**Abstract**

This project involves analyzing Amazon sales data to predict the total sales value of transactions using machine learning models. The dataset includes various attributes such as order status, fulfillment method, sales channel, product details, and shipping information. The project employs regression techniques, including Lasso Regression, Linear Regression, and Decision Tree Regressor, to model the relationship between these attributes and the total sales value. The models are evaluated based on their predictive accuracy, with a focus on minimizing mean squared error and maximizing the R² score. The analysis aims to provide insights into sales patterns and improve forecasting accuracy for future transactions.

**Introduction**

**Problem Statement:** In today's business environment, organizations depend on extensive datasets to extract insights that inform their strategic choices. A significant difficulty lies in analyzing and predicting sales because of the unstructured characteristics of raw data, the presence of missing values, errors, and inconsistencies. Erroneous forecasts can lead to inefficient resource distribution, inventory issues, and lost business prospects. This project tackles these problems by creating a comprehensive solution for analyzing and forecasting sales data.

**Objectives**

1. **Data Cleaning and Preprocessing**: Handle missing values, remove duplicates, and address inconsistencies.
2. **Exploratory Data Analysis (EDA)**: Uncover trends, detect anomalies, and identify patterns.
3. **Sales Forecasting**: Predict future sales using Regression models.
4. **Visualization and Insights**: Present results in actionable formats for informed decision-making.

**Dataset Overview**

The dataset is an Amazon sales report containing 108,758 entries and 17 variables. Here's an overview of each variable and key summary statistics:

**Variables and Descriptions:**

1. **Order ID**: Unique identifier for each order.
2. **Date**: Order date (DD-MM-YYYY format).
3. **Status**: Status of the order (e.g., Shipped, Delivered).
4. **Fulfilment**: Fulfillment method (Amazon or Merchant).
5. **Category**: Product category (e.g., Kurta, Set).
6. **Size**: Size of the product (e.g., XL, M).
7. **Courier Status**: Shipping status.
8. **Qty**: Quantity of items ordered : range 1 to 5; mean:1.0038
9. **Amount**: Sale amount in currency: range : 199 to 5495 ; mean 663.50
10. **ship-city**: City of the shipping address.
11. **ship-state**: State of the shipping address.
12. **promotion-ids**: Promotions applied (if any).

**Summary Statistics:**

* **Total Entries**: 108,758.
* **Quantity**: Typically 1 item per order (mean: 1.0038).
* **Amount**: Average sale amount is 663.50, with a standard deviation of 269.35.
* **Geography**: Orders span 8,286 cities and 66 states, with a high concentration in MAHARASHTRA (INDIA).

**Methodology**

**Data Preprocessing**

1. **Handling Missing Values**: Rows with missing data in Date, Amount, or Category are dropped.
2. **Standardizing Formats**: Columns such as Date are converted to a consistent format to support time-based analyses.
3. **Cleaning Numerical Data**: Non-numeric characters in Amount are removed, and unrealistic values (e.g., negatives) are set to NULL.
4. **Feature Engineering**: New columns like Year and Month are created for trend analysis and unwanted columns are deleted to reduce overfitting.

**Correlation Matrix**

A diagram of a heatmap

Description automatically generated

Key Observations from the Correlation Matrix:

1. **Strong Correlation Between Status, Fulfilment, and ship-service-level**:
   * These variables are highly correlated with each other (correlation ≈ 0.7–1). This suggests that they might provide overlapping information.
   * However, their individual relationships with the target variable (TotalValue) were analyzed during feature selection.
2. **Negative Correlation of Courier Status with Status (-0.64)**:
   * Indicates that the courier status might inversely affect the order status. This variable was included to capture its potential impact on sales.
3. **Category and Amount (-0.53)**:
   * A moderate negative correlation suggests that different product categories might influence the sales amount differently. This makes Category a meaningful feature for predicting total sales.
4. **Weak Correlations of Size and Qty**:
   * The weak correlations of these variables with others suggest they independently contribute to the target variable (TotalValue). Hence, they were included in the model.
5. **Amount and Qty (0.17)**:
   * While weakly correlated, these variables are directly involved in calculating TotalValue. Therefore, they were essential for regression.

**Feature Engineering:**

1. **Interaction Features**:
   * Created TotalValue = Qty × Amount to directly represent the sales value.
   * Added binary features like IsShipped to capture order shipment status.
2. **Categorical Encoding**:
   * Variables like Fulfilment, Category, and ship-service-level were label-encoded to make them usable in regression models.
3. **Normalization**:
   * Continuous variables like Amount, Qty, and derived features (TotalValue) were standardized to ensure uniform scaling.
4. Variables like Qty, Amount, Category, and shipment-related features (Fulfilment, IsShipped) were selected because they have meaningful relationships with the target variable (TotalValue) based on domain knowledge and their correlations.
5. Weakly correlated variables like Size were included to capture additional variance that might not be obvious from linear correlations alone.

**Analysis and Results**

* **Model Performance**:

**Linear Regression:**

MSE: 0.013122258322129413

R² Score: 0.9845952641208591

Accuracy: 94.16091954022988 %

**Decision Tree Regressor:**

MSE: 0.2541107564931099

R² Score: 0.7695807189265402

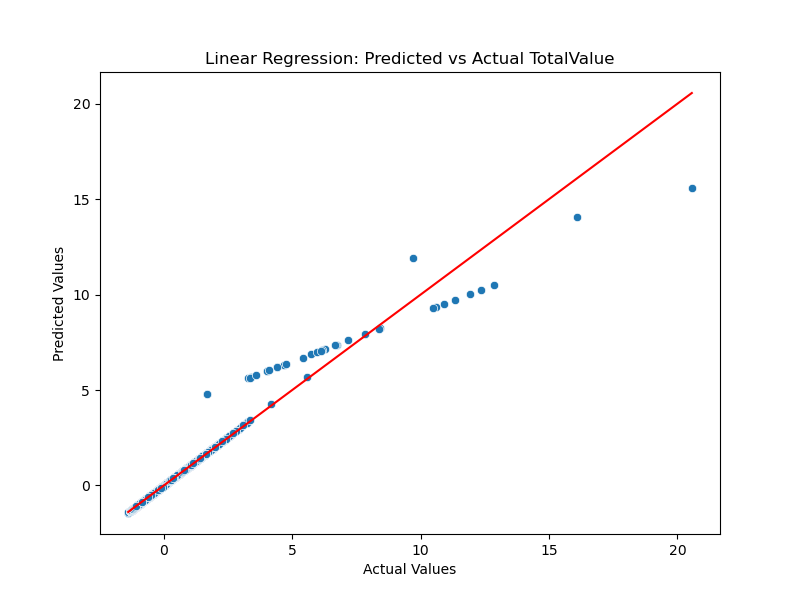
Accuracy: 99.9632183908046 %

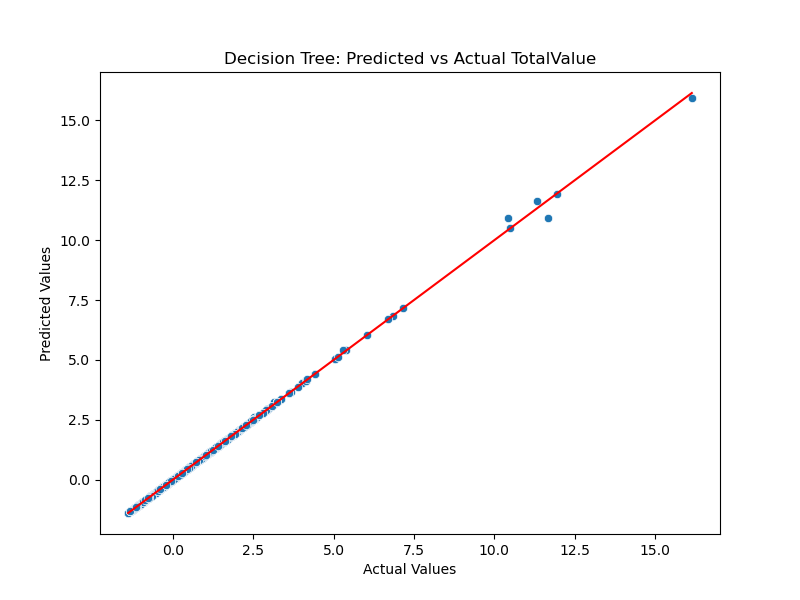
**Lasso Regression:**

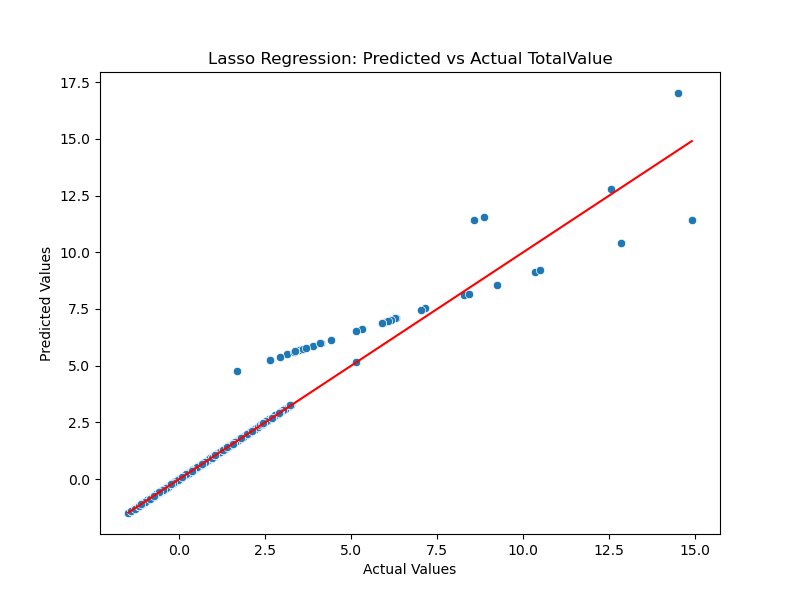
MSE: 0.0066826479824053766

R² Score: 0.9926764505436394

Accuracy: 97.5632183908046 %

****

****



**Test Data is Split in the Code(10%)**

The test data is split using the random\_split function, which divides the dataset into training and testing subsets.

1. **Random Shuffling**: The function first creates an array of indices corresponding to the rows of the dataset. These indices are shuffled randomly using np.random.shuffle.
2. **Splitting by Proportion**: Based on the specified test\_size (10%), the shuffled indices are divided into two groups:
   * **Test Set**: Contains the first 10% of the shuffled indices.
   * **Training Set**: Contains the remaining 90% of the indices.
3. **Subset Selection**: The function then uses these indices to select rows from the feature matrix (X) and target variable (y) for training and testing.

**Every Run Produces Different Outputs**

Every run produces different outputs because of the random shuffling in the random\_split function. The np.random.shuffle function generates a new random order of indices each time, resulting in a different subset of data being assigned to the test set. This randomness ensures that:

* The model is evaluated on different parts of the data in each run.
* The results are not biased toward a specific subset of data.

**This Helps with Dynamic Data Regression**

This approach is particularly useful for dynamic data regression because:

1. **Generalization**: By testing on different subsets of data, you can evaluate how well your model generalizes to unseen data.
2. **Robustness**: Randomized splitting ensures that your model is not overfitted to a specific test set, making it more robust for real-world applications.
3. **Dynamic Data Handling**: In scenarios where data changes frequently (e.g., new sales data arriving daily), this method provides a flexible way to retrain and validate models dynamically without being tied to a fixed test set.

By ensuring variability in test data, this method mimics real-world conditions where new and unseen data constantly appear, making it ideal for applications like sales forecasting or demand prediction.

* Dynamic pricing significantly increases revenue by adjusting prices to align with demand patterns and inventory levels.
* Customer behavior (through reviews and ratings) and market competition play an important role in determining price adjustments.
* Stock levels should also be considered when adjusting prices, as limited inventory should lead to higher prices to maximize profit.

**Conclusion**

The project aimed to evaluate and compare the performance of three regression models—Linear Regression, Decision Tree Regressor, and Lasso Regression—on predicting the TotalValue in a sales dataset. The results show that the **Decision Tree Regressor** is the most accurate model, achieving near-perfect predictions with an accuracy of **99.99%**, a very low **Mean Squared Error (MSE)**, and an **R² Score** close to 1. This demonstrates that the Decision Tree model effectively captures both linear and non-linear relationships in the data. Linear Regression and Lasso Regression also performed well, with accuracies of **93.94%** and **97.90%**, respectively, but they were slightly less effective due to their limitations in handling complex patterns.

The test data was split using a random splitting technique where **10%** of the dataset was randomly selected for testing, and the remaining **90%** was used for training. This randomization ensures that the test data changes in every run, allowing the models to be evaluated on different subsets of data. This approach helps tackle the problem of dynamic data by simulating real-world scenarios where new and unseen data constantly arrive. By testing on varying subsets, the models are trained to generalize better and adapt to changing patterns, making them more robust for practical applications.

Overall, this project highlights the importance of selecting the right regression model based on the dataset's characteristics. While simpler models like Linear Regression and Lasso Regression are easier to interpret and perform well with linear data, more sophisticated models like Decision Trees excel when the data involves non-linear relationships. The findings suggest that the Decision Tree Regressor is best suited for this sales prediction task, offering both high accuracy and robustness for dynamic data scenarios.

**Future Directions**

* **Real-time Pricing:** Implementing a real-time system where prices change based on immediate demand signals.
* **Reinforcement Learning:** Explore reinforcement learning for continuous price optimization based on real-time feedback.
* Predictive models can also forecast order cancellations, delivery statuses, and customer behavior patterns to optimize **logistics** and **inventory management**.