra\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is dep
recated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\ishra\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is dep
recated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



In [6]:

```python
# === 2. DATA PREPROCESSING ===

# Convert Quarter format
train_df["Quarter"] = train_df["Quarter"].str.replace("Q", "").astype(int)
test_df["Quarter"] = test_df["Quarter"].str.replace("Q", "").astype(int)
economic_df["Quarter"] = ((economic_df["Month"] - 1) // 3 + 1)

# Economic indicators rolling averages
economic_df = economic_df.sort_values("Month")
economic_df["ConsumerSentiment_RollingMean"] = economic_df["Consumer Sentiment"].rolling(window=3, min_periods=
economic_df["InterestRate_RollingMean"] = economic_df["Interest Rate"].rolling(window=3, min_periods=1).mean()
economic_df["PMI_RollingMean"] = economic_df["PMI"].rolling(window=3, min_periods=1).mean()
economic_df["MoneySupply_RollingMean"] = economic_df["Money Supply"].rolling(window=3, min_periods=1).mean()
quarterly_econ = economic_df.groupby("Quarter").mean(numeric_only=True).reset_index()

# Handle missing InventoryRatio
for df in [train_df, test_df]:
    df["InventoryRatio"] = df.groupby("Company")["InventoryRatio"].transform(lambda x: x.fillna(x.median()))
    df["InventoryRatio"].fillna(df["InventoryRatio"].median(), inplace=True)

# Merge economic data
train_df = train_df.merge(quarterly_econ, on="Quarter", how="left")
test_df = test_df.merge(quarterly_econ, on="Quarter", how="left")

# Fill remaining missing values
train_df.fillna(train_df.median(numeric_only=True), inplace=True)
test_df.fillna(test_df.median(numeric_only=True), inplace=True)

# Keep only Q1–Q7 for training
train_df = train_df[train_df["Quarter"] <= 7]
```

```
C:\Users\ishra\AppData\Local\Temp\ipykernel_4544\1913812346.py:19: FutureWarning: A value is trying to be set o
n a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on
which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)
' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.


  df["InventoryRatio"].fillna(df["InventoryRatio"].median(), inplace=True)
C:\Users\ishra\AppData\Local\Temp\ipykernel_4544\1913812346.py:19: FutureWarning: A value is trying to be set o
n a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on
which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)
' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.


  df["InventoryRatio"].fillna(df["InventoryRatio"].median(), inplace=True)
```

In [7]:
```
# === 3. FEATURE ENGINEERING ===

# Lag features
train_df = train_df.sort_values(by=["Company", "Quarter"])
train_df["Sales_Lag_1"] = train_df.groupby("Company")["Sales"].shift(1)
train_df["Sales_Lag_2"] = train_df.groupby("Company")["Sales"].shift(2)
train_df["Sales_Lag_3"] = train_df.groupby("Company")["Sales"].shift(3)
train_df["Average_Sales_Last_2"] = (train_df["Sales_Lag_1"] + train_df["Sales_Lag_2"]) / 2
train_df["Sales_Trend"] = train_df["Sales_Lag_1"] - train_df["Sales_Lag_2"]

for col in ["Sales_Lag_1", "Sales_Lag_2", "Sales_Lag_3", "Average_Sales_Last_2", "Sales_Trend"]:
    train_df[col].fillna(train_df[col].median(), inplace=True)

# Company-level stats
company_stats = train_df.groupby("Company")["Sales"].agg(["mean", "std"]).reset_index()
company_stats.rename(columns={"mean": "Company_Avg_Sales", "std": "Company_Sales_STD"}, inplace=True)
train_df = train_df.merge(company_stats, on="Company", how="left")
test_df = test_df.merge(company_stats, on="Company", how="left")
test_df.fillna(test_df.median(numeric_only=True), inplace=True)

# Categorical encoding
for col in ["Bond rating", "Stock rating", "Industry"]:
    le = LabelEncoder()
    train_df[col] = le.fit_transform(train_df[col])
    test_df[col] = le.transform(test_df[col])

# Region one-hot encoding
train_df = pd.get_dummies(train_df, columns=["Region"], drop_first=False)
test_df = pd.get_dummies(test_df, columns=["Region"], drop_first=False)
for col in set(train_df.columns).union(set(test_df.columns)):
    if "Region_" in col:
        train_df[col] = train_df.get(col, 0)
        test_df[col] = test_df.get(col, 0)

# Final training data
X_train = train_df.drop(columns=["Company", "Sales"])
y_train = train_df["Sales"]
```

```
C:\Users\ishra\AppData\Local\Temp\ipykernel_4544\940224722.py:12: FutureWarning: A value is trying to be set on
a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on
which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)
' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.


  train_df[col].fillna(train_df[col].median(), inplace=True)
```

In [8]:
```
# === 4. MODEL TRAINING WITH GRIDSEARCHCV ===

param_grid = {
    "n_estimators": [300, 400, 500],
    "learning_rate": [0.01, 0.02, 0.03],
    "max_depth": [5, 6, 7],
    "subsample": [0.8, 1.0],
    "colsample_bytree": [0.8, 1.0]
}

grid_search = GridSearchCV(
    XGBRegressor(random_state=42),
    param_grid=param_grid,
    scoring='neg_mean_absolute_error',
    cv=5,
    verbose=2,
    n_jobs=1
)
```

```python
grid_search.fit(X_train, y_train)
best_model = grid_search.best_estimator_

print("\n Best Hyperparameters Found:", grid_search.best_params_)
```

```
Fitting 5 folds for each of 108 candidates, totalling 540 fits
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=5, n_estimators=300, subsample=0.8; total time=
0.8s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=5, n_estimators=300, subsample=0.8; total time=
0.2s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=5, n_estimators=300, subsample=0.8; total time=
0.1s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=5, n_estimators=300, subsample=0.8; total time=
0.1s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=5, n_estimators=300, subsample=0.8; total time=
0.1s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=5, n_estimators=300, subsample=1.0; total time=
0.2s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=5, n_estimators=300, subsample=1.0; total time=
0.1s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=5, n_estimators=300, subsample=1.0; total time=
0.2s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=5, n_estimators=300, subsample=1.0; total time=
0.2s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=5, n_estimators=300, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=5, n_estimators=400, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=5, n_estimators=400, subsample=0.8; total time=
0.2s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=5, n_estimators=400, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=5, n_estimators=400, subsample=0.8; total time=
0.2s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=5, n_estimators=400, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=5, n_estimators=400, subsample=1.0; total time=
0.2s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=5, n_estimators=400, subsample=1.0; total time=
0.2s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=5, n_estimators=400, subsample=1.0; total time=
0.2s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=5, n_estimators=400, subsample=1.0; total time=
0.2s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=5, n_estimators=400, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=5, n_estimators=500, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=5, n_estimators=500, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=5, n_estimators=500, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=5, n_estimators=500, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=5, n_estimators=500, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=5, n_estimators=500, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=5, n_estimators=500, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=5, n_estimators=500, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=5, n_estimators=500, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=5, n_estimators=500, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=6, n_estimators=300, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=6, n_estimators=300, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=6, n_estimators=300, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=6, n_estimators=300, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=6, n_estimators=300, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=6, n_estimators=300, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=6, n_estimators=300, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=6, n_estimators=300, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=6, n_estimators=300, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=6, n_estimators=300, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=6, n_estimators=400, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=6, n_estimators=400, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=6, n_estimators=400, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=6, n_estimators=400, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=6, n_estimators=400, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=6, n_estimators=400, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=6, n_estimators=400, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=6, n_estimators=400, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=6, n_estimators=400, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=6, n_estimators=400, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=6, n_estimators=500, subsample=0.8; total time=
0.5s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=6, n_estimators=500, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=6, n_estimators=500, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=6, n_estimators=500, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=6, n_estimators=500, subsample=0.8; total time=
0.6s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=6, n_estimators=500, subsample=1.0; total time=
0.5s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=6, n_estimators=500, subsample=1.0; total time=
0.5s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=6, n_estimators=500, subsample=1.0; total time=
0.5s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=6, n_estimators=500, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=6, n_estimators=500, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=7, n_estimators=300, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=7, n_estimators=300, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=7, n_estimators=300, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=7, n_estimators=300, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=7, n_estimators=300, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=7, n_estimators=300, subsample=1.0; total time=
0.5s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=7, n_estimators=300, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=7, n_estimators=300, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=7, n_estimators=300, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=7, n_estimators=300, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=7, n_estimators=400, subsample=0.8; total time=
0.5s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=7, n_estimators=400, subsample=0.8; total time=
0.5s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=7, n_estimators=400, subsample=0.8; total time=
0.5s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=7, n_estimators=400, subsample=0.8; total time=
0.5s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=7, n_estimators=400, subsample=0.8; total time=
0.5s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=7, n_estimators=400, subsample=1.0; total time=
0.5s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=7, n_estimators=400, subsample=1.0; total time=
0.5s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=7, n_estimators=400, subsample=1.0; total time=
0.5s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=7, n_estimators=400, subsample=1.0; total time=
0.5s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=7, n_estimators=400, subsample=1.0; total time=
0.5s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=7, n_estimators=500, subsample=0.8; total time=
0.7s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=7, n_estimators=500, subsample=0.8; total time=
0.6s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=7, n_estimators=500, subsample=0.8; total time=
0.6s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=7, n_estimators=500, subsample=0.8; total time=
0.6s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=7, n_estimators=500, subsample=0.8; total time=
0.6s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=7, n_estimators=500, subsample=1.0; total time=
0.6s
```

```
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=7, n_estimators=500, subsample=1.0; total time=
0.6s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=7, n_estimators=500, subsample=1.0; total time=
0.7s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=7, n_estimators=500, subsample=1.0; total time=
0.6s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=7, n_estimators=500, subsample=1.0; total time=
0.6s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=5, n_estimators=300, subsample=0.8; total time=
0.2s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=5, n_estimators=300, subsample=0.8; total time=
0.1s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=5, n_estimators=300, subsample=0.8; total time=
0.2s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=5, n_estimators=300, subsample=0.8; total time=
0.2s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=5, n_estimators=300, subsample=0.8; total time=
0.2s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=5, n_estimators=300, subsample=1.0; total time=
0.2s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=5, n_estimators=300, subsample=1.0; total time=
0.1s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=5, n_estimators=300, subsample=1.0; total time=
0.2s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=5, n_estimators=300, subsample=1.0; total time=
0.2s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=5, n_estimators=300, subsample=1.0; total time=
0.2s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=5, n_estimators=400, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=5, n_estimators=400, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=5, n_estimators=400, subsample=0.8; total time=
0.2s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=5, n_estimators=400, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=5, n_estimators=400, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=5, n_estimators=400, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=5, n_estimators=400, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=5, n_estimators=400, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=5, n_estimators=400, subsample=1.0; total time=
0.2s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=5, n_estimators=400, subsample=1.0; total time=
0.2s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=5, n_estimators=500, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=5, n_estimators=500, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=5, n_estimators=500, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=5, n_estimators=500, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=5, n_estimators=500, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=5, n_estimators=500, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=5, n_estimators=500, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=5, n_estimators=500, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=5, n_estimators=500, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=5, n_estimators=500, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=6, n_estimators=300, subsample=0.8; total time=
0.2s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=6, n_estimators=300, subsample=0.8; total time=
0.2s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=6, n_estimators=300, subsample=0.8; total time=
0.2s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=6, n_estimators=300, subsample=0.8; total time=
0.2s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=6, n_estimators=300, subsample=0.8; total time=
0.2s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=6, n_estimators=300, subsample=1.0; total time=
0.2s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=6, n_estimators=300, subsample=1.0; total time=
0.2s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=6, n_estimators=300, subsample=1.0; total time=
0.2s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=6, n_estimators=300, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=6, n_estimators=300, subsample=1.0; total time=
0.2s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=6, n_estimators=400, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=6, n_estimators=400, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=6, n_estimators=400, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=6, n_estimators=400, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=6, n_estimators=400, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=6, n_estimators=400, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=6, n_estimators=400, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=6, n_estimators=400, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=6, n_estimators=400, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=6, n_estimators=400, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=6, n_estimators=500, subsample=0.8; total time=
0.5s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=6, n_estimators=500, subsample=0.8; total time=
0.5s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=6, n_estimators=500, subsample=0.8; total time=
0.5s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=6, n_estimators=500, subsample=0.8; total time=
0.5s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=6, n_estimators=500, subsample=0.8; total time=
0.5s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=6, n_estimators=500, subsample=1.0; total time=
0.5s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=6, n_estimators=500, subsample=1.0; total time=
0.5s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=6, n_estimators=500, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=6, n_estimators=500, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=6, n_estimators=500, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=7, n_estimators=300, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=7, n_estimators=300, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=7, n_estimators=300, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=7, n_estimators=300, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=7, n_estimators=300, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=7, n_estimators=300, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=7, n_estimators=300, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=7, n_estimators=300, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=7, n_estimators=300, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=7, n_estimators=300, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=7, n_estimators=400, subsample=0.8; total time=
0.6s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=7, n_estimators=400, subsample=0.8; total time=
0.5s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=7, n_estimators=400, subsample=0.8; total time=
0.5s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=7, n_estimators=400, subsample=0.8; total time=
0.5s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=7, n_estimators=400, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=7, n_estimators=400, subsample=1.0; total time=
0.5s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=7, n_estimators=400, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=7, n_estimators=400, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=7, n_estimators=400, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=7, n_estimators=400, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=7, n_estimators=500, subsample=0.8; total time=
0.6s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=7, n_estimators=500, subsample=0.8; total time=
0.6s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=7, n_estimators=500, subsample=0.8; total time=
0.7s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=7, n_estimators=500, subsample=0.8; total time=
0.7s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=7, n_estimators=500, subsample=0.8; total time=
0.7s
```

```
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=7, n_estimators=500, subsample=1.0; total time=
0.7s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=7, n_estimators=500, subsample=1.0; total time=
0.6s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=7, n_estimators=500, subsample=1.0; total time=
0.5s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=7, n_estimators=500, subsample=1.0; total time=
0.5s
[CV] END colsample_bytree=0.8, learning_rate=0.02, max_depth=7, n_estimators=500, subsample=1.0; total time=
0.6s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=5, n_estimators=300, subsample=0.8; total time=
0.2s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=5, n_estimators=300, subsample=0.8; total time=
0.2s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=5, n_estimators=300, subsample=0.8; total time=
0.1s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=5, n_estimators=300, subsample=0.8; total time=
0.1s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=5, n_estimators=300, subsample=0.8; total time=
0.1s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=5, n_estimators=300, subsample=1.0; total time=
0.2s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=5, n_estimators=300, subsample=1.0; total time=
0.1s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=5, n_estimators=300, subsample=1.0; total time=
0.1s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=5, n_estimators=300, subsample=1.0; total time=
0.1s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=5, n_estimators=300, subsample=1.0; total time=
0.2s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=5, n_estimators=400, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=5, n_estimators=400, subsample=0.8; total time=
0.2s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=5, n_estimators=400, subsample=0.8; total time=
0.2s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=5, n_estimators=400, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=5, n_estimators=400, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=5, n_estimators=400, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=5, n_estimators=400, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=5, n_estimators=400, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=5, n_estimators=400, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=5, n_estimators=400, subsample=1.0; total time=
0.2s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=5, n_estimators=500, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=5, n_estimators=500, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=5, n_estimators=500, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=5, n_estimators=500, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=5, n_estimators=500, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=5, n_estimators=500, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=5, n_estimators=500, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=5, n_estimators=500, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=5, n_estimators=500, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=5, n_estimators=500, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=6, n_estimators=300, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=6, n_estimators=300, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=6, n_estimators=300, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=6, n_estimators=300, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=6, n_estimators=300, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=6, n_estimators=300, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=6, n_estimators=300, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=6, n_estimators=300, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=6, n_estimators=300, subsample=1.0; total time=
0.2s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=6, n_estimators=300, subsample=1.0; total time=
0.2s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=6, n_estimators=400, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=6, n_estimators=400, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=6, n_estimators=400, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=6, n_estimators=400, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=6, n_estimators=400, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=6, n_estimators=400, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=6, n_estimators=400, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=6, n_estimators=400, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=6, n_estimators=400, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=6, n_estimators=400, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=6, n_estimators=500, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=6, n_estimators=500, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=6, n_estimators=500, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=6, n_estimators=500, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=6, n_estimators=500, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=6, n_estimators=500, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=6, n_estimators=500, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=6, n_estimators=500, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=6, n_estimators=500, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=6, n_estimators=500, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=7, n_estimators=300, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=7, n_estimators=300, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=7, n_estimators=300, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=7, n_estimators=300, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=7, n_estimators=300, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=7, n_estimators=300, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=7, n_estimators=300, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=7, n_estimators=300, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=7, n_estimators=300, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=7, n_estimators=300, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=7, n_estimators=400, subsample=0.8; total time=
0.6s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=7, n_estimators=400, subsample=0.8; total time=
0.6s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=7, n_estimators=400, subsample=0.8; total time=
0.5s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=7, n_estimators=400, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=7, n_estimators=400, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=7, n_estimators=400, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=7, n_estimators=400, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=7, n_estimators=400, subsample=1.0; total time=
0.5s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=7, n_estimators=400, subsample=1.0; total time=
0.5s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=7, n_estimators=400, subsample=1.0; total time=
0.5s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=7, n_estimators=500, subsample=0.8; total time=
0.7s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=7, n_estimators=500, subsample=0.8; total time=
0.7s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=7, n_estimators=500, subsample=0.8; total time=
0.6s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=7, n_estimators=500, subsample=0.8; total time=
0.5s
```

```
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=7, n_estimators=500, subsample=0.8; total time=
0.6s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=7, n_estimators=500, subsample=1.0; total time=
0.5s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=7, n_estimators=500, subsample=1.0; total time=
0.6s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=7, n_estimators=500, subsample=1.0; total time=
0.5s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=7, n_estimators=500, subsample=1.0; total time=
0.5s
[CV] END colsample_bytree=0.8, learning_rate=0.03, max_depth=7, n_estimators=500, subsample=1.0; total time=
0.6s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=5, n_estimators=300, subsample=0.8; total time=
0.2s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=5, n_estimators=300, subsample=0.8; total time=
0.2s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=5, n_estimators=300, subsample=0.8; total time=
0.2s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=5, n_estimators=300, subsample=0.8; total time=
0.2s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=5, n_estimators=300, subsample=0.8; total time=
0.2s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=5, n_estimators=300, subsample=1.0; total time=
0.2s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=5, n_estimators=300, subsample=1.0; total time=
0.2s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=5, n_estimators=300, subsample=1.0; total time=
0.2s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=5, n_estimators=300, subsample=1.0; total time=
0.2s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=5, n_estimators=300, subsample=1.0; total time=
0.2s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=5, n_estimators=400, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=5, n_estimators=400, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=5, n_estimators=400, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=5, n_estimators=400, subsample=0.8; total time=
0.2s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=5, n_estimators=400, subsample=0.8; total time=
0.2s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=5, n_estimators=400, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=5, n_estimators=400, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=5, n_estimators=400, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=5, n_estimators=400, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=5, n_estimators=400, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=5, n_estimators=500, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=5, n_estimators=500, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=5, n_estimators=500, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=5, n_estimators=500, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=5, n_estimators=500, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=5, n_estimators=500, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=5, n_estimators=500, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=5, n_estimators=500, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=5, n_estimators=500, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=5, n_estimators=500, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=6, n_estimators=300, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=6, n_estimators=300, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=6, n_estimators=300, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=6, n_estimators=300, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=6, n_estimators=300, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=6, n_estimators=300, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=6, n_estimators=300, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=6, n_estimators=300, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=6, n_estimators=300, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=6, n_estimators=300, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=6, n_estimators=400, subsample=0.8; total time=
0.5s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=6, n_estimators=400, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=6, n_estimators=400, subsample=0.8; total time=
0.5s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=6, n_estimators=400, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=6, n_estimators=400, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=6, n_estimators=400, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=6, n_estimators=400, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=6, n_estimators=400, subsample=1.0; total time=
0.5s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=6, n_estimators=400, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=6, n_estimators=400, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=6, n_estimators=500, subsample=0.8; total time=
0.5s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=6, n_estimators=500, subsample=0.8; total time=
0.5s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=6, n_estimators=500, subsample=0.8; total time=
0.5s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=6, n_estimators=500, subsample=0.8; total time=
0.5s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=6, n_estimators=500, subsample=0.8; total time=
0.5s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=6, n_estimators=500, subsample=1.0; total time=
0.5s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=6, n_estimators=500, subsample=1.0; total time=
0.5s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=6, n_estimators=500, subsample=1.0; total time=
0.5s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=6, n_estimators=500, subsample=1.0; total time=
0.5s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=6, n_estimators=500, subsample=1.0; total time=
0.5s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=7, n_estimators=300, subsample=0.8; total time=
0.5s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=7, n_estimators=300, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=7, n_estimators=300, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=7, n_estimators=300, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=7, n_estimators=300, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=7, n_estimators=300, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=7, n_estimators=300, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=7, n_estimators=300, subsample=1.0; total time=
0.5s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=7, n_estimators=300, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=7, n_estimators=300, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=7, n_estimators=400, subsample=0.8; total time=
0.7s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=7, n_estimators=400, subsample=0.8; total time=
0.6s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=7, n_estimators=400, subsample=0.8; total time=
0.7s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=7, n_estimators=400, subsample=0.8; total time=
0.7s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=7, n_estimators=400, subsample=0.8; total time=
0.6s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=7, n_estimators=400, subsample=1.0; total time=
0.7s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=7, n_estimators=400, subsample=1.0; total time=
0.5s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=7, n_estimators=400, subsample=1.0; total time=
0.5s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=7, n_estimators=400, subsample=1.0; total time=
0.5s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=7, n_estimators=400, subsample=1.0; total time=
0.6s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=7, n_estimators=500, subsample=0.8; total time=
0.7s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=7, n_estimators=500, subsample=0.8; total time=
0.7s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=7, n_estimators=500, subsample=0.8; total time=
0.7s
```

```
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=7, n_estimators=500, subsample=0.8; total time=
0.7s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=7, n_estimators=500, subsample=0.8; total time=
0.7s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=7, n_estimators=500, subsample=1.0; total time=
0.7s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=7, n_estimators=500, subsample=1.0; total time=
0.7s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=7, n_estimators=500, subsample=1.0; total time=
0.7s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=7, n_estimators=500, subsample=1.0; total time=
0.7s
[CV] END colsample_bytree=1.0, learning_rate=0.01, max_depth=7, n_estimators=500, subsample=1.0; total time=
0.7s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=5, n_estimators=300, subsample=0.8; total time=
0.2s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=5, n_estimators=300, subsample=0.8; total time=
0.2s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=5, n_estimators=300, subsample=0.8; total time=
0.2s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=5, n_estimators=300, subsample=0.8; total time=
0.2s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=5, n_estimators=300, subsample=0.8; total time=
0.2s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=5, n_estimators=300, subsample=1.0; total time=
0.2s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=5, n_estimators=300, subsample=1.0; total time=
0.2s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=5, n_estimators=300, subsample=1.0; total time=
0.2s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=5, n_estimators=300, subsample=1.0; total time=
0.2s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=5, n_estimators=300, subsample=1.0; total time=
0.2s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=5, n_estimators=400, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=5, n_estimators=400, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=5, n_estimators=400, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=5, n_estimators=400, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=5, n_estimators=400, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=5, n_estimators=400, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=5, n_estimators=400, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=5, n_estimators=400, subsample=1.0; total time=
0.2s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=5, n_estimators=400, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=5, n_estimators=400, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=5, n_estimators=500, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=5, n_estimators=500, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=5, n_estimators=500, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=5, n_estimators=500, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=5, n_estimators=500, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=5, n_estimators=500, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=5, n_estimators=500, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=5, n_estimators=500, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=5, n_estimators=500, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=5, n_estimators=500, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=6, n_estimators=300, subsample=0.8; total time=
0.2s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=6, n_estimators=300, subsample=0.8; total time=
0.2s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=6, n_estimators=300, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=6, n_estimators=300, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=6, n_estimators=300, subsample=0.8; total time=
0.2s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=6, n_estimators=300, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=6, n_estimators=300, subsample=1.0; total time=
0.2s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=6, n_estimators=300, subsample=1.0; total time=
0.2s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=6, n_estimators=300, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=6, n_estimators=300, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=6, n_estimators=400, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=6, n_estimators=400, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=6, n_estimators=400, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=6, n_estimators=400, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=6, n_estimators=400, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=6, n_estimators=400, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=6, n_estimators=400, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=6, n_estimators=400, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=6, n_estimators=400, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=6, n_estimators=400, subsample=1.0; total time=
0.5s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=6, n_estimators=500, subsample=0.8; total time=
0.5s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=6, n_estimators=500, subsample=0.8; total time=
0.5s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=6, n_estimators=500, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=6, n_estimators=500, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=6, n_estimators=500, subsample=0.8; total time=
0.5s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=6, n_estimators=500, subsample=1.0; total time=
0.5s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=6, n_estimators=500, subsample=1.0; total time=
0.5s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=6, n_estimators=500, subsample=1.0; total time=
0.5s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=6, n_estimators=500, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=6, n_estimators=500, subsample=1.0; total time=
0.5s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=7, n_estimators=300, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=7, n_estimators=300, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=7, n_estimators=300, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=7, n_estimators=300, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=7, n_estimators=300, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=7, n_estimators=300, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=7, n_estimators=300, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=7, n_estimators=300, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=7, n_estimators=300, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=7, n_estimators=300, subsample=1.0; total time=
0.5s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=7, n_estimators=400, subsample=0.8; total time=
0.6s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=7, n_estimators=400, subsample=0.8; total time=
0.5s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=7, n_estimators=400, subsample=0.8; total time=
0.5s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=7, n_estimators=400, subsample=0.8; total time=
0.5s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=7, n_estimators=400, subsample=0.8; total time=
0.5s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=7, n_estimators=400, subsample=1.0; total time=
0.5s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=7, n_estimators=400, subsample=1.0; total time=
0.6s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=7, n_estimators=400, subsample=1.0; total time=
0.6s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=7, n_estimators=400, subsample=1.0; total time=
0.6s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=7, n_estimators=400, subsample=1.0; total time=
0.6s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=7, n_estimators=500, subsample=0.8; total time=
0.6s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=7, n_estimators=500, subsample=0.8; total time=
0.7s
```

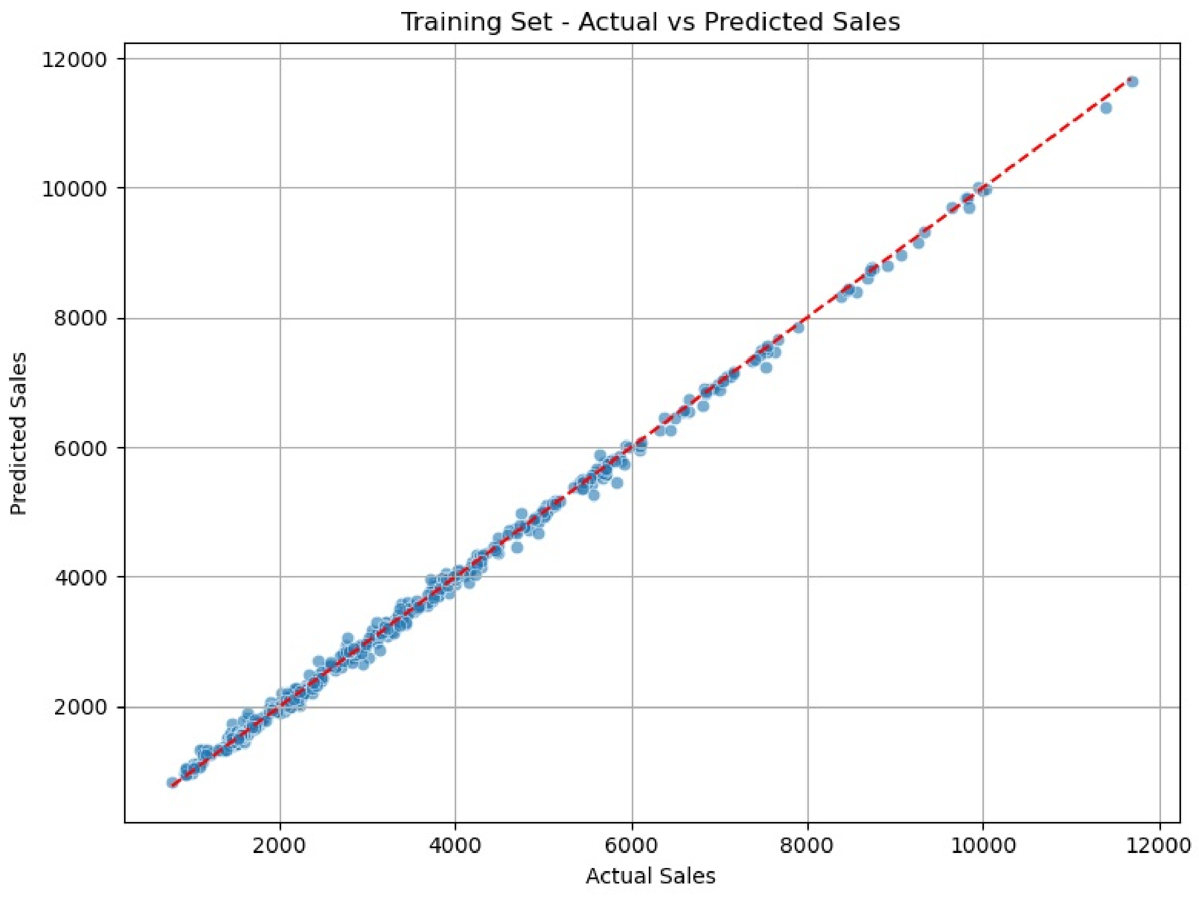
```
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=7, n_estimators=500, subsample=0.8; total time=
0.6s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=7, n_estimators=500, subsample=0.8; total time=
0.6s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=7, n_estimators=500, subsample=0.8; total time=
0.7s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=7, n_estimators=500, subsample=1.0; total time=
0.7s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=7, n_estimators=500, subsample=1.0; total time=
0.7s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=7, n_estimators=500, subsample=1.0; total time=
0.6s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=7, n_estimators=500, subsample=1.0; total time=
0.6s
[CV] END colsample_bytree=1.0, learning_rate=0.02, max_depth=7, n_estimators=500, subsample=1.0; total time=
0.7s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=5, n_estimators=300, subsample=0.8; total time=
0.2s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=5, n_estimators=300, subsample=0.8; total time=
0.2s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=5, n_estimators=300, subsample=0.8; total time=
0.2s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=5, n_estimators=300, subsample=0.8; total time=
0.1s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=5, n_estimators=300, subsample=0.8; total time=
0.2s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=5, n_estimators=300, subsample=1.0; total time=
0.2s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=5, n_estimators=300, subsample=1.0; total time=
0.2s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=5, n_estimators=300, subsample=1.0; total time=
0.2s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=5, n_estimators=300, subsample=1.0; total time=
0.2s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=5, n_estimators=300, subsample=1.0; total time=
0.2s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=5, n_estimators=400, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=5, n_estimators=400, subsample=0.8; total time=
0.2s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=5, n_estimators=400, subsample=0.8; total time=
0.2s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=5, n_estimators=400, subsample=0.8; total time=
0.2s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=5, n_estimators=400, subsample=0.8; total time=
0.2s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=5, n_estimators=400, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=5, n_estimators=400, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=5, n_estimators=400, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=5, n_estimators=400, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=5, n_estimators=400, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=5, n_estimators=500, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=5, n_estimators=500, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=5, n_estimators=500, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=5, n_estimators=500, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=5, n_estimators=500, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=5, n_estimators=500, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=5, n_estimators=500, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=5, n_estimators=500, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=5, n_estimators=500, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=5, n_estimators=500, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=6, n_estimators=300, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=6, n_estimators=300, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=6, n_estimators=300, subsample=0.8; total time=
0.2s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=6, n_estimators=300, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=6, n_estimators=300, subsample=0.8; total time=
0.2s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=6, n_estimators=300, subsample=1.0; total time=
0.2s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=6, n_estimators=300, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=6, n_estimators=300, subsample=1.0; total time=
0.2s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=6, n_estimators=300, subsample=1.0; total time=
0.2s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=6, n_estimators=300, subsample=1.0; total time=
0.2s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=6, n_estimators=400, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=6, n_estimators=400, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=6, n_estimators=400, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=6, n_estimators=400, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=6, n_estimators=400, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=6, n_estimators=400, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=6, n_estimators=400, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=6, n_estimators=400, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=6, n_estimators=400, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=6, n_estimators=400, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=6, n_estimators=500, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=6, n_estimators=500, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=6, n_estimators=500, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=6, n_estimators=500, subsample=0.8; total time=
0.5s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=6, n_estimators=500, subsample=0.8; total time=
0.5s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=6, n_estimators=500, subsample=1.0; total time=
0.5s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=6, n_estimators=500, subsample=1.0; total time=
0.5s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=6, n_estimators=500, subsample=1.0; total time=
0.5s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=6, n_estimators=500, subsample=1.0; total time=
0.5s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=6, n_estimators=500, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=7, n_estimators=300, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=7, n_estimators=300, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=7, n_estimators=300, subsample=0.8; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=7, n_estimators=300, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=7, n_estimators=300, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=7, n_estimators=300, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=7, n_estimators=300, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=7, n_estimators=300, subsample=1.0; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=7, n_estimators=300, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=7, n_estimators=300, subsample=1.0; total time=
0.3s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=7, n_estimators=400, subsample=0.8; total time=
0.5s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=7, n_estimators=400, subsample=0.8; total time=
0.4s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=7, n_estimators=400, subsample=0.8; total time=
0.5s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=7, n_estimators=400, subsample=0.8; total time=
0.5s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=7, n_estimators=400, subsample=0.8; total time=
0.5s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=7, n_estimators=400, subsample=1.0; total time=
0.5s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=7, n_estimators=400, subsample=1.0; total time=
0.5s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=7, n_estimators=400, subsample=1.0; total time=
0.5s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=7, n_estimators=400, subsample=1.0; total time=
0.5s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=7, n_estimators=400, subsample=1.0; total time=
0.5s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=7, n_estimators=500, subsample=0.8; total time=
0.6s
```

```
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=7, n_estimators=500, subsample=0.8; total time=
0.6s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=7, n_estimators=500, subsample=0.8; total time=
0.7s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=7, n_estimators=500, subsample=0.8; total time=
0.7s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=7, n_estimators=500, subsample=0.8; total time=
0.7s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=7, n_estimators=500, subsample=1.0; total time=
0.7s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=7, n_estimators=500, subsample=1.0; total time=
0.8s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=7, n_estimators=500, subsample=1.0; total time=
0.7s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=7, n_estimators=500, subsample=1.0; total time=
0.6s
[CV] END colsample_bytree=1.0, learning_rate=0.03, max_depth=7, n_estimators=500, subsample=1.0; total time=
0.7s

✅ Best Hyperparameters Found: {'colsample_bytree': 1.0, 'learning_rate': 0.03, 'max_depth': 5, 'n_estimators':
500, 'subsample': 0.8}
```

In [12]:

```python
# === 5. MODEL PERFORMANCE EVALUATION ===

X_train_part, X_val_part, y_train_part, y_val_part = train_test_split(X_train, y_train, test_size=0.2, random_s

y_val_pred = best_model.predict(X_val_part)
val_mae = mean_absolute_error(y_val_part, y_val_pred)
print(f"\n Validation MAE (20% split): {val_mae:.2f}")

y_train_pred = best_model.predict(X_train)
train_mae = mean_absolute_error(y_train, y_train_pred)
print(f" Training MAE (100%): {train_mae:.2f}")

# Plot: Actual vs Predicted (Training)
plt.figure(figsize=(8, 6))
sns.scatterplot(x=y_train, y=y_train_pred, alpha=0.6)
plt.plot([y_train.min(), y_train.max()], [y_train.min(), y_train.max()], 'r--')
plt.xlabel("Actual Sales")
plt.ylabel("Predicted Sales")
plt.title("Training Set - Actual vs Predicted Sales")
plt.grid(True)
plt.tight_layout()
plt.show()
```

```
 Validation MAE (20% split): 69.29
 Training MAE (100%): 60.85
```



In [13]:

```python
# === 6. Q8 & Q9 PREDICTION ===

# Predict Q8
test_q8 = test_df[test_df["Quarter"] == 8].copy()
```

```
test_q9 = test_df[test_df["Quarter"] == 9].copy()

q7 = train_df[train_df["Quarter"] == 7][["Company", "Sales"]].rename(columns={"Sales": "Sales_Lag_1"})
q6 = train_df[train_df["Quarter"] == 6][["Company", "Sales"]].rename(columns={"Sales": "Sales_Lag_2"})
q5 = train_df[train_df["Quarter"] == 5][["Company", "Sales"]].rename(columns={"Sales": "Sales_Lag_3"})

test_q8 = test_q8.merge(q7, on="Company", how="left")
test_q8 = test_q8.merge(q6, on="Company", how="left")
test_q8 = test_q8.merge(q5, on="Company", how="left")
test_q8["Average_Sales_Last_2"] = (test_q8["Sales_Lag_1"] + test_q8["Sales_Lag_2"]) / 2
test_q8["Sales_Trend"] = test_q8["Sales_Lag_1"] - test_q8["Sales_Lag_2"]

X_q8 = test_q8[X_train.columns]
test_q8["Sales"] = best_model.predict(X_q8)

# Predict Q9 using Q8's predicted Sales
q8_pred = test_q8[["Company", "Sales"]].rename(columns={"Sales": "Sales_Lag_1"})
q7_actual = train_df[train_df["Quarter"] == 7][["Company", "Sales"]].rename(columns={"Sales": "Sales_Lag_2"})
q6_actual = train_df[train_df["Quarter"] == 6][["Company", "Sales"]].rename(columns={"Sales": "Sales_Lag_3"})

q8q9 = test_q9.merge(q8_pred, on="Company", how="left")
q8q9 = q8q9.merge(q7_actual, on="Company", how="left")
q8q9 = q8q9.merge(q6_actual, on="Company", how="left")
q8q9["Average_Sales_Last_2"] = (q8q9["Sales_Lag_1"] + q8q9["Sales_Lag_2"]) / 2
q8q9["Sales_Trend"] = q8q9["Sales_Lag_1"] - q8q9["Sales_Lag_2"]

X_q9 = q8q9[X_train.columns]
q8q9["Sales"] = best_model.predict(X_q9)

# Combine Q8 and Q9 predictions
final_preds = pd.concat([test_q8, q8q9], axis=0).sort_values(by=["Company", "Quarter"]).reset_index(drop=True)
```

In [14]:

```
# === 7. SAVE SUBMISSION FILE ===
submission = sample_submission_df.copy()
submission["Sales"] = final_preds["Sales"].values
submission.to_csv("submission_q8_q9_gridsearch.csv", index=False)
print("\n Submission file saved: submission_q8_q9_gridsearch.csv")
```

```
 Submission file saved: submission_q8_q9_gridsearch.csv
```

In [ ]:

Loading [MathJax]/extensions/Safe.js