MARKET MAVEN: AI-DRIVEN SALES FORECASTING FOR SMARTER SUPERMARKET

A Project Report Submitted in Partial Fulfillment of the Requirements for the Completion of the Infosys Springboard Internship Program

Project by

E. CHANDANA

Under the esteemed guidance of

Dr. N JAGAN MOHAN

Senior Software Developer

Infosys

ABSTRACT

Market Maven is an AI-driven sales forecasting solution designed to empower supermarkets with smarter decision-making capabilities. Leveraging advanced machine learning algorithms, this system predicts sales trends by analyzing historical sales data, product attributes, customer demographics, and external factors such as seasonal demand and market conditions. The solution integrates data preprocessing techniques, feature engineering, and hyper parameter-tuned models, such as Random Forest Regressions, to provide accurate and actionable insights.

By normalizing data, encoding categorical variables, and addressing missing values, Market Maven ensures robust and reliable predictions. It incorporates feature importance analysis to identify key factors influencing sales, enabling supermarket managers to optimize inventory, pricing strategies, and promotional campaigns. The system also detects anomalies, mitigates potential losses, and supports strategic decision-making to maximize profitability.

Designed with scalability and user-friendliness in mind, Market Maven outputs intuitive visualizations and detailed reports, offering a seamless interface for data-driven business operations. This project represents a significant advancement in retail analytics, demonstrating the transformative power of AI in enhancing supermarket efficiency and competitiveness.

INDEX

CHAPTER 1

Understand about the topic: AI-Driven sales forecasting for smarter supermarket

INTRODUCTION

The goal of every supermarket is to make profit. This is achieved when more goods are sold and the turnover is high. A major challenge to increasing sales of a supermarket lies in the ability of the manager to forecast sales pattern and know readily beforehand when to order and replenish inventories as well as plan for manpower and staffs. The amount of sales data has steadily been on the increase in recent years and the ability to leverage this gold of data separates high performing supermarket from the others. One of the most valuable assets a supermarket can have is data generated by customers as they interact with various supermarkets. Within these data, lies important patterns and variables that can be modeled using a machine learning algorithm; and this can to a very high degree of accuracy correctly forecast sales [1][2]. There exist several techniques to forecasting supermarket sales and historically, many supermarkets have relied on these traditional statistical models [3]. However, machine learning has grown to be an important area of data science that has gained ground due to its high predictive and forecasting powers and as such as become the go-to for highly accurate sales forecasting as well as other important areas [3] [4,[5]. To correctly forecast a future event, a machine learning model is trained on data from which it learns patterns that are used to predict future instances. An accurate forecasting model can greatly increase supermarket revenue and is generally of great importance to the organization as it improves profit as well as provides insights into the way customers can be better served [3].

1.1 Project Definition:

The project focuses on leveraging advanced machine learning algorithms and big data analytics to provide precise, actionable predictions of sales trends. By integrating this solution into supermarket operations, businesses can optimize inventory management, minimize waste, and improve profitability while meeting customer demands effectively. The primary objective of the project is to develop a robust AI-driven system that forecasts sales patterns with high accuracy, enabling supermarkets to make data-informed decisions.

1.2 Project Overview :

The supermarket industry faces dynamic challenges in maintaining optimal inventory levels and meeting customer demands while minimizing operational costs. Traditional sales forecasting methods often fail to capture the complexity of modern retail environments, leading to inefficiencies such as overstocking, stockouts, and increased waste.

Market Maven is an innovative AI-driven solution designed to transform the way supermarkets forecast sales and manage inventory. By harnessing the power of advanced machine learning algorithms and data analytics, this system provides accurate, data-informed sales predictions. These insights enable supermarkets to respond proactively to changing market conditions, optimize their supply chains, and enhance the overall customer experience.

1.3 Software Requirements:

1. **Operating System:**
    - Windows 10/11, macOS, or Linux
2. **Programming Languages and Frameworks:**
    - Python
    - Libraries: Pandas, NumPy, Scikit-learn, TensorFlow/PyTorch, Matplotlib/Seaborn
    - Web Framework: Django or Flask
3. **Database Management System:**
    - PostgreSQL, MySQL, or MongoDB
4. **Cloud Platform :**
    - AWS, Google Cloud, or Microsoft Azure
5. **Data Visualization Tools:**
    - Plotly, Tableau, or integrated web-based visualizations (e.g., Dash)
6. **Version Control System:**

- Git (with GitHub or GitLab for repository management)

7. **Integrated Development Environment (IDE):**
   - PyCharm, VS Code, or Jupyter Notebook
8. **APIs and Integration Tools:**
   - REST APIs for external data sources (e.g., weather, market trends)
   - Integration with POS (Point of Sale) systems for real-time sales data

1.4 Hardware Requirements:

1. **For Development Environment:**
   - **Processor:** Intel i5/i7 or AMD Ryzen 5/7 (or higher)
   - **RAM:** 16 GB (minimum)
   - **Storage:** 512 GB SSD (minimum)
   - **GPU:** NVIDIA GTX 1050 or higher (for model training and testing)
2. **For Deployment Environment:**
   - **Server Configuration (Cloud or On-Premises):**
     - **Processor:** 8-Core CPU (Intel Xeon or equivalent)
     - **RAM:** 32 GB or higher
     - **Storage:** 1 TB SSD (minimum)
     - **GPU:** NVIDIA Tesla T4 or A100 (for training/deployment of complex models)
3. **Network Requirements:**
   - High-speed internet
4. **Backup and Storage:**
   - External or cloud storage with at least 2 TB capacity for data backup.
5. **End-User Hardware:**
   - Devices with a modern browser (desktop, laptop, tablet) for accessing the dashboard.
   - Optional: Tablets for in-store inventory management integration.

CHAPTER 2

LITERATURE SURVEY

2.1Existing System:

- Many supermarkets still rely on basic tools like spreadsheets to analyze historical sales data manually. They are time-consuming, prone to human error, and lacks real-time adaptability.
- Techniques like linear regression, time-series analysis (e.g., ARIMA), and moving averages are used for demand forecasting. These methods struggle to capture complex patterns, nonlinear trends, or external factors like weather or events.
- Challenges such as fragmented data, scalability issues, and high costs still limit widespread adoption of AI-driven sales forecasting system in smaller supermarkets.

2.2Proposed System:

- The proposed system combines predictive power, real time data processing and automation to optimize supermarkets operations, reduce waste, improve inventory management and provide a better shopping experience.
- It aims to create a more intelligent, responsive, and customer-centric supermarket environment. Incorporate ensemble learning techniques to combine multiple models and improve prediction reliability.
- Supermarket use machine learning algorithms such as decision trees, random forests, and gradient boosting for more accurate sales predictions, relying on historical data.

CHAPTER 3

METHODOLOGY

Understand the terms: Classification, Regression, Clustering, Time series Modelling.

Classification: In **AI-driven sales forecasting** for supermarkets, **classification** plays a critical role in analysing and predicting sales trends, identifying consumer behaviours, and segmenting products or customers into meaningful groups. By applying classification techniques, supermarkets can make better business decisions about inventory, promotions, and customer engagement strategies.

Classification Techniques in Sales Forecasting:

- **Algorithms**: Common algorithms for classification in sales forecasting include:
    - **Decision Trees**: Useful for classifying products or periods based on decision rules from historical data.
    - **Random Forests**: Helpful for improving accuracy and handling a variety of features in customer behaviour or product types.
    - **Neural Networks**: Useful when there's a large amount of complex data, like customer transaction history or real-time demand patterns.
    - **Support Vector Machines (SVM)**: Effective for defining precise boundaries in high-dimensional customer or product data.

Regression: In **AI-driven sales forecasting in supermarkets**, **regression** is a technique used to predict continuous numerical outcomes, such as the expected sales volume or revenue for a given product or category. Unlike classification, which assigns data to categories, regression outputs a real number (e.g., the quantity of an item sold or the total sales amount). In sales forecasting, regression models analyse historical sales data and other related variables to make accurate predictions about future sales, allowing supermarkets to optimize inventory, reduce waste, and improve revenue.

1.Linear Regression: Basic approach; predicts sales based on a single or multiple factors like time or ad spend.

2.Polynomial Regression: Captures non-linear trends (e.g., seasonality or product lifecycle effects).

3.Ridge & Lasso Regression: Regularized models; reduce overfitting in complex datasets and perform feature selection to focus on key predictors.

4.Logistic Regression: For predicting categorical outcomes (e.g., likelihood of meeting a sales target).

5.Time Series Models:

- **ARIMA**: Predicts based on historical trends.
- **SARIMA**: Adds seasonality; ideal for cyclical sales data.
- **Exponential Smoothing**: Captures trends and seasonality in stable sales patterns.

6.Decision Trees & Ensembles:

- **Random Forests**: Ensemble of trees; handles non-linear relationships well.
- **Gradient Boosting**: Sequentially improves predictions; effective for complex patterns.

7.Support Vector Regression (SVR): Handles noisy data; ideal for outlier-prone sales data.

8.Neural Networks:

- **LSTMs**: Capture long-term patterns; ideal for time-series forecasting with complex trends.

9.Bayesian Regression: Provides predictions with confidence intervals, useful for uncertain markets.

10.Quantile Regression: Predicts specific sales scenarios, e.g., high-demand periods.

11.PCR & PLS: Reduce dimensionality in high-feature datasets, improving accuracy by focusing on key factors.

Clustering: Clustering plays an essential role in AI-driven sales forecasting by helping to segment data into meaningful groups, enabling more accurate forecasting based on these segments. By identifying patterns among customer behaviours, product types, or time periods, clustering can uncover valuable insights that improve forecast quality.

Clustering Techniques in Sales Forecasting:

- **K-means Clustering**: Often used for customer and product segmentation due to its efficiency and simplicity.

- **Hierarchical Clustering**: Useful for creating a nested hierarchy of clusters, allowing for multi-level segmentation.
- **DBSCAN**: This density-based clustering method can handle noise in sales data and is useful for detecting anomalies.

Time series Modelling:

Time series modelling is crucial in AI-driven sales forecasting because it captures patterns in sales data over time, allowing businesses to make accurate predictions based on past behaviours. Here's a summary of key time series modelling techniques used in sales forecasting.

Text pre-processing

Text pre-processing is the first step in preparing raw text data for analysis or modeling. It involves cleaning and transforming the text to reduce noise and structure it in a way that algorithms can process effectively. In natural language processing (NLP) and text analytics, pre-processing is essential to improve the accuracy and efficiency of tasks like classification, sentiment analysis, translation, and summarization.

Text pre-processing include:

1. **Lowercasing**: Converting all text to lowercase to avoid treating words like "Apple" and "apple" as different entities, unless case distinctions are important for the application.
2. **Removing Punctuation and Special Characters**: Eliminating symbols (e.g., commas, periods, hashtags) that may not contribute meaningful information to the analysis. Special characters often add noise to text data.
3. **Removing Stop Words**: Excluding common, uninformative words (e.g., "the," "is," "at") that don't carry significant meaning but occur frequently. Removing stop words helps focus on more meaningful content.
4. **Tokenization**: Splitting text into individual words, phrases, or tokens. This step enables analyzing each part of the text independently.
5. **Stemming and Lemmatization**: Reducing words to their base forms. Stemming cuts words to their root (e.g., "running" to "run"), while lemmatization uses context to convert words to their canonical form (e.g., "better" to "good").
6. **Handling Outliers and Rare Words:** Identify and manage outliers or rare words that could introduce noise, such as uncommon slang in reviews, misspellings, or unusual product descriptions.
7. **Handling Spelling and Grammar Issues**: Correcting misspellings or normalizing grammar can improve consistency, especially if there are frequent spelling errors or informal language.

8. **Removing Numeric Values or Converting Them**: Depending on the application, numbers can be removed, retained, or converted to words if they hold significance. For example, “2024” might be converted to “year 2024.”
9. **Removing Duplicate Text**: Filtering out duplicate entries in the dataset helps avoid bias in the analysis.

Preprocessing dataset

1) Load the excel file: It is not always possible to get the dataset in CSV format. So, Pandas provides us the functions to convert datasets in other formats to the Data frame. An excel file has a ‘.xlsx’ format.

2) Convert Date and Create Date Time Column: Date Time is a collection of dates and times in the format of “yyyy-mm-dd HH:MM:SS” where yyyy-mm-dd is referred to as the date and HH:MM:SS is referred to as Time.

3) Handle Missing Values: Using the dropna() function is the easiest way to remove observations or features with missing values from the data frame.

4) Encode Categorical Variables: Categorical features are variables that can take on a limited, fixed number of values or categories. These features are commonly encountered in datasets and can present challenges when working with machine learning algorithms, as many algorithms require numerical input.

5) Feature Engineering: Date-Time Features: Feature engineering is the process of creating features (also called "attributes") that don't already exist in the dataset. This means that if your dataset already contains enough "useful" features, you don't necessarily need to engineer 11 additional features. In other words, it's a feature that helps your model to make better predictions!

6) Add additional columns: Create entirely new features based on business logic or domain-specific calculations.

7) Log transformation: Log transformation is a data transformation method in which it replaces each variable x with a log(x). When our original continuous data do not follow the bell curve, we can log transform this data to make it as “normal” as possible so that the statistical analysis results from this data become more valid.

8) Interaction Feature (Quantity * Unit Price): It was also known as feature interaction. Feature interaction can be described as a phenomenon that involves two or more features meeting and influencing each other during a prediction task. Owing to the presence of feature interaction,

the overall prediction performance of a model is not equal to a simple sum of the performance of each constituent feature.

9) Scaling Numerical Features: Feature Scaling is a technique to standardize the independent features present in the data in a fixed range. It is performed during the data pre-processing to handle highly varying magnitudes or values or units. If feature scaling is not done, then a machine learning algorithm tends to weigh greater values, higher and consider smaller values as the lower values, regardless of the unit of the values.

10) save preprocessed data in excel file: We can save the data in an excel file.

Exploratory Data Analysis(EDA)

Introduction to EDA

The main objective of this article is to cover the steps involved in Data pre-processing, Feature Engineering, and different stages of Exploratory Data Analysis, which is an essential step in any research analysis. Data pre-processing, Feature Engineering, and EDA are fundamental early steps after data collection. Still, they are not limited to where the data is simply visualized, plotted, and manipulated, without any assumptions, to assess the quality of 12 the data and build models. This article will guide you through data pre-processing, feature engineering, and exploratory data analysis (EDA) using Python.

What is Exploratory Data Analysis?

Exploratory Data Analysis (EDA) is a method of analyzing datasets to understand their main characteristics. It involves summarizing data features, detecting patterns, and uncovering relationships through visual and statistical techniques. EDA helps in gaining insights and formulating hypotheses for further analysis. [23]

Types of Exploratory Data Analysis EDA

1.Univariate Analysis

Univariate analysis focuses on a single variable to understand its internal structure. It is primarily concerned with describing the data and finding patterns existing in a single feature.

Histograms: Used to visualize the distribution of a variable.

Box plots: Useful for detecting outliers and understanding the spread and skewness of the data.

Bar charts: Employed for categorical data to show the frequency of each category.

2.Bivariate Analysis

Bivariate evaluation involves exploring the connection between variables. It enables find associations, correlations, and dependencies between pairs of variables. Bivariate analysis is a crucial form of exploratory data analysis that examines the relationship between two variables.

Scatter Plots: These are one of the most common tools used in bivariate analysis. A scatter plot helps visualize the relationship between two continuous variables.

Correlation Coefficient: This statistical measure (often Pearson's correlation coefficient for linear relationships) quantifies the degree to which two variables are related.

Cross-tabulation: Also known as contingency tables, cross-tabulation is used to analyze the relationship between two categorical variables. It shows the frequency distribution of categories of one variable in rows and the other in columns, which helps in understanding the relationship between the two variables.

Line Graphs: In the context of time series data, line graphs can be used to compare two variables over time. This helps in identifying trends, cycles, or patterns that emerge in the interaction of the variables over the specified period.

Covariance: Covariance is a measure used to determine how much two random variables change together. However, it is sensitive to the scale of the variables, so it's often supplemented by the correlation coefficient for a more standardized assessment of the relationship.

3.Multivariate Analysis

Multivariate analysis examines the relationships between two or more variables in the dataset. It aims to understand how variables interact with one another, which is crucial for most statistical modeling techniques. Techniques include:

Pair plots: Visualize relationships across several variables simultaneously to capture a comprehensive view of potential interactions.

Principal Component Analysis (PCA): A dimensionality reduction technique used to reduce the dimensionality of large datasets, while preserving as much variance as possible.

Tools for Performing Exploratory Data Analysis

Exploratory Data Analysis (EDA) can be effectively performed using a variety of tools and software, each offering unique features suitable for handling different types of data and analysis requirements.

Python Libraries

Pandas: Provides extensive functions for data manipulation and analysis, including data structure handling and time series functionality.

Matplotlib: A plotting library for creating static, interactive, and animated visualizations in Python.

Seaborn: Built on top of Matplotlib, it provides a high-level interface for drawing attractive and informative statistical graphics.

Plotly: An interactive graphing library for making interactive plots and offers more sophisticated visualization capabilities.

Steps for performing EDA

Step 1: Understand the Problem and the Data: Define the objective and understand the data structure and requirements.

Step 2: Import and Inspect the Data: Load the data into the environment and check its structure, types, and sample values.

Step 3: Handle Missing Data: Identify and deal with missing values using imputation or removal techniques.

Step 4: Explore Data Characteristics: Analyze key statistics, distributions, and relationships in the data.

Step 5: Perform Data Transformation: Clean and modify the data (e.g., normalization, encoding) for better analysis.

Step 6: Visualize Data Relationships: Use charts and graphs to identify patterns and correlations between variables.

Step 7: Handling Outliers: Detect and address extreme values that could distort the analysis.

Step 8: Communicate Findings and Insights: Summarize and present the results clearly to stakeholders.

Feature Engineering

Feature Engineering is the process of creating new features or transforming existing features to improve the performance of a machine-learning model. It involves selecting relevant information from raw data and transforming it into a format that can be easily understood by a model. The goal is to improve model accuracy by providing more meaningful and relevant information.

Steps in Feature Engineering:

The steps of feature engineering may vary as per different data scientists and ML engineers. However, there are some common steps that are involved in most machine learning algorithms, and these steps are as follows:

Data Preparation: The first step is data preparation. In this step, raw data acquired from different resources are prepared to make it in a suitable format so that it can be used in the ML model. The data preparation may contain cleaning of data, delivery, data augmentation, fusion, ingestion, or loading.

Exploratory Analysis: Exploratory analysis or Exploratory data analysis (EDA) is an important step of features engineering, which is mainly used by data scientists. This step involves analysis, investing dataset, and summarization of the main characteristics of data. Different data visualization techniques are used to better understand the manipulation of data sources, to find the most appropriate statistical technique for data analysis, and to select the best features for the data.

Benchmark: Benchmarking is a process of setting a standard baseline for accuracy to compare all the variables from this baseline. The benchmarking process is used to improve the predictability of the model and reduce the error rate.

There are two main approaches to feature engineering for most tabular datasets:

The checklist approach: using tried and tested methods to construct features.

The domain-based approach: incorporating domain knowledge of the dataset's subject matter into constructing new features.

Dataset Understanding

Data is the foundation of any AI-driven system, and it is essential to collect, organize, and clean critical sales data before feeding it into the AI sales forecasting tools.

Effective demand forecasting relies on diverse and comprehensive data inputs, including:

Historical sales data: Detailed records of past sales performance, which help identify trends and patterns essential for forecasting.

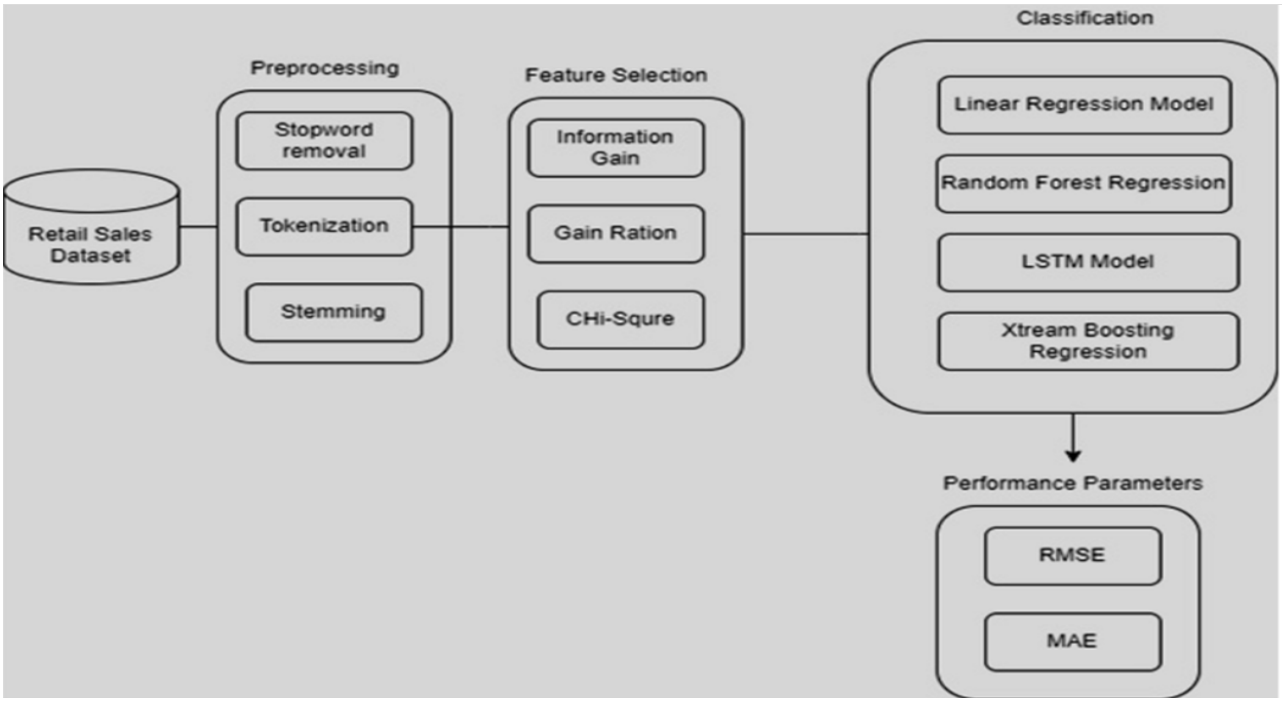
Market trends data: Insights into current market conditions and emerging trends that affect consumer demand.

Consumer behavior data: Data from customer interactions and purchasing behaviors to understand preferences and predict future buying trends.

Competitor activity: Insights into competitor promotions, pricing strategies, and product launches, which can influence market dynamics

CHAPTER 4

SYSTEM ARCHITECTURE



1. Retail Sales Dataset

- **Purpose**: This serves as the input data source containing historical sales records and related information (e.g., products, stores, dates, and external factors like promotions and weather).
- **Role**: Provides raw data for preprocessing and further analysis.

2. Preprocessing

- **Stopword Removal**: Eliminates irrelevant words or information (e.g., "and," "the") to clean the text fields or descriptions.
- **Tokenization**: Breaks down data (such as text fields) into smaller meaningful units for analysis.
- **Stemming**: Reduces words to their root form (e.g., "selling" becomes "sell") to unify similar terms.

3. Feature Selection

- **Information Gain**: Measures the significance of a feature by evaluating how much it contributes to reducing uncertainty.
- **Gain Ratio**: A normalization of information gain to address bias towards features with more levels.
- **Chi-Square Test**: A statistical test to assess the relationship between features and the target variable (e.g., sales).

4. Classification/Regression

- **Linear Regression Model**: A basic predictive model that assumes a linear relationship between features and the target variable.
- **Random Forest Regression**: An ensemble method that combines multiple decision trees to improve accuracy and handle non-linear relationships.
- **LSTM Model (Long Short-Term Memory)**: A type of recurrent neural network (RNN) suitable for time-series forecasting, capable of capturing long-term dependencies in sales data.
- **XGBoost Regression**: A powerful gradient-boosting framework that is efficient, scalable, and excels at handling large datasets with complex patterns.

5. Performance Parameters

- **RMSE (Root Mean Square Error)**: Measures the average magnitude of errors in predictions, with greater penalties for larger errors.
- **MAE (Mean Absolute Error)**: Calculates the average absolute error, providing a straightforward measure of prediction accuracy.

The model with the best performance metrics (lowest RMSE and MAE) is chosen for deployment in the sales forecasting system.

CHAPTER 5

IMPLEMENTATION

The implementation of the **Market Maven** system involves systematic execution of the proposed methodology, integrating data preprocessing, machine learning models, and real-time forecasting tools.

1. Data Preparation

- **Data Collection**:
  - Gather historical sales data, inventory records, customer purchase patterns, and external factors like promotions and weather.
  - Store data in a centralized database using cloud platforms (e.g., AWS, Azure, or Google Cloud).
- **Data Cleaning**:
  - Handle missing values, remove duplicates, and correct inconsistencies.
  - Normalize numerical features and encode categorical variables.
- **Data Preprocessing**:
  - Stopword Removal**,** Tokenization**,** and Stemming for textual data.
  - Extract time-based features (e.g., day of the week, season) and external variables (e.g., holidays).

2. Feature Engineering

- Perform feature selection using techniques such as:
  - Information Gain**,** Gain Ratio**,** and Chi-Square Tests**.**
  - Remove irrelevant or redundant features to enhance model efficiency.
- Add derived features like moving averages, lagged sales, and trends for better prediction accuracy.

3. Model Development

- **Model Selection**:
  - Implement multiple algorithms to handle regression-based sales forecasting:
    - **Linear Regression** for a baseline model.
    - **Random Forest Regressor** for capturing non-linear relationships.

        - **LSTM (Long Short-Term Memory)** for time-series forecasting.
        - **XGBoost Regressor** for complex, large-scale datasets.
- **Model Training**:
    - Split data into training, validation, and test sets.
    - Use cross-validation to avoid overfitting.
    - Perform hyperparameter tuning (e.g., grid search or random search) to optimize model performanc.
- **Model Evaluation**:
    - Measure model accuracy using metrics such as **RMSE (Root Mean Square Error)** and **MAE (Mean Absolute Error)**.
    - Compare models to select the best-performing algorithm.

4. System Integration

- **Real-Time Data Pipeline**:
    - Build an ETL pipeline to ingest real-time data from POS systems, weather APIs, and inventory logs.
    - Use frameworks like Apache Kafka or Apache Airflow for seamless data integration.
- **Forecasting Module**:
    - Deploy the selected machine learning model to predict sales at SKU, category, or store levels.
    - Update forecasts dynamically as new data arrives.
- **Inventory Optimization**:
    - Integrate forecasting outputs with inventory systems to automate replenishment orders and reduce stockouts.

5. Dashboard and Reporting

- **Interactive Dashboards**:
    - Use tools like **Tableau**, **Power BI**, or custom web applications (built with frameworks like Flask or Django) to display:
        - Predicted sales trends.
        - Inventory alerts and reorder points.
        - Key performance indicators (KPIs).
- **Actionable Insights**:
    - Provide recommendations for pricing, promotions, and inventory management through user-friendly interfaces.

6. Deployment

- Deploy the system on a scalable cloud platform (e.g., AWS Sagemaker, Google AI Platform).
- Implement APIs to connect the forecasting system with supermarket management software.

7. Monitoring and Maintenance

- **Performance Monitoring**:
  - Continuously track forecast accuracy using RMSE/MAE metrics.
  - Identify and address any drift in model performance due to changes in data patterns.
- **Regular Updates**:
  - Retrain models periodically with new data.
  - Update features and algorithms as needed to adapt to evolving business needs.

CHAPTER 6

MODELTRAINING

Random forest is a supervised learning algorithm. It builds a forest with an ensemble of decision trees. It is an easy to use machine learning algorithm that produces a great result most of the time even without hyperparameter tuning. Random Forests can be used for both classification and regression tasks. Random Forests work well with both categorical and numerical data. No scaling or transformation of variables is usually necessary. Random Forests implicitly perform feature selection and generate uncorrelated decision trees. It does this by choosing a random set of features to build each decision tree. This also makes it a great model when you have to work with a high number of features in the data. Random Forests are not influenced by outliers to a fair degree. It does this by binning the variables. Random Forests can handle linear and non-linear relationships well. Random Forests generally provide high accuracy and balance the bias-variance trade-off well. Since the model's principle is to average the results across the multiple decision trees it builds, it averages the variance as well.

Random forest creates decision trees that are independent of each other using random samples of the data through "bagging", which results in a "forest" of decision trees. When each of the trees in the forest produce their own predictions, a vote occurs with the majority prediction class resulting in the final prediction of the model.

HowDoes RandomForest Work?

The random Forest algorithm works in several steps which are discussed below

Ensemble of Decision Trees: Random Forest leverages the power of ensemble learning by constructing an army of Decision Trees. These trees are like individual experts, eachspecializing in a particular aspect of the data. Importantly, they operate independently, minimizing the risk of the model being overly influenced by the nuances of a single tree.

Random Feature Selection: To ensure that each decision tree in the ensemble brings a unique perspective, Random Forest employs random feature selection. During the training of each tree, a random subset of features is chosen. This randomness ensures that each tree focuses on different aspects of the data, fostering a diverse set of predictors within the ensemble.

Bootstrap Aggregating or Bagging: The technique of bagging is a cornerstone of Random Forest's training strategy which involves creating multiple bootstrap samples from the original dataset, allowing instances to be sampled with replacement. This results in different subsets of

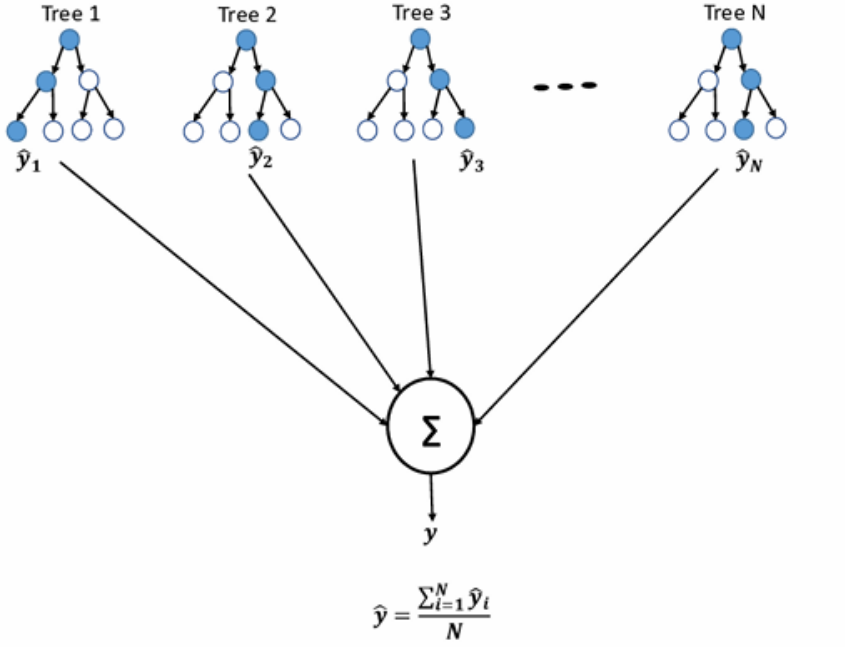
data for each decision tree, introducing variability in the training process and making the model more robust.

Decision Making and Voting: When it comes to making predictions, each decision tree in the Random Forest casts its vote. For classification tasks, the final prediction is determined by the mode (most frequent prediction) across all the trees. In regression tasks, the average of the individual tree predictions is taken. This internal voting mechanism ensures a balanced and collective decision-making process.

We can specify the number of decision trees using 'n_estimators' and the maximum depth per decision tree using 'max_depth'. Once we have specified the parameters of the Random Forest, we can then fit the model on the training data and predict the output sale difference values

Now we need to revert the original scale of the predicted sale values using the 'inverse_transform' function of the MinMaxScaler()

Now, we can evaluate the model metrics by comparing the predicted sale amount with the actual sale value



CHAPTER 7

SOURCE CODE

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from datetime import datetime

# Load the dataset
data = pd.read_csv('SuperKart.csv')  # Replace with your dataset filename

# Handle missing values
data['Product_Weight'].fillna(data['Product_Weight'].mean(), inplace=True)
data['Product_Sugar_Content'].fillna('Unknown', inplace=True)
data['Product_MRP'].fillna(data['Product_MRP'].mean(), inplace=True)
data['Store_Size'].fillna(data['Store_Size'].mode()[0], inplace=True)

# Encode categorical variables
encoder = LabelEncoder()
categorical_columns = ['Product_Type', 'Store_Type', 'Store_Location_City_Type',
'Product_Sugar_Content']
for col in categorical_columns:
    data[col] = encoder.fit_transform(data[col])

# Feature engineering: Calculate Store Age
current_year = datetime.now().year
data['Store_Age'] = current_year - data['Store_Establishment_Year']

# Normalize numerical data
scaler = MinMaxScaler()
numerical_columns = ['Product_Weight', 'Product_MRP', 'Product_Store_Sales_Total']
data[numerical_columns] = scaler.fit_transform(data[numerical_columns])

# Save the preprocessed data to an Excel file
output_file = 'preprocessed_data.xlsx'
data.to_excel(output_file, index=False)
print(f"Preprocessed data saved to {output_file}")
```

Preprocessed data saved to preprocessed_data.xlsx

```
import pandas as pd
```

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from datetime import datetime
import numpy as np

# Load the dataset
data = pd.read_csv('SuperKart.csv')  # Replace with your dataset filename

# Handle missing values
data['Product_Weight'].fillna(data['Product_Weight'].mean(), inplace=True)
data['Product_Allocated_Area'].fillna(data['Product_Allocated_Area'].mean(), inplace=True)
data['Product_Sugar_Content'].fillna('Unknown', inplace=True)  # Handle missing sugar content
data['Product_MRP'].fillna(data['Product_MRP'].mean(), inplace=True)
data['Store_Size'].fillna(data['Store_Size'].mode()[0], inplace=True)

# Convert Product_Sugar_Content to numeric
sugar_mapping = {
    'No Sugar': 0,
    'Low Sugar': 1,
    'Medium': 2,
    'High Sugar': 3,
    'Unknown': -1
}  # Define mapping
data['Product_Sugar_Content'] = data['Product_Sugar_Content'].map(sugar_mapping)

# Convert Store_Size to numeric
size_mapping = {'Small': 0, 'Medium': 1, 'Large': 2, 'Unknown': -1}
data['Store_Size'] = data['Store_Size'].map(size_mapping)

# Encode categorical variables
encoder = LabelEncoder()
categorical_columns = ['Product_Type', 'Store_Type', 'Store_Location_City_Type']
for col in categorical_columns:
    if col in data.columns:
        data[col] = encoder.fit_transform(data[col])

# Feature engineering: Calculate Store Age
current_year = datetime.now().year
```

```
data['Store_Age'] = current_year - data['Store_Establishment_Year']

# Drop irrelevant columns
irrelevant_columns = ['Product_Id', 'Store_Id', 'Store_Establishment_Year']  # Exclude identifier columns
data = data.drop(columns=irrelevant_columns)

# Normalize numerical data
scaler = MinMaxScaler()
numerical_columns = ['Product_Weight', 'Product_MRP', 'Store_Age', 'Product_Allocated_Area']
data[numerical_columns] = scaler.fit_transform(data[numerical_columns])

# Define features (X) and target (y)
if 'Product_Store_Sales_Total' in data.columns:
    X = data.drop(columns=['Product_Store_Sales_Total'])
    y = data['Product_Store_Sales_Total']
else:
    raise ValueError("'Product_Store_Sales_Total' column is missing in the dataset.")

# Verify data types (ensure all columns are numeric)
print("Data Types After Preprocessing:")
print(X.dtypes)

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train a Random Forest Regressor
model = RandomForestRegressor(random_state=42)
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)
# Evaluate the model
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

# Print evaluation metrics
```

```
print("\nModel Evaluation Results:")
print(f"Mean Absolute Error (MAE): {mae}")
print(f"Mean Squared Error (MSE): {mse}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"R-squared (R2): {r2}")

# Save predictions and evaluation results to Excel
results = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
results_file = 'model_results.xlsx'
results.to_excel(results_file, index=False)


print(f"\nModel predictions and results saved to {results_file}")
```

```
Data Types After Preprocessing:
Product_Weight              float64
Product_Sugar_Content       float64
Product_Allocated_Area      float64
Product_Type                int64
Product_MRP                 float64
Store_Size                  float64
Store_Location_City_Type    int64
Store_Type                  int64
Store_Age                   float64
dtype: object

Model Evaluation Results:
Mean Absolute Error (MAE): 109.46834911580144
Mean Squared Error (MSE): 81317.92994120122
Root Mean Squared Error (RMSE): 285.162988378929
R-squared (R2): 0.9287322480091669

Model predictions and results saved to model_results.xlsx
```

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import pandas as pd
import numpy as np
from scipy.stats import zscore
```

```
from sklearn.preprocessing import MinMaxScaler

# Load the dataset
data = pd.read_csv('SuperKart.csv')  # Replace with your dataset file

# Handle missing values
data['Product_Weight'].fillna(data['Product_Weight'].mean(), inplace=True)
data['Product_Allocated_Area'].fillna(data['Product_Allocated_Area'].mean(), inplace=True)
data['Product_Sugar_Content'].fillna('Unknown', inplace=True
data['Product_MRP'].fillna(data['Product_MRP'].mean(), inplace=True)
data['Store_Size'].fillna(data['Store_Size'].mode()[0], inplace=True)

# Map categorical string columns to numeric
sugar_mapping = {'No Sugar': 0, 'Low Sugar': 1, 'Medium': 2, 'High Sugar': 3, 'Unknown': -1}
data['Product_Sugar_Content'] = data['Product_Sugar_Content'].map(sugar_mapping)

size_mapping = {'Small': 0, 'Medium': 1, 'Large': 2, 'Unknown': -1}
data['Store_Size'] = data['Store_Size'].map(size_mapping)

# Label encode categorical columns
from sklearn.preprocessing import LabelEncoder
categorical_columns = ['Product_Type', 'Store_Type', 'Store_Location_City_Type']
encoder = LabelEncoder()
for col in categorical_columns:
    data[col] = encoder.fit_transform(data[col])

# Feature engineering: Add store age
data['Store_Age'] = 2024 - data['Store_Establishment_Year']

# Drop irrelevant columns
data = data.drop(columns=['Product_Id', 'Store_Id', 'Store_Establishment_Year'])

# Handle outliers using Z-score
numerical_columns = ['Product_Weight', 'Product_Allocated_Area', 'Product_MRP',
'Store_Age', 'Product_Store_Sales_Total']
for col in numerical_columns:
    z_scores = zscore(data[col])
    data = data[(np.abs(z_scores) < 3)]  # Keep rows where z-score is less than 3

# Normalize numerical columns
scaler = MinMaxScaler()
```

```
data[numerical_columns] = scaler.fit_transform(data[numerical_columns])

# Split data into features (X) and target (y)
X = data.drop(columns=['Product_Store_Sales_Total'])
y = data['Product_Store_Sales_Total']

# Split into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Hyperparameter tuning for Random Forest with RandomizedSearchCV
param_grid = {

    'n_estimators': [100, 200],
    'max_depth': [10, 20, None],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2]
}
rf = RandomForestRegressor(random_state=42)
random_search = RandomizedSearchCV(estimator=rf, param_distributions=param_grid,
n_iter=10, cv=3, n_jobs=-1, scoring='r2')
random_search.fit(X_train, y_train)

# Best model after tuning
best_model = random_search.best_estimator_

# Make predictions
y_pred = best_model.predict(X_test)

# Evaluate the model
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)
print("\nModel Evaluation After Hyperparameter Tuning:")
print(f"Mean Absolute Error (MAE): {mae}")
print(f"Mean Squared Error (MSE): {mse}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"R-squared (R2): {r2}")
```

```
Model Evaluation After Hyperparameter Tuning:
Mean Absolute Error (MAE): 0.016128750329528145
```

Mean Squared Error (MSE): 0.0017978834237489649
Root Mean Squared Error (RMSE): 0.042401455443757646
R-squared (R2): 0.9335912890560294

```
# Save feature combinations ranked by importance
feature_importances = pd.DataFrame({
    'Feature': X.columns,
    'Importance': best_model.feature_importances_
}).sort_values(by='Importance', ascending=False)

# Cumulative importance for selecting top features
feature_importances['Cumulative_Importance'] = feature_importances['Importance'].cumsum()

# Select features that account for a significant portion (e.g., 95%) of importance
selected_features = feature_importances[feature_importances['Cumulative_Importance'] <= 0.95]

# Save selected features to an Excel file
selected_features_file = 'selected_feature_combinations.xlsx'
selected_features.to_excel(selected_features_file, index=False)
print(f"Selected feature combinations saved to {selected_features_file}")

# Train the model again using only the selected features to validate loss reduction
X_selected = X[selected_features['Feature']]
X_train_selected, X_test_selected, y_train, y_test = train_test_split(X_selected, y, test_size=0.2, random_state=42)
# Retrain the model with selected features
best_model.fit(X_train_selected, y_train)
y_pred_selected = best_model.predict(X_test_selected)
# Evaluate the new model with reduced features
mae_selected = mean_absolute_error(y_test, y_pred_selected)
mse_selected = mean_squared_error(y_test, y_pred_selected)
rmse_selected = np.sqrt(mse_selected)
r2_selected = r2_score(y_test, y_pred_selected)

# Save results of reduced feature model
reduced_model_results = pd.DataFrame({
    'Metric': ['Mean Absolute Error (MAE)', 'Mean Squared Error (MSE)', 'Root Mean Squared Error (RMSE)', 'R-squared (R2)'],
```

```
    'Value': [mae_selected, mse_selected, rmse_selected, r2_selected]
})

reduced_model_results_file = 'reduced_model_results.xlsx'

reduced_model_results.to_excel(reduced_model_results_file, index=False)
print(f"Reduced feature model results saved to {reduced_model_results_file}")
```

```
Selected feature combinations saved to selected_feature_combinations.xlsx
Reduced feature model results saved to reduced_model_results.xlsx
```

CHAPTER 8

DATA VISUALIZATIONS

Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs, and maps, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data.

What is Data Visualization?

Data visualization translates complex data sets into visual formats that are easier for the human brain to comprehend. This can include a variety of visual tools such as:

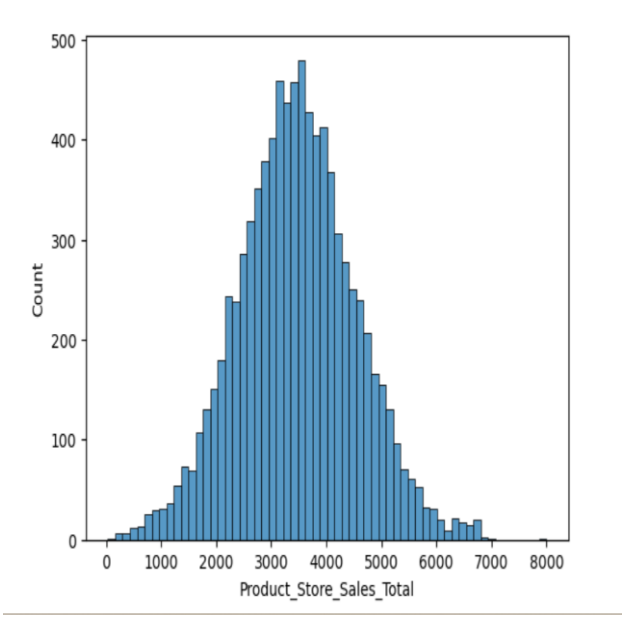
Charts: Bar charts, line charts, pie charts, etc.

Graphs: Scatter plots, histograms, etc.

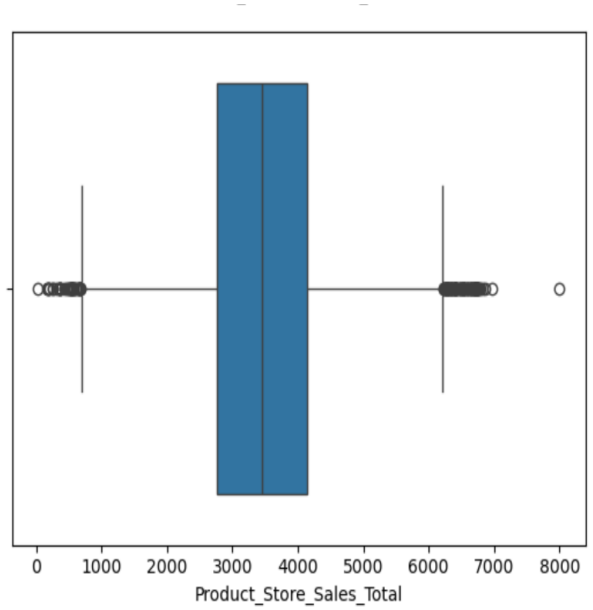
Maps: Geographic maps, heat maps, etc.

Dashboards: Interactive platforms that combine multiple visualizations.

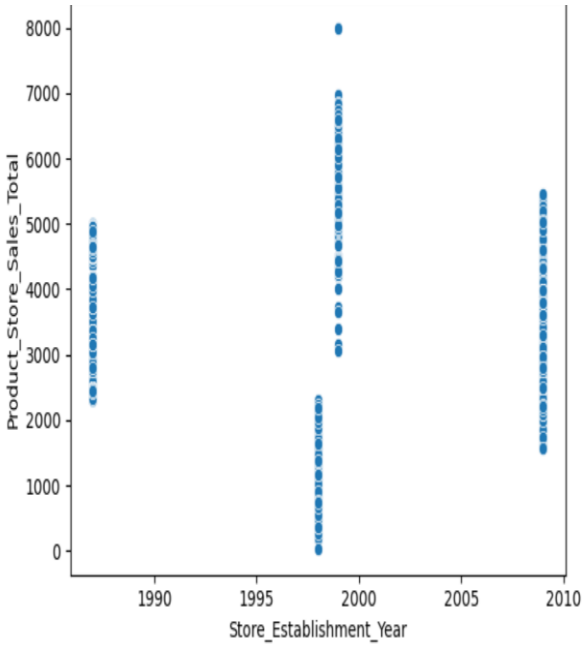
The primary goal of data visualization is to make data more accessible and easier to interpret, allowing users to identify patterns, trends, and outliers quickly. This is particularly important in the context of big data, where the sheer volume of information can be overwhelming without effective visualization techniques.



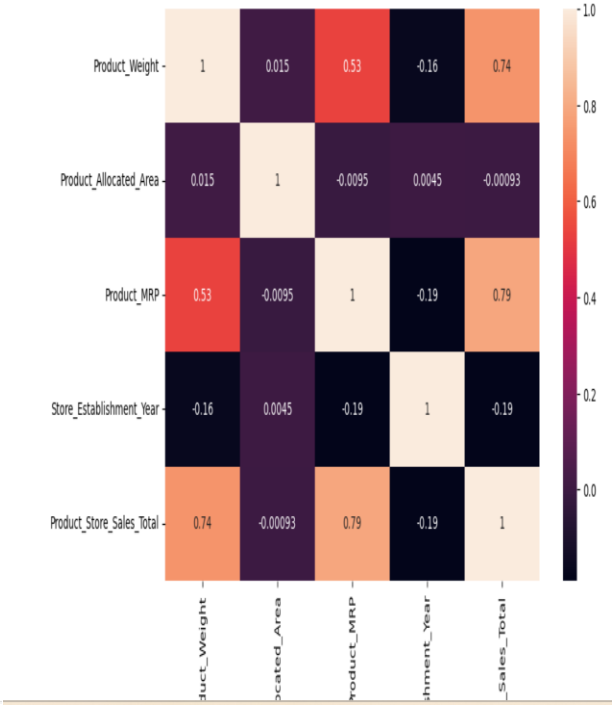
Histogram



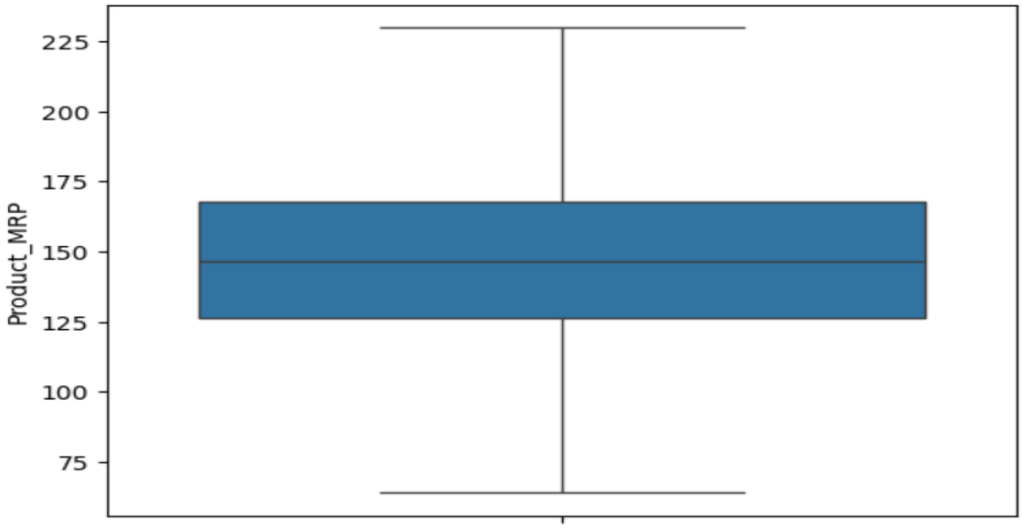
BoxplotS

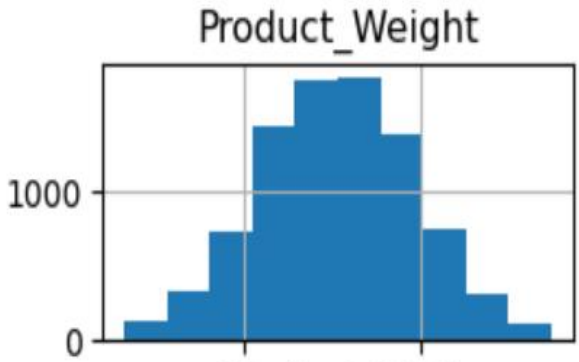


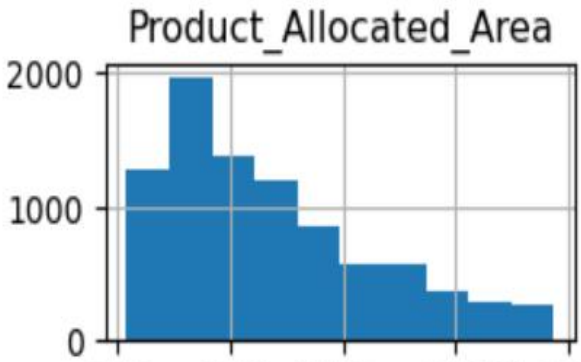
Scatter plot

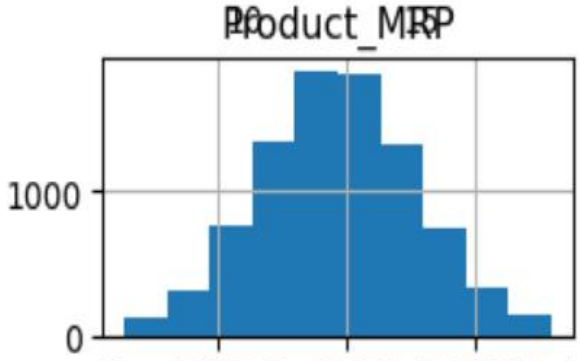


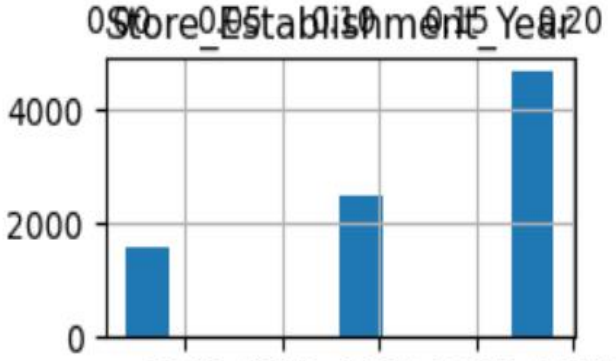
Heatmap

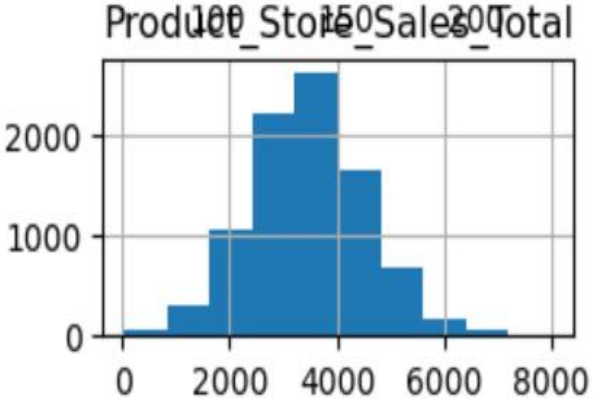

Product_MRP
225
200
175
150
125
100
75

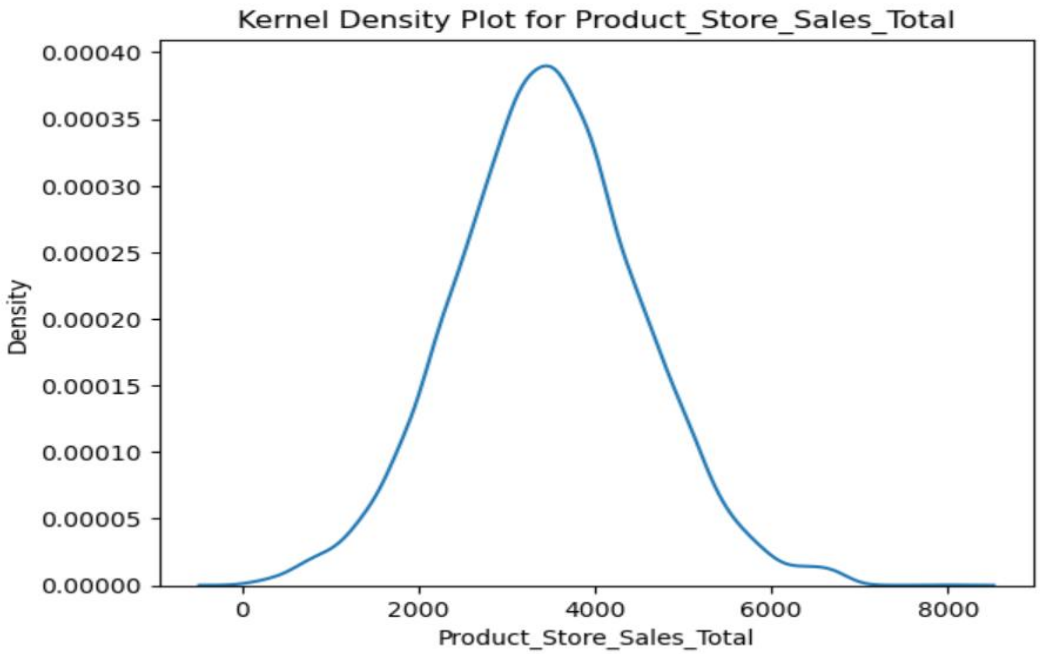

Product_Weight
1000
0


Product_Allocated_Area
2000
1000
0


Product_MRP
1000
0


Store_Establishment_Year
4000
2000
0
1990
1995
2000
2005
2010


Product_Store_Sales_Total
2000
1000
0
0
2000
4000
6000
8000


Kernel Density Plot for Product_Store_Sales_Total
Density
0.00040
0.00035
0.00030
0.00025
0.00020
0.00015
0.00010
0.00005
0.00000
0
2000
4000
6000
8000
Product_Store_Sales_Total

CONCLUSION AND FUTURESCOPE

The Market Maven system is an innovative AI-driven solution aimed at enhancing inventory management and supporting smarter decision-making for supermarkets. By leveraging a robust methodology that integrates feature engineering, time series data analysis, and advanced machine learning models, the system provides precise and actionable sales forecasts. Techniques such as Random Forest modeling and hyperparameter tuning are employed to optimize model performance and ensure high forecasting accuracy. This systematic approach enables businesses to identify key patterns in historical sales data, consumer behavior, and external factors like seasonality and economic trends. Additionally, the system is designed with scalability and adaptability in mind, making it an ideal tool not only for supermarkets but also for a variety of retail and e-commerce domains. Its ability to adapt to diverse business environments ensures that organizations can streamline inventory processes, minimize wastage, and make data-driven decisions to maximize profitability. The basics of machine learning and the associated data processing and modeling algorithms have been described, followed by their application for the task of sales prediction in Supermarket centers at different locations. On implementation, the prediction results show the correlation among different attributes considered and how a particular location of medium size recorded the highest sales, suggesting that other shopping locations should follow similar patterns for improved sales .

The project can be further collaborated in a web-based application or in any device supported with an in-built intelligence by virtue of Internet of Things (IoT), to be more feasible for use.Expanding the dataset to include additional external factors like weather, holidays, and economic indicators. Enhancing customer segmentation using clustering algorithms for targeted marketing strategies. We can implement sentiment analysis and neutral networks to enhance the model. We can modify the same system to an online learning system that adapts in real-time.

References

[1] Kim Brynjolfsson Hitt. “Strength in Numbers: How Does DataDriven Decision making Affect Firm Performance”. In:
(2011). URL: http://ebusiness.mit.edu/research/papers
[2] Orinna Cortes and Vladimir Vapnik. “Support-vector networks”. In: Machine Learning 20(3) (1995), pp. 273–297.
[3] Nari Sivanandam Arunraj and Diane Ahrens. “A hybrid seasonal autoregressive integrated moving average and quantile
regression for daily food sales forecasting”. In: International Journal Production Economics 170 (2015), pp. 321–335
[4] Philip Doganis et al. “Time series sales forecasting for short shelf life food products based on artificial neural networks and
evolutionary computing”. In: Journal of Food Engineering 75 (2006), pp. 196–20.
[5] Maike Krause-Traudes et al. Spatial data mining for retail sales forecasting. Tech.rep. Fraunhofer-Institut Intelligente
Analyse- und Informationssysteme (IAIS), 2008.