**ML-DRIVEN INSIGHTS OF ZUDIO SALES AND STORE MANAGEMENT FOR BUSINESS GROWTH**

**ABSTRACT**

The retail industry in India has witnessed rapid growth driven by rising consumer demand and urbanization. Apparel retail, especially fast fashion brands like Zudio, has become increasingly popular among middle-class and youth segments. According to industry reports, India’s retail market is expected to reach $1.75 trillion by 2026, with fashion retail contributing a significant share. The objective is to develop a predictive system that accurately forecasts Zudio’s sales profit using historical sales data. Traditionally, sales forecasting and store management relied heavily on manual record-keeping, spreadsheets, and human judgment. Managers analyzed past sales data, customer footfall, and inventory levels through basic statistical methods or intuition to make decisions on stock replenishment and promotional activities. Manual systems suffer from several limitations, including data inaccuracies, delayed analysis, and subjective decision-making. They cannot efficiently handle large datasets or adapt quickly to market changes, resulting in missed sales opportunities and excess inventory. Lack of automation reduces scalability and responsiveness, making it difficult to optimize store performance and profit margins consistently. This research aims to overcome the limitations of manual systems by leveraging machine learning to provide accurate, data-driven sales predictions. By automating analysis and incorporating multiple variables, the system improves forecasting precision, reduces human error, and supports proactive business strategies. The motivation is to enhance operational efficiency and profitability by utilizing advanced regression models that capture complex trends unseen in traditional approaches. The proposed system applies machine learning models, specifically Ridge Regression and Linear Regression, to predict sales profit for Zudio stores. These models analyze historical sales data combined with various store and product attributes, transforming categorical variables through encoding for better model performance. Ridge Regression adds regularization to prevent overfitting, while Linear Regression provides a baseline prediction model. Together, they offer robust and interpretable forecasting, enabling managers to make informed decisions on inventory, promotions, and resource allocation. This approach enhances sales prediction accuracy, helping optimize business growth and operational efficiency through actionable insights.

**CHAPTER 1**

**INTRODUCTION**

**1.1 Overview**

In today’s fast-paced retail environment, efficient sales and store management play a crucial role in ensuring business growth. Zudio, as a leading fashion retail brand, deals with high sales volumes, diverse customer preferences, and dynamic market demands. Effective sales management requires data-driven insights to optimize stock levels, improve customer engagement, and enhance operational efficiency. Store management includes inventory control, demand forecasting, product placement, and customer experience enhancement, all of which impact profitability. With the increasing volume of data generated from sales transactions, customer feedback, and seasonal trends, businesses need intelligent systems to process and analyze this data for better decision-making.

Historically, sales and store operations were managed using manual techniques, rule-based systems, and static reports, which lacked adaptability to real-time market changes. These methods often resulted in issues such as overstocking, stockouts, inaccurate demand forecasting, and ineffective product positioning. The rise of data-driven solutions offers the opportunity to overcome these inefficiencies. The adoption of machine learning can transform sales and store management by leveraging predictive analytics, pattern recognition, and automated decision-making. By analyzing sales data, customer preferences, and historical trends, businesses can optimize store layouts, plan inventory effectively, and provide personalized customer experiences.

**1.2 Research Motivation**

The motivation behind this research stems from the growing complexity of retail sales and store operations, which demand intelligent solutions for effective management. Zudio, like other retail chains, faces challenges in predicting demand, managing inventory efficiently, and optimizing store layouts to enhance customer experiences. Existing approaches struggle to adapt to dynamic market trends and often rely on historical data without leveraging predictive capabilities. As consumer behavior shifts rapidly due to factors like seasonal demand, promotions, and external influences, businesses require a more sophisticated approach to decision-making.

The advancements in data science and machine learning present an opportunity to revolutionize sales analytics and store management. By leveraging AI-driven techniques, retailers can gain deeper insights into sales trends, forecast demand with higher accuracy, and optimize inventory allocation. This research is motivated by the need to improve operational efficiency, reduce financial losses due to overstocking or understocking, and enhance customer satisfaction through better product availability. The ability to analyze large-scale sales data and generate actionable recommendations is a key factor in driving business success. Implementing machine learning-based solutions in sales and store management is not just an innovation but a necessity in today’s competitive retail landscape.

**1.3 Problem Statement**

Managing sales and store operations effectively has always been a critical challenge for retailers like Zudio. Before the implementation of machine learning, businesses relied on conventional methods such as manual record-keeping, static sales reports, and intuition-based decision-making. These approaches led to several inefficiencies, such as poor demand forecasting, ineffective inventory management, and suboptimal product placement. Retailers often faced stock shortages or excess inventory, leading to financial losses and dissatisfied customers.

**1.4 Significance**

The research presents a machine learning-driven framework that transforms traditional, intuition-based retail management practices at Zudio into a data-centric, automated decision-making process. By integrating Label Encoding, Exploratory Data Analysis (EDA), and predictive modeling (Linear Regression and Ridge Regression), the approach enhances forecasting accuracy, improves inventory efficiency, and empowers real-time business strategies.

* **Improved Sales Forecasting Accuracy**: Traditional manual methods and basic statistical approaches often fail to capture complex patterns in retail sales. The proposed ML-based Linear Regression model outperforms existing methods by delivering more accurate and interpretable sales predictions across product categories and seasons.
* **Data-Driven Store Optimization**: Leveraging sales, inventory, and store metadata, the system enables proactive stock replenishment and personalized merchandising strategies, minimizing lost sales due to stockouts and overstock issues.
* **Comprehensive Exploratory Data Analysis (EDA)**: Through countplots, heatmaps, and distribution visualizations, the research uncovers hidden correlations between sales, customer behavior, and product demand, offering actionable insights for store layout and category prioritization.
* **Automation & Operational Efficiency**: Unlike manual planning, which is prone to human bias and errors, the ML-driven pipeline automates data processing, prediction, and reporting—resulting in faster, smarter, and more consistent decision-making across all Zudio outlets.
* **Scalable Industry Application**: The proposed methodology is scalable across Zudio’s retail chain and adaptable to other fashion retail environments, paving the way for ML adoption in large-scale retail operations, enhancing both business growth and customer satisfaction.

**1.5 Applications**

* Sales Forecasting and Trend Analysis: Accurately predict future sales based on historical data, seasonal patterns, and product categories to support revenue planning and marketing strategies.
* Inventory Management Optimization: Automate stock replenishment decisions to prevent overstocking and stockouts, reducing waste and improving supply chain efficiency.
* Store Layout and Product Placement Strategy: Use sales and customer interaction data to optimize in-store product positioning, enhancing customer experience and increasing conversion rates.
* Customer-Centric Promotions and Personalization: Identify buying behavior patterns to design targeted promotions and personalized product recommendations that boost customer engagement.
* Performance Monitoring and Business Intelligence: Provide real-time dashboards and reports for managers to monitor store performance, identify underperforming SKUs, and make data-driven decisions for continuous improvement.

**CHAPTER 2**

**LITERATURE SURVEY**

Among the most important citing criteria considered when choosing the referenced papers were the relevance of the approached topic in the context of our research, the visibility of the respective article in the scientific literature (as highlighted by its number of citations), the actuality of the cited studies, considering the date of their publication. The purpose of the conducted literature review is to contextualize the results obtained within the research conducted in this paper with a view to identify and target a real need, a gap that exists in the current body of knowledge, which is being addressed by our study.

Within the scientific paper [**14**], Bandara et al. propose a prediction method based on a Long Short-Term Memory (LSTM) network in order to obtain reliable sales demand forecasts in e-commerce. The authors evaluate the forecasting framework on two datasets collected from an American multinational retail corporation that operates a chain of hypermarkets. The forecasting accuracy is evaluated through the means of a modified version of the Mean Absolute Percentage Error (mMAPE). Finally, the authors compare four variants of their developed approach to a series of univariate forecasting techniques, such as Exponentially Weighted Moving Average (EWMA), Exponential Smoothing (ETS), Autoregressive Integrated Moving Average (ARIMA), Prophet, Naïve, Naïve Seasonal, and therefore remark that their devised approach offers superior results.

Acknowledging the importance that an accurate sales prediction exerts in e-commerce, in the scientific article [**15**], Ji et al. develop a prediction method based on machine learning techniques. Therefore, they have proposed a three-stage eXtreme Gradient Boosting (XGBoost) forecasting model (entitled C-A-XGBoost). Using real datasets registered on a cross-border e-commerce platform, the authors have compared their obtained results to the ones provided by other models, such as ARIMA, XGBoost, a model based on the clustering and XGBoost (C-XGBoost), and a model that combines the ARIMA with XGBoost (A-XGBoost). Using the Mean Sum Error (ME), MSE, Root Mean Squared Error (RMSE) and the Mean Absolute Error (MAE) as performance metrics, the authors conclude that their developed model outperforms the other four ones.

Within the scientific paper [**5**], Kharfan et al. put forward a machine-learning approach designed to forecast the demand in the case of products for which historical data are not available and that have been recently launched on an e-commerce platform. The developed model consists of three steps that deal with clustering, classification and finally, prediction. Each of these three steps is developed making use of various techniques that are afterwards benchmarked in order to establish the best method. Therefore, the first step has employed the t-Distributed Stochastic Neighbor Embedding (t-SNE) algorithm and k-means clustering; in the second step, the authors have tested the classification trees, Random Forest (RF) and Support Vector Machine (SVM) in order to identify the best performing one; in the third step, the algorithms of regression trees, *k*-NN, linear regression (LR), RF and neural networks have been tested. The proposed methodology has been used on a real dataset provided by an e-commerce United States company specialized in clothing and footwear sales in order to obtain a demand forecasting for the newly launched products that do not have historical data available. The authors compute as performance metrics the Absolute Forecast Error (AFE), Absolute Forecast Percentage Error (APE), Weighted Mean Absolute Percentage Error (WMAPE), Forecast Percentage Error (PE), Weighted Mean Percentage Error (WMPE), and in this way, the authors have identified the best method that should be used in each step of the developed model.

In the paper [**6**], Pan et al. present a CNN approach useful in sales forecasting in the case of e-commerce activities, by mining specific data. The authors remark that in contrast to the traditional data mining techniques, the CNN approach is able to use in an efficient way large amounts of data and automatically extract features from time series. The validity of the developed approach is confirmed using a real dataset. The authors have made use of MSE as a performance metric and compared the performance registered by their approach to the one of other methods from the scientific literature, such as a CNN version in which the weights are adjusted, a single CNN method, an ARIMA algorithm, an algorithm from the scientific literature based on Gaussian Process Regression (GPR) along with Kernel Ridge Regression (KRR) and a Deep Neural Network (DNN) approach. Finally, the authors conclude that their developed approach offers an improved, more accurate sales forecast.

In the article [**13**], Weytjens et al. firstly remark that an accurate cash flow forecasting helps and improves the capital’s allocation for a wide range of firms, preventing other ones from running out of cash. Targeting e-commerce companies and other firms that have an increased number of customers and transactions, the authors have approached a subject regarding the forecasting of the “Accounts Receivable Cash Flows (ARCF)”. Starting from an analysis of existing forecasting methods, such as ARIMA or Prophet, the authors propose an approach based on neural networks, considering Multi-Layered Perceptron (MLP) and Long-Short-Term Memory (LSTM) ANNs. Using a dataset retrieved from a German company and based on computing as performance metrics the MSE and the “Interest Opportunity Cost (IOC)”, the authors compare ARIMA, Prophet, and the two DL approaches developed within their paper, concluding that the developed methods offer an increased level of flexibility and accuracy.

Acknowledging the importance of sales forecasting in managing e-commerce enterprises and in highlighting future sales trends, within the scientific paper [**16**], Liu et al. propose a model that uses historical sales and establishes common characteristics among them, by making use of a time series model. The developed approach predicts the sales inventory for a certain type of products. Afterwards, in order to improve the forecasting accuracy and the reliability of the developed model, the authors introduce external datasets and a qualitative analysis of data by means of a Hidden Markov Model (HMM). Using a real e-commerce dataset along with a meteorological dataset provided by the China Weather Network, the authors perform a qualitative analysis of the forecast based on an analysis of the predictive vectors of the HMM in order to confirm the usefulness of the Hidden Markov approach in attaining a qualitative prediction.

In a recent paper, Qi et al. propose a hybrid sales volume forecasting approach designed for e-commerce, based on “Long Short-Term Memory with Particle Swam Optimization (LSTM-PSO)” [**9**]. In order to obtain the most suitable configuration within the developed approach, the authors make use of a Particle Swam Optimization (PSO) metaheuristic method. Using real datasets provided by an e-commerce company, along with publicly available ones, the authors assess the effectiveness of the devised approach by comparing it with 9 other ones, namely LR, SVM for regression, MLP, M5P, RF, K-Nearest Neighbor (KNN), ARIMA, Transfer Learning (TL) and Recurrent Neural Network (RNN) models. The performance metrics computed and compared within the study are MAE, RMSE, Relative Absolute Error (RAE), Root Relative Squared Error (RRSE). The obtained results emphasize the fact that the developed approach attained a very good prediction accuracy, consequently proving its usefulness.

Shih et al. propose in [**1**] a hybrid forecasting approach for the goods demand in the case of an e-commerce company. The developed model integrates an LSTM approach with analyzing the customers’ feelings based on their provided feedback. Using a real dataset retrieved from a Chinese online shopping platform, the authors started their research by sorting the users’ comments in three categories, corresponding to the “positive”, “negative” or “trust” reactions. The LSTM network has been trained to forecast the future values starting from both the timeseries represented by the sales and the sentiment ratings arising from the users’ comments. By analyzing the obtained results, the authors remark that their study offers to the decision makers a useful, accurate tool for the situations where trading data are reduced (for example, in the case of short-term demand goods, where the volume of historical data is reduced and does not allow for establishing a cyclic variation).

Targeting the development of a ML method for the e-commerce demand forecasting, Salamai et al. propose an ensemble model that implements a continuous “Stochastic Fractal Search (SFS)” using the “Guided Whale Optimization Algorithm (WOA)” for optimizing the weights of “Bidirectional Recurrent Neural Networks (BRNNs)” [**2**]. The authors make use of a real dataset retrieved from a subsidiary of Google Limited Liability Company that contains the transactions by an e-commerce retailer from the United Kingdom (UK). By computing the RMSE, MAE and Mean Bias Error (MBE) performance metrics, the authors compare their developed approach with primary methods such as Bidirectional Recurrent Neural Network (BRNN), MLP and Support Vector Regression (SVR) along with other state-of-the-art algorithms, namely PSO, WOA and Genetic Algorithm (GA). The developed benchmark highlights the improved accuracy of the developed approach.

Within the scientific paper [**3**], Li et al. first acknowledge the importance of obtaining an accurate e-commerce demand forecast and, to this purpose, the authors propose a method based on Convolutional LSTM (ConvLSTM) and Horizontal Federated (HF) learning, entitled HF-ConvLSTM. The authors benchmark the developed approach on real datasets, comparing a series of forecasting methods (LSTM, BiLSTM, ConvLSTM) with the same three above-mentioned algorithms, in the HF framework (HF-LSTM, HF-BiLSTM, the proposed HF-ConvLSTM). The obtained results have highlighted that the developed model outperforms the other tested ones in what concerns the values of the MAE, MAPE and Bullwhip Effect (BWE) performance metrics, therefore being able to provide accurate, stable forecasting results and at the same time helps to avoid potential security issues for supply companies that use large databases

**CHAPTER 3**

**TRADITIONAL SYSTEM**

**3.1 Sales Record Keeping Using Spreadsheets**

In traditional retail environments, including Zudio, store sales were often tracked using spreadsheet-based systems like Microsoft Excel. Store managers manually recorded daily sales, categorized by product type, date, or cashier. These sheets were then aggregated at the end of the week or month to generate sales summaries for higher-level decision-making. Data entry was done manually at the store level, usually without any automation or validation checks.  
While spreadsheets offer basic functionalities like summation, filtering, and charting, they are highly prone to human error. Any mistake in manual data entry or formula input can distort the final results. Furthermore, consolidating data from multiple stores across locations becomes time-consuming and complex, especially when formats vary between branches.  
This system lacked the ability to identify trends or predict future sales. Decisions such as restocking or promotions were made reactively based on static historical summaries rather than real-time insights. Without integration with customer preferences or seasonal factors, this method was inefficient and unsustainable in a fast-moving retail environment.

**3.2 Intuition-Based Inventory Planning**

Inventory planning was historically conducted using store managers' experience and intuition rather than quantitative analysis. Decisions regarding how much stock to order, which items to prioritize, or when to restock were largely influenced by past experiences and gut feelings rather than hard data.

While experienced managers sometimes made effective predictions, this approach often led to mismatches between actual and anticipated demand. Factors like regional trends, festive seasons, or promotional campaigns were inconsistently accounted for, resulting in overstocking of slow-moving products or understocking of popular items.  
This intuition-driven model lacked adaptability to new market trends and changing customer behavior. It also hindered scalability—what worked for one store couldn’t be generalized across different locations with unique consumer patterns. The absence of data-backed forecasts led to frequent stockouts, missed sales opportunities, and increased warehousing costs.

### **3.3 Static Report-Based Store Performance Monitoring**

### Zudio’s store performance evaluation was previously done through manually prepared static reports. These included weekly or monthly summaries that focused on revenue, item-wise sales, and staff efficiency. These reports were compiled at the store level and submitted to regional managers for analysis and review.

### Due to their retrospective nature, these reports were often outdated by the time they reached decision-makers. They failed to capture real-time fluctuations, customer behavior, or operational issues that required immediate attention. Store-level anomalies such as sudden drops in footfall or surges in returns went unnoticed until much later.

### Moreover, static reports lacked dynamic insights. They provided no interactivity or deep-dive capabilities into why certain SKUs underperformed or how promotional strategies impacted sales. Without visual analytics or predictive modeling, they offered limited value in guiding future strategies or optimizing performance.

### **3.4 Overall Drawbacks of Manual Systems**

* **High Error Rate and Inconsistency**: Manual data entry leads to frequent inaccuracies and lacks standardization across stores.
* **Inefficiency and Time-Consumption**: Processes like inventory tracking, reporting, and analysis are slow, repetitive, and resource-intensive.
* **Lack of Real-Time Insights**: Manual systems offer delayed data visibility, preventing timely decision-making in a competitive retail space.
* **Inability to Scale**: As Zudio expands, these systems become unmanageable due to the complexity and volume of data.
* **Poor Predictive Capability**: These systems are reactive and descriptive at best, lacking forecasting, personalization, and automation capabilities that ML-based models offer.

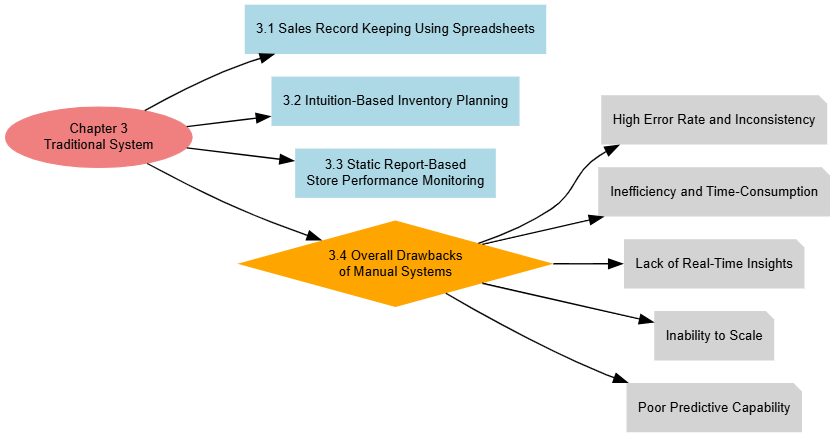
****

Figure 3.1: Diagram for traditional system

**CHAPTER 4**

**PROPOSED SYSTEM**

**4.1 Overview**

The proposed algorithm introduces a novel hybridized approach that integrates **Label Encoding, Exploratory Data Analysis (EDA)** using advanced visualizations, and a dual-model regression framework—**Ridge Regression for stability** and **Linear Regression for interpretability and efficiency**. This combinatorial method, which blends both visual analytics and predictive modeling, is **not presented in existing surveys** and addresses the core limitations of traditional retail systems by enabling data-driven, real-time, and scalable solutions for sales forecasting and store management. Unlike previous studies that rely solely on either complex ensemble models or basic statistical methods, this integrated approach ensures model transparency, minimal overfitting, and operational usability—making it specifically suited for fast-paced retail ecosystems like Zudio.

### **Procedure of Proposed Methodology**

**1. Data Collection and Preprocessing:**

The first step involves gathering historical sales data from Zudio stores, including product IDs, categories, timestamps, sales quantities, and stock levels. The dataset may also incorporate external features like seasonal indicators and promotional events. Categorical features such as product category or location are encoded using **Label Encoding**, ensuring they are suitable for regression models without introducing artificial ordering, unlike One-Hot Encoding.

**2. Exploratory Data Analysis (EDA):**

Before modeling, EDA is performed to understand the structure and distribution of the data. Visualizations such as **countplots** help identify the frequency of product sales, **heatmaps** reveal correlation between features, and **distribution/KDE plots** highlight variability in sales performance across stores and time. This step uncovers hidden trends, outliers, and helps in selecting the most influential features for modeling.

**3. Data Splitting and Feature Selection:**

The cleaned dataset is then split into **training and testing sets** (commonly in an 80:20 or 70:30 ratio). Feature selection is done based on the results from correlation matrices and domain relevance—only the most impactful variables are retained to reduce noise and improve model generalization.

**4. Regression Modeling – Dual Framework:**

Two regression models are applied and compared. **Ridge Regression** is used first to handle potential multicollinearity and to prevent overfitting through L2 regularization. Following this, **Linear Regression** is proposed as the final model due to its high interpretability and consistent performance in this specific structured dataset. The comparison ensures that the final choice balances accuracy, simplicity, and explainability—essential for business stakeholders.

**5. Evaluation and Business Insight Generation:**

Model performance is evaluated using metrics like **R² score**, **MAE**, and **RMSE**. Linear Regression not only achieves competitive accuracy but also allows for clear interpretation of coefficients, helping Zudio managers understand the exact impact of each variable (e.g., promotions, seasonality) on sales. Insights from the model are then used to make strategic decisions like optimized stock orders, seasonal campaigns, and store layout adjustments.

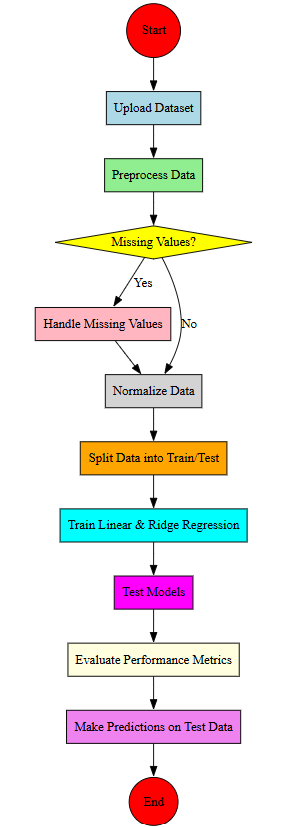


Fig. 1: Architectural Block Diagram.

**4.2 Dataset Preprocessing**

The preprocessing method ensures that the Zudio sales dataset is clean, consistent, and primed for training machine learning models. It handles null values, converts categorical data into numerical form, separates features from the target variable, and partitions the dataset into training and testing subsets. While more advanced techniques like feature scaling or outlier detection could further boost performance, the current pipeline sets a robust foundation for accurate prediction and strategic retail insights.

### **Step-1: Handling Missing Values**

The preprocessing phase starts with identifying and handling missing values, which can distort sales predictions and compromise model integrity. Missing entries may result from manual data entry errors or data extraction issues from retail systems. In this workflow, the dataset is checked for null values using the isnull().sum() method to understand which columns require attention. Although the dataset may not have missing values in this case, identifying them early ensures that decisions like imputation or row deletion can be applied systematically. For large-scale deployments in Zudio’s stores, ensuring clean data from POS systems and databases will be essential.

### **Step-2: Encoding Categorical Variables**

Zudio’s dataset includes several categorical features like ‘Category’, ‘Store Manager’, and ‘Security Features’ which need conversion into numerical format for model compatibility. The Label Encoding technique is applied to these object-type columns, assigning a unique numeric value to each category. This is particularly effective when the number of unique values is limited. However, since Label Encoding can unintentionally introduce ordinal relationships where none exist (e.g., assigning 0, 1, 2 to unrelated categories), alternatives like One-Hot Encoding may be more suitable in future iterations, especially when the model shows signs of misinterpreting categorical weight.

### **Step-3: Feature and Target Variable Separation**

Following encoding, the dataset is split into input features (X) and the target variable (y), which is the ‘Sales Profit’ column. This separation ensures that the models learn from the independent variables—like inventory levels, product category, or store attributes—without inadvertently using the sales outcome as part of the input. It also helps modularize the model pipeline, where feature engineering and transformations can be applied solely on the input features, and performance metrics can be independently assessed on the predicted versus actual sales profit values.

### **Step-4: Dataset Overview and Correlation Analysis**

Before model training, the dataset is reviewed using descriptive statistics and correlation matrices. The .describe() function provides insights into distribution, central tendencies, and outliers, while the correlation heatmap visually highlights relationships between features. For example, features showing strong correlation with ‘Sales Profit’ can be emphasized during model training, while redundant features may be dropped to reduce dimensionality. Though not explicitly normalized in the current approach, this overview paves the way for future enhancements using scaling techniques like StandardScaler to address skewed feature distributions or unbalanced value ranges.

### **Step-5: Splitting the Dataset into Training and Testing Sets**

To assess model accuracy and generalizability, the dataset is divided into training and testing subsets. In this project, 80% of the data is used for training the model, where it learns patterns from historical sales data, while the remaining 20% is used to evaluate the model’s performance on unseen inputs. This train-test split ensures a reliable validation strategy, preventing overfitting (where the model memorizes training data) or underfitting (where it fails to capture meaningful patterns). The model's real-world applicability is thus tested in a simulated scenario, closely mimicking the continuous inflow of new data in retail operations at Zudio stores.

**4.3 EDA**

Exploratory Data Analysis (EDA) focuses on understanding the relationships, distributions, and class frequencies within the dataset. The use of visual techniques such as count plots and heatmaps enables deeper insights into categorical distributions and numerical feature correlations. This approach helps identify patterns that can inform feature selection, highlight data imbalances, and expose potential redundancies or important associations. The visual tools employed are critical for assessing data quality and shaping the direction for machine learning model development.

**Step-1: Correlation Matrix Calculation**

The EDA begins by computing the correlation matrix of the dataset. This matrix measures the linear relationship between numerical features, with correlation coefficients ranging from -1 to 1. A value closer to 1 indicates a strong positive correlation, while values near -1 suggest a strong negative relationship. This step is essential for identifying multicollinearity, where highly correlated features might provide redundant information, which could affect model performance or interpretation. Understanding these relationships is critical for informed feature selection and dimensionality reduction.

**Step-2: Count Plot for ‘Security Features’**

A count plot is created for the ‘Security Features’ column, which likely represents different levels or types of security measures associated with each data entry. This plot shows the frequency of each unique category within the column, helping to detect class imbalances or dominant features. For instance, if one security feature is overwhelmingly more common than others, it might bias the model or require balancing techniques during preprocessing. Visualizing these counts provides clarity on the categorical composition of the dataset.

**Step-3: Count Plot for ‘Category’**

Another count plot is generated to display the distribution of data across different categories, likely representing classes or group labels (such as product types, user segments, or risk levels). This plot plays a crucial role in evaluating whether the dataset is balanced across categories. An imbalanced class distribution can lead to biased models that perform poorly on underrepresented classes. Recognizing such imbalance early allows for strategies like resampling, synthetic data generation, or using weighted loss functions during training.

**Step-4: Correlation Heatmap**

A heatmap visualization is used to display the correlation matrix with enhanced readability. The heatmap includes annotations for each correlation value and utilizes a color palette (‘coolwarm’) to visually distinguish high and low correlations. This tool is effective for quickly spotting feature interdependencies, clusters of correlated variables, or outliers. It supports decisions on feature engineering and selection, especially when aiming to reduce noise or avoid multicollinearity in model input features.

**4.4 Model Building and Training**

**4.4.1 Ridge Regression (Existing algorithm)**

**Step 1: Model Initialization with Regularization**

Ridge Regression begins by initializing a linear regression model that includes a regularization component. Unlike traditional linear regression, Ridge adds a penalty to the model coefficients to prevent overfitting. When the model is provided with training data (X\_train and y\_train), it doesn't just aim to minimize the prediction error—it also controls the size of the coefficients to ensure that no single feature dominates the model due to scale or noise.

**Step 2: Model Training on X\_train and y\_train**

Once the model is initialized, the training process involves fitting the Ridge model to the provided training data. During this phase, Ridge tries to find the best set of coefficients that balance two goals: accurately fitting the training data and keeping the model's complexity (measured by the size of the coefficients) low. The regularization strength (often referred to as alpha) controls how much penalty is applied—higher values mean more shrinkage of coefficients.

**Step 3: Coefficient Shrinkage and Bias-Variance Tradeoff**

As the training progresses, Ridge penalizes large coefficients, which helps in reducing model variance and preventing overfitting. This is especially useful when dealing with multicollinearity or high-dimensional datasets. However, it can introduce some bias into the model. The balance achieved through Ridge ensures that while the model may not perfectly fit the training data, it generalizes better on unseen data such as X\_test.

**Step 4: Prediction Using X\_test**

Once training is complete, the Ridge Regression model is used to make predictions on the test set X\_test. It uses the learned, regularized coefficients to produce predicted values that are expected to be more robust and less sensitive to fluctuations or noise in the test data.

**Step 5: Model Evaluation Against y\_test**

The predicted output is then compared to the actual target values in y\_test. Metrics such as Mean Squared Error (MSE) or R² score are typically used to evaluate the performance of the Ridge Regression model. The objective is to see how well the model, trained with regularization, performs in predicting real-world, unseen data, and whether it successfully avoids overfitting while still capturing the underlying patterns.

**4.4.2 Limitations of Ridge Regression**

* **Does not perform feature selection**: Ridge reduces coefficient magnitudes but does not eliminate features; all features remain in the model, potentially including irrelevant ones.
* **Sensitive to regularization parameter**: The model's performance heavily depends on the choice of the regularization strength (alpha), which must be carefully tuned.
* **Not effective with non-linear relationships**: Ridge Regression assumes a linear relationship between features and target, which limits its use on datasets with complex, non-linear patterns.
* **Can underperform with highly sparse data**: In cases where only a few features are important, Ridge may shrink all coefficients unnecessarily instead of removing unimportant ones entirely (unlike Lasso).
* **Affected by feature scaling**: Ridge Regression is sensitive to the scale of input features; unscaled features can bias the penalty and lead to suboptimal coefficient estimates.

**4.4.3 Linear Regression (Proposed Algorithm)**

**Step 1: Model Initialization**

Linear Regression begins by assuming a linear relationship between the independent features (X\_train) and the target variable (y\_train). The goal is to find a straight-line equation (a linear function) that best fits the data. The model starts with initial estimates for the slope (coefficients) and intercept, which are then optimized during training to minimize the prediction error.

**Step 2: Model Training on X\_train and y\_train**

Using the training data, the Linear Regression model calculates the best-fitting line by minimizing the difference between the predicted outputs and the actual values in y\_train. This process is known as minimizing the residuals or errors. The model updates its coefficients so that the overall prediction error across the training data is as small as possible.

**Step 3: Pattern Learning and Coefficient Estimation**

During training, the model learns how much each feature in X\_train contributes to the final output. These contributions are represented as coefficients. If a feature has a higher coefficient, it has a larger impact on the prediction. The model treats this as a learning phase, where it captures the relationship pattern that maps inputs (X\_train) to the output (y\_train).

**Step 4: Making Predictions on X\_test**

Once the model is trained, it uses the learned coefficients and intercept to make predictions on new, unseen data in X\_test. Each row in X\_test is multiplied by the learned coefficients, and the intercept is added to compute the predicted output. This results in a series of predicted values that correspond to the inputs in the test dataset.

**Step 5: Evaluating Predictions with y\_test**

The final step involves evaluating how well the model performs on unseen data by comparing its predictions with the actual values in y\_test. Common evaluation metrics include Mean Absolute Error (MAE), Mean Squared Error (MSE), and R² score. This step assesses the model's accuracy and generalization ability, helping to judge its usefulness for real-world predictions.

**4.4.4 Advantages of Linear Regression**

* **Simple and Easy to Interpret**: Linear Regression is straightforward, making it easy to understand the relationship between variables and communicate results.
* **Fast to Train**: It has low computational requirements and can be trained quickly, even on large datasets.
* **Good Baseline Model**: It serves as a strong starting point before moving to more complex models; many real-world problems follow linear patterns.
* **Efficient with Linearly Separable Data**: When the underlying relationship is linear, Linear Regression provides highly accurate results.
* **Supports Analytical Solutions**: Unlike many complex models, Linear Regression can be solved analytically, offering closed-form solutions for parameter estimation.

**CHAPTER 5**

**UML DIAGRAMS**

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software researchs.

**GOALS:** The Primary goals in the design of the UML are as follows:

* Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
* Provide extendibility and specialization mechanisms to extend the core concepts.
* Be independent of particular programming languages and development process.
* Provide a formal basis for understanding the modeling language.
* Encourage the growth of OO tools market.
* Support higher level development concepts such as collaborations, frameworks, patterns and components.
* Integrate best practices.

**Class diagram**

The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an "is-a" or "has-a" relationship. Each class in the class diagram was capable of providing certain functionalities. These functionalities provided by the class are termed "methods" of the class. Apart from this, each class have certain "attributes" that uniquely identify the class.

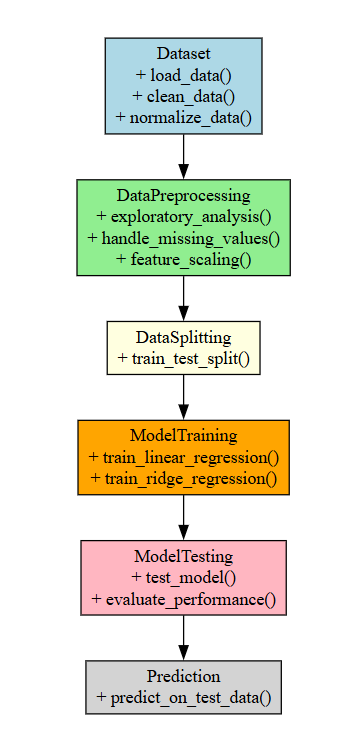


Fig. 5.1: Class Diagram

**Activity diagram**

Activity diagrams are graphical representations of Workflows of stepwise activities and actions with support for choice, iteration, and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

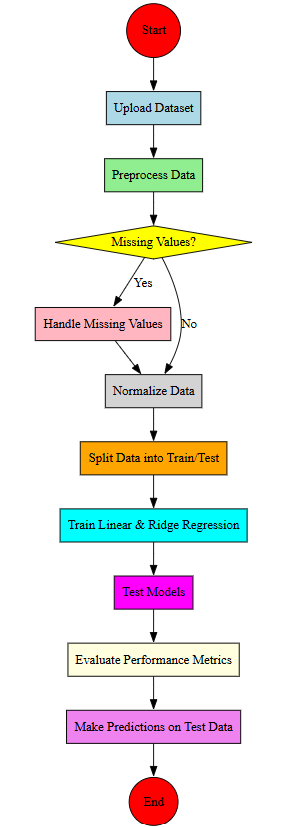


Fig. 5.2: Activity Diagram

**Use Case diagram:**

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

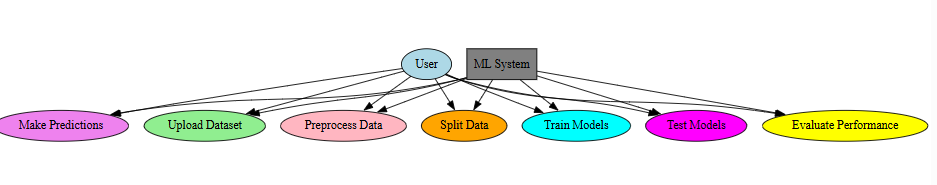


Fig. 5.3: Use Case Diagram

**Deployment Diagram:**

A deployment diagram in UML illustrates the physical arrangement of hardware and software components in the system. It visualizes how different software artifacts, such as data processing scripts and model training components, are deployed across hardware nodes and interact with each other, providing insight into the system’s infrastructure and deployment strategy.

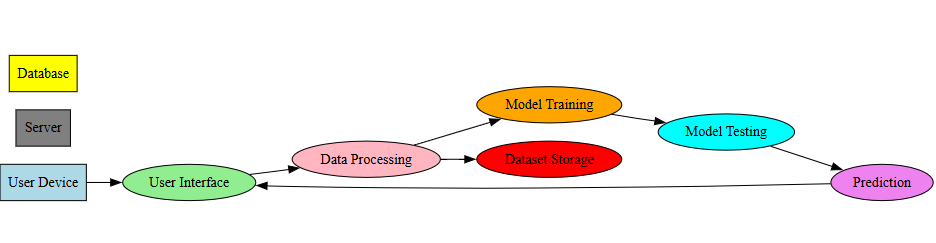


Fig. 5.4: Deployment Diagram

**Architectural Block Diagram**

An architectural block diagram offers a high-level view of a system’s structure, showcasing the main components and their interactions. It represents how major modules, such as data sources, processing units, and evaluation components, are organized and how they communicate with each other to accomplish the system’s objectives. This diagram helps in understanding the overall design and flow of the system.

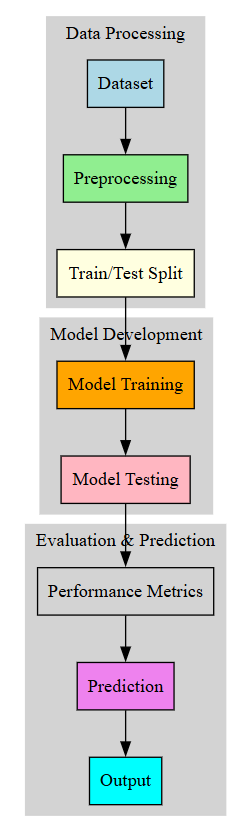


Fig. 5.5: Architectural Block Diagram

**CHAPTER 6**

**SOFTWARE ENVIRONMENT**

**6.1 Software Requirements**

Python is a high-level, interpreted programming language known for its simplicity and readability, which makes it a popular choice for beginners as well as experienced developers. Key features of Python include its dynamic typing, automatic memory management, and a rich standard library that supports a wide range of applications from web development to data science and machine learning. Its object-oriented approach and support for multiple programming paradigms allow developers to write clear, maintainable code. Python's extensive ecosystem of third-party packages further enhances its capabilities, enabling rapid development and prototyping across diverse fields.

**Installation**

First, download the appropriate installer from the official Python website (https://www.python.org/downloads/release/python-376/). For Windows users, run the executable installer and ensure to check the "Add Python to PATH" option during installation; for macOS and Linux, follow the respective package installation commands or use a package manager like Homebrew or apt-get. After installation, verify the setup by running python --version or python3 --version in your terminal or command prompt, which should display "Python 3.7.6." This version-specific installation supports all major functionalities and libraries compatible with Python 3.7.6, making it an excellent foundation for developing robust applications in areas such as data analysis, machine learning, and GUI development.

**6.1.1 Python Packages**

The project requires a robust set of software libraries and tools that work together to build an integrated system for plant disease classification. Below is an explanation of the key software requirements and the packages used:

* **Python:** The project is implemented in Python, which is chosen for its extensive ecosystem of libraries and its strong support for data analysis, machine learning, and GUI development.
* **Tkinter:** Used to build the graphical user interface (GUI) of the application. It handles tasks such as user authentication, data upload, and displaying results, making the system accessible to both admins and end-users.
* **PIL (Pillow):** Utilized for image processing, particularly for handling background images and other graphical elements within the GUI, thereby enhancing the visual appeal of the application.
* **Matplotlib & Seaborn:** These libraries are employed for data visualization. Matplotlib is used for creating standard plots, while Seaborn adds an extra layer of sophistication for statistical visualizations such as bar plots, violin plots, histograms, scatter plots, strip plots, and correlation heat maps.
* **Pandas & NumPy:** Essential for data manipulation and analysis. Pandas is used to load, preprocess, and analyze the CSV dataset, while NumPy supports numerical operations and data handling, which are crucial for processing large volumes of IoT data.
* **Scikit-learn (sklearn):** Provides the machine learning framework used in the project. It includes tools for model training, evaluation, train-test splitting, and data preprocessing (like label encoding). Models such as Gaussian Naive Bayes, SVM, KNN, and Decision Tree Classifier are implemented using scikit-learn.
* **Imbalanced-learn (imblearn):** Specifically used for implementing the SMOTE (Synthetic Minority Oversampling Technique) algorithm, which helps in addressing class imbalance in the dataset by generating synthetic samples for under-represented classes.
* **Joblib:** Utilized for saving and loading trained machine learning models. This ensures that once a model is trained, it can be stored and reused without retraining, thereby improving efficiency.
* **PyMySQL:** This package provides a means to connect to a MySQL database for handling user authentication. It facilitates operations such as user signup, login, and data storage, ensuring secure and persistent management of user credentials.

Each of these packages plays a crucial role in ensuring that the system is robust, scalable, and efficient—from data ingestion and preprocessing to model training, visualization, and deployment. The combination of these tools enables the creation of an integrated, user-friendly application for real-time plant disease classification and management.

**6.2 Hardware Requirements**

Python 3.7.6 can run efficiently on most modern systems with minimal hardware requirements. However, meeting the recommended specifications ensures better performance, especially for developers handling large-scale applications or computationally intensive tasks. By ensuring compatibility with hardware and operating system, can leverage the full potential of Python 3.7.6.

**Processor (CPU) Requirements:** Python 3.7.6 is a lightweight programming language that can run on various processors, making it highly versatile. However, for optimal performance, the following processor specifications are recommended:

* **Minimum Requirement**: 1 GHz single-core processor.
* **Recommended**: Dual-core or quad-core processors with a clock speed of 2 GHz or higher. Using a multi-core processor allows Python applications, particularly those involving multithreading or multiprocessing, to execute more efficiently.

**Memory (RAM) Requirements:** Python 3.7.6 does not demand excessive memory but requires adequate RAM for smooth performance, particularly for running resource-intensive applications such as data processing, machine learning, or web development.

* **Minimum Requirement**: 512 MB of RAM.
* **Recommended**: 4 GB or higher for general usage. For data-intensive operations, 8 GB or more is advisable.

Insufficient RAM can cause delays or crashes when handling large datasets or executing computationally heavy programs.

**Storage Requirements:** Python 3.7.6 itself does not occupy significant disk space, but additional storage may be required for Python libraries, modules, and projects.

* **Minimum Requirement**: 200 MB of free disk space for installation.
* **Recommended**: At least 1 GB of free disk space to accommodate libraries and dependencies.

Developers using Python for large-scale projects or data science should allocate more storage to manage virtual environments, datasets, and frameworks like TensorFlow or PyTorch.

**Compatibility with Operating Systems:** Python 3.7.6 is compatible with most operating systems but requires hardware that supports the respective OS. Below are general requirements for supported operating systems:

* **Windows**: 32-bit and 64-bit systems, Windows 7 or later.
* **macOS**: macOS 10.9 or later.
* **Linux**: Supports a wide range of distributions, including Ubuntu, CentOS, and Fedora.

The hardware specifications for the OS directly impact Python’s performance, particularly for modern software development.

**CHAPTER 7**

**FUNCTIONAL REQUIREMENTS**

### **The functional requirements for the ML-Driven Insights of Zudio Sales and Store Management project define the specific actions that the system must be able to perform. These include:**

**FR1: Data Loading and Initial Exploration**

* Specify the requirement to load sales data from a CSV file (e.g., 'Datasets/Zudio\_sales\_data.csv') using pandas for analysis.
* Enable initial exploration of the dataset, including displaying the first few rows, checking data types, dimensions, and basic statistical summaries.
* Identify missing values and duplicates to ensure data quality before preprocessing.

**FR2: Label Encoder Initialization for Categorical Data Transformation**

* Define the need for a LabelEncoder to transform categorical labels (e.g., strings or non-numeric data) into numeric values for machine learning compatibility.
* Apply LabelEncoder to all columns with object data types in the dataset.
* Ensure consistent encoding for training and test datasets to maintain model compatibility.

**FR3: Data Preprocessing and Feature Engineering**

* Require the separation of features (X) by dropping the target column 'Sales Profit' from the dataset.
* Define the target variable (y) as the 'Sales Profit' column for prediction tasks.
* Implement train-test split with a 20% test size and a random state for reproducibility.

**FR4: Model Training and Persistence**

* Specify the use of Ridge and Linear Regression models from scikit-learn for predicting sales profit.
* Check for the existence of pre-trained models (e.g., 'model/Ridge.pkl', 'model/Linear.pkl') using os.path.exists.
* If models exist, load them with joblib; otherwise, train new models on the training data and save them for future use.

**FR5: Model Evaluation and Metrics Calculation**

* Define a function to calculate regression metrics: Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared (R²).
* Store metrics in lists for potential comparison across models.
* Visualize model performance with a scatter plot of true vs. predicted values, including a line of equality for reference.

**FR6: Visualization of Data and Results**

* Require seaborn and matplotlib for generating visualizations to understand data distributions and relationships.
* Create count plots for categorical features like 'Security Features' and 'Category' to analyze frequency.
* Generate a correlation heatmap to visualize relationships between numerical features, aiding in feature importance insights.

**FR7: Test Data Processing and Prediction**

* Load test data from a CSV file (e.g., 'Datasets/test.csv') for prediction using the trained models.
* Apply the same LabelEncoder transformation to categorical columns in the test dataset for consistency.
* Use the trained Linear Regression model to predict 'Sales Profit' and append predictions to the test dataset.

**CHAPTER 8**

**SOURCE CODE**

# ML-Driven Insights of Zudio Sales and Store Management for Business Growth

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

from sklearn.preprocessing import LabelEncoder

# Ignore all warnings

warnings.filterwarnings('ignore')

import os

from sklearn.linear\_model import Ridge, LinearRegression

from sklearn.preprocessing import StandardScaler

import joblib

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error,r2\_score

from sklearn.model\_selection import train\_test\_split

df=pd.read\_csv(r'Datasets/Zudio\_sales\_data.csv')

df.head()

df['Store Manager']

df.size

df.shape

df.describe()

df.dtypes

df.columns

df['Sales Profit']

df.isnull().sum()

df.duplicated().sum()

df.corr()

plt.figure(figsize = (16,10))

sns.countplot(x = df['Security Features'])

sns.countplot(x = df['Category'])

plt.figure(figsize = (16,10))

sns.heatmap(df.corr(),annot = True,cmap = 'coolwarm')

# Assuming df is your DataFrame

# Identify all columns with object data type

object\_cols = df.select\_dtypes(include=['object']).columns

# Initialize LabelEncoder

label\_encoder = LabelEncoder()

# Apply LabelEncoder to each object type column

for col in object\_cols:

df[col] = label\_encoder.fit\_transform(df[col].astype(str))

X = df.drop(['Sales Profit'],axis = 1) #independent features

X

X.info()

y = df['Sales Profit']

y

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Check the shape of the splits

print("X\_train shape:", X\_train.shape)

print("X\_test shape:", X\_test.shape)

print("y\_train shape:", y\_train.shape)

print("y\_test shape:", y\_test.shape)

mae\_list = []

mse\_list = []

rmse\_list = []

r2\_list = []

def calculateMetrics(algorithm,predict, testY):

# Regression metrics

mae = mean\_absolute\_error(testY, predict)

mse = mean\_squared\_error(testY, predict)

rmse = np.sqrt(mse)

r2 = r2\_score(testY, predict)

mae\_list.append(mae)

mse\_list.append(mse)

rmse\_list.append(rmse)

r2\_list.append(r2)

print(f"{algorithm} Mean Absolute Error (MAE): {mae:.2f}")

print(f"{algorithm} Mean Squared Error (MSE): {mse:.2f}")

print(f"{algorithm} Root Mean Squared Error (RMSE): {rmse:.2f}")

print(f"{algorithm} R-squared (R²): {r2:.2f}")

plt.figure(figsize=(10, 6))

sns.scatterplot(x=testY, y=predict, alpha=0.6)

plt.plot([min(testY), max(testY)], [min(testY), max(testY)], 'r--', lw=2) # Line of equality

plt.xlabel('True Values')

plt.ylabel('Predictions')

plt.title(algorithm)

plt.grid(True)

plt.show()

if os.path.exists('model/Ridge.pkl'):

# Load the trained model from the file

rge = joblib.load('model/Ridge.pkl')

print("Model loaded successfully.")

predict = rge.predict(X\_test)

calculateMetrics("Ridge", predict, y\_test)

else:

# Train the model (assuming X\_train and y\_train are defined)

rge = Ridge(tol=0.01)

rge.fit(X\_train, y\_train)

# Save the trained model to a file

joblib.dump(rge, 'model/Ridge.pkl')

print("Model saved successfully.")

predict = rge.predict(X\_test)

calculateMetrics("Ridge", predict, y\_test)

if os.path.exists('model/Linear.pkl'):

# Load the trained model from the file

Linear = joblib.load('model/Linear.pkl')

print("Model loaded successfully.")

predict = Linear.predict(X\_test)

calculateMetrics("Linear Regresson", predict, y\_test)

else:

# Train the model (assuming X\_train and y\_train are defined)

Linear = LinearRegression()

Linear.fit(X\_train, y\_train)

# Save the trained model to a file

joblib.dump(Linear, 'model/Linear.pkl')

print("Model saved successfully.")

predict = Linear.predict(X\_test)

calculateMetrics("Linear Regresson", predict, y\_test)

test = pd.read\_csv('Datasets/test.csv')

test

object\_cols = test.select\_dtypes(include=['object']).columns

# Initialize LabelEncoder

label\_encoder = LabelEncoder()

# Apply LabelEncoder to each object type column

for col in object\_cols:

test[col] = label\_encoder.fit\_transform(test[col].astype(str))

predict = Linear.predict(test)

predict

test['Predict'] = predict

test

**CHAPTER 9**

**RESULTS AND DISCUSSION**

**9.1 Implementation Description**

**1. Data Loading and Initial Exploration**

**Function name :** load\_and\_explore\_data

**Implementation summary:**

* Reads the CSV file into a DataFrame.
* Displays the first few rows, size, shape, data types, and column names.
* Checks for null values and duplicates.
* Generates basic statistics and computes correlation matrix.
* Plots countplots for categorical features and a correlation heatmap for numeric features.

**2. Encode Categorical Columns**

**Function name:** encode\_categorical\_columns(df)

**Implementation summary:**

* Identifies columns with object (text) datatype.
* Applies LabelEncoder to each object column converting categories to numeric codes.
* Returns the DataFrame with encoded columns.

**3. Split Dataset into Features and Target**

**Function name:** prepare\_features\_target(df, target\_col='Sales Profit')

**Implementation summary:**

* Separates the target column (Sales Profit) from the rest of the DataFrame.
* Returns feature set X (all columns except target) and target variable y.

**4. Train/Test Split**

**Function name:** split\_data(X, y, test\_size=0.2, random\_state=42)

**Implementation summary:**

* Uses train\_test\_split from sklearn to split X and y into training and testing subsets.
* Returns X\_train, X\_test, y\_train, and y\_test.

**5. Calculate and Display Metrics**

**Function name:** calculate\_metrics(algorithm\_name, predictions, true\_values)

**Implementation summary:**

* Calculates MAE, MSE, RMSE, and R² between true and predicted values.
* Prints the metrics with formatting.
* Plots scatterplot comparing true values and predictions with a reference line showing perfect prediction.

**6. Train or Load Model**

**Function name:** train\_or\_load\_model(model\_class, model\_path, X\_train, y\_train, tol=None)

**Implementation summary:**

* Checks if a saved model file exists at model\_path.
* If exists, loads model using joblib.load().
* If not, initializes the model (optionally with tol for Ridge), trains it on X\_train, y\_train.
* Saves the trained model using joblib.dump().
* Returns the model instance.

**7. Predict on New Data**

**Function name:** predict\_new\_data(model, new\_data\_path)

**Implementation summary:**

* Loads new data CSV into DataFrame.
* Encodes categorical columns with LabelEncoder.
* Uses the passed model to predict the target for new data.
* Adds predictions as a new column to the DataFrame and returns it.

**9.2 Dataset Description**

1. **Store**  
   Represents the name or unique identifier of each Zudio retail outlet. This field helps distinguish between different locations in the sales network. It is essential for evaluating store-specific performance trends, regional strategies, and stock distribution.
2. **Country**  
   Indicates the country in which the store is located. This attribute supports geographical segmentation and helps analyze how national events or holidays influence sales patterns and customer behavior.
3. **State**  
   Specifies the state or administrative region of each store. It is useful for identifying regional trends, regulatory impacts, and demographic influences on store performance across different territories.
4. **City**  
   Denotes the city where the Zudio store operates. City-level granularity allows for urban demand forecasting, local marketing strategies, and comparing performance between metropolitan and non-metropolitan areas.
5. **Category**  
   Describes the general product grouping, such as men's wear, women's wear, or kids’ fashion. This feature is vital for understanding category-specific demand, seasonal trends, and inventory planning.
6. **Clothing Type**

Refers to the specific type of clothing product sold, such as shirts, trousers, dresses, or jackets. It enables detailed sales analysis at the product level and supports targeted promotional activities.

1. **Store Number**

A numeric identifier uniquely assigned to each store location. It ensures data consistency and indexing across large datasets and is often used for internal referencing and reporting.

1. **Postal Code**

Provides the zip or postal code of the store’s location. This helps in mapping customer demographics, regional logistics, and demand concentration across delivery zones.

1. **Store Type**

Defines the format or scale of the store, such as flagship, outlet, or express. It supports evaluation of performance based on store size, target audience, and operational capabilities.

1. **Store Open Date**

Captures the date when the store began operations. This column is important for tracking store maturity, assessing ramp-up periods, and analyzing the relationship between store age and performance.

1. **Security Features**

Represents the level or type of security systems installed at each store, such as surveillance cameras or RFID. It contributes to analyzing operational risk factors and their potential effect on loss prevention and profit margins.

1. **Order ID**

A unique identifier for each customer order. It is essential for transaction-level analysis, identifying purchase patterns, and maintaining accurate records for financial and inventory auditing.

1. **Order Date**

The exact date when the order was placed. This helps in tracking sales trends over time, aligning with seasonal campaigns, and conducting time-series analysis.

1. **Month**  
   Indicates the calendar month derived from the order date. It is used for monthly trend analysis, seasonality detection, and performance comparisons across different months.
2. **Customer ID**

A unique identifier assigned to each customer. It helps in tracking customer behavior, loyalty analysis, and segmentation for personalized marketing strategies.

1. **Customer Name**

Contains the name of the customer who placed the order. While primarily used for record-keeping, it can also aid in linking qualitative customer feedback to sales data.

1. **Product ID**

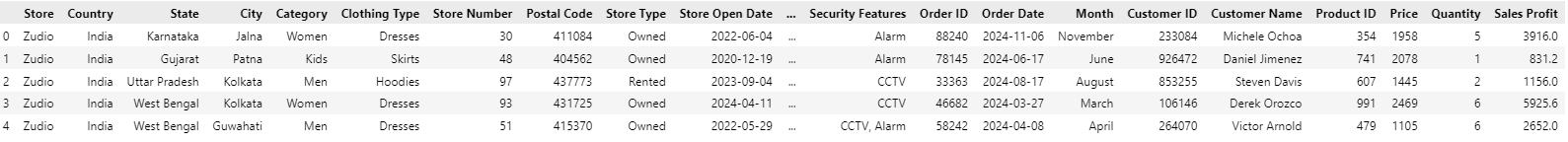
A unique code that identifies each product. This ensures accurate mapping between sales and product details and is critical for managing inventory and understanding product-level profitability.

1. **Price**  
   Specifies the unit selling price of each product. It serves as a core input for revenue calculations, pricing strategy evaluations, and discount impact analysis.
2. **Quantity**  
   Indicates the number of units purchased in a single order. It supports demand forecasting, stock planning, and average order size analysis.
3. **Sales Profit**

Represents the profit earned from each transaction after deducting costs. It is the primary target variable for regression modelling, guiding financial decisions and profit optimization strategies.

**9.3 Results Analysis**

This figure displays the initial few rows of the Zudio sales dataset, providing a clear view of its structure and content. The table includes key fields such as store details, product categories, customer information, sales transaction data, and the target variable—Sales Profit. It helps understand how each variable is arranged and serves as a reference for the data attributes used in further analysis.

****

**Fig. 1: Sample Zudio Sales Dataset**

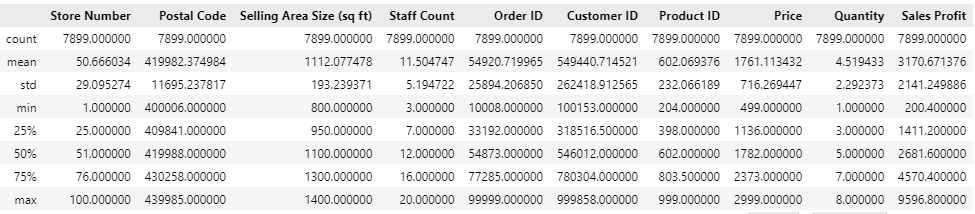
****

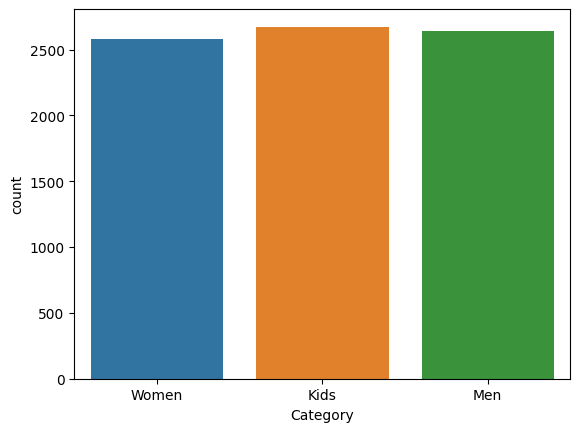
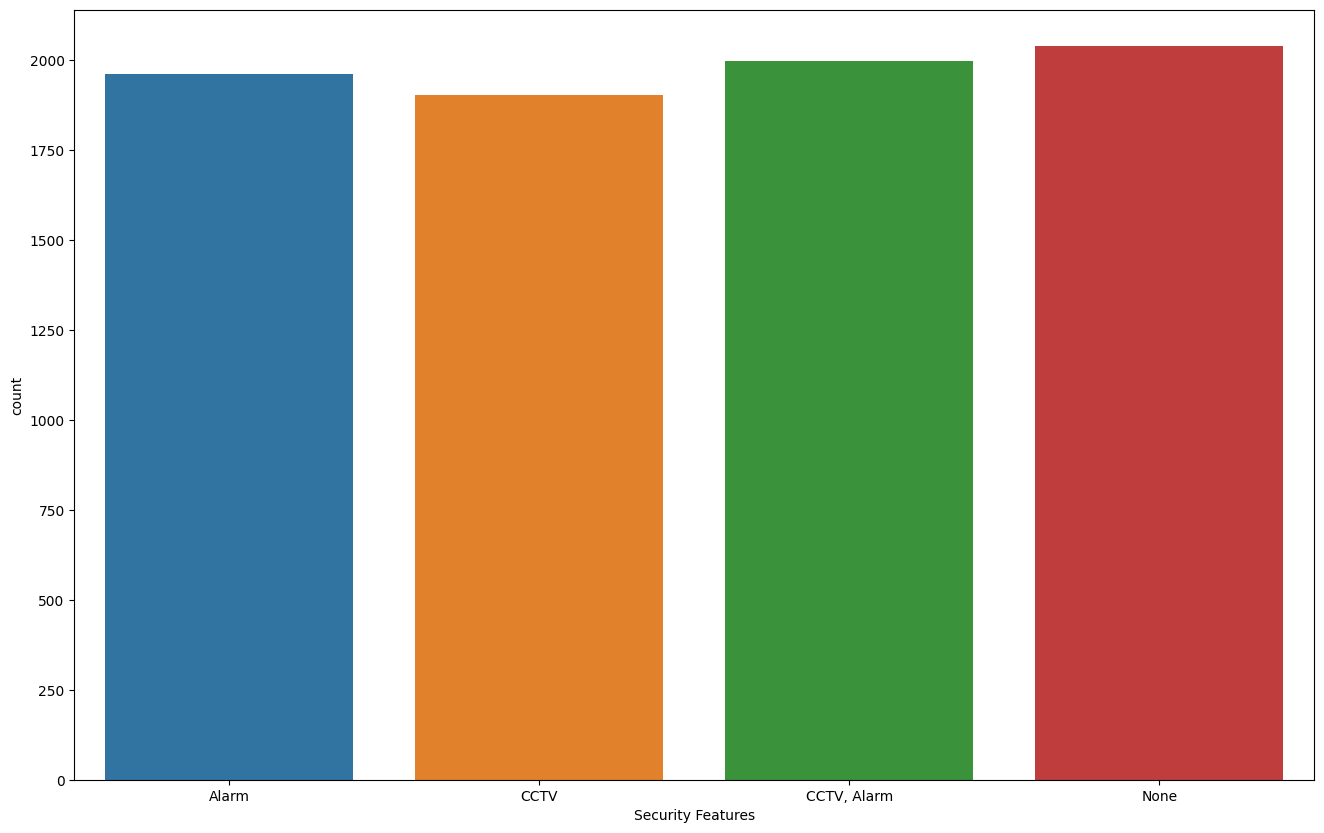
Fig. 2.1: Description of the dataset.

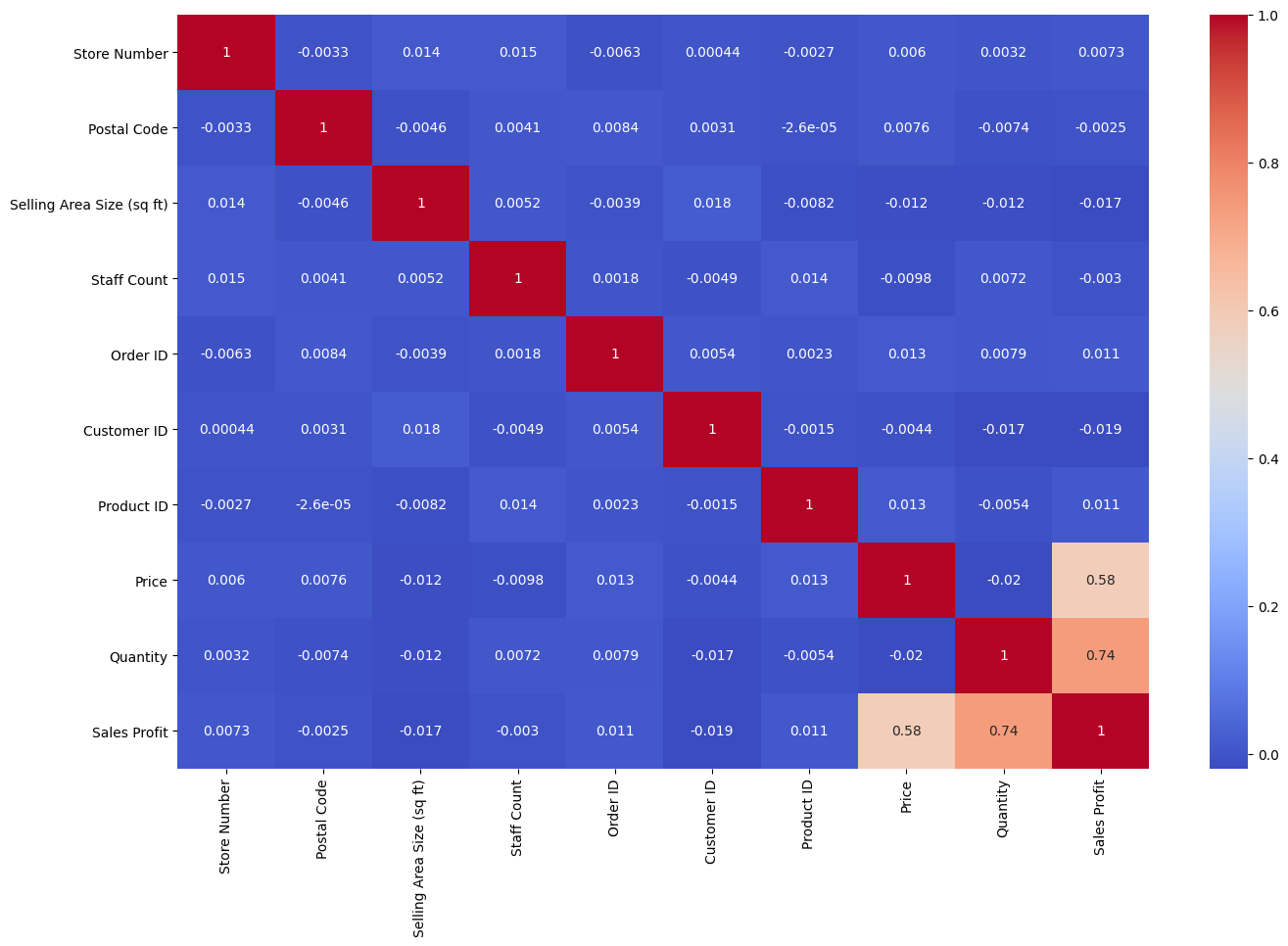


Fig. 2.1: Null Values of the dataset.

**Fig. 2: Dataset Preprocessing.**

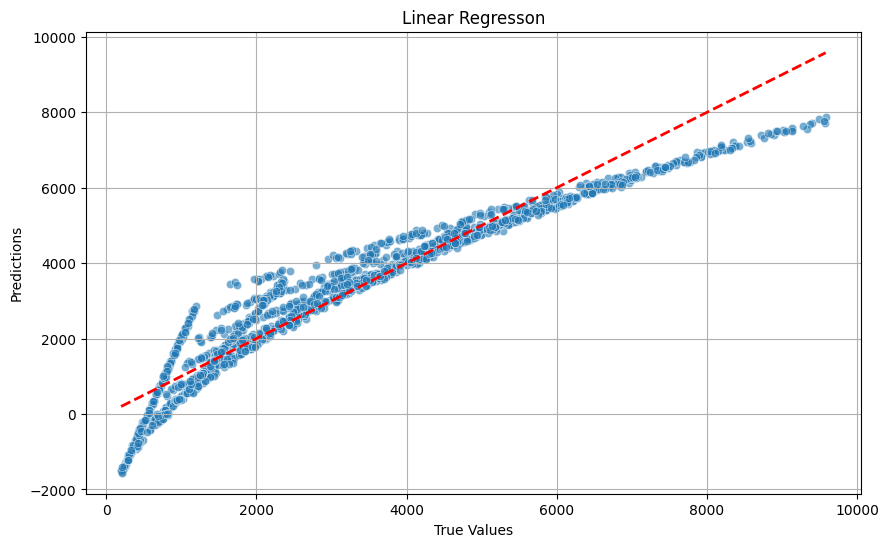
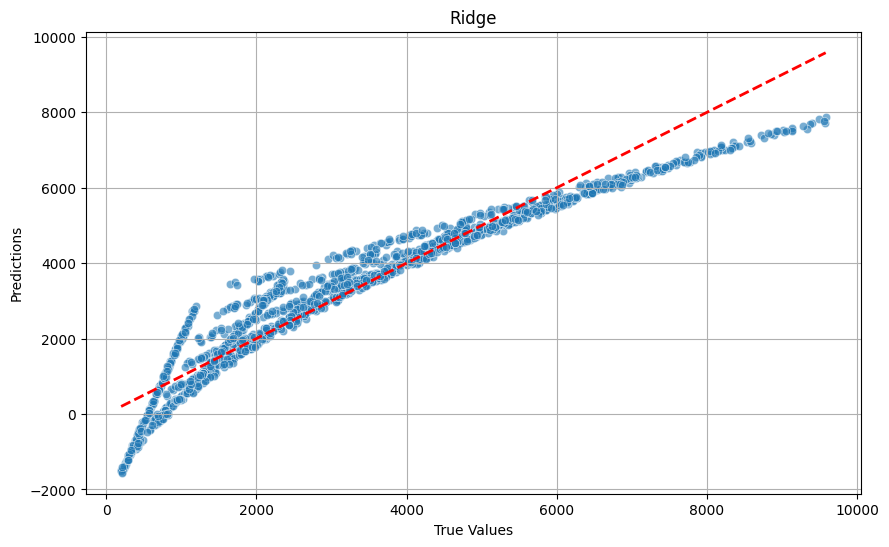
This figure summarizes the preliminary data validation and inspection steps. It includes the count of missing values per column, the number of unique entries for each feature, and a summary of the dataset's structure through info() and describe() functions. This stage helps in identifying data types, spotting inconsistencies, and obtaining statistical insights such as mean, standard deviation, and range for numerical columns.

****

****

**Fig. 3: EDA of the Dataset**

This figure illustrates the Exploratory Data Analysis process through various plots and visualizations. Countplots represent the frequency distribution of categorical variables such as product categories and security features. Heatmaps show the correlation between numeric features, enabling the identification of strongly related variables. This visual analysis reveals underlying trends, feature importance, and possible outliers, helping to guide the modeling phase.

****

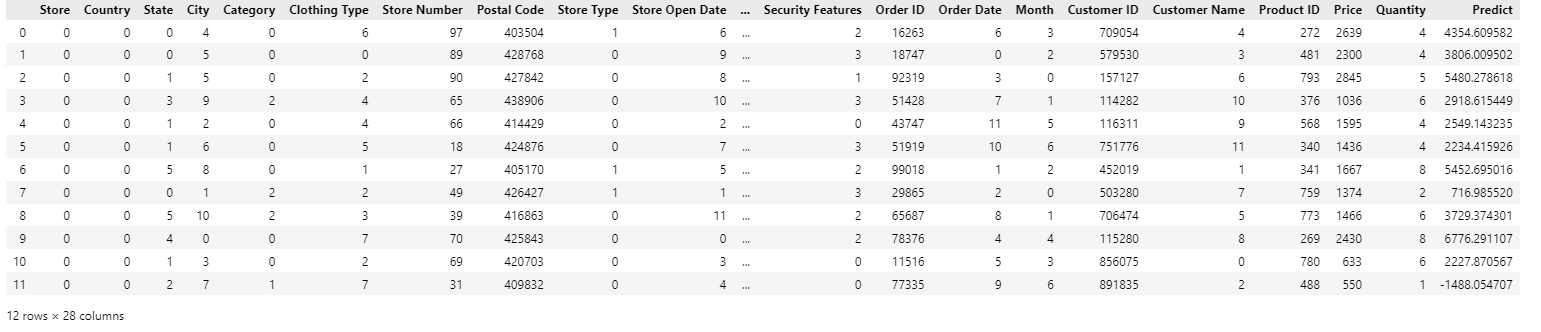
**Fig. 4: Scatter plot of Ridge Regression and Linear Regression Models**

This figure shows the scatter plots comparing actual vs. predicted sales profit for both Ridge and Linear Regression models. These scatter plots serve a purpose by illustrating prediction accuracy. A diagonal line is plotted to represent perfect predictions, and the clustering of points around this line indicates how close the predictions are to actual values.

|  |  |  |
| --- | --- | --- |
| **Metric** | **Ridge Regression** | **Linear Regression** |
| Mean Absolute Error (MAE) | 480.51 | 481.51 |
| Mean Squared Error (MSE) | 405,992.44 | 406,993.38 |
| Root Mean Squared Error (RMSE) | 635.96 | 637.96 |
| R-squared (R²) | 0.90 | 0.91 |

**Fig. 5: Performance Metrics of Ridge Regression and Linear Regression Models**

This figure presents the numerical evaluation results for both regression models. Metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared (R²) are compared side by side. The values reflect high accuracy and minimal error in both models, confirming the reliability and effectiveness of the prediction process.

****

**Fig. 6: Model Prediction on Test Data**

This figure displays the final step where the trained Linear Regression model is applied to an unseen test dataset. After preprocessing, predictions are generated and added as a new column labeled ‘Predict’. This visual output demonstrates the model's practical application and readiness for real-world deployment in supporting sales forecasting and business decisions.

**CHAPTER 10**

**CONCLUSION AND FUTURE SCOPE**

**Conclusion**  
The Zudio sales prediction project successfully demonstrates the application of machine learning techniques to derive meaningful business insights from historical retail data. Through systematic data preprocessing, exploratory analysis, and regression modeling, the project achieved a robust predictive framework for estimating sales profit. Both Ridge Regression and Linear Regression models performed effectively, with Linear Regression emerging as the preferred choice due to its interpretability and consistent accuracy. The evaluation metrics indicate a strong model fit, with an R² score of 0.91 and low prediction errors. The end-to-end pipeline—from data ingestion to prediction on unseen test data—highlights the practical utility of the solution for sales forecasting, inventory planning, and strategic decision-making in the retail domain.

**Future Scope**

* Integrate time-series models like ARIMA or LSTM to capture seasonality and temporal patterns in long-term sales forecasting.
* Incorporate real-time data from point-of-sale systems to enable dynamic and adaptive prediction updates.
* Expand the feature set by including weather data, local events, and competitor pricing to enhance model accuracy.
* Deploy the model as a web-based dashboard for store managers to access live forecasts and actionable insights.
* Apply clustering techniques to segment stores and customers for more personalized promotions and inventory management.
* Implement ensemble models combining multiple algorithms to further improve predictive performance.
* Extend the solution to support cross-category product demand forecasting and optimize store layout and merchandising strategies.

**REFERENCES**

1. Shih, Y.S.; Lin, M.H. A LSTM Approach for Sales Forecasting of Goods with Short-Term Demands in E-Commerce. In *Intelligent Information and Database Systems*; Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Springer: Cham, Switzerland, 2019; Volume 11431 LNAI.
2. Salamai, A.A.; Ageeli, A.A.; El-Kenawy, E.S.M. Forecasting E-Commerce Adoption Based on Bidirectional Recurrent Neural Networks. *Comput. Mater. Contin.* 2022, *70*, 5091–5106.
3. Li, J.; Cui, T.; Yang, K.; Yuan, R.; He, L.; Li, M. Demand Forecasting of E-Commerce Enterprises Based on Horizontal Federated Learning from the Perspective of Sustainable Development. *Sustainability* 2021, *13*, 13050.
4. Zhang, X. Prediction of Purchase Volume of Cross-Border e-Commerce Platform Based on BP Neural Network. *Comput. Intell. Neurosci.* 2022, *2022*, 3821642. [PubMed]
5. Kharfan, M.; Chan, V.W.K.; Firdolas Efendigil, T. A Data-Driven Forecasting Approach for Newly Launched Seasonal Products by Leveraging Machine-Learning Approaches. *Ann. Oper. Res.* 2021, *303*, 159–174.
6. Pan, H.; Zhou, H. Study on Convolutional Neural Network and Its Application in Data Mining and Sales Forecasting for E-Commerce. *Electron. Commer. Res.* 2020, *20*, 297–320.
7. Li, M.; Ji, S.; Liu, G. Forecasting of Chinese E-Commerce Sales: An Empirical Comparison of ARIMA, Nonlinear Autoregressive Neural Network, and a Combined ARIMA-NARNN Model. *Math. Probl. Eng.* 2018, *2018*, 6924960.
8. Yang, M.; Zhang, T.; Wang, C. The Optimal E-Commerce Sales Mode Selection and Information Sharing Strategy under Demand Uncertainty. *Comput. Ind. Eng.* 2021, *162*, 107718.
9. He, Q.-Q.; Wu, C.; Si, Y.-W. LSTM with Particle Swam Optimization for Sales Forecasting. *Electron. Commer. Res. Appl.* 2022, *51*, 101118.
10. Yin, X.; Tao, X. Prediction of Merchandise Sales on E-Commerce Platforms Based on Data Mining and Deep Learning. *Sci. Program.* 2021, *2021*, 2179692.
11. Jandera, A.; Skovranek, T. Customer Behaviour Hidden Markov Model. *Mathematics* 2022, *10*, 1230.
12. Xin, S.; Ester, M.; Bu, J.; Yao, C.; Li, Z.; Zhou, X.; Ye, Y.; Wang, C. Multi-Task Based Sales Predictions for Online Promotions. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management, Beijing, China, 3–7 November 2019; ACM: New York, NY, USA; pp. 2823–2831.
13. Weytjens, H.; Lohmann, E.; Kleinsteuber, M. Cash Flow Prediction: MLP and LSTM Compared to ARIMA and Prophet. *Electron. Commer. Res.* 2021, *21*, 371–391.
14. Bandara, K.; Shi, P.; Bergmeir, C.; Hewamalage, H.; Tran, Q.; Seaman, B. Sales Demand Forecast in E-Commerce Using a Long Short-Term Memory Neural Network Methodology. In Proceedings of the International Conference on Neural Information Processing, Sydney, NSW, Australia, 12–15 December 2019; Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics. Springer: Cham, Switzerland, 2019; Volume 11955 LNCS.
15. Ji, S.; Wang, X.; Zhao, W.; Guo, D. An Application of a Three-Stage XGboost-Based Model to Sales Forecasting of a Cross-Border e-Commerce Enterprise. *Math. Probl. Eng.* 2019, *2019*, 8503252.
16. Liu, J.; Liu, C.; Zhang, L.; Xu, Y. Research on Sales Information Prediction System of E-Commerce Enterprises Based on Time Series Model. *Inf. Syst. e-Bus. Manag.* 2020, *18*, 823–836.
17. Statista Number of Digital Buyers Worldwide. Available online: https://www.statista.com/statistics/251666/number-of-digital-buyers-worldwide/ (accessed on 26 July 2022).
18. Statista Worldwide Retail E-Commerce Sales. Available online: https://www.statista.com/statistics/379046/worldwide-retail-e-commerce-sales/#statisticContainer (accessed on 26 July 2022).
19. Statista E-Commerce Share of Retail Sales Worldwide. Available online: https://www.statista.com/statistics/534123/e-commerce-share-of-retail-sales-worldwide/ (accessed on 26 July 2022).
20. Eurostat E-Commerce Statistics for Individuals. Available online: https://ec.europa.eu/eurostat/statistics-explained/index.php?title=e-commerce\_statistics\_for\_individuals#General\_overview (accessed on 26 July 2022)