Project: Sales Forecasting and Demand Prediction

What is the problem we are trying to solve?

Many businesses struggle with accurately predicting how much of a product they will need in the future. Without reliable sales and demand forecasts, companies often face challenges like overstocking, running out of products, or missing opportunities to increase revenue. These issues not only affect day-to-day operations but can also lead to long-term financial losses.

This project focuses on developing a forecasting system that uses historical sales data to better anticipate future demand. The goal is to build a system that helps businesses make **informed, data-driven decisions** about:

- How much inventory to stock.
- When to launch promotions.
- Where to allocate resources across stores or regions.
- How to optimize sales strategies for revenue growth

Data Exploration:

We began by exploring the dataset, which included historical records of product sales, store locations, discount rates, and several other business-related variables. The initial analysis helped us understand the distribution of data, identify missing values, detect outliers, and uncover patterns such as seasonal trends or relationships between variables like discounts and sales volume. Interactive visualizations were also created to offer dynamic views of key metrics and patterns across time and different stores or product categories.

Feature Engineering:

To improve model performance, we performed feature engineering by creating new variables that capture hidden patterns in the data. These included time-based features like month, day of the week, and holiday indicators, as well as derived metrics like lagged sales values, rolling averages, and discount ratios. These additional features helped the model better understand the underlying trends and seasonality in customer behavior.

Model Development:

For the forecasting model, we experimented with several machine learning algorithms such as Linear Regression, Random Forest, and Gradient Boosting. Each model was evaluated based on accuracy metrics such as RMSE (Root Mean Squared Error) and MAE (Mean Absolute Error). After tuning hyperparameters and comparing performance, the best-performing model was selected to make future sales predictions. The results were integrated into a dashboard to allow end users to interact with the forecasts and visualize the effects of variables like discounts on sales.

- *Throughout this project, we set out to understand and predict sales and demand patterns in a way that could genuinely help businesses make smarter decisions. By carefully exploring the data, engineering meaningful features, and developing accurate forecasting models, we were able to build a system that does more than just make predictions, it tells a story about what drives customer behavior.*

  This work isn't just about numbers. It's about giving businesses the tools to avoid overstocking or running out of products, to prepare better for seasonal changes, and to respond more confidently to market shifts. While there's always room to grow, like integrating live data or automating updates, we believe this project is a strong step toward smarter, data-backed planning.

About Dataset

The dataset used is the "Global Super Store Dataset" available on Kaggle. This dataset contains historical sales data for a global superstore, providing insights into consumer behavior and product performance across different regions. It is well-suited for time-series analysis and forecasting due to its detailed transactional data. The Global Super Store Dataset contains approximately 51,291 entries and 24 columns.

Column Descriptions:

1. Row ID: Row identifier.
2. Order ID: Order identifier.
3. Order Date: Date of order.
4. Ship Date: Date of shipment.
5. Ship Mode: Shipping method.
6. Customer ID: Customer identifier.
7. Customer Name: Customer's name.
8. Segment: Customer type.
9. City: City of order.
10. State: State of order.
11. Country: Country of order.
12. Postal Code: Postal code.
13. Market: Market region.
14. Region: Geographical region.
15. Product ID: Product identifier.
16. Category: Product category.
17. Sub-Category: Product subcategory.
18. Product Name: Product name.
19. Sales: Sales amount.
20. Quantity: Number of items.
21. Discount: Discount applied.
22. Profit: Profit earned.
23. Shipping Cost: Shipping cost.
24. Order Priority: Order priority level.

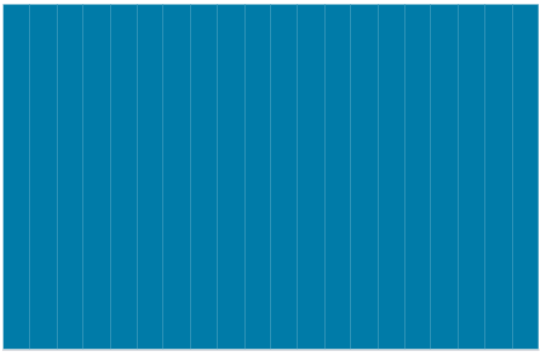
Dataset Characteristics:

The dataset is diverse, containing both categorical and numerical data. It includes temporal information with "Order Date" and "Ship Date" in datetime format. Some columns like "Cost Price," "Retail Price," and others related to monetary values are currently stored as objects, which may need conversion for accurate numerical analysis. The dataset provides a comprehensive snapshot of the sales process, making it suitable for various analytical and exploratory tasks.

A sample of the Data:

40155	CA-2014-1	14-10-201	21-10-201	Standard (	JW-15220	Jane Wacc	Corporate	Sacramen	California
40936	CA-2012-1	28-01-201	31-01-201	Second Cl	JH-15985	Joseph Ho	Consumer	Concord	North Car
34577	CA-2011-1	5/4/2011	9/4/2011	Second Cl	GM-14695	Greg Max	Corporate	Alexandri	Virginia
28879	ID-2012-2	19-04-201	22-04-201	First Class	AJ-10780	Anthony J	Corporate	Kabul	Kabul
45794	SA-2011-1	27-12-201	29-12-201	Second Cl	MM-7260	Magdelen	Consumer	Jizan	Jizan
4132	MX-2012-	13-11-201	13-11-201	Same Day	VF-21715	Vicky Frey	Home Offi	Toledo	Parana
27704	IN-2013-7	6/6/2013	8/6/2013	Second Cl	PF-19120	Peter Fulle	Consumer	Mudanjiar	Heilongjia
13779	ES-2014-5	31-07-201	3/8/2014	Second Cl	BP-11185	Ben Peter	Corporate	Paris	Ile-de-Fra
36178	CA-2014-1	########	########	Second Cl	TB-21175	Thomas B	Corporate	Henderso	Kentucky
12069	ES-2014-1	8/9/2014	14-09-201	Standard (	PJ-18835	Patrick Jor	Corporate	Prato	Tuscany
22096	IN-2014-1	31-01-201	1/2/2014	First Class	JS-15685	Jim Sink	Corporate	Townsville	Queenslar
49463	TZ-2014-8	########	########	Second Cl	RH-9555	Ritsa High	Consumer	Uvinza	Kigoma
46630	PL-2012-7	8/8/2012	########	First Class	AB-600	Ann Blum	Corporate	Bytom	Silesia

Row ID



Valid	51.3k	100%
Mismatched	0	0%
Missing	0	0%
Mean	25.6k	
Std. Deviation	14.8k	
Quantiles	1	Min
	12.8k	25%
	25.6k	50%
	38.5k	75%
	51.3k	Max

Order ID

25035
unique values

Valid	51.3k	100%
Mismatched	0	0%
Missing	0	0%
Unique	25.0k	
Most Common	CA-2014-1...	0%

Order Date

1430
unique values

Valid	51.3k	100%
Mismatched	0	0%
Missing	0	0%
Unique	1430	
Most Common	18-06-2014	0%

A more detailed information about some of the columns in the data (Missing values, unique values and most common)

Data Collection, Exploration, and Preprocessing:

Objective:
The goal of this phase was to prepare a clean and structured dataset that is ready for building forecasting models.

What We Did:

1. Data Collection:

- Loaded a dataset containing historical sales data for multiple stores and items.

```
In [58]:
sales = pd.read_csv('../Data/Original Data.csv', encoding='latin-1')
sales.shape
```

```
Out[58]: (51290, 24)
```

> Discribe Dataset

- The data included columns 24 columns.

```
In [59]:
sales.describe()
```

Out[59]:

	Row ID	Postal Code	Sales	Quantity	Discount	Profit	Shipping Cost
count	51290.00000	9994.000000	51290.000000	51290.000000	51290.000000	51290.000000	51290.000000
mean	25645.50000	55190.379428	246.490581	3.476545	0.142908	28.610982	26.375915
std	14806.29199	32063.693350	487.565361	2.278766	0.212280	174.340972	57.296804
min	1.00000	1040.000000	0.444000	1.000000	0.000000	-6599.978000	0.000000
25%	12823.25000	23223.000000	30.758625	2.000000	0.000000	0.000000	2.610000
50%	25645.50000	56430.500000	85.053000	3.000000	0.000000	9.240000	7.790000
75%	38467.75000	90008.000000	251.053200	5.000000	0.200000	36.810000	24.450000
max	51290.00000	99301.000000	22638.480000	14.000000	0.850000	8399.976000	933.570000

- Then we dropped useless columns like ‘Row ID', 'Order ID', 'Customer ID', 'Postal Code’, ‘Product ID'
- Then added some features like the holidays using `USFederalHolidayCalendar`

2. Data Exploration:

Investigate relationships between product types, promotional activities, and sales volume. In addition to understanding sales trends, seasonality, and external factors influencing demand.

- Reviewed the structure and content of the dataset.

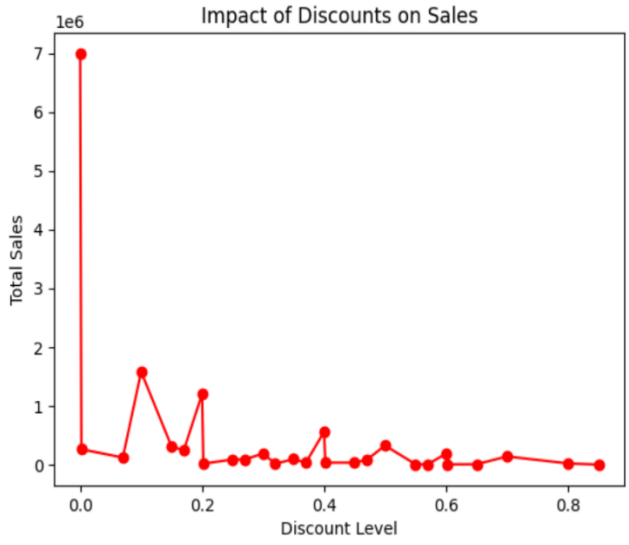
  Like getting the highest selling category

```
In [68]:
category_sales = sales.groupby('Category')['Sales'].sum().round(2).sort_values(ascending=False)
category_sales = pd.DataFrame(category_sales).reset_index()
category_sales.index += 1
category_sales
```

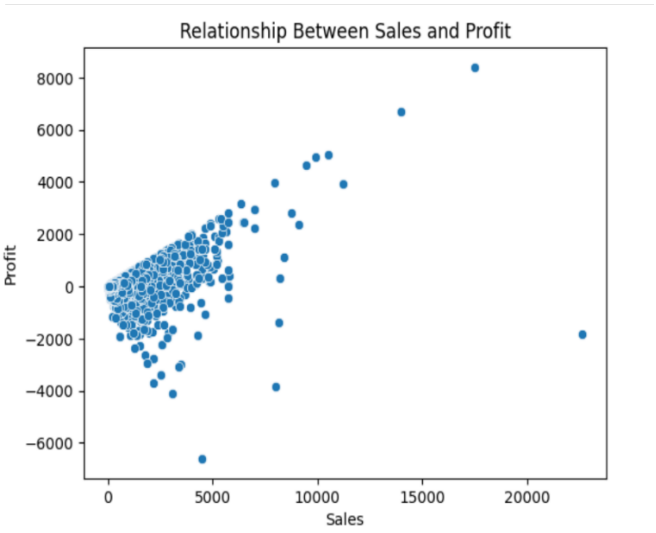
Out[68]:

	Category	Sales
1	Technology	4744557.50
2	Furniture	4110874.19
3	Office Supplies	3787070.23

The highest selling category is Technology



total sales initially drop sharply with a small increase in discount level but then fluctuate and gradually decline



as sales increase, profit generally goes up too, but there is some scatter, and sometimes higher sales can still lead to a loss.

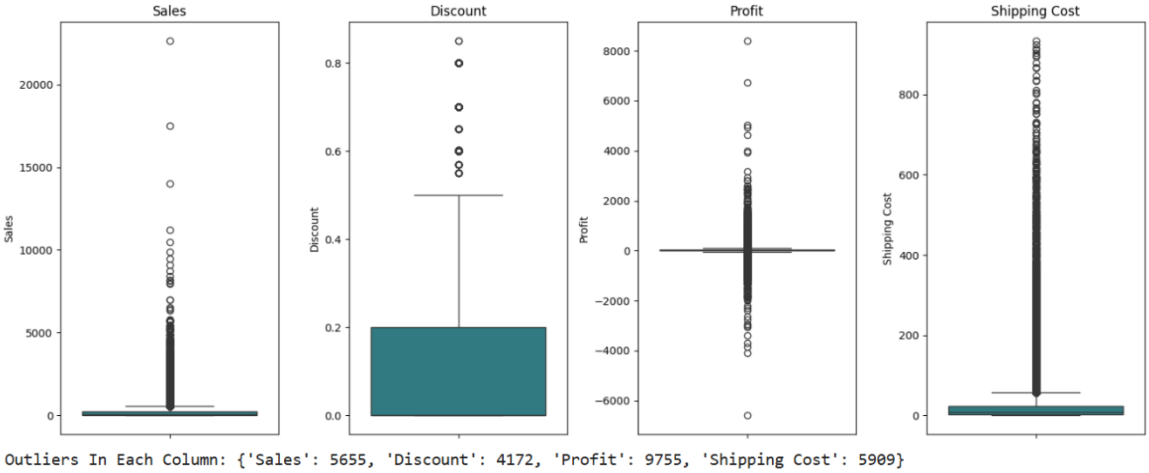
- Then we started Handling missing values, duplicates, and outliers, and compute basic summary statistics.

  1-The data did not have any missing values or any duplicates.

```
missing_values = pd.DataFrame(sales.isna().sum())
missing_values
```

  2-Identifying the outliers

  By Plotting the Outliers in each numerical column and calculating the IQR




```
Outliers In Each Column: {'Sales': 5655, 'Discount': 4172, 'Profit': 9755, 'Shipping Cost': 5909}
```

The sales and shipping costs have some extreme outliers, discounts are mostly consistent with some high values, and profits vary widely with some significant negative outliers.

- Visualized general sales trends over time to understand seasonality and growth patterns.

3. Data Preprocessing:

- Converted the `date` column to a proper datetime format.

```
# Convert Order Date to DateTime
sales['Order Date'] = pd.to_datetime(sales['Order Date'])

# Convert Ship Date to Date time
sales['Ship Date'] = pd.to_datetime(sales['Ship Date'])
```

- Added new time-related features like `year`, `month`, and `day` to support better analysis.
- Verified data consistency and ensured there were no missing values or extreme outliers that could affect modeling.

4. Feature Engineering:

- Created time-based features such as `month`, `day`, `week and adding the seasons` to help capture patterns in the sales.

```python
# Identify the Seasons
def get_season(month):
    if month in [12, 1, 2]:
        return 'Winter'
    elif month in [3, 4, 5]:
        return 'Spring'
    elif month in [6, 7, 8]:
        return 'Summer'
    else:
        return 'Fall'

sales['Season'] = sales['Order Date'].dt.month.apply(get_season)

# Holiday Flags
holidays = ['12-25', '01-01']  # Christmas and New Year
sales['Holiday'] = sales['Order Date'].dt.strftime('%m-%d').isin(holidays).astype(int)

# Extract Months
sales['Month'] = sales['Order Date'].dt.month

# Extract Weeks
sales['Week'] = sales['Order Date'].dt.isocalendar().week

# Extract Day of The Month
sales['Day_of_Month'] = sales['Order Date'].dt.day

# Extract Day of The Week
sales['Day_of_Week'] = sales['Order Date'].dt.dayofweek  # 0 = Monday, 6 = Sunday
```

Adding the seasons based on the month

Setting the day of the Month and the day of the week for the data

- Encode categorical variables, normalize numerical features, and create lag features (e.g., sales from the previous month).

```python
encoder = LabelEncoder()

# Columns that Label Encoder will Applied to
categorical_columns = ["Order Priority", "Ship Mode", "Segment", "Market", "Category", "Sub-Category", "Region", "Season"]

# Appling Label Encoder
for col in categorical_columns:
    sales[col + "_Encoded"] = encoder.fit_transform(sales[col])

# Show Modified Columns
sales[[*categorical_columns, *[col + "_Encoded" for col in categorical_columns]]].head()
```

	Order Priority	Ship Mode	Segment	Market	Category	Sub-Category	Region	Season	Order Priority_Encoded	Ship Mode_Encoded	Segment_Encodec
0	Critical	Same Day	Consumer	US	Technology	Accessories	East	Summer	0	1	
1	Critical	Second Class	Corporate	APAC	Furniture	Chairs	Oceania	Winter	0	2	
2	Medium	First Class	Consumer	APAC	Technology	Phones	Oceania	Fall	3	0	
3	Medium	First Class	Home Office	EU	Technology	Phones	Central	Winter	3	0	

- Encode categorical variables, normalize numerical features, and create lag features (e.g., sales from the previous month).

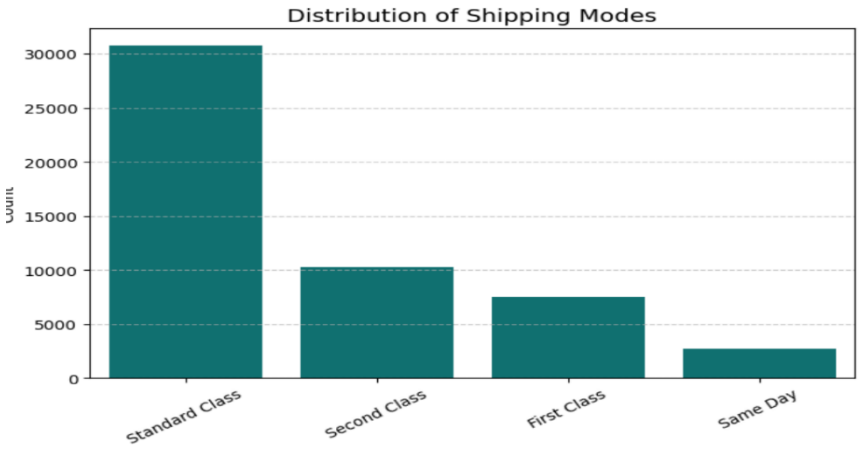
In [183…

```python
sales = sales.sort_index()

# Create lag features
sales['Sales_Lag_1D'] = sales['Sales'].shift(1)   # 1 day before
sales['Sales_Lag_7D'] = sales['Sales'].shift(7)   # 1 week before
sales['Sales_Lag_30D'] = sales['Sales'].shift(30) # 1 month before

sales.head()
```

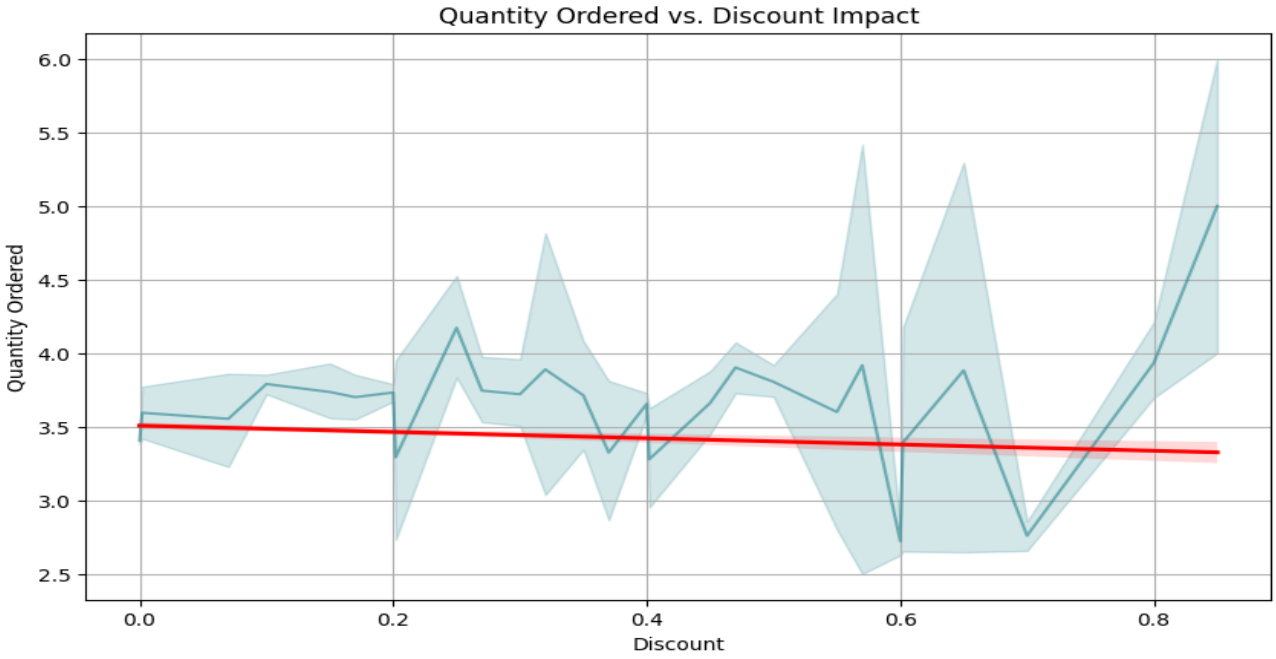
Deliverables Prepared:

- **EDA Report** (within the notebook): Includes visualizations and observations about trends and seasonality.
- **Interactive Visualizations**: Simple plots were used to display sales distribution and time trends.
- **Cleaned Dataset**: The data is now well-structured and ready for use in building predictive models.

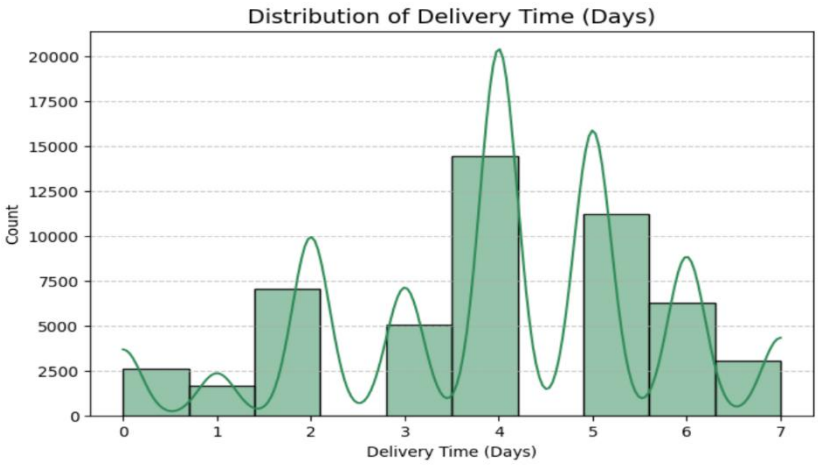
➢ A sample of the EDA



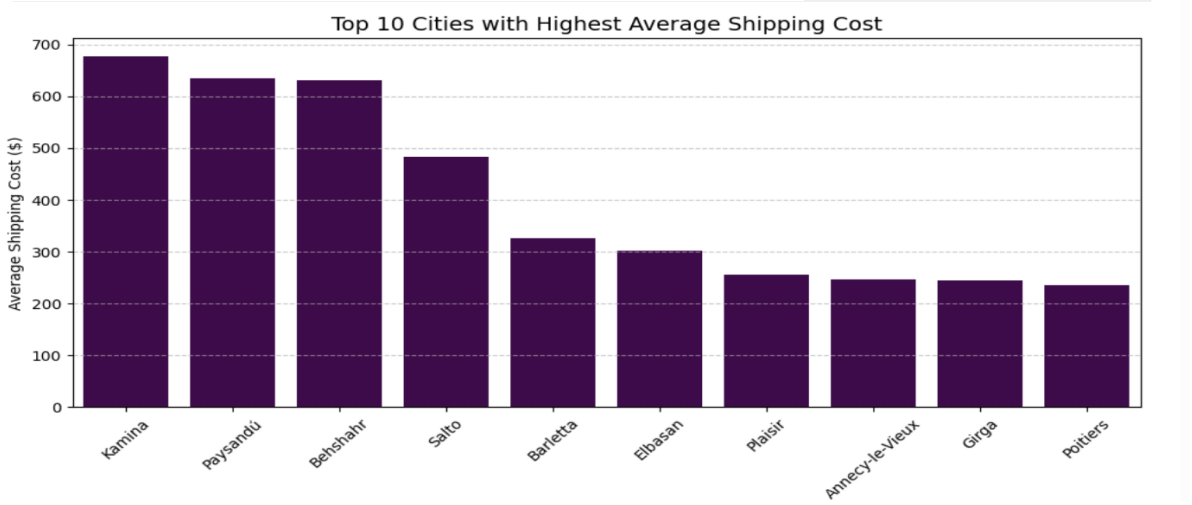
The most frequently used shipping method is the standard class which indicate customer preferences and cost efficiency.



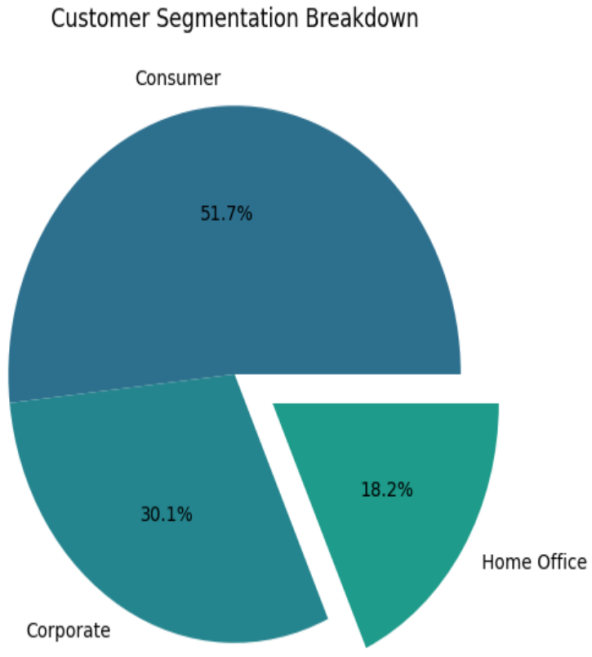
The plot suggests that higher discounts do not strongly correlate with an increase in quantity ordered. Indicating that larger discounts might not necessarily lead to higher sales volumes on average.



This shows whether shipping times are consistent or if delays frequently occur. It shows that most orders arrive after 4 days.



The highest average shipping cost for orders is in Kamina.



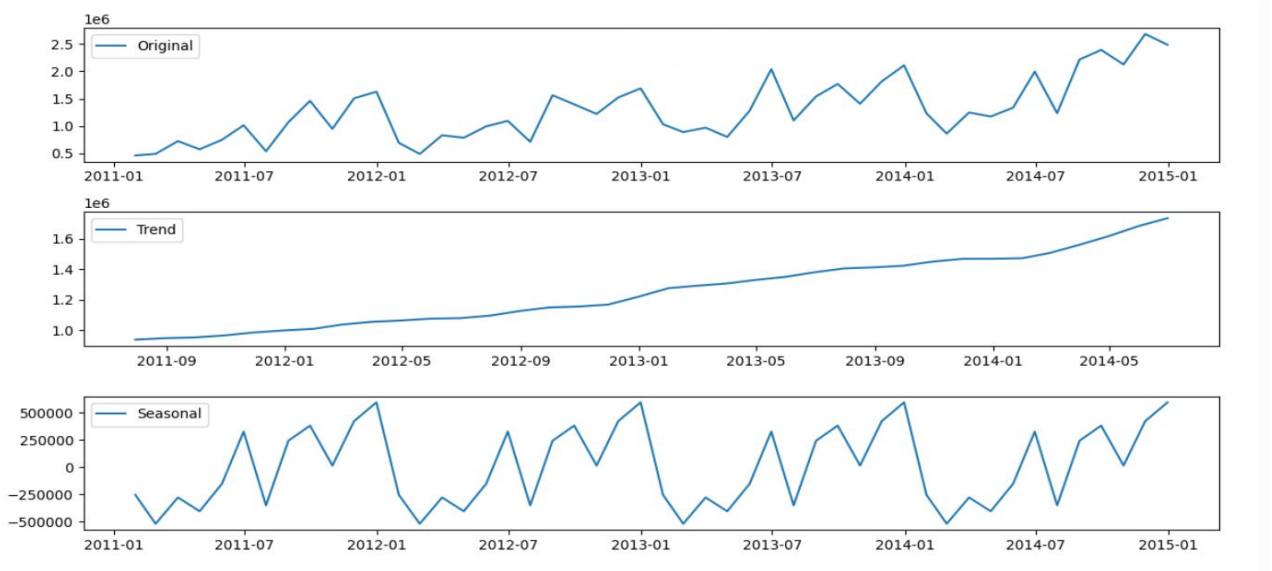
The majority of the sales is due to individual costumers, then the corporate companies, the smallest segment Is due to the home office.

➢ Advanced Data Analysis and Feature Engineering

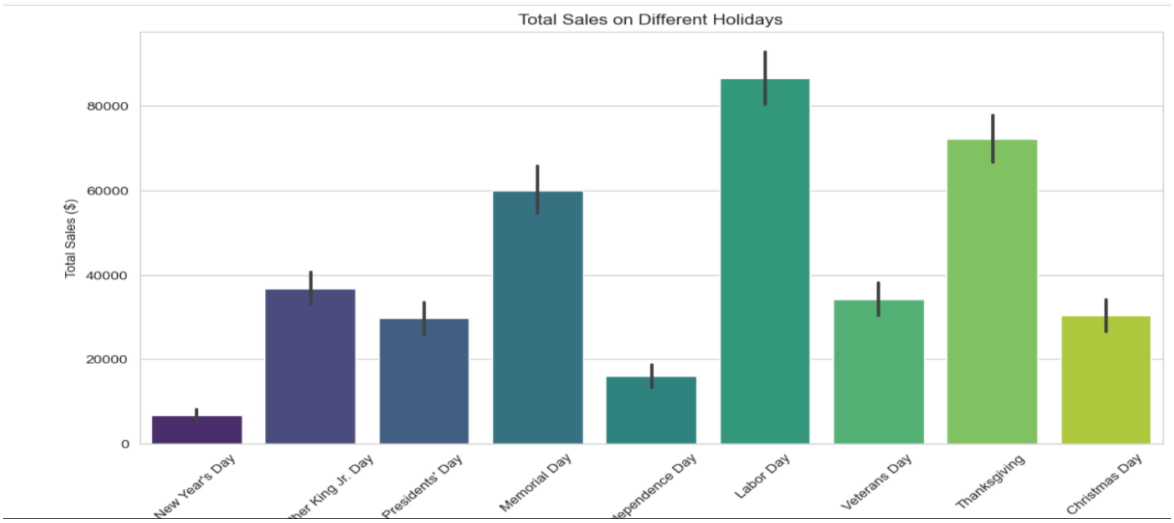
Objectives:
Perform deeper analysis and enhance feature selection to improve the forecasting model's accuracy.

1- Advanced Data Analysis

- We analyzed **sales over time** to identify patterns such as **seasonality and trends**. This helps in understanding whether sales go up or down during certain times of the year.



- We used **correlation analysis** to check how features like promotions or holidays are related to sales. This revealed which factors might be driving demand.



This shows that the highest holiday is sales is Labor Day followed by Thanksgiving

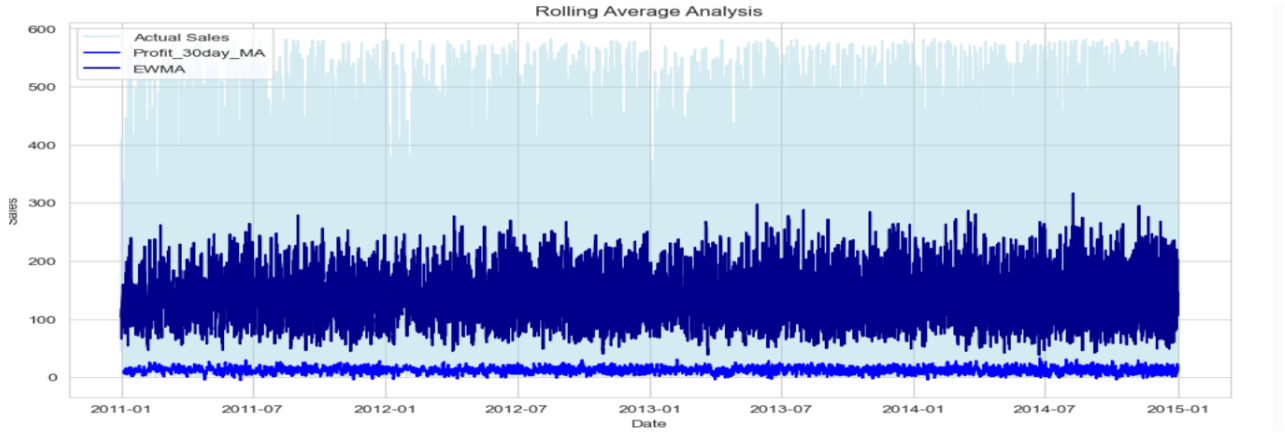
- Use statistical tests (e.g., **ADF** test for stationarity) to ensure data suitability for time series modeling.

```python
def adf_test(series):
    result = adfuller(series, autolag='AIC')
    print("ADF Test Results:")
    print(f"ADF Statistic: {result[0]}")
    print(f"p-value: {result[1]}")
    print("Critical Values:")
    for key, value in result[4].items():
        print(f"   {key}: {value}")

    if result[1] <= 0.05:
        print("\n The data is stationary (reject null hypothesis).")
    else:
        print("\n The data is non-stationary (fail to reject null hypothesis).")

adf_test(sales['Sales'])
```

```python
# Understanding the ADF Test Results
# ADF Statistic: -158.29
# This is the test statistic calculated by the ADF test.
# The more negative this value is, the stronger the evidence that the data is stationary.
#-----------------------------------------------------------------------------------------------------
# p-value: 0.0
# The p-value tells us whether to reject or fail to reject the null hypothesis:
# If p-value ≤ 0.05 → Reject the null hypothesis → Data is stationary .
# If p-value > 0.05 → Fail to reject the null hypothesis → Data is non-stationary .
# Since p-value is 0.0, we strongly reject the null hypothesis, confirming that the data is already stationary.
#-----------------------------------------------------------------------------------------------------
# Critical Values
# These values help compare the ADF Statistic:
# 1% Level: -3.430
# 5% Level: -2.862
# 10% Level: -2.567
# If the ADF Statistic (-158.29) is lower (more negative) than the critical values, it confirms stationarity.
# In our case, -158.29 is much lower than all critical values, so we confirm stationarity.
# What Does This Mean for our Forecasting?
# Because our data is already stationary, I do NOT need to transform it like differencing or log transformation.
```

2-Feature Engineering

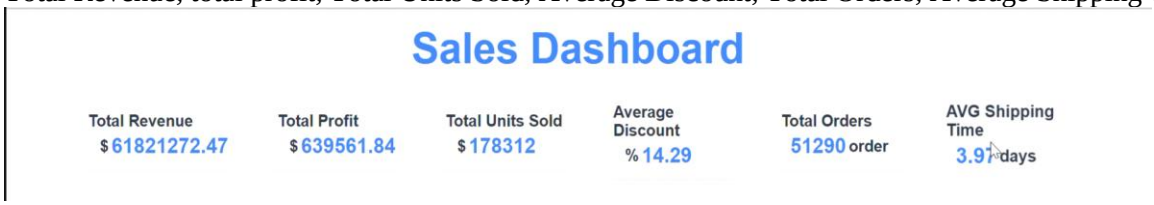
- Added **rolling averages** to smooth out short-term fluctuations and highlight longer-term trends.



The 30-day moving average (MA) and exponentially weighted moving average (EWMA) smooth out these changes,

3-Sales Dashboard

- **KPI Summary Cards (Always Visible)**
  Metrics:
  Total Revenue, total profit, Total Units Sold, Average Discount, Total Orders, Average Shipping Time



- **Sales & Profit Over Time**
  Chart Type: Line Chart

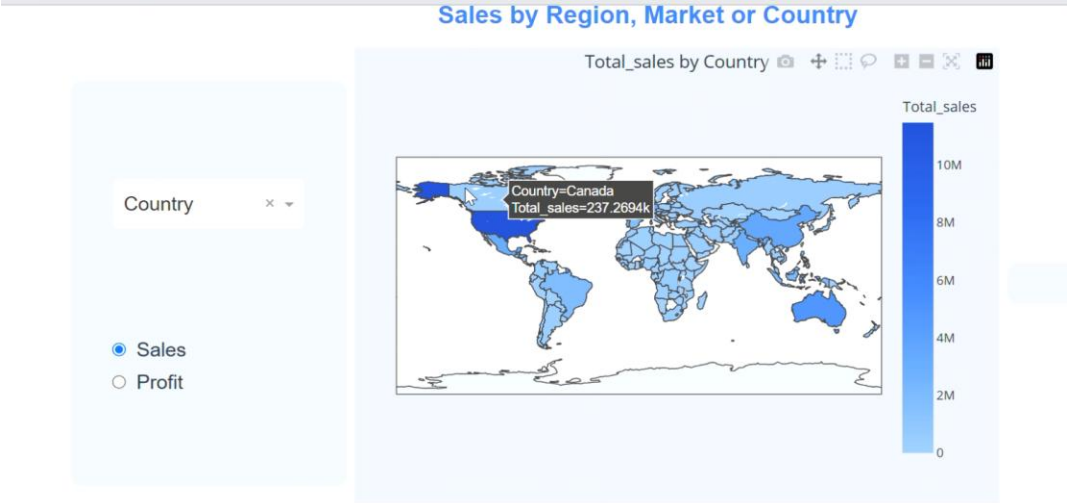
  Interactivity: dcc.DatePickerRange to filter by date, dcc.Slider to change moving average window (e.g., 7, 14, 30 days), dcc.Dropdown to select measure: Sales or Profit

  Goal: Analyze how sales and profit evolve over time and detect trends or seasonality.

- **Sales by region, Market and Country**
  Chart Type: Bar Chart by region/market/country

  Interactivity: dcc.Dropdown to choose Region, Market, map to choose between countries, dcc.RadioItems to toggle between Sales and Profit

  Goal: Identify geographic strengths or weaknesses.



- **Top preforming products and categories**
  Chart Type: Horizontal Bar Chart

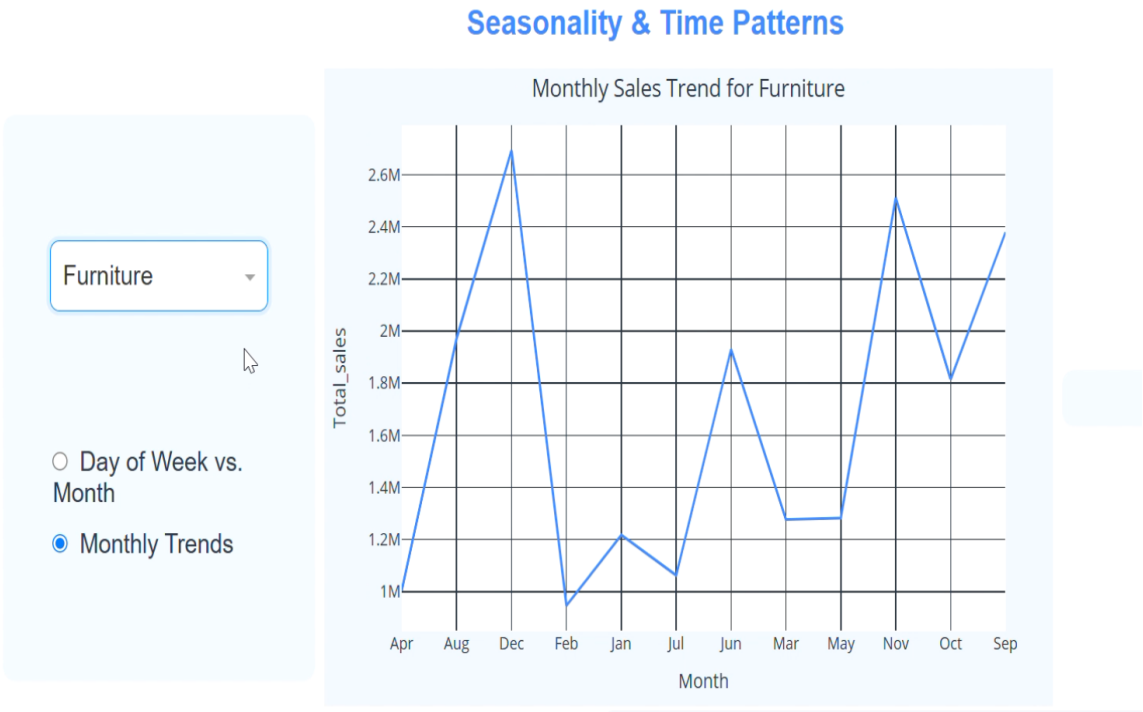
  Interactivity: dcc.Dropdown to select metric: Sales, Profit, or Quantity, dcc.Checklist to filter by Category or Sub-Category

  Goal: Discover the best and worst selling products to inform inventory decisions.

- **Seasonality & Time Patterns**
  Chart Type: Heatmap

  Interactivity: dcc.RadioItems to view by Day, Month, or Season, dcc.Dropdown to filter by Category

  Goal: Identify when sales spike or slow down to adjust strategy accordingly.



- **Discount Impact on Profit**
  Chart Type: Scatter Plot (Discount vs Profit)

  Interactivity: dcc.RangeSlider to filter discount levels, dcc.Dropdown to choose Category/Sub-Category

  Goal: Evaluate how discounts affect profitability and whether they are effective.

➢ Machine Learning Model Development and Optimization

Objectives: Build, train, and optimize forecasting models to predict future sales and demand.

1- Data Preprocessing

To prepare the dataset for modeling, we began with feature selection. We selected the following relevant columns based on correlation analysis and domain understanding:

['Quantity', 'Discount', 'Profit', 'Shipping Cost', 'Month', 'Year', 'Day_of_Week', 'Is_Weekend', 'Total_sales', 'Day_of_Month', 'Order Priority_Encoded', 'Ship Mode_Encoded', 'Market_Encoded', 'Category_Encoded', 'Sub-Category_Encoded']

2-Encoding

For categorical variables, we used Label Encoding. This method converts each category into a unique integer. It is computationally efficient and works well for tree-based models which are not sensitive to the ordinal nature of encoded labels.

Example Code:

```
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
categorical_columns = ["Order Priority", "Ship Mode", "Segment", "Market", "Category", "Sub-Category", "Region"]
for col in categorical_columns:
    sales[col + '_Encoded'] = encoder.fit_transform(sales[col])
```

- We did not apply scaling, as our selected models (particularly tree-based models) are not affected by feature scaling.

3- Model Evaluation

We trained and evaluated multiple models to predict the target variable. The evaluation metrics included RMSE, MAE, and R^2 Score for both train and test sets.

Models and Results Comparison:

Model	Train RMSE	Test RMSE	Train MAE	Test MAE	Train R^2	Test R^2
Linear Regression	289.29	291.76	195.20	194.63	0.6957	0.6862
K-Nearest Neighbors (KNN)	216.96	266.70	125.78	154.50	0.8288	0.7378
Random Forest	64.01	175.18	34.49	93.75	0.9851	0.8869
Extra Trees	0.00	177.89	0.00	96.72	1.0000	0.8834
Gradient Boosting	184.94	191.20	107.85	109.77	0.8756	0.8653
XGBoost	122.52	**171.52**	70.93	**95.26**	0.9454	**0.8916**

4- Final Model Selection: XGBoost

We selected XGBoost as the final model because it achieved the best overall performance on the test set, balancing high accuracy with reasonable generalization.

Initial XGBoost Results:
Train R^2: 0.9440 | Test R^2: 0.9003
Train RMSE: 124.10 | Test RMSE: 164.44

While the model showed a slight gap between train and test R^2, indicating possible overfitting, it still outperformed the other models on the test set.

5- Hyperparameter Tuning with MLflow

To address the overfitting and optimize the model, we performed hyperparameter tuning using MLflow. The updated model showed improved generalization:

Tuned XGBoost (via MLflow):
Train R^2: 0.9103 | Test R^2: 0.8916
Train RMSE: 157.04 | Test RMSE: 171.51

- **Next Step: Deployment**

After finalizing the model with optimized parameters, we moved on to the deployment phase to make the model accessible for real-world usage.

Conclusion:

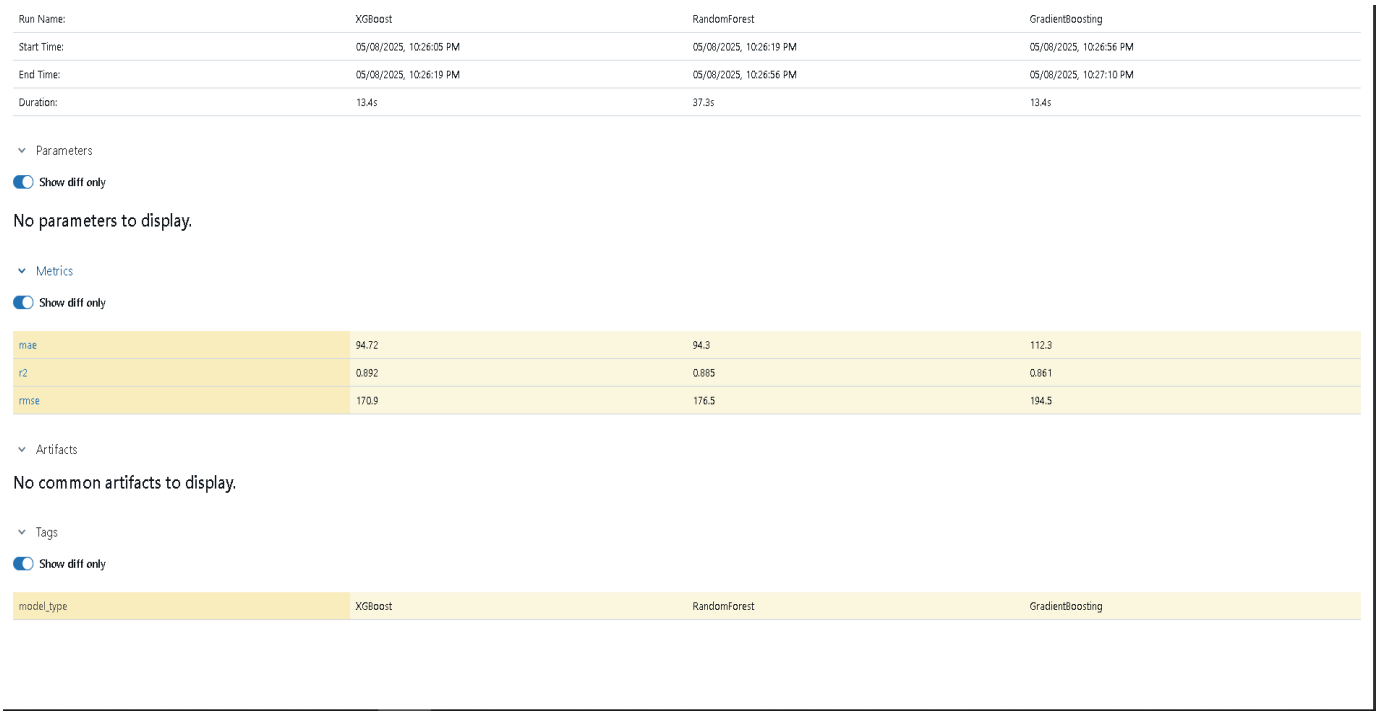
The final model used for sales forecasting was saved as a serialized.pkl file with Python's pickle library. The trained XGBoost Regressor, which has already discovered patterns from the historical sales data, such as seasonal trends, promotional effects, and other influencing factors, is contained in this file, xgboost_sales_model.pkl.
The model can be quickly reused and deployed without retraining if it is saved in this format. If the input features are preprocessed similarly to how they were during training, it can be imported straight into any Python environment to generate predictions on fresh data.
With this method, real-time demand forecasting applications like dashboards or web services can be seamlessly integrated.

MLOps, Deployment, and Monitoring

Objectives: Implement MLOps practices and deploy the forecasting model for real-time or batch predictions.

1-MLOps implementation

Multiple runs of the XGBoost model were conducted, along with RandomForest and GradientBoosting models, to compare their performance. MLflow was used to log each run, capturing details like run ID, start time, end time, and duration.



Run Name:	XGBoost	RandomForest	GradientBoosting
Start Time:	05/08/2025, 10:26:05 PM	05/08/2025, 10:26:19 PM	05/08/2025, 10:26:56 PM
End Time:	05/08/2025, 10:26:19 PM	05/08/2025, 10:26:56 PM	05/08/2025, 10:27:10 PM
Duration:	13.4s	37.3s	13.4s

Parameters

Show diff only

No parameters to display.

Metrics

Show diff only

mae	94.72	94.3	112.3
r2	0.892	0.885	0.861
rmse	170.9	176.5	194.5

Artifacts

No common artifacts to display.

Tags

Show diff only

model_type	XGBoost	RandomForest	GradientBoosting

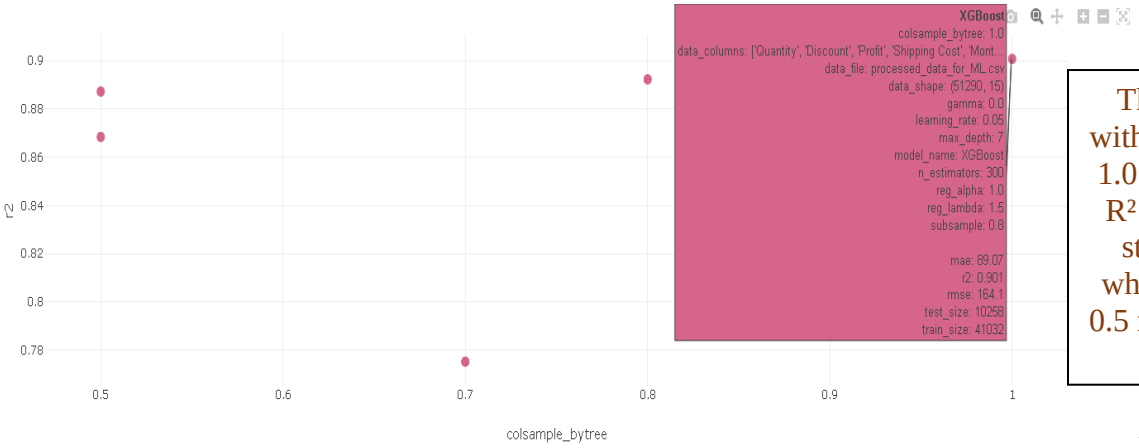
The XGBoost model has the best R^2 score of 0.892, indicating it explains 89.2% of the variance in the data, slightly outperforming RandomForest (0.885) and GradientBoosting (0.861). It also has a

- Parameters Logged: For each XGBoost run, parameters such as colsample_bytree (ranging from 0.5 to 1.0), gamma (0 to 5.0), learning_rate (0.05 to 0.5), max_depth (3 to 7), n_estimators (100 to 500), reg_alpha (0 to 4.0), reg_lambda (1.0 to 2.0), and subsample (0.5 to 0.9) were logged. These parameters were varied across runs to evaluate their impact on performance.

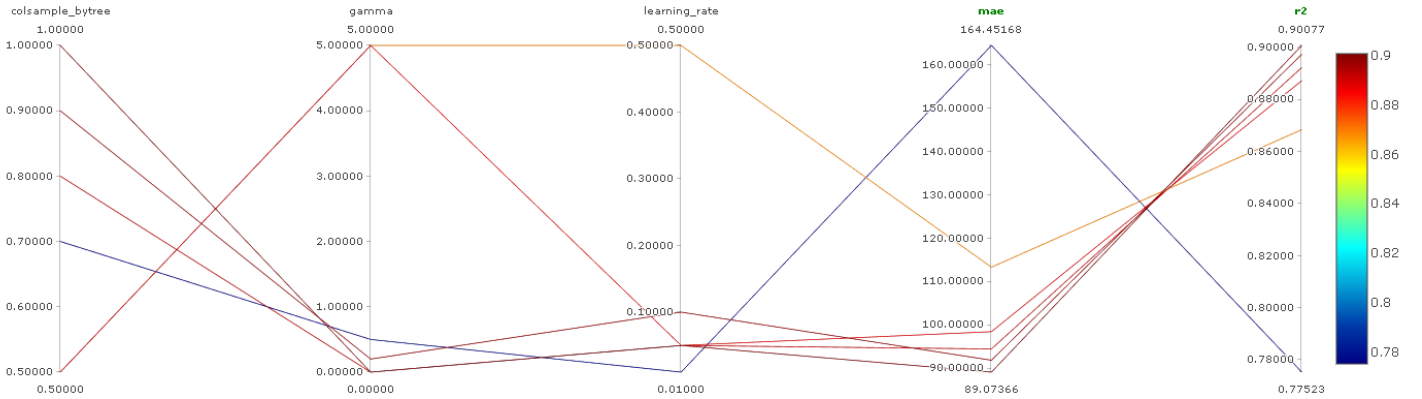
2- XGBOOST performance

- **Hyperparameter Tuning:** Multiple XGBoost models were trained with varying hyperparameters to explore their impact on performance. MLflow was used to log each experiment run.
- **Parameters Logged:** The following parameters were varied and logged in MLflow:

  - colsample_bytree: Values ranged from 0.5 to 1.0.
  - gamma: Values ranged from 0 to 5.0.
  - learning_rate: Values ranged from 0.05 to 0.5.
  - max_depth, n_estimators, reg_alpha, reg_lambda, and subsample were also logged but not shown in the axes of the plot.



The XGBoost model with colsample_bytree at 1.0 achieves the highest R^2 of 0.901, indicating strong performance, while lower values like 0.5 result in a reduced R^2 of 0.78.



This plot illustrates how varying parameters like colsample_bytree, gamma, and learning_rate impacts the model's performance. The R^2 values range from 0.77523 to 0.90077, and MAE from 89.07366 to 164.45168. Higher R^2 and lower MAE are observed with certain parameter combinations (colsample_bytree around 0.8-1.0 and lower gamma), indicating that tuning these parameters can significantly improve the model's accuracy and generalization.

➢ In conclusion:

With a test R2 of 0.8921 and a training R2 of 0.9150, the XGBoost model performed the best, showing good generalization to unknown data and strong explanatory power. That is the reason we ultimately decided on it.

2-Model deployment

A user-friendly Streamlit web application was created to support the deployment of the machine learning model and enable interactive sales forecasting for non-technical users.
How does the app work?

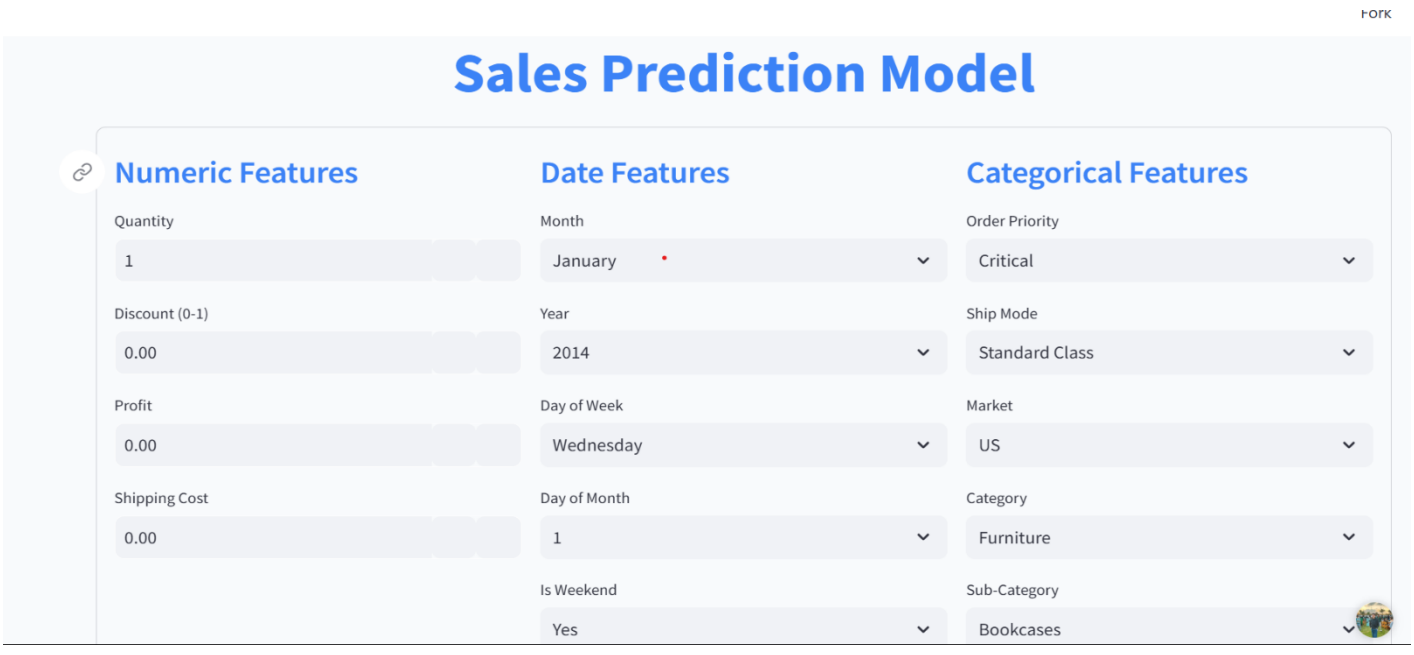
- The app takes user input for various features:

  a) **Numerical features**: Quantity, Discount, Profit, Shipping Cost
  b) **Date-based features**: Month, Year, Day of Week, Day of Month, Weekend indicator
  c) **Categorical features**: Order Priority, Ship Mode, Market, Category, Sub-Category

```
# Define feature mappings
FEATURE_MAPPINGS = {
    'Order Priority': ["Critical", "High", "Medium", "Low"],
    'Ship Mode': ["Standard Class", "Second Class", "First Class", "Same Day"],
    'Market': ["US", "EU", "APAC", "LATAM", "Africa"],
    'Category': ["Furniture", "Office Supplies", "Technology"],
    'Sub-Category': ["Bookcases", "Chairs", "Tables", "Furnishings", "Paper", "Binders",
                     "Storage", "Appliances", "Phones", "Accessories", "Machines", "Copiers"]
}
```

- Inputs are transformed into the proper format for the model.
- The app uses the model we trained **xgboost_sales_model.pkl** to make a prediction
- A clean **sales prediction output** is displayed using st.metric.
- A **summary table** shows all input values for reference.
- **Features:**

  - Styled using custom CSS for a modern, consistent look.
  - Includes a link to a separate **interactive dashboard** for deeper exploration.
  - Fully functional and published on streamlit cloud



TEAM 3 (DEPI _GHR2_AIS4_S2)

- Abdelwahab Mohamed Abdelwahab
- Mohamed Asaad Elbadawy
- Mohamed Ramadan Mohamed Radwan
- Abdelazim Reda Abdelazim Rashwan
- Mohamed Kamal Ahmed El Ashmawy
- Nouran Khaled Ali Wafa

Thank you