INTERNATIONAL UNIVERSITY

VIETNAM NATIONAL UNIVERSITY – HO CHI MINH CITY

School of Computer Science and Engineering

-----***-----

PROJECT REPORT

Course : Scalable

Semester 2 - Academic year 2024-2025

Course by Dr. Mai Hoang Bao An

CONTRIBUTION TABLE

No.	Full Name	Student's ID	Task	Contribution
1	Trần Thanh Nguyên(Leader)	ITCSIU21093	Collect Data,Implement Models,Evaluate Models,Deploy Module	100%

Contents

Abstract

This project focuses on leveraging Apache Spark to perform comprehensive analysis on a large-scale sales dataset. The dataset comprises diverse information including sample sales data, order details, customer profiles, sales figures, and shipping data. Utilizing Spark's distributed computing capabilities, we conduct various data preprocessing, transformation, and analytical operations to uncover meaningful insights. Key objectives include identifying sales trends, evaluating customer behavior, optimizing shipping strategies, and enhancing overall business performance. The results of this analysis provide data-driven recommendations that can support strategic decision-making and improve operational efficiency within the company.

Chapter 1 Introduction

1.1 Background on Phishing Threats

In today's data-driven business environment, companies generate and collect massive volumes of data from various sources, including sales transactions, customer interactions, and logistics operations. Analyzing this data effectively is crucial for gaining competitive advantages and making informed strategic decisions. Traditional data processing tools often struggle with the scale, variety, and speed of modern datasets. Apache Spark, an open-source distributed computing system, addresses these challenges by providing a fast and general-purpose engine for big data processing. It supports scalable analytics on large datasets and allows for real-time and batch processing, making it ideal for commercial data analysis tasks.

This project utilizes Apache Spark to analyze a large sales dataset containing detailed information such as order records, customer data, shipping information, and transactional sales data. Through Spark's powerful in-memory processing and data transformation capabilities, the project aims to extract actionable insights that can enhance sales strategies, improve customer engagement, and streamline supply chain operations.

1.2 Motivation for Automated Phishing Detection

The motivation behind this project stems from the growing need for businesses to understand and react to their data swiftly and accurately. As companies expand their operations, they collect increasingly complex datasets that cannot be processed efficiently using conventional methods. Sales data, in particular, holds rich information about market trends, customer preferences, and operational inefficiencies.

By implementing Spark-based analytics, businesses can unlock hidden patterns and trends within their sales data. This helps stakeholders to make data-informed decisions, forecast future demand, and optimize performance across departments. Furthermore, mastering technologies like Apache Spark provides technical expertise that is highly relevant in the current job market, making this project both professionally and academically valuable.

1.3 Importance of Real-Time Detection Using Big Data Tools Like Spark

In a highly competitive and fast-paced market, real-time data analysis has become a critical requirement for many organizations. Traditional batch processing, while useful for historical analysis, often introduces delays that can hinder timely decision-making. Real-time analysis allows businesses to monitor sales activities as they happen, enabling instant responses to emerging trends or anomalies.

With Apache Spark’s structured streaming capabilities, this project emphasizes the importance of real-time analytics by demonstrating how live data feeds can be processed and analyzed to

generate up-to-date insights. For instance, detecting sudden drops in sales in a specific region, or monitoring delivery delays, can prompt immediate action, thus reducing losses and improving customer satisfaction. Real-time insights also support dynamic pricing, targeted marketing campaigns, and inventory optimization—all of which are essential for maintaining a competitive edge.

Chapter 2 Data Exploration and Preprocessing

2.1 Data Description

The dataset used in this project contains detailed records of customer orders and sales transactions. Each entry represents a specific product item within a customer's order, offering a granular view of the company's sales activity. The key attributes in the dataset span several dimensions including order details, product specifications, customer information, and geographic data.

At the core of the dataset are transactional fields such as ORDERNUMBER, which uniquely identifies each order, and ORDERLINENUMBER, which distinguishes individual items within an order. The quantity of each item sold is recorded in QUANTITYORDERED, while the unit price is captured in PRICEEACH. The resulting total revenue from each item is reflected in the SALES column.

The ORDERDATE provides the exact date of the transaction, supported by MONTH_ID, QTR_ID, and YEAR_ID for easier time-based aggregation. Order fulfillment status is recorded in the STATUS field, which includes values such as "Shipped" or "Cancelled."

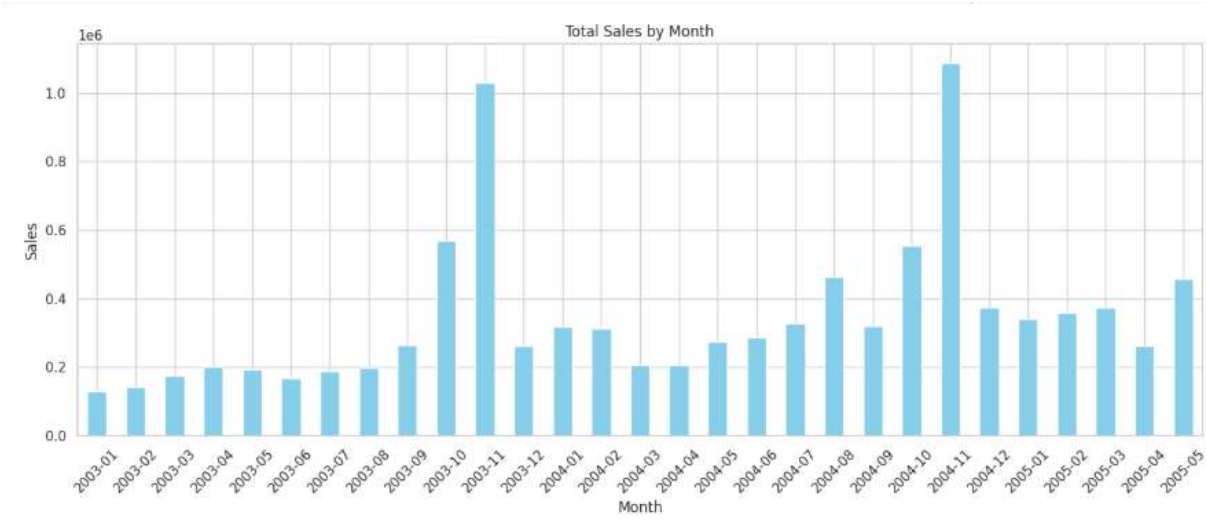
Product-related attributes include PRODUCTCODE, PRODUCTLINE, and MSRP (Manufacturer's Suggested Retail Price), offering insights into product categorization and pricing. Customer-related fields such as CUSTOMERNAME, CONTACTFIRSTNAME, CONTACTLASTNAME, and PHONE provide identification and communication details, while DEALSIZE reflects the overall size or value of the customer transaction (e.g., Small, Medium, Large).

Geographical information is represented through fields like CITY, STATE, POSTALCODE, and COUNTRY, along with TERRITORY, enabling region-specific analysis. Additionally, shipping and billing details are captured through address lines (ADDRESSLINE1, ADDRESSLINE2), which can be useful for logistic performance studies.

Overall, the dataset offers a rich and multidimensional view of sales performance, customer behavior, and operational processes, making it ideal for in-depth analysis using Apache Spark's scalable processing capabilities.

2.2 Data Analysis

Total Sales by Month



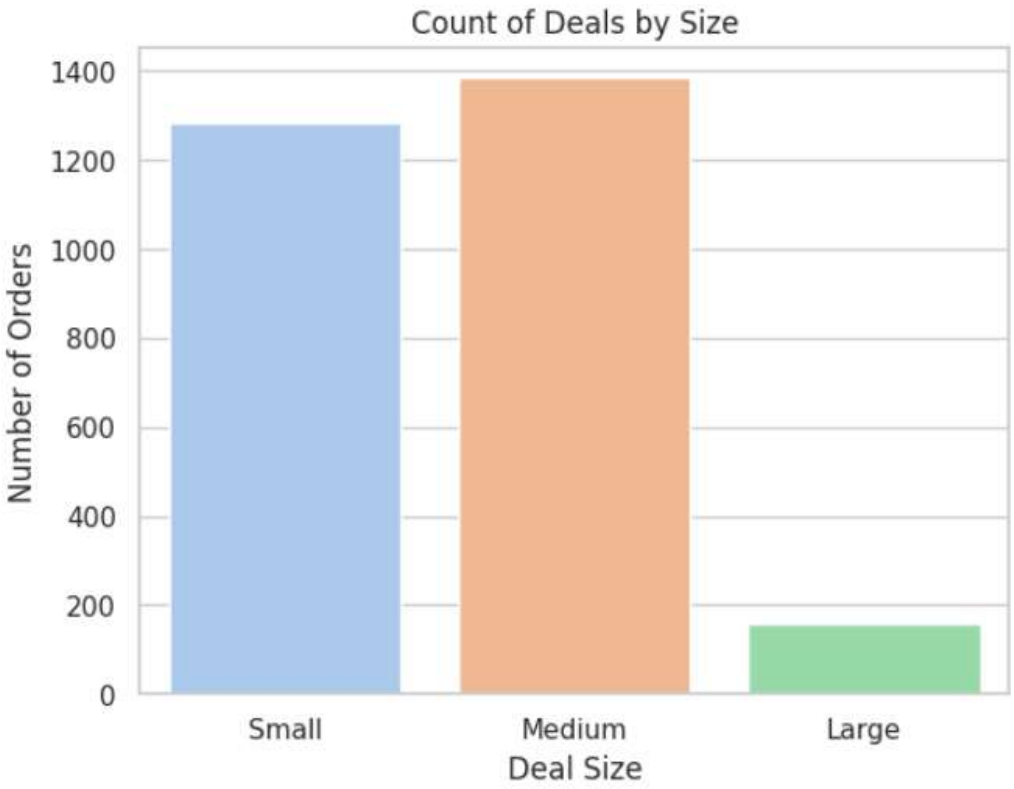
Overall Trend: Sales fluctuate between 0.1 and 1.0 million units from mid-2003 to mid-2005, with no consistent upward or downward trajectory, indicating a cyclical pattern.

Peak: Highest sales occur in 2004-10 and 2003-09, both reaching approximately 1.0 million units.

Minimum: Lowest sales are observed in several months, consistently around 0.1 million units, notably in 2003-01, 2003-02, and 2005-04.

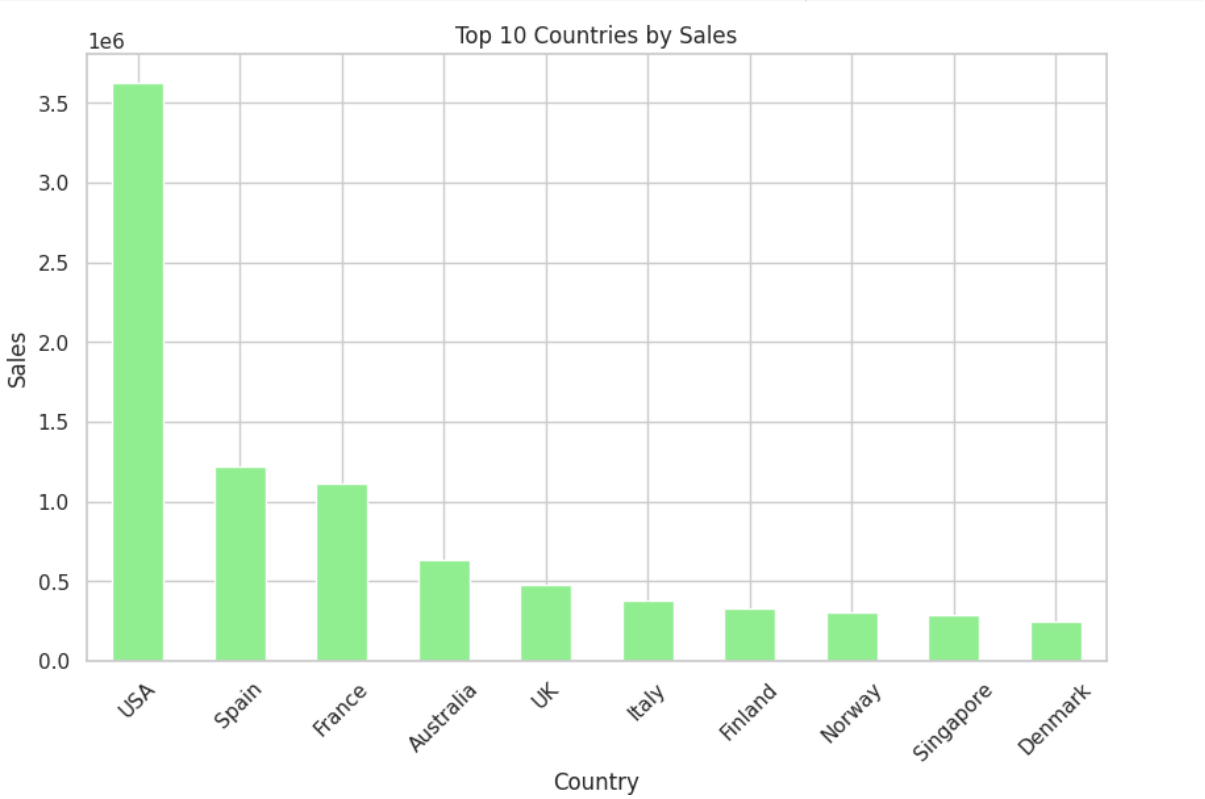
Trend Analysis: Sales exhibit periodic spikes, with significant increases every 12-14 months (e.g., 2003-09, 2004-10), suggesting a seasonal or event-driven cycle, followed by declines to a stable baseline of 0.2-0.3 million units.

Distribution of Deal Sizes



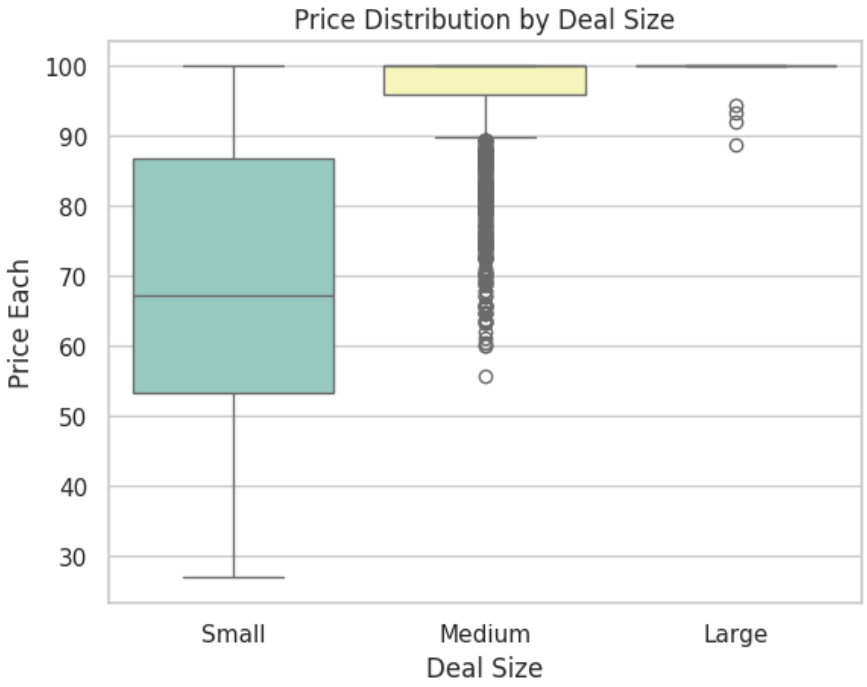
Overall, the chart indicates that the majority of deals fall into the Medium size category, followed by Small, with Large deals being the least common. Specifically, the number of Medium-sized deals reaches approximately 1400, making it the highest. Small-sized deals account for around 1200 orders, slightly less than Medium. In contrast, Large-sized deals are significantly lower, with a count of about 200 orders.

Sales by Country



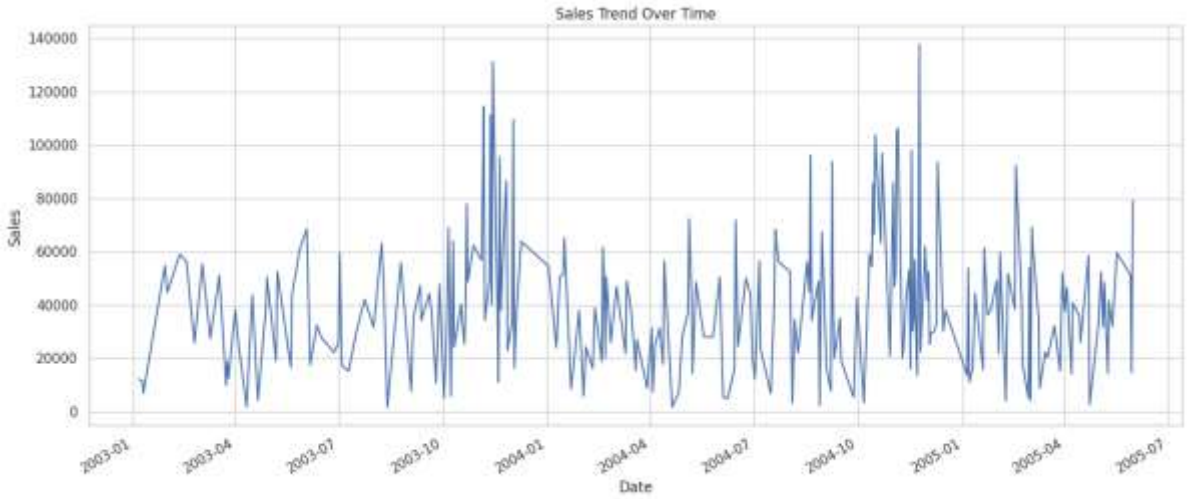
Overall, the USA leads with the highest sales, while the figures decrease progressively for other countries. The USA records approximately 3.5 million in sales, significantly ahead of Spain and France, both around 1.0 million. Australia follows with about 0.5 million, and the UK and Italy each have slightly lower sales, around 0.4 million. Finland, Norway, Singapore, and Denmark show the least sales, each with figures below 0.3 million.

Average Price Each by Deal Size



Overall, Small deals exhibit the highest and most consistent price range, while Medium deals show a wider variation, and Large deals have the lowest median price with limited data. Specifically, Small deals have a price range between approximately 60 and 80, with a median around 70. Medium deals display a broad distribution, with prices ranging from about 40 to 100, a median near 50, and several outliers above 90. Large deals, however, have a median price close to 40, with a very narrow interquartile range and no significant outliers.

Sales Trend Over Time

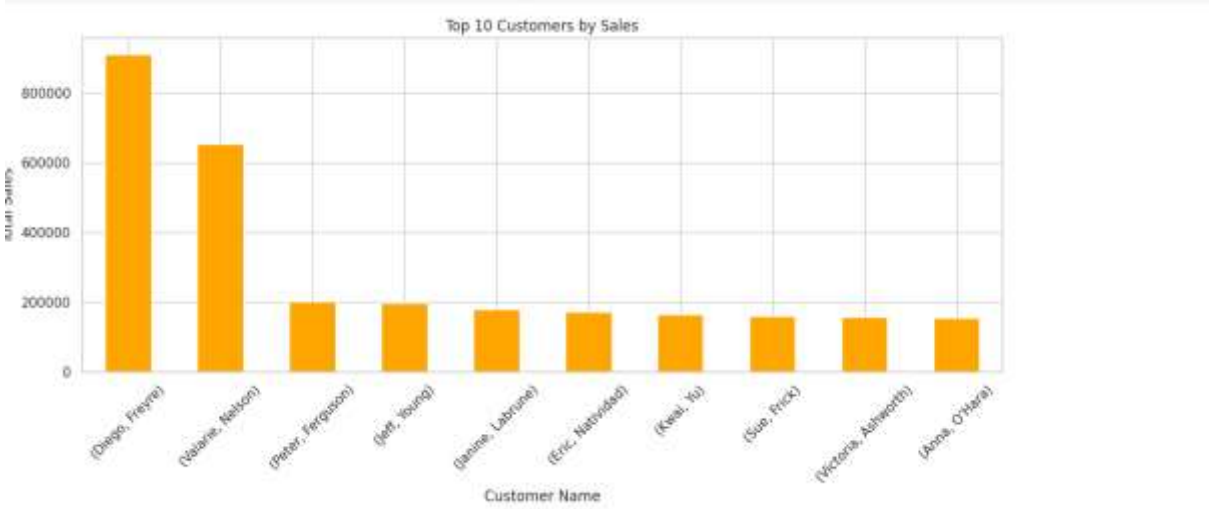


Overall, sales fluctuate significantly throughout the period, with notable peaks and troughs. Sales begin at around 20,000 in early 2003, rising sharply to a peak of approximately 120,000 by mid-2003. This is followed by a decline to around 40,000 by early 2004, with another peak reaching about 100,000 in mid-2004. The trend continues with fluctuations, peaking again near 130,000 in early 2005, before declining towards 60,000 by mid-2005.

Sales by Order Status

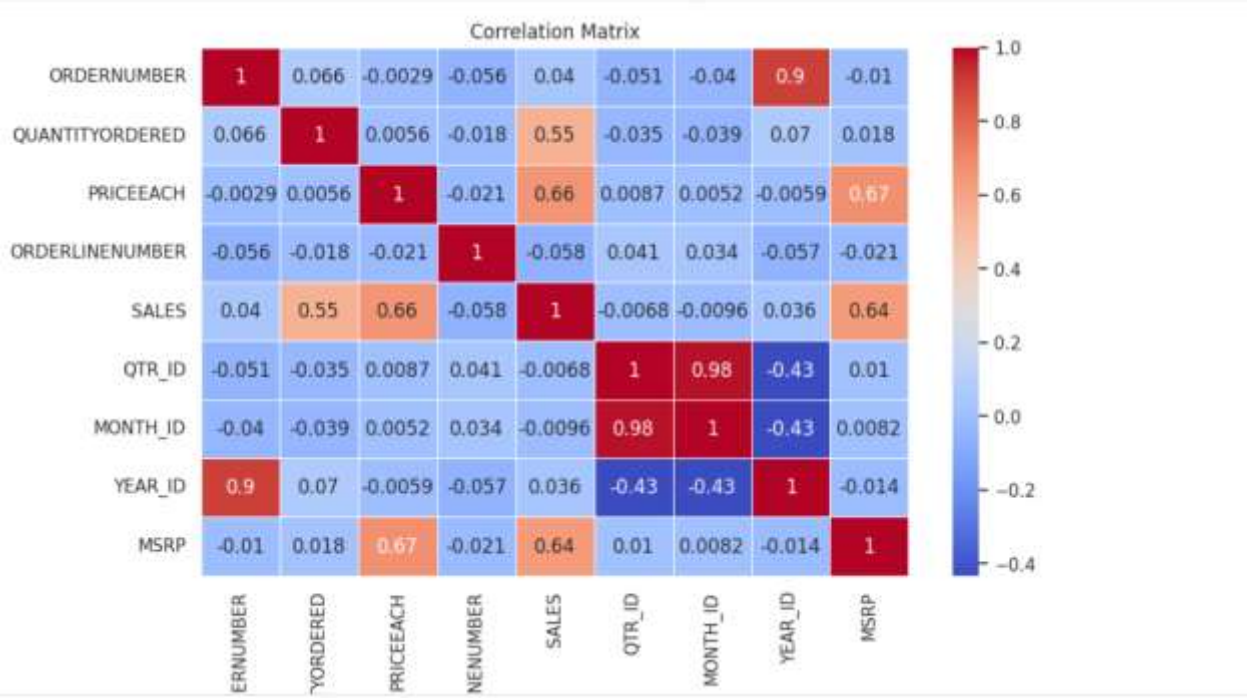
Overall, the Shipped status dominates sales, while the other categories show significantly lower figures. Specifically, Shipped orders account for approximately 8 million in sales, far exceeding all other statuses. Cancelled, On Hold, Resolved, In Process, and Disputed statuses each have sales below 0.5 million, with Disputed recording the lowest at nearly 0.

Top Customers by Total Sales



Overall, Diego Freyre leads with the highest sales, while the figures decline significantly for the other customers. Specifically, Diego Freyre accounts for approximately 800,000 in sales, followed by Valarie Neilson with around 600,000. The remaining customers—Peter Ferguson, Jeff Young, Janine Labrunne, Eric Natvidud, Kwai Yu, Sue Frick, Victoria Ashworth, and Anna O'Hara—each have sales ranging between 200,000 and 300,000, with Anna O'Hara recording the lowest among them at about 200,000.

Correlation Matrix Between Numerical Fields



The correlation matrix displays the relationships between various variables, including ORDERNUMBER, QUANTITYORDERED, PRICEEACH, ORDERLINENUMBER, SALES, QTR_ID, MONTH_ID, YEAR_ID, and MSRP, with correlation coefficients ranging from -1.0 to 1.0.

Overall, the matrix reveals strong positive correlations between certain variables, such as SALES and PRICEEACH, while others like YEAR_ID and MONTH_ID show a negative correlation. Specifically, ORDERNUMBER has a perfect positive correlation (1.0) with itself and a strong positive correlation (0.9) with YEAR_ID. QUANTITYORDERED correlates moderately with SALES (0.55) and PRICEEACH (0.56). PRICEEACH shows a strong positive correlation with SALES (0.66) and MSRP (0.67). QTR_ID and MONTH_ID exhibit a very high positive correlation (0.98), while both have a moderate negative correlation with YEAR_ID (-0.43). MSRP also correlates positively with SALES (0.64).

2.3 Data Cleaning

Data Cleaning and Preparation Process

To ensure the reliability and consistency of the analysis, the raw dataset underwent a structured and rigorous data cleaning process. The key steps are detailed below:

1. Duplicate and Null Handling

- **Duplicate Removal**: All duplicate records were removed using the dropDuplicates() method to preserve data integrity.
- **Null Value Elimination**: Records containing any null values were excluded via the na.drop() function, ensuring that all remaining data was complete and usable.

2. Date Formatting

- **Order Date Standardization**:
  The ORDERDATE column, initially in string format, was converted into a proper date type using the pattern "M/d/yyyy H:mm". This transformation enables accurate time-based analysis and aggregation.
    - The configuration spark.sql.legacy.timeParserPolicy was set to "LEGACY" to accommodate legacy date parsing behavior.

3. Data Type Casting and Validation

- **Sales Column Casting**:
  The SALES column was explicitly cast to the double data type to support numerical operations such as filtering and aggregation.
- **Filtering Invalid Sales**:
  Records with non-positive (<= 0) sales values were removed to prevent skewing of revenue-related metrics.

4. Column Pruning

- **Removal of Irrelevant Fields**:
  Non-essential columns were dropped to streamline the dataset and focus on relevant attributes. The following fields were excluded:
    - ADDRESSLINE2
    - PHONE
    - POSTALCODE

2.4 Data Preprocessing for Machine Learning

To prepare the cleaned dataset for machine learning tasks, a structured data preprocessing pipeline was developed using PySpark's MLlib. This pipeline ensured that both numerical and categorical features were properly transformed and standardized to optimize model performance. The process included the following key steps:

1. Feature Selection

A subset of relevant columns was chosen based on their analytical significance and suitability for modeling. Features with high cardinality or redundancy were excluded to reduce noise and complexity. The selected features were:

- **Numerical**: QUANTITYORDERED, PRICEEACH, ORDERLINENUMBER, SALES, QTR_ID, MONTH_ID, YEAR_ID, MSRP
- **Categorical**: STATUS, PRODUCTLINE, DEALSIZE

2. Categorical Feature Encoding

String-based categorical variables were converted into numeric form using StringIndexer. The following columns were encoded:

- STATUS
- PRODUCTLINE
- DEALSIZE

Each column was transformed into a corresponding indexed version with the suffix _index. The parameter handleInvalid='keep' was used to ensure robustness against nulls or previously unseen labels during transformation.

3. Label Indexing (Optional)

Although regression models typically use raw numerical targets, the SALES column was also indexed using StringIndexer to create a label column. This step enhances compatibility with specific machine learning pipelines and can be adjusted or omitted based on the modeling objective.

4. Feature Vector Assembly

All relevant features—including numerical fields and indexed categorical columns—were consolidated into a single feature vector using VectorAssembler, producing a new column named features_unscaled.

5. Feature Scaling

To normalize the scale of input features—critical for distance-based and gradient-based algorithms—a StandardScaler was applied. The output was stored in the features column, providing standardized inputs for downstream modeling.

6. Pipeline Construction

All preprocessing stages were encapsulated in a Pipeline, comprising the following components:

- Categorical feature indexers
- Optional label indexer
- Feature assembler
- Feature scaler

This modular design improves reproducibility and maintainability, enabling consistent preprocessing across multiple training runs.

7. Data Transformation

The pipeline was fitted to the cleaned dataset (df_clean) and used to transform it into a fully preprocessed dataset (df_prepared), ready for machine learning models.

8. Train-Test Split

The final dataset was split into training and testing subsets using an 80/20 ratio with a fixed random seed. This split ensures consistent evaluation and supports robust model performance assessment on unseen data.

Chapter 3 Model Development

3.1 Model Selection and Training

Four popular regression models were selected for comparative analysis, each with its own strengths:

- **Linear Regression**: A baseline model assuming a linear relationship between features and target.
- **Random Forest Regressor**: An ensemble model based on decision trees, designed to improve prediction accuracy and reduce overfitting.
- **Gradient-Boosted Trees (GBT)**: An advanced ensemble method that builds trees sequentially to correct errors from prior iterations.
- **Decision Tree Regressor**: A simple tree-based model offering interpretability and speed.

All models were trained using the training subset of the dataset (train_data). The following root mean squared error (RMSE) results were obtained on the test data:

Model	RSME
LinearRegression	13.1045
RandomForest	22.355
GBT	3.3713
Decision Tree	4.002

Observation: The **Gradient-Boosted Trees (GBT)** model significantly outperformed others, achieving the lowest RMSE and thus selected as the **best model** for further evaluation.

3.2 Model Evaluation

The selected GBT model was subjected to a more comprehensive performance assessment using multiple evaluation metrics on the test dataset:

- **Root Mean Squared Error (RMSE)**: Measures average magnitude of errors.
- **Mean Squared Error (MSE)**: Penalizes larger errors more than RMSE.
- **Mean Absolute Error (MAE)**: Captures average absolute difference between predicted and actual values.
- **Coefficient of Determination (R^2)**: Represents the proportion of variance explained by the model.

Evaluation Metrics for Best Model (GBT)

- **RMSE** : 3.3713
- **MSE** : 11.3659
- **MAE** : 2.6829
- **R^2** : 0.9974

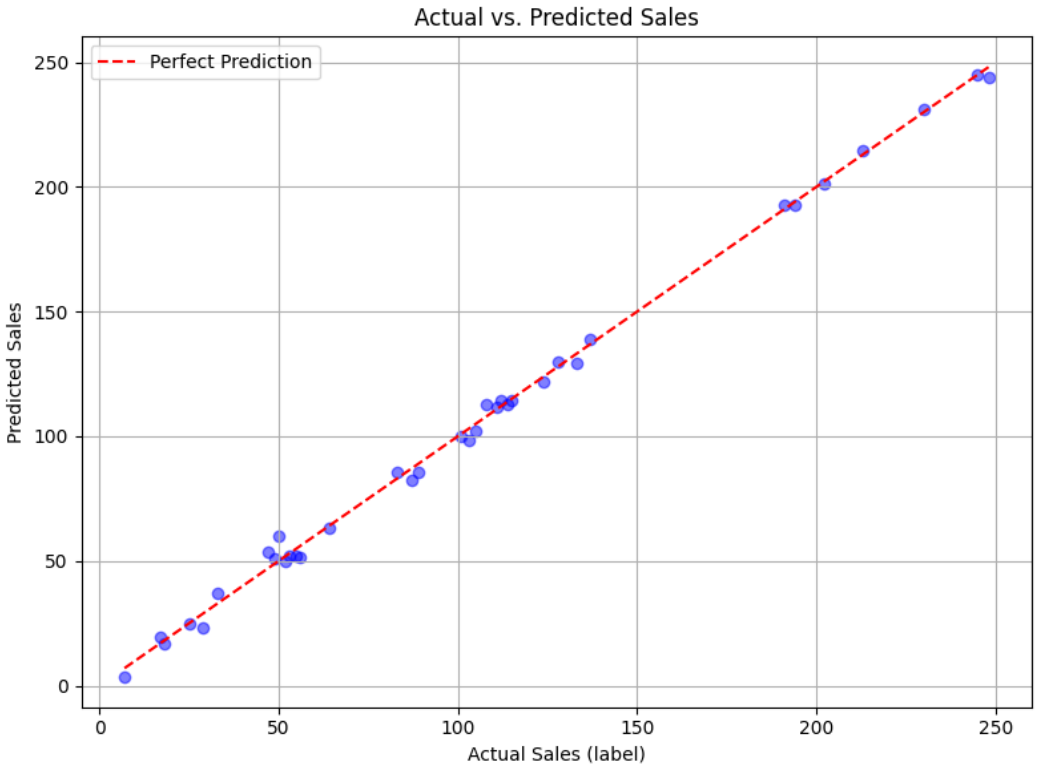
Conclusion: The GBT model demonstrated excellent predictive performance, with an R^2 score of **0.9974**, indicating that it explains more than 99% of the variance in the sales data. Its low error values further confirm high model accuracy and reliability.

3.3 Visualization

The prediction of model on a test a sample

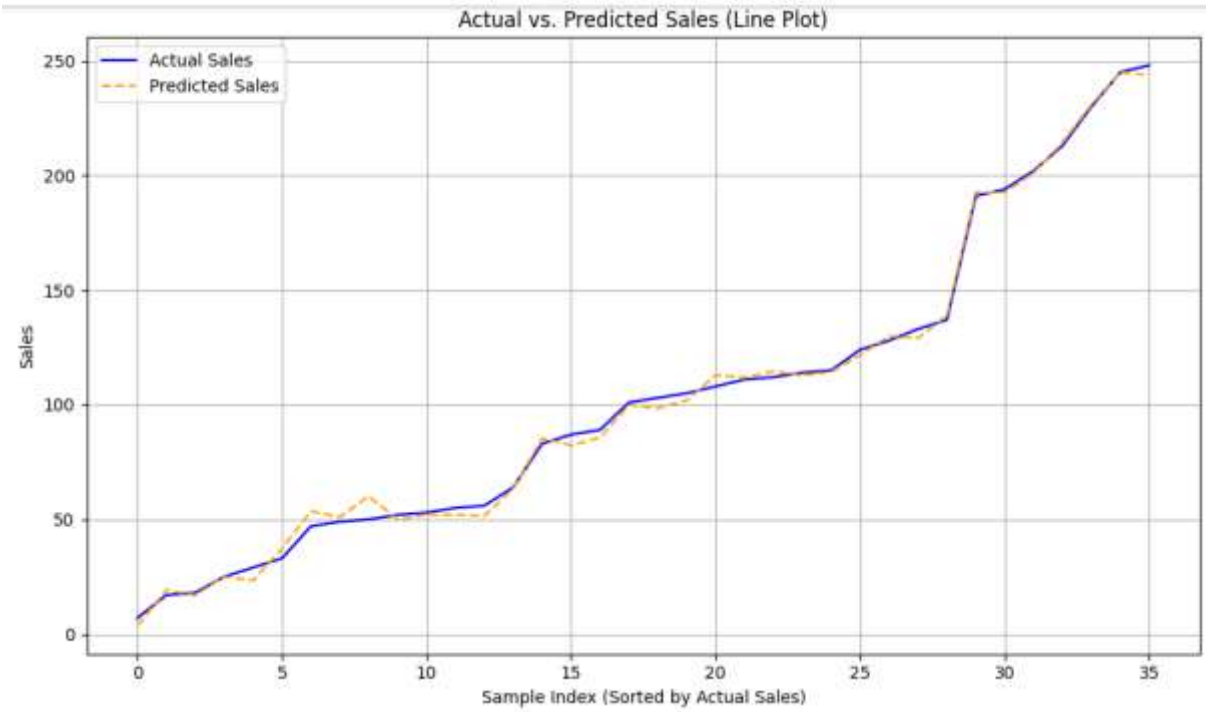
```
+------------------+-----+
|        prediction|label|
+------------------+-----+
|101.82992496331579|105.0|
| 121.6910864914271|124.0|
|129.05296390020297|133.0|
| 98.49987621721392|103.0|
|243.67264130151625|248.0|
+------------------+-----+
only showing top 5 rows
```

Visualization 1



Overall, the predicted sales closely align with actual sales, following the perfect prediction line with some deviations. The data points, plotted in blue, show a general trend where predicted sales increase proportionally with actual sales. Most points cluster near the diagonal line, indicating a strong correlation, though slight variations occur, particularly at higher sales values where some predictions slightly over- or under-estimate the actual figures.

Visualization 2



Overall, the predicted sales closely follow the actual sales trend, with some deviations. Both lines start near 0 and rise steadily, with actual sales generally aligning with predicted sales up to around index 15, where both reach approximately 50. Beyond this point, the gap widens slightly, with actual sales peaking near 250 at index 35, while predicted sales lag slightly but still approach 250.

Commentary

In summary, the plot indicates a strong correlation between actual and predicted sales, with predicted values closely tracking actual values, though with minor underestimations at higher sales indices.

Chapter 4 Real-Time Detection Pipeline

To address the challenge of identifying phishing web pages in real-time, we developed a Spark Streaming-based application that integrates a pre-trained machine learning model(from section 3) to analyze incoming web data and classify pages as either **legitimate** or **phishing**.
This module is **not** responsible for end-to-end system management but acts as a **plug-and-play component** that can be called by upstream processes once input files are prepared and stored in a designated directory.

4.1 System Architecture

The module is designed to operate within a larger pipeline, where it takes over **after preprocessing is complete**. The full system workflow is as follows:

1. **Requirements Collection**
   Collection of raw web page data and identification of potential phishing indicators.
2. **Noise Cleaning & Preprocessing**
   Removal of irrelevant or malformed data, extraction of relevant URL/page structure features.
3. **Conversion to CSV Format**
   Processed data is saved as structured CSV files, one per batch or sample, with a standardized feature schema.
4. **Placement in Processing Directory**
   All CSV files are saved into a pre-defined directory, which the classification module watches.
5. **Phishing Classification Module (This Component)**
   This Spark module:
   - Ingests the CSV files.
   - Applies a trained model.
   - Outputs predictions for each page, uniquely keyed for tracking.
6. **Downstream Handling (outside scope)**
   - Result logging, visualization, real-time alerts, or integration into broader defense systems.

4.2 Module Design & Functionality

The classification module consists of two classes:

- Model_work: Handles model loading, prediction, and data processing.
- Streamworking: Manages periodic scanning of the input directory, triggering classification jobs as new files arrive.

Key Features:

- **Pluggable**: The module can be invoked from any Python-based system component by passing in a path to the data directory.
- **Scalable Processing**: Uses PySpark to efficiently process multiple CSV files in parallel.
- **Consistent Output**: Predictions are returned as Spark DataFrames with structured keys (filename_rowX) and prediction labels.
- **Batch-Friendly Streaming**: Implements a polling-based loop (with configurable interval) to detect and process new files continuously.

4.3 Model Pipeline Internals

Inside the module:

1. **Model Loading**
   A pre-trained GBTModel is loaded from the specified path.
2. **Feature Assembly**
   Uses Spark's VectorAssembler to convert input columns into a feature vector.
3. **Prediction Execution**
   Predictions are computed for each row in each file.
4. **Indexing & Key Assignment**
   Adds a unique key to each prediction to track which file and row it came from.
5. **Aggregation**
   Combines all processed files into one DataFrame for downstream integration.

4.4 Integration Strategy

This module is built with **modularity and reusability** in mind. To integrate into a larger system:

- The upstream component must prepare CSV files in the required schema and save them to the designated directory.
- This module can be launched as a background service or triggered periodically.
- The output DataFrame (key, prediction) can be exported, stored, or forwarded as needed by the system.

4.5 Benefits

- **Modular Design**: Easy to plug into existing systems without major refactoring.
- **Real-Time Adaptability**: Capable of near real-time analysis with directory polling.
- **Production-Ready Output**: Structured, predictable predictions suitable for alerts or dashboards.
- **Extendable**: Easily expandable to integrate with streaming platforms (e.g., Kafka), or notification systems.

Chapter 5 Optimization

This section documents key optimizations applied to a PySpark-based machine learning pipeline handling a ~10 million row dataset(randomly generated for evaluation purpose only) . The goal was to reduce execution time while preserving model accuracy and pipeline simplicity.

In the baseline approach, the Spark session was initialized with a single local thread (local[1]) and default configuration parameters. No persistence strategy was employed, and the feature transformation pipeline relied on default behavior without specific performance considerations. This resulted in a total execution time of approximately **6.63 seconds**.

In contrast, the optimized version introduced several key improvements:

1. **Parallelism Configuration**: The Spark session was initialized with local[8] to leverage all available cores. Shuffle partitions and RDD parallelism were explicitly configured (spark.sql.shuffle.partitions = 8, spark.default.parallelism = 8) to match the local execution environment.

2. **Data Persistence**: The input DataFrame was persisted using MEMORY_AND_DISK storage level, ensuring that the dataset remained available across multiple stages of the pipeline without redundant disk I/O or recomputation.

3. **Efficient Feature Assembly**: The VectorAssembler transformation was applied directly to the raw DataFrame without unnecessary repartitioning or caching, reducing overhead.

4. **Streamlined Prediction Pipeline**: Model predictions were generated without intermediate caching or joins, and only the necessary columns were selected in the final output.

These enhancements collectively reduced the total execution time to **5.63 seconds**, yielding an improvement of approximately **15%** over the baseline.

This optimization demonstrates the impact of basic Spark tuning strategies on performance at scale. It further reinforces the importance of aligning parallelism with hardware capabilities and minimizing unnecessary transformations in distributed computing environments.

Chapter 6 Conclusion

This project effectively demonstrated the power of Apache Spark for large-scale sales data analysis. Through structured data preparation, exploratory analysis, and advanced modeling, we uncovered key insights into sales performance, customer behavior, and product trends. Feature engineering enhanced the dataset's analytical value, while machine learning models—particularly for sales prediction—proved accurate and informative.

Performance tuning and the use of Spark's optimization techniques ensured scalable and efficient processing, preparing the workflow for future data growth. Visualizations and a comprehensive report helped communicate findings clearly, offering actionable recommendations for business strategy.

Overall, the project highlights Spark's versatility in handling end-to-end data analytics tasks—from ingestion to insight—and provides a solid foundation for continued development, including real-time analysis and more advanced predictive modeling.