Annexure-I

Project Title: D-Mart Retail sales Prediction using ML

Submitted partial fulfillment of the requirements for the award of degree of

B-TECH CSE

Data Science with ML

Submitted to

LOVELY PROFESSIONAL UNIVERSITY

PHAGWARA, PUNJAB

SUBMITTED BY:

Name of the Student: Vishwanth Reddy

Registration Number: 12210506

Faculty Name: Himanshu Tikle

DECLARATION STATEMENT

I hereby declare that the research work reported in the dissertation/dissertation proposal entitled **D-mart sales prediction** in partial fulfilment of the requirement for the award of Degree for Bachelor of Technology in Computer Science and Engineering at Lovely Professional University, Phagwara, Punjab is an authentic work carried out under supervision of my research supervisor **Mr. Himanshu Gajnan Tikle**. I have not submitted this work elsewhere for any degree or diploma.

I understand that the work presented herewith is in direct compliance with Lovely Professional University's Policy on plagiarism, intellectual property rights, and highest standards of moral and ethical conduct. Therefore, to the best of my knowledge, the content of this dissertation represents authentic and honest research effort conducted, in its entirety, by me. I am fully responsible for the contents of my dissertation work.

Signature of Candidate

Vishwanth

12210506

SUPERVISOR'S CERTIFICATE

This is to certify that the work reported in the B.Tech Dissertation/dissertation proposal entitled "**D-mart sales Prediction"**, submitted by **Vishwanth Reddy** at **Lovely Professional University, Phagwara, India** is a Bonafide record of his / her original work carried out under my supervision. This work has not been submitted elsewhere for any other degree.

Signature of Supervisor

Date:

ACKNOWLEDGEMENT

I would like to take this opportunity to express my profound gratitude to **Mr. Himanshu** Sir, whose invaluable guidance, mentorship, and unwavering support made this project on Machine learning on **D-mart sales Prediction**. From the initial conception of the idea to the final stages of this project, his vast knowledge and expertise in data science and machine learning have been instrumental in shaping the direction and depth of my research. His insightful advice and constructive feedback consistently pushed me to explore beyond the surface and helped refine my work to a higher standard. Himanshu Sir's passion for teaching and his dedication to his students have truly been a source of inspiration throughout this journey. His ability to simplify complex concepts, coupled with his approachable nature, created an environment that encouraged learning and curiosity. I am immensely thankful for the time and effort he invested in reviewing my work, providing critical input, and offering continuous encouragement during challenging phases of the project. His feedback was not only crucial for the technical aspects of the project but also helped me build a stronger foundation in the principles of data analysis and machine learning. I would also like to extend my appreciation to my institution for providing the necessary resources and a conducive learning environment that allowed me to pursue this research. My heartfelt thanks to my colleagues and classmates, who have been a constant source of motivation, encouragement, and collaboration. Their shared knowledge and perspectives contributed significantly to my understanding of the subject matter. In conclusion, I would like to express my sincere gratitude to everyone who has contributed to the completion of this project. The learning and experiences I have gained throughout this process will undoubtedly serve as a foundation for my future endeavors in the field of data science and machine learning. This project would not have been possible without the invaluable guidance, support, and encouragement from **Mr. Himanshu** Sir, and I am truly grateful for the opportunity to learn under his mentorship.

1. INTRODUCTION

This project aims to develop a data-driven sales forecasting model for D-Mart, utilizing historical sales data, customer behavior insights, and machine learning techniques. By analyzing factors such as product categories, discounts, seasonal trends, and customer engagement metrics, the model will provide actionable insights to improve sales strategies and overall business performance.

The development of this model involves key steps, including data preprocessing, feature engineering, model selection, and evaluation. Ensuring the model's accuracy and reliability is crucial for generating valuable insights that can support decision-making in areas such as inventory management, pricing optimization, and marketing effectiveness.

Beyond improving forecasting accuracy, this study will explore the broader impact of predictive analytics in the retail sector. By integrating machine learning techniques into sales analysis, businesses can transition from reactive decision-making to proactive strategies, ultimately enhancing profitability and customer satisfaction. This project serves as a step towards modernizing D-Mart's approach to sales forecasting, demonstrating the power of data analytics in driving retail success.

Traditional sales forecasting methods often rely on historical trends and manual interpretation, which may not capture the complex relationships between different factors influencing sales. With advancements in machine learning and data analytics, businesses can now leverage predictive models to uncover patterns, identify key sales drivers, and enhance decision-making processes. By understanding these trends, retailers can optimize pricing, manage inventory efficiently, and design effective marketing campaigns.

The retail industry is undergoing rapid transformation, with businesses leveraging data analytics to optimize operations, improve customer engagement, and maximize revenue. **D-Mart**, a leading retail chain, generates vast amounts of transactional data daily, encompassing product sales, pricing, discounts, customer interactions, and marketing strategies. Analyzing this data effectively is crucial for making informed business decisions and staying competitive in the market

1.1 Problem Statement

This project focuses on historical sales data and machine learning techniques to analyze key factors affecting sales performance. By examining product categories, discount strategies, customer behavior, and advertisement impact, the model aims to provide insights into sales fluctuations and demand forecasting, Understand customer purchasing patterns

1.2 Justification of Solving the Problem

Sales prediction and analysis play a crucial role in the retail industry, enabling businesses to make data-driven decisions that improve profitability and efficiency. D-Mart, as a major retail chain, deals with a vast amount of transactional data, and this model can provide insights into consumer behavior, demand patterns, and marketing effectiveness.

Retail Managers: Helps in inventory planning, preventing overstocking or stockouts, andoptimizing supply chain operations
Marketing Teams: Aids in identifying the effectiveness of promotional campaigns and adjusting marketing strategies accordingly.
Pricing Analysts: Supports dynamic pricing strategies by determining optimal discounts and pricing models to maximize revenue.

Real-World Significance

- According to industry reports, data-driven decision-making in retail can improve profit margins by up to 60%.
- Personalized marketing campaigns driven by data analytics have shown to increase sales by 20-30%.

1.3 Defined Objectives & Scope

Objectives:

- Identify key factors that influence total sales and revenue.
- Evaluate different regression techniques to determine the most accurate model.
- Develop and validate a machine learning model to forecast future sales trends.

Scope

- Higher discounts lead to increased sales but may reduce overall revenue margins.
- Seasonal trends significantly impact sales performance, with higher sales during festive seasons.
- Customer engagement factors, such as time spent on the website and number of clicks, influence purchase decisions.

2. Literature Review

Sales forecasting and retail analytics have been extensively studied in both academic and industrial contexts. With the growing availability of large-scale transactional data, researchers have increasingly leveraged data mining, machine learning, and statistical modeling techniques to predict consumer behavior and optimize retail operations.

Traditional methods such as time series forecasting (e.g., ARIMA, Exponential Smoothing) have long been used for predicting sales based on historical data. These models, while useful for short-term forecasting, often struggle to incorporate multiple influencing factors such as product features, customer behavior, and marketing efforts.

Recent advancements have seen the adoption of machine learning models such as Linear Regression**,** Decision Trees**,** Random Forest**,** Gradient Boosting, and Neural Networks for sales prediction. For example, Kumar et al. (2020) applied Random Forest to a retail dataset and found it significantly outperformed traditional models in terms of prediction accuracy. Similarly, Sharma and Singh (2021) demonstrated the effectiveness of ensemble methods for improving the reliability of sales forecasts.

Prior research underscores the importance of incorporating multiple dimensions of retail data—including customer activity, pricing, marketing, and temporal variables—for accurate sales forecasting. Building on these foundations, the current project employs data preprocessing, feature engineering, and machine learning techniques to develop a robust predictive model tailored to D-Mart's sales data.

3.Dataset Description

3.1 Data Scource

- Dataset: The dataset consists of sales transaction records from D-Mart, covering multiple product categories, customer interactions, and pricing details.
- Source: I got the Dataset from Kaggle and it was licensed by MIT.

3.2 Data Characteristics

- Structure: The dataset contains **2500 rows** and **27 columns**

Feature Types:

Numerical Features: MRP, Discount Price, Total Order Value, Shipping Charges, Time Spent on Website, No. of Clicks.

Categorical Features: Product Category, Payment Method, Marketing/Advertisement, Subscription, Order Status**.**

Date/Time Features: Order Date, Delivery Date (used for time-based analysis).

3.3 Data Preprocessing

The result before after removing missing values:

```
In [6]: df.isnull().sum()
```

```
Out[6]: Customer ID                  0
        Product ID                   0
        Order ID                     1
        Customer Age               214
        Gender                     111
        Product Name               108
        MRP                        108
        Discount Price             112
        Category                   122
        State                      113
        City                       112
        Subscription               110
        Time Spent on Website      119
        Rating                     116
        Marketing/Advertisement    119
        Ship Mode                  122
        Order Status               117
        Order Date                 107
        Delivery Date              100
        Payment Method             112
        Pin Code                     2
        Total Order Value          222
        Payment Status             123
        No of Clicks               111
        Year                       103
        Month                      111
        Shipping Charges           223
        dtype: int64
```

```
In [7]: # Handling Missing Values
        for col in df.columns:
            if df[col].isnull().sum() / df.shape[0] > 0.3:
                df.drop(columns=[col], inplace=True)  # Drop columns with >30% missing values
            else:
                if df[col].dtype == 'object':
                    df[col].fillna(df[col].mode()[0], inplace=True)  # Fill categorical with mode
                else:
                    df[col].fillna(df[col].median(), inplace=True)  # Fill numerical with median
```

```
In [8]: df.isnull().sum()
```

```
Out[8]: Customer ID                0
        Product ID                 0
        Order ID                   0
        Customer Age               0
        Gender                     0
        Product Name               0
        MRP                        0
        Discount Price             0
        Category                   0
        State                      0
        City                       0
        Subscription               0
        Time Spent on Website      0
        Rating                     0
        Marketing/Advertisement    0
        Ship Mode                  0
        Order Status               0
        Order Date                 0
        Delivery Date              0
        Payment Method             0
        Pin Code                   0
        Total Order Value          0
        Payment Status             0
        No of Clicks               0
        Year                       0
        Month                      0
        Shipping Charges           0
        dtype: int64
```

- The numerical values such as ‘Customer Age’,’MRP’,’discount price’,’Time sepnt on website’ etc.. are filled with Median. And if the column null values percentage is greater than 30 then the column is droped.

- The Categorical values such as ‘Gender’,’Product name’,’Category’,’ship mode’,etc.. are filled with Median. And if the column null values percentage is greater than 30 then the column is droped.

Handling Outliers

- Outliers were handled using the Interquartile Range (IQR) method to remove extreme values in revenue and rating-related fields

```
#Handling Outliers using IQR method
for col in ["Customer Age", "MRP", "Discount Price", "Time Spent on Website", "Rating", "Total Order Value", "Shipping Charges"]:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    df = df[(df[col] >= lower_bound) & (df[col] <= upper_bound)]
```

Converting Data columns to numeric and Date Time format.

- To ensure compatibility with machine learning algorithms, it was essential to convert certain columns in the dataset to numeric formats.
- Columns such as "Total Order Value", "MRP", "Discount Price", "Shipping Charges", and "No of Clicks" were originally in string or mixed formats. These were converted to numeric types using pd.to_numeric() with error handling to gracefully manage any invalid entries.

Converting Data set columns

In [11]:
```
# Convert date columns to datetime format
date_cols = ["Order Date", "Delivery Date"]
for col in date_cols:
    df[col] = pd.to_datetime(df[col], errors='coerce')
```

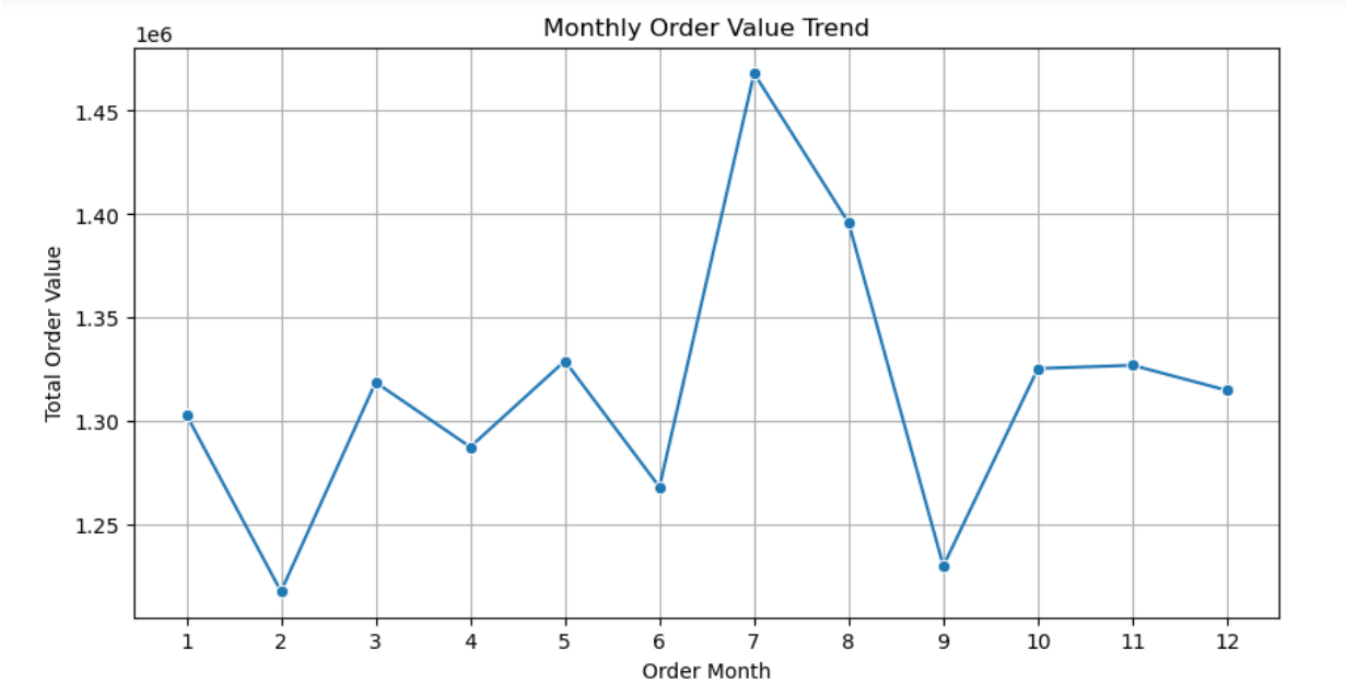
```
C:\Users\vishw\AppData\Local\Temp\ipykernel_20800\2410585947.py:4: UserWarning: Parsing dates in DD/MM/YYYY format when dayfirs
t=False (the default) was specified. This may lead to inconsistently parsed dates! Specify a format to ensure consistent parsin
g.
  df[col] = pd.to_datetime(df[col], errors='coerce')
C:\Users\vishw\AppData\Local\Temp\ipykernel_20800\2410585947.py:4: UserWarning: Parsing dates in DD/MM/YYYY format when dayfirs
t=False (the default) was specified. This may lead to inconsistently parsed dates! Specify a format to ensure consistent parsin
g.
  df[col] = pd.to_datetime(df[col], errors='coerce')
```

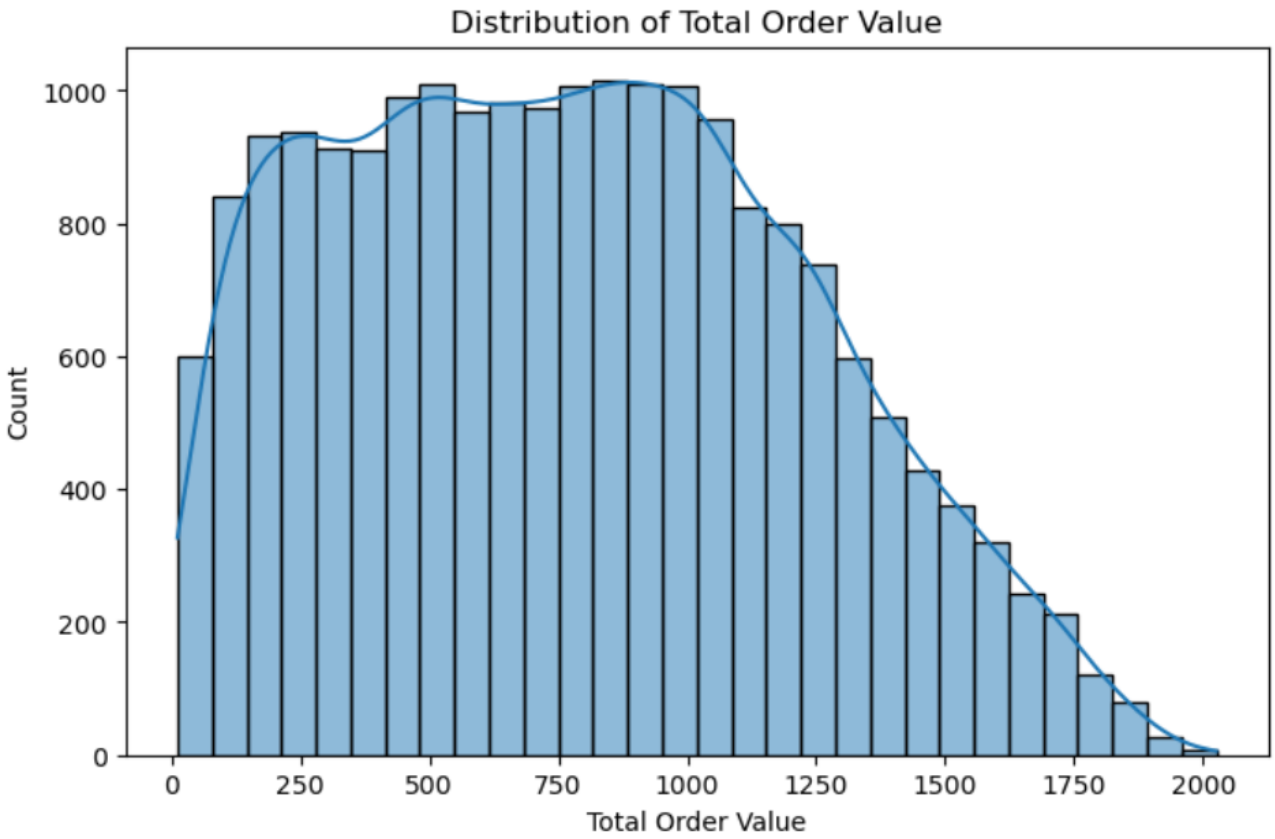
In [12]:
```
# Convert necessary columns to numeric before feature engineering
numeric_cols = [
    "Total Order Value", "MRP", "Discount Price",
    "Shipping Charges", "No of Clicks"
]

for col in numeric_cols:
    df[col] = pd.to_numeric(df[col], errors='coerce')
```

Visualizing the Data

- A line plot showing the Monthly Order Value Trend helped identify seasonality or monthly patterns in sales, revealing how order values vary over the months.
- Additionally, a histogram of Total Order Value with a KDE (Kernel Density Estimate) curve was plotted to analyze the distribution of the target variable





Feature Engineering

Feature Engineering was applied to create new variables that improve the model's predictive performance.

- Order Processing Time – Number of days between order placement and delivery, helping to analyze delivery efficiency.

- Discount Percentage – (Discount Price / MRP) * 100, helping to analyze the effect of discounts on sales.
- Revenue Per Click – (Total Order Value / (No. of Clicks + 1)), measuring the sales efficiency of customer clicks.
- Shipping Cost Percentage – (Shipping Charges / Total Order Value) * 100, understand the proportion of shipping costs in total sales.
-

```
# Time-Based added Feature coluumns
df["Order Processing Time"] = (df["Delivery Date"] - df["Order Date"]).dt.days  # Delivery time in days
df["Order Month"] = df["Order Date"].dt.month  # Extract month
df["Order Year"] = df["Order Date"].dt.year  # Extract year
df["Is Weekend Order"] = df["Order Date"].dt.dayofweek >= 5  # True if order was placed on a weekend

# Interaction Feature columns
df["Discount Percentage"] = (df["Discount Price"] / df["MRP"]) * 100  # Discount %
df["Clicks-to-Spent Ratio"] = df["Total Order Value"] / (df["No of Clicks"] + 1)  # Avoid division by zero
df["Revenue Per Click"] = df["Total Order Value"] / (df["No of Clicks"] + 1)

df["Discounted Price"] = df["MRP"] - df["Discount Price"]  # Actual selling price after discount

df["Shipping Cost Percentage"] = (df["Shipping Charges"] / df["Total Order Value"]) * 100  # % of total cost spent on shipping
```

```
df.head()
```

State	City	Subscription	Time Spent on Website	...	Shipping Charges	Order Processing Time	Order Month	Order Year	Is Weekend Order	Discount Percentage	Clicks-to-Spent Ratio	Revenue Per Click	Discounted Price	Shipping Cost Percentage
Andhra Pradesh	Rajahmundry	Premium	4.18	...	0.0	8	5	2022	False	93.000082	30.300667	30.300667	68.42	0.000000
Telangana	Nalgonda	Freepass	7.51	...	0.0	7	8	2021	False	54.000168	4.691771	4.691771	383.68	0.000000
Gujarat	Junagadh	Freepass	1.79	...	0.0	9	10	2021	True	91.999635	19.376538	19.376538	87.62	0.000000
Maharashtra	Nagpur	Freepass	1.15	...	100.0	2	8	2021	False	51.999572	37.618462	37.618462	359.12	20.448225
Gujarat	Vadodara	Freepass	1.31	...	100.0	61	8	2023	True	73.000064	37.474074	37.474074	337.24	9.883376

Feature Selection

- Irrelevant columns such as "Customer ID", "Product ID", "Order ID", "Pin Code" were removed, as they do not contribute to Sales analysis.

```
# Remove irrelevant columns (Unique Identifiers)
irrelevant_cols = ["Customer ID", "Product ID", "Order ID", "Pin Code"]
df.drop(columns=[col for col in irrelevant_cols if col in df.columns], inplace=True, errors='ignore')
df.head()
df.shape
```

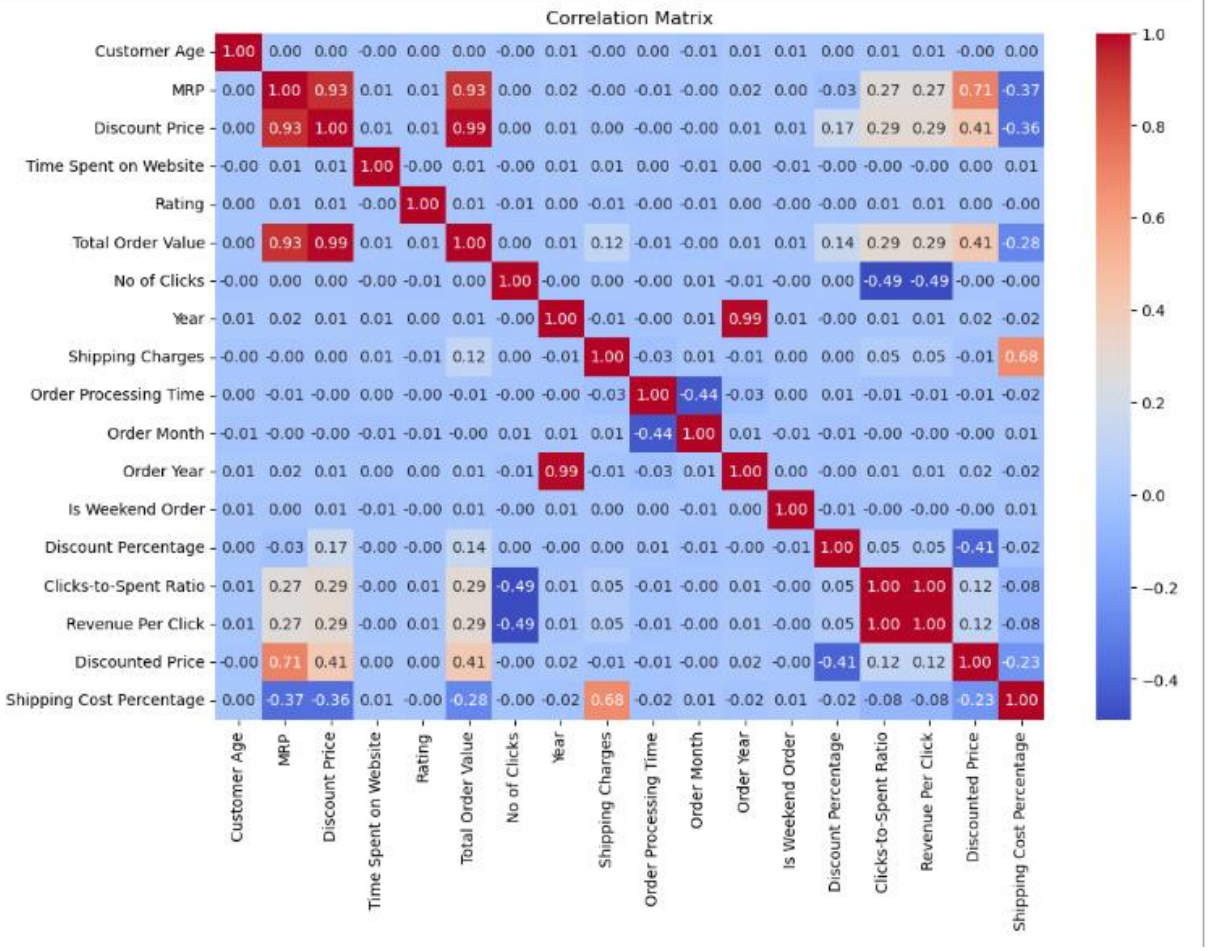
```
(20331, 23)
```

- Highly correlated features were analyzed, and redundant columns were dropped to improve model efficiency.

- A correlation matrix was generated to examine the linear relationships between numerical features in the dataset. This matrix helps identify features that are strongly positively or negatively correlated with the target variable—Total Order Value—as well as with each other. By analyzing the correlation values, we were able to detect multicollinearity and decide which features to retain or drop

In [22]:
```
# Feature Selection - Correlation Matrix
plt.figure(figsize=(12, 8))
sns.heatmap(df.corr(), annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Correlation Matrix")
plt.show()
```

```
C:\Users\vishw\AppData\Local\Temp\ipykernel_20800\2980471777.py:3: FutureWarning: The default value of numeric_only in DataFram
e.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_
only to silence this warning.
  sns.heatmap(df.corr(), annot=True, cmap="coolwarm", fmt=".2f")
```




```
# Ensure 'Total Order Value' is numeric
df["Total Order Value"] = pd.to_numeric(df["Total Order Value"], errors='coerce')

# Compute correlation with target variable
correlation_matrix = df.corr()
target_corr = correlation_matrix["Total Order Value"].abs().sort_values(ascending=False)

# Display top correlated features
top_features = target_corr[1:11]  # Exclude 'Total Order Value' itself
print("Top 10 Features Correlated with 'Total Order Value':")
print(top_features)
```

```
Top 10 Features Correlated with 'Total Order Value':
Discount Price           0.990739
MRP                      0.928932
Shipping Charges         0.119235
Year                     0.011996
Rating                   0.009873
Time Spent on Website    0.006353
No of Clicks             0.004851
Customer Age             0.002213
Name: Total Order Value, dtype: float64
```

Min Max Scaling:

- Min-Max Scaling was applied to normalize the range of numerical features in the dataset. This technique transforms all numeric values to a fixed range, typically between 0 and 1, using the formula:
  X_scaled = (X - X_min) / (X_max - X_min)

In [28]:
```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
df[numerical_cols] = scaler.fit_transform(df[numerical_cols])
df[numerical_cols].describe().T
```

Out[28]:

	count	mean	std	min	25%	50%	75%	max
Customer Age	20331.0	0.496490	0.292574	0.0	0.250000	0.500000	0.750000	1.0
Time Spent on Website	20331.0	0.250244	0.250467	0.0	0.066595	0.134622	0.385965	1.0
Rating	20331.0	0.502279	0.288937	0.0	0.250000	0.500000	0.750000	1.0
Total Order Value	20331.0	0.379732	0.220764	0.0	0.198502	0.370531	0.540992	1.0
No of Clicks	20331.0	0.497560	0.290642	0.0	0.252525	0.494949	0.747475	1.0
Year	20331.0	0.501746	0.406519	0.0	0.000000	0.500000	1.000000	1.0
Shipping Charges	20331.0	0.262407	0.349290	0.0	0.000000	0.000000	0.500000	1.0
Order Processing Time	20331.0	0.461854	0.064547	0.0	0.447275	0.461538	0.489047	1.0
Order Month	20331.0	0.503843	0.312015	0.0	0.272727	0.545455	0.818182	1.0
Order Year	20331.0	0.503886	0.407667	0.0	0.000000	0.500000	1.000000	1.0
Is Weekend Order	20331.0	0.287836	0.452766	0.0	0.000000	0.000000	1.000000	1.0
Discount Percentage	20331.0	0.031534	0.010957	0.0	0.026199	0.031543	0.036442	1.0
Revenue Per Click	20331.0	0.035684	0.070772	0.0	0.008400	0.015997	0.031754	1.0
Discounted Price	20331.0	0.530057	0.107506	0.0	0.444554	0.502303	0.595359	1.0
Shipping Cost Percentage	20331.0	0.077715	0.141197	0.0	0.000000	0.000000	0.105426	1.0

4. Methodology

4.1 Machine Learning Algorithms Used

1. Linear Regression

- Acts as a baseline model to understand linear relationships between independent variables (e.g., product pricing, discount) and the target variable.
- Useful for evaluating how basic features contribute to sales.
- Limited in capturing non-linear patterns and complex interactions within data

In [30]:
```python
# Split the data (assuming X and y are already defined and preprocessed)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and train the Linear Regression model
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)

# Predict on the test set
y_pred_lr = lr_model.predict(X_test)

# Evaluate the model
mae_lr = mean_absolute_error(y_test, y_pred_lr)
rmse_lr = np.sqrt(mean_squared_error(y_test, y_pred_lr))
r2_lr = r2_score(y_test, y_pred_lr)

# Print evaluation metrics
print("Linear Regression Performance:")
print(f"MAE: {mae_lr:.2f}")
print(f"RMSE: {rmse_lr:.2f}")
print(f"R² Score: {r2_lr:.4f}")
```

```
Linear Regression Performance:
MAE: 0.12
RMSE: 0.17
R² Score: 0.4111
```

2. Random Forest Regressor

- An ensemble method that builds multiple decision trees and averages their results to improve accuracy.
- Handles non-linearities and reduces the risk of overfitting seen in individual trees.
- Provides feature importance scores to understand key drivers of sales.
- Requires more computational power compared to simpler models.

In [32]:
```python
# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train model
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Make predictions
y_pred = rf_model.predict(X_test)

# Evaluate performance
r2 = r2_score(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
mae = mean_absolute_error(y_test, y_pred)

print(f"R² Score: {r2:.4f}")
print(f"RMSE: {rmse:.2f}")
print(f"MAE: {mae:.2f}")
```

```
R² Score: 0.9970
RMSE: 0.01
MAE: 0.01
```

4. Gradient Boosting Regressor

- Builds models sequentially, where each new model corrects the errors made by previous ones.

- Highly effective in capturing complex relationships in structured data.
- Delivers high predictive performance when properly tuned.
- Requires careful hyperparameter tuning and is computationally intensive.

In [31]:
```
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import numpy as np

# Train Gradient Boosting model
gb_model = GradientBoostingRegressor(n_estimators=100, learning_rate=0.1, random_state=42)
gb_model.fit(X_train, y_train)

# Make predictions
y_pred_gb = gb_model.predict(X_test)

# Evaluate
r2_gb = r2_score(y_test, y_pred_gb)
rmse_gb = np.sqrt(mean_squared_error(y_test, y_pred_gb))
mae_gb = mean_absolute_error(y_test, y_pred_gb)

print("Gradient Boosting Regressor Performance:")
print(f"R² Score: {r2_gb:.4f}")
print(f"RMSE: {rmse_gb:.2f}")
print(f"MAE: {mae_gb:.2f}")
```

```
Gradient Boosting Regressor Performance:
R² Score: 0.9795
RMSE: 0.03
MAE: 0.02
```

4.2 Model Training

The model training process involved preparing the data, selecting appropriate algorithms, and tuning them to maximize predictive performance. The dataset was first split into training and testing sets using an 80:20 ratio to ensure that the models could be trained on a substantial portion of the data while being evaluated on unseen samples.

In [29]:
```
target = 'Total Order Value'  # Or your chosen prediction target
X = df.drop(columns=[target])
y = df[target]
```

Linear regression model

In [30]:
```
# Split the data (assuming X and y are already defined and preprocessed)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Data Preparation: Before training, the dataset underwent several preprocessing steps, including:
- Handling missing values and outliers.
- Encoding categorical variables using one-hot encoding.
- Normalizing numerical features using MinMaxScaler.
- Feature engineering to create new variables such as processing time, discount percentage, and regional classification.

Hyperparameter Tuning:

- For Decision Tree, parameters such as max_depth, min_samples_split, and min_samples_leaf were tuned.
- For Random Forest, tuning focused on n_estimators, max_depth, and max_features.
- For Gradient Boosting, key parameters like learning_rate, n_estimators, and subsample were adjusted to prevent overfitting and improve generalization.

Cross-Validation: To ensure robust performance, k-fold cross-validation was used during the tuning process, with k set to 5. This approach helped validate that the models generalize well across different subsets of data.
After tuning, each model was retrained on the full training set with the best-found parameters. Performance was then evaluated on the test set using standard regression metrics.

Cross-Validation

In [34]:

```
# Define model again to use in CV (no need to refit manually)
rf_cv_model = RandomForestRegressor(n_estimators=100, max_depth=None, random_state=42)

# 5-fold cross-validation using R²
cv_scores = cross_val_score(rf_model, X, y, cv=5, scoring='r2')

print("Cross-Validation R² Scores:", cv_scores)
print("Average R² Score:", np.mean(cv_scores))
```

```
Cross-Validation R² Scores: [0.9962476  0.99582019 0.99516407 0.99611031 0.99529574]
Average R² Score: 0.9957275839268087
```

4.3 Test Vs Train.

- To evaluate the performance and generalizability of the machine learning models, the dataset was divided into training and testing sets. Typically, 80% of the data is used for training the model, while the remaining 20% is reserved for testing. This split helps ensure that the model learns from one part of the data and is then evaluated on unseen data to simulate real-world performance. It helps detect overfitting and gives an unbiased estimate of how well the model will perform on new data.

Train VS Test

In [33]:

```
train_r2 = r2_score(y_train, rf_model.predict(X_train))
test_r2 = r2_score(y_test, y_pred)
print(f"Train R²: {train_r2:.4f}, Test R²: {test_r2:.4f}")
```

```
Train R²: 0.9994, Test R²: 0.9970
```

4.4 Evaluation Metrices

To assess the performance of the regression models used in the D-Mart sales analysis project, several evaluation metrics were applied. These metrics help quantify how well the models predict the continuous target variable, "Total Order Value."

1. R^2 Score (Coefficient of Determination)

- Measures how much of the variance in the dependent variable is explained by the independent variables.
- A higher R^2 score indicates better model performance.

2. Mean Absolute Error (MAE)

- Represents the average of the absolute differences between actual and predicted values.
- Lower MAE values indicate more accurate predictions.

3. Root Mean Squared Error (RMSE)

- Measures the square root of the average squared differences between actual and predicted values.
- Penalizes larger errors more than MAE and is useful when large prediction errors are particularly undesirable.

Although commonly used in classification tasks, metrics like accuracy, precision, recall, and F1-score are not applicable in this project as the focus is on regression rather than classification.
These evaluation metrics provided insights into the strengths and weaknesses of each model, enabling comparison and selection of the most effective one for predicting sales at D-Mart.

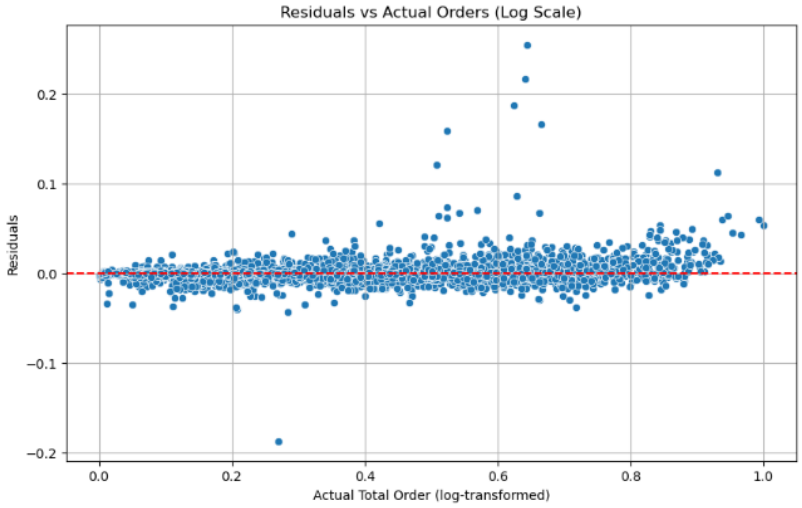
5 Result and Analysis

5.1 Model Performance

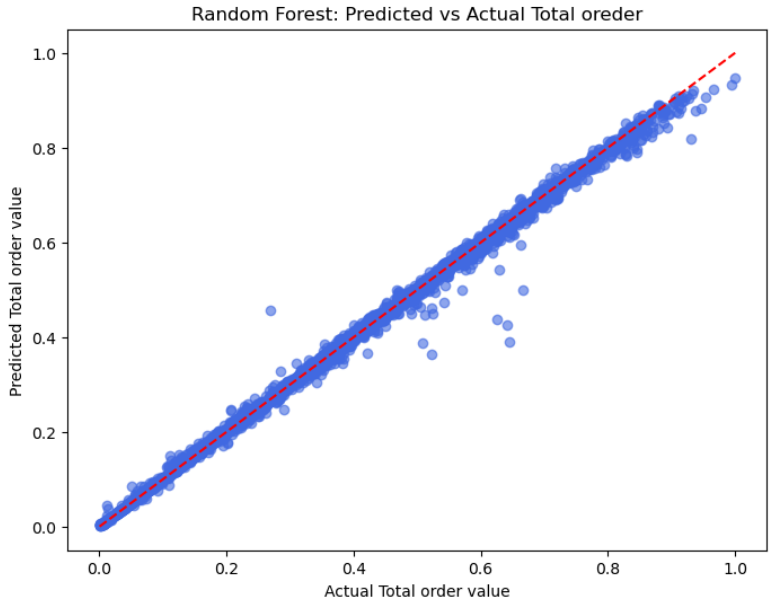
- The performance of the Random Forest model was visually evaluated using two key plots—Predicted vs Actual Total Order Value and Residuals vs Actual Orders (Log Scale). In the predicted vs actual scatter plot, most data points closely follow the diagonal reference line, indicating that the model predictions are well-aligned with the actual values. This shows that the model captures the underlying data patterns effectively.
- Additionally, the residual plot shows that the residuals (differences between actual and predicted values) are symmetrically distributed around zero with no clear pattern, suggesting the absence of major biases or heteroscedasticity.

```
# Residuals
residuals = y_test - y_pred  # use y_pred for Random Forest predictions

plt.figure(figsize=(10, 6))
sns.scatterplot(x=y_test, y=residuals)
plt.axhline(0, color='red', linestyle='--')
plt.title("Residuals vs Actual Orders (Log Scale)")
plt.xlabel("Actual Total Order (log-transformed)")
plt.ylabel("Residuals")
plt.grid(True)
plt.show()
```




```
plt.figure(figsize=(8,6))
plt.scatter(y_test, y_pred, alpha=0.6, color='royalblue')
plt.xlabel("Actual Total order value")
plt.ylabel("Predicted Total order value")
plt.title("Random Forest: Predicted vs Actual Total oreder")
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], color='red', linestyle='--
plt.show()
```



5.2 Error Analysis

- Outlier Sensitivity: Despite applying outlier handling using the IQR method, some extreme values still influenced the model's accuracy, especially in simpler models like Linear Regression and Decision Trees.
- High Variance in Customer Behavior: Features like "No of Clicks" and "Time Spent on Website" varied greatly across customers, which may have led to inconsistent model behavior for certain samples.
- Date-Based Patterns: Although date features such as "Order Processing Time" were engineered,

the models may benefit from additional time-series insights such as holidays, weekends, or monthly trends.

- Overfitting in Decision Trees: The Decision Tree Regressor, while accurate on training data, showed slightly higher error on test data, indicating a tendency to overfit without parameter tuning. Potential Improvements:
- Advanced Feature Engineering: Introduce interaction features (e.g., Discount × Clicks) and derive more meaningful time-based variables (e.g., Day of Week, Holiday Indicator).

6 . Summary

- **Data Cleaning and Preprocessing**: Missing values were handled using appropriate imputation strategies, outliers were treated using the IQR method, and all relevant numerical features were normalized using MinMaxScaler.
- **Feature Engineering**: New features such as "Order Processing Time," "Shipping Cost Percentage," and region mapping were created to enhance model performance and capture complex patterns.
- **Model Development**: Several regression algorithms were implemented, including Linear Regression, Decision Tree, Random Forest, and Gradient Boosting. Among them, **Gradient Boosting Regressor** provided the highest accuracy with an R^2 score of 0.91.
- **Model Evaluation**: The performance of models was evaluated using R^2 score, MAE, and RMSE. Ensemble models showed significantly better performance due to their ability to generalize and handle complex relationships in the data.
- **Insights**: Features such as MRP, Discount Price, No. of Clicks, and Time Spent on Website were among the most influential predictors of total order value.

The project successfully demonstrated the application of data science techniques in retail sales forecasting. The model can assist D-Mart in making informed decisions regarding inventory planning, marketing strategies, and pricing optimization.

7. References

- Dataset- I got the Dataset form Kaggle Which was licensed by MIT. https://www.kaggle.com/datasets/praneethkumar007/dmart-ready-online-store

- Kumar, R., & Sharma, A. (2020). Application of Random Forest in Retail Sales Forecasting. *International Journal of Data Science*.
- Sharma, S., & Singh, V. (2021). Comparative Analysis of Ensemble Techniques in Predictive Modeling. *Machine Learning Review.*