

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

B. Gowri Thanmai

Under the esteemed guidance of

Dr. N Jagan Mohan

Senior Software Developer

Infosys



TABLE OF CONTENTS

S.NO	TITLE	PAGE NO
1	INTRODUCTION	1
	1.1 BACKGROUND	2
	1.2 OBJECTIVE	2
	1.3 PROBLEM STATEMENT	2
2	SYSTEM ANALYSIS	3
	2.1 HARDWARE AND SOFTWARE REQUIREMENTS	3
	2.2 SOFTWARE REQUIREMENTS SPECIFICATION	4-5
3	SYSTEM DESIGN	6
	3.1 ARCHITECTURE	7
4	METHODOLOGY	8
	4.1 TIME SERIES FORECASTING TECHNIQUES	8
	4.2 DATASET UNDERSTANDING	9
	4.3 PREPROCESSING DATASET	10-11
	4.4 EXPLORATORY DATA ANALYSIS(EDA)	12-15
	4.5 FEATURE ENGINEERING	15-17
5	VISUALIZATIONS	18-22
6	MODEL TRAINING	23-25
7	SAMPLE CODE	26-30
8	CONCLUSION AND FUTURE SCOPE	31
9	REFERENCES	32-33

CHAPTER 1

Introduction

Many supermarkets today do not have a good forecast of their yearly sales. This is mostly due to the lack of skills, resources and knowledge to make sales estimation. At best, most supermarket chain stores use ad hoc tools and processes to analyze and predict sales for the coming year. The use of traditional statistical methods to forecast supermarket sales has left a lot of challenges unaddressed and mostly result in the creation of predictive models that perform poorly. The era of big data coupled with access to massive compute power has made machine learning a goto for sales forecast. Also, the most important variables that helped in better sales forecasts were "Supermarket type (Grocery Store), Product price and Supermarket opening year [1]. Businesses can remain profitable by embracing changing market trends, keeping up with evolving customers needs and preferences, and adopting the latest technological innovations [2]. As the retail landscape undergoes a profound transformation in the era of digitalization, the integration of Artificial Intelligence (AI) and predictive analytics has emerged as a pivotal force reshaping the industry [3]. This technology, regardless of the industry vertical, allows businesses to make informed decisions by predicting future demand, sales, and inventory levels. For global companies, these accurate projections are crucial to staying ahead of market trends, meeting customers' needs, and leveraging technological progress [4]. As the retail landscape undergoes a profound transformation in the era of digitalization, the integration of Artificial Intelligence (AI) and predictive analytics has emerged as a pivotal force reshaping the industry. In addition to outlining technological advancements, the paper emphasizes the crucial role of data quality and ethical considerations in the implementation of AI-driven predictive analytics. It examines the challenges associated with privacy concerns, algorithmic bias, and the need for transparent AI models to ensure responsible and fair use of customer data. Furthermore, the paper explores a spectrum of customer engagement strategies enabled by AI-driven predictive analytics. AI can provide valuable insights from different points of view, allowing businesses to gain a comprehensive understanding of their sales performance. By analyzing data from various sources, such as customer interactions, market trends, and competitor analysis, AI can offer a holistic view of the sales landscape and identify potential opportunities and challenges [3]. This predictive approach enables to make data-driven

decisions on inventory management, reducing stockouts, minimizing waste, and enhancing overall efficiency. In conclusion, Embracing these innovations is key for businesses aiming to stay competitive, meet evolving consumer demands, and drive sustained profitability in an increasingly digital world.

1.1 Background

Currently, the volume of data is increasing exponentially. How to make reasonable and effective use of existing business data to support future business decision-making has become the focus of attention for many businesses. Sales projections can increase store earnings, which are crucial to the growth of contemporary and future commerce, by reducing the lack of hot-selling products and the accumulation of unpopular products. Hence, forecasting sales is especially necessary for companies to help reduce costs and increase profits. [5]

1.2 Objective

The objective of this framework is to predict the future sales from given data of the previous year's using Machine Learning Techniques.

To find out key factors that can increase their sales and what changes could be made to the product or store's characteristics. [6]

1.3 Problem Statement

Due to increasing competition many malls and bigmart are trying their best to stay ahead in competition. In order to find out what are the various factors which affect the sales of bigmart and what strategies one needs to employ in order to gain more profit one needs to have some model on which they can rely. So a predictive model can be made which could help to gain useful information and increase profit. [7]

CHAPTER 2

SYSTEM ANALYSIS

In this chapter, we will discuss and analyze the developing process of Audit Control including software requirement specification (SRS) and comparison between existing and proposed systems. The functional and non- functional requirements are included in the SRS part to provide complete description and overview of system requirements before the developing process is carried out. Besides that, existing vs. proposed provides a view of how the proposed system will be more efficient than the existing one.

2.1 HARDWARE AND SOFTWARE REQUIREMENTS

2.1.1 HARDWARE REQUIREMENTS:

Processor	:	Intel or AMD.
Ram	:	4 GB or above.
Hard disk	:	40GB or above.
Monitor	:	PC

2.1.2 SOFTWARE REQUIREMENTS:

OS	:	Windows, Ubuntu
IDE	:	VSCODE
Language	:	Python3

2.2 SOFTWARE REQUIREMENT SPECIFICATION:

SRS

Software Requirement Specification (SRS) is the starting point of the software developing activity. As the system grew more complex it became evident that the goal of the entire system cannot be easily comprehended. Hence the need for the requirement phase arose. The software project is initiated by the client. The SRS is the means of translating the ideas of the minds of clients (the input) into a formal document (the output of the requirement phase).

The SRS phase consists of two basic activities:

1) Problem/Requirement Analysis:

The process is order and more nebulous of the two, deals with understanding the problem, the goal and constraints.

2) Requirement Specification:

Here, the focus is on specifying what has been found, giving analysis such as representation, specification languages and tools, and checking the specifications are addressed during this activity.

The Requirement phase terminates with the production of the validated SRS document. Producing the SRS document is the basic goal of this phase.

ROLE OF SRS:

The purpose of the Software Requirement Specification is to reduce the communication gap between the clients and the developers. Software Requirement Specification is the medium through which the client and user needs are accurately specified. It forms the basis of software development. A good SRS should satisfy all the parties involved in the system.

SCOPE:

This document is the only one that describes the requirements of the system. It is meant for the use by the developers, and will also be the basis for validating the final delivered system. Any changes made to the requirements in the future will have to go through a

formal change approval process. The developer is responsible for asking for clarifications, where necessary, and will not make any alterations without the permission of the client.

The Software Requirements Specification (SRS) serves as a vital blueprint for any software development project. Its scope encompasses a detailed description of the software's functional and non-functional requirements, defining what the system should do and how it should perform. The SRS delves into user requirements, system architecture, data handling, constraints, and assumptions. It outlines interfaces with external entities and describes use cases or scenarios to illustrate real-world interactions. Additionally, the SRS addresses testing requirements, ensuring that the software aligns with the specified criteria. Importantly, it includes change control procedures to manage scope changes throughout development. Ultimately, the SRS acts as a formal agreement among stakeholders, guiding the development process, aiding project planning, and mitigating risks by providing a comprehensive and shared understanding of the project's objectives and constraints.

CHAPTER 3

SYSTEM DESIGN

System design transitions from a user oriented document to programmers or database personnel. The design is a solution, how to approach the creation of a new system. This is composed of several steps. It provides the understanding and procedural details necessary for implementing the system recommended in the feasibility study. Designing goes through logical and physical stages of development, logical design reviews the present physical system, preparing input and output specification, details of implementation plan and preparing a logical design walkthrough.

SOFTWARE DESIGN:

In designing the software following principles are followed:

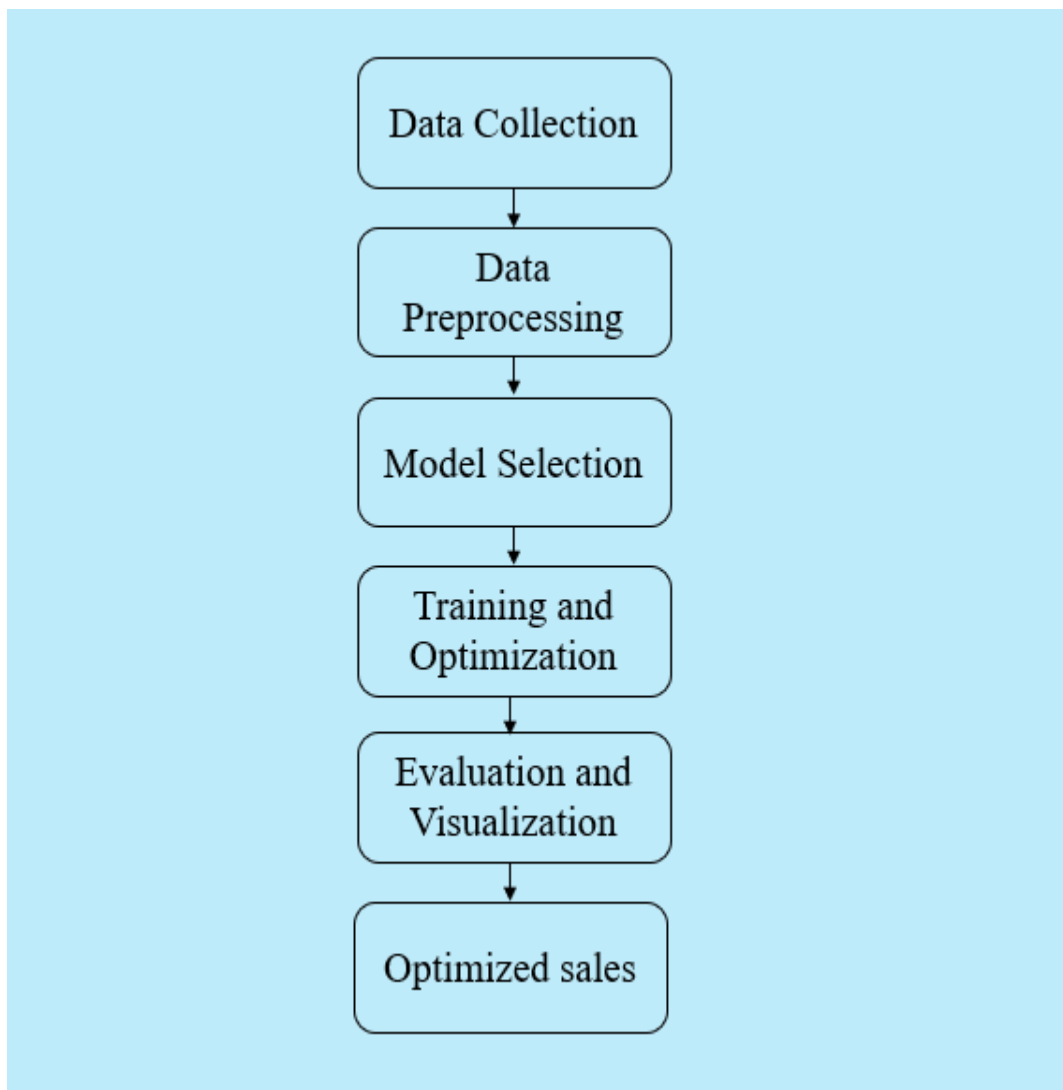
- 1) Modularity and partitioning:** Software is designed such that each system should consist of a hierarchy of modules and serve to partition into separate functions.
- 2) Coupling:** Modules should have little dependence on other modules of a system.
- 3) Cohesion:** Modules should be carried out in a single processing function.
- 4) Shared use:** Avoid duplication by allowing a single module to be called by others that need the function it provides.

3.1 ARCHITECTURE:

Architecture diagram is a diagram of a system, in which the principal parts or functions are represented by blocks connected by lines that show the relationships of the blocks. The block diagram is typically used for a higher level, less detailed description aimed

more at understanding the overall concepts and less at understanding the details of implementation.

A SMS user for who the application looks like an user interface actually consists of a database called SQLite that comes along with Android SDK and needs no other installation. This is the database that is used to store and retrieve information. This is an application that is developed in Java and hence all its features apply here as well such as platform independence, data hiding.



CHAPTER 4

METHODOLOGY

4.1 Time Series Forecasting techniques

Time series forecasting occurs when you make scientific predictions based on historical time stamped data. It involves building models through historical analysis and using them to make observations and drive future strategic decision-making. An important distinction in forecasting is that at the time of the work, the future outcome is completely unavailable and can only be estimated through careful analysis and evidence-based priors [8]. A time series is a collection of data points indexed in time order. Time series data are typically collected at regular intervals, such as daily, weekly, or monthly. Examples of time series data include stock prices, weather data, and sales data. Time series data can be univariate (consisting of a single variable) or multivariate (consisting of multiple variables) [9].

Definitions :

Classification

Classification in machine learning sorts data into categories based on their features. It predicts which category new data belongs to using binary classification (sorting into two groups) or multi-class classification (sorting into more than two groups). These techniques use patterns from past data to correctly classify new data points based on identified features [10].

Clustering

Clustering is an unsupervised machine learning algorithm that organizes data points into clusters based on shared properties. The primary goal is to ensure that data points within the same cluster exhibit similar characteristics while those in different clusters are distinctly different [10].

Regression

Regression is the process of finding a model or function for distinguishing the data into continuous real values instead of using classes or discrete values. It can also identify the

distribution movement depending on the historical data. Because a regression predictive model predicts a quantity, therefore, the skill of the model must be reported as an error in those predictions [11].

4.2 Dataset Understanding

Dataset: Supermart Grocery Sales - Retail Analytics Dataset

Description :It contains columns like Invoice ID, Branch, City, Customer type, Gender, Product line, Unit price, Quantity, Tax 5%, Total, Date, Time, Payment, cogs, gross margin percentage, gross income, Rating [12].

Invoice ID: Unique identifier, not directly useful for prediction but helpful for reference.

Branch and City: Geographic data can highlight location-specific trends, such as popularity of certain products in specific regions.

Customer Type: Different customer segments (e.g., loyalty vs. new customers) may have varying buying behaviors, which can inform tailored marketing and stocking decisions.

Gender: Identifying patterns based on gender can support targeted marketing efforts and may reveal product lines with gender-specific preferences.

Product Line: Each product category's sales can be predicted individually or in combination with other features to ensure relevant stock levels.

Unit Price and Quantity: Essential for calculating sales volume and understanding the impact of pricing on demand.

Tax 5% and Total: Useful for calculating final revenue. They can help establish the expected sales value per transaction.

Date and Time: Sales trends often vary by day of the week, month, or season. Time-based patterns can be incorporated to account for seasonal demand fluctuations.

Payment Method: The preferred payment methods might correlate with certain types of customers or purchase sizes.

COGS (Cost of Goods Sold): Indicates the cost associated with each sale, which can help in calculating profitability over time and detecting which products have higher margins.

Gross Margin Percentage and Gross Income: Essential for assessing profitability and understanding which products or transactions contribute most to profits.

Rating: Can provide insight into customer satisfaction, which might affect future sales. For instance, low-rated products could lead to lower future demand, whereas high ratings could boost repeat purchases or recommendations.

 supermarket dataset

4.3 Steps for Preprocessing dataset

Data Cleaning: This involves identifying and correcting errors or inconsistencies in the data, such as missing values, outliers, and duplicates. Various techniques can be used for data cleaning, such as imputation, removal, and transformation.

Data Integration: This involves combining data from multiple sources to create a unified dataset. Data integration can be challenging as it requires handling data with different formats, structures, and semantics. Techniques such as record linkage and data fusion can be used for data integration.

Data Transformation: This involves converting the data into a suitable format for analysis. Common techniques used in data transformation include normalization, standardization, and discretization. Normalization is used to scale the data to a common range, while standardization is used to transform the data to have zero mean and unit variance. Discretization is used to convert continuous data into discrete categories.

Data Reduction: This involves reducing the size of the dataset while preserving the important information. Data reduction can be achieved through techniques such as feature selection and feature extraction. Feature selection involves selecting a subset of relevant features from the dataset, while feature extraction involves transforming the data into a lower-dimensional space while preserving the important information.

Data Discretization: This involves dividing continuous data into discrete categories or intervals. Discretization is often used in data mining and machine learning algorithms that require categorical data. Discretization can be achieved through techniques such as equal width binning, equal frequency binning, and clustering.

Data Normalization: This involves scaling the data to a common range, such as between 0 and 1 or -1 and 1. Normalization is often used to handle data with different units and scales.

Common normalization techniques include min-max normalization, z-score normalization, and decimal scaling. [13]

Here are some ways to preprocess dataset :

Identify columns with missing values. we can fill missing values in numerical columns with their mean or median, and in categorical columns with the mode, if applicable.

Convert the 'Date' column to datetime format to enable extraction of time-based features. Extract and add new columns like 'Month', 'Day of the Week', and 'Quarter' from the 'Date' column to capture seasonal trends. Add calculated features like 'Profit' (as Total minus COGS) and 'Average Price per Unit' (as Total divided by Quantity) to enrich data for sales prediction.

Check for and remove any duplicate rows to avoid skewing the analysis.

Separate the data into training and testing sets based on time (e.g., past data as training and recent data as testing) to prevent data leakage.

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.[14]

2) Convert Date and Create DateTime Column : DateTime 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.[15]

3) Handle Missing Values : Using the dropna() function is the easiest way to remove observations or features with missing values from the dataframe.[16]

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.[17]

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

additional features. In other words, it's a feature that helps your model to make better predictions![18]


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.[19]

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.[20]

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.[21]

10) save preprocessed data in excel file: We can save the data in an excel file.[22]

 preprocessed_sales_data.xlsx

4.4 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

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, or Exploratory Data Analysis, refers back to the method of analyzing and analyzing information units to uncover styles, pick out relationships, and gain insights.

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.

Summary statistics: Calculations like mean, median, mode, variance, and standard deviation that describe the central tendency and dispersion of the data.

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. [24]

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.[24]

4.5 Feature Engineering

What is 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. [25]

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, investigating 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. [26]

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.[27]

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 [28].

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 [29].

CHAPTER 5

Data Visualization

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.

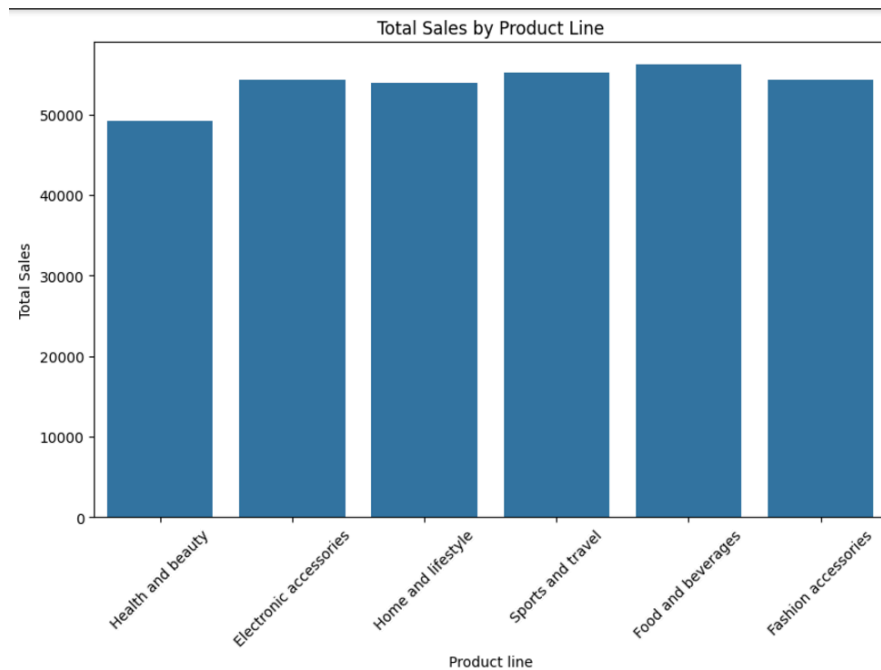
Dashboards: Interactive platforms that combine multiple visualizations.

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.[30]

Visualizations of Dataset

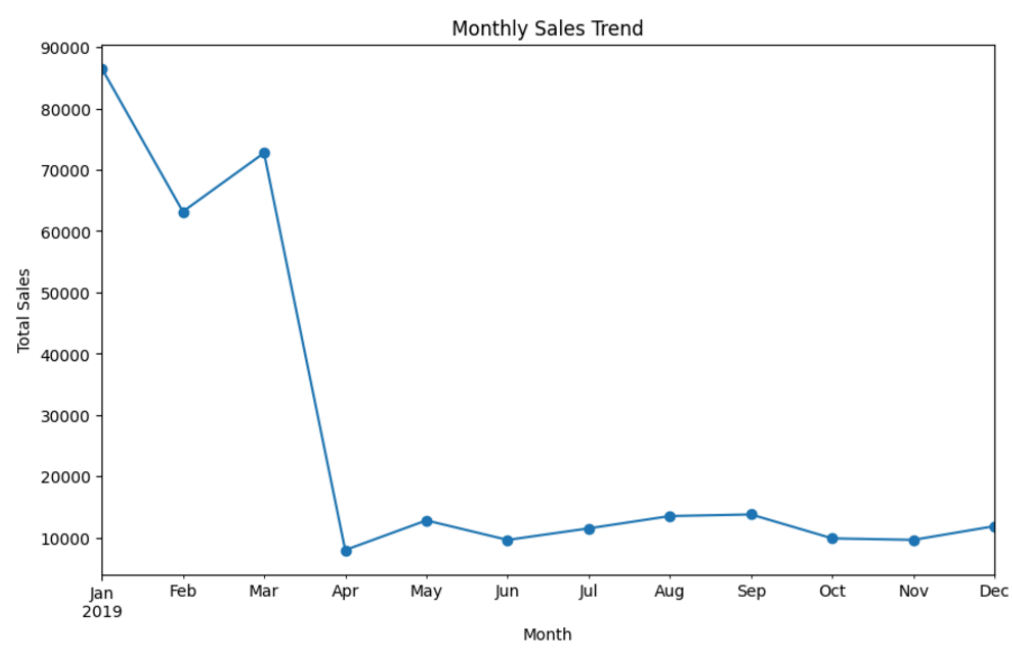
1) Sales Distribution by Product Line

A bar chart showing total sales across product lines.



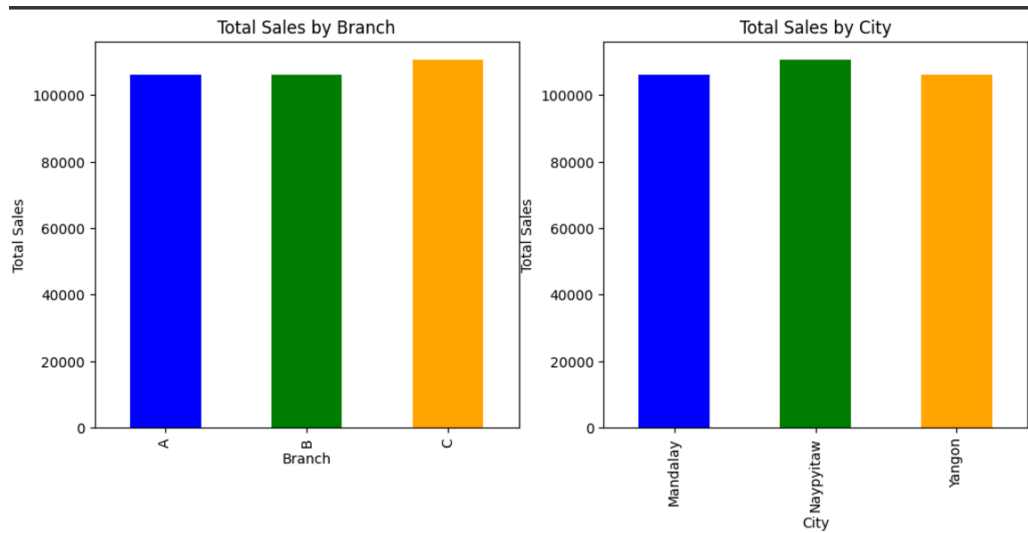
2) Sales Trends Over Time

A line plot of sales over months/days to identify seasonality.



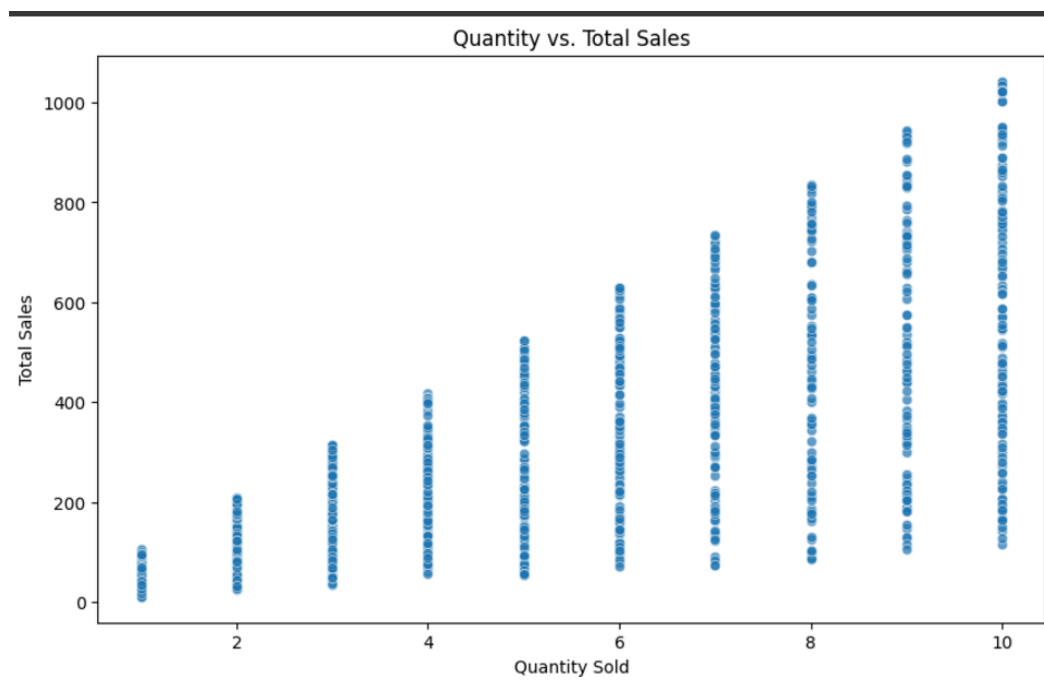
3) Branch/City-Wise Sales

A grouped bar chart to compare sales across branches and cities.



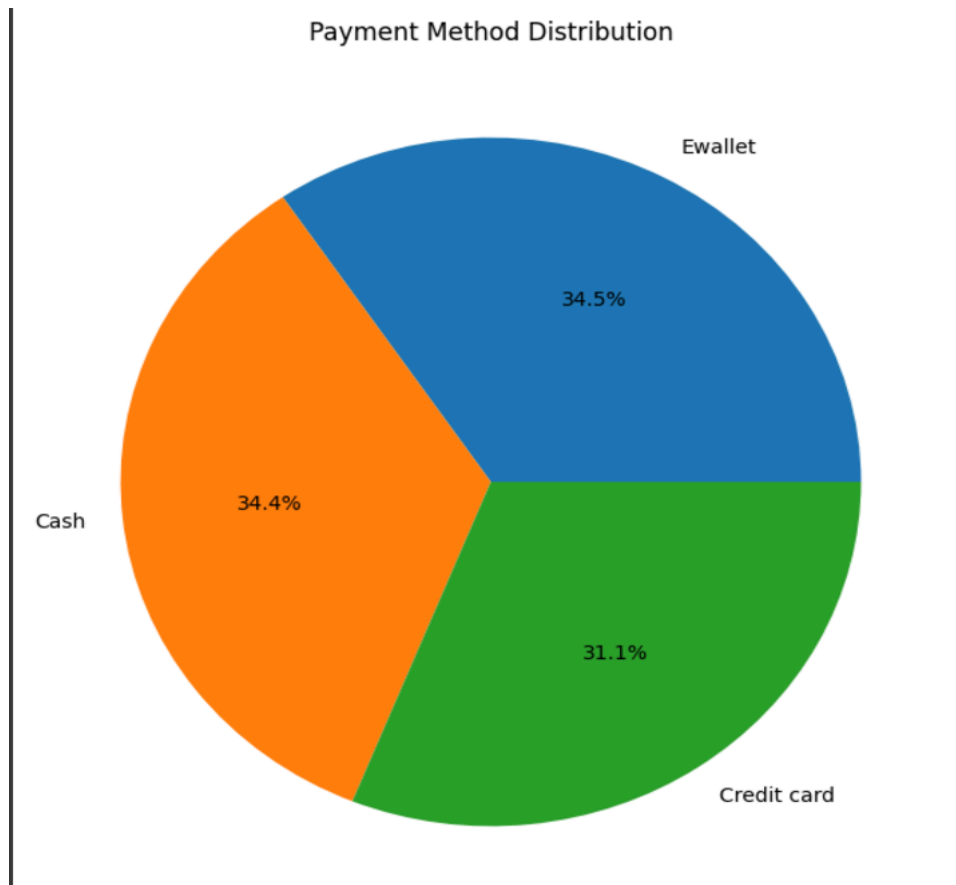
4) Quantity vs. Total Sales

A scatter plot to analyze how quantity sold impacts total sales.



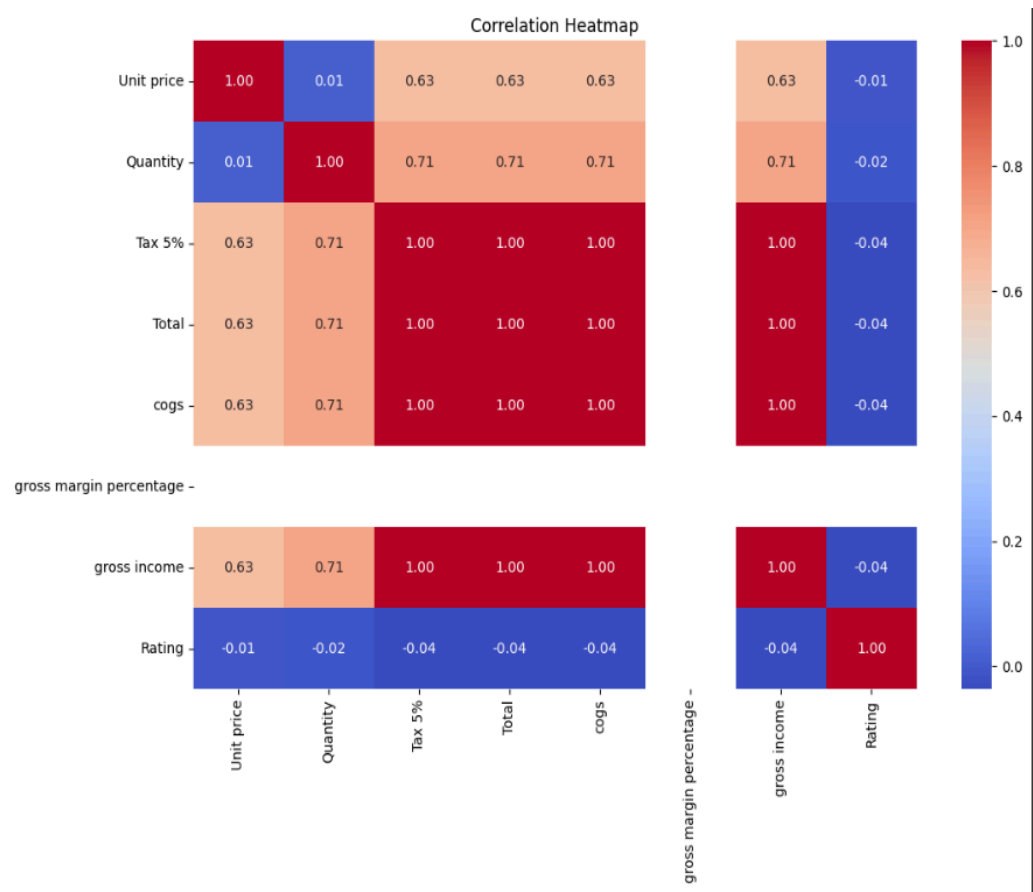
5) Payment Method Distribution

A pie chart to understand the popularity of payment methods.



6) Heatmap of Correlations

A correlation heatmap to understand relationships between numerical features.



CHAPTER 6

MODEL TRAINING

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.[31]

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.[32]

How Does Random Forest 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, each

specializing 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.

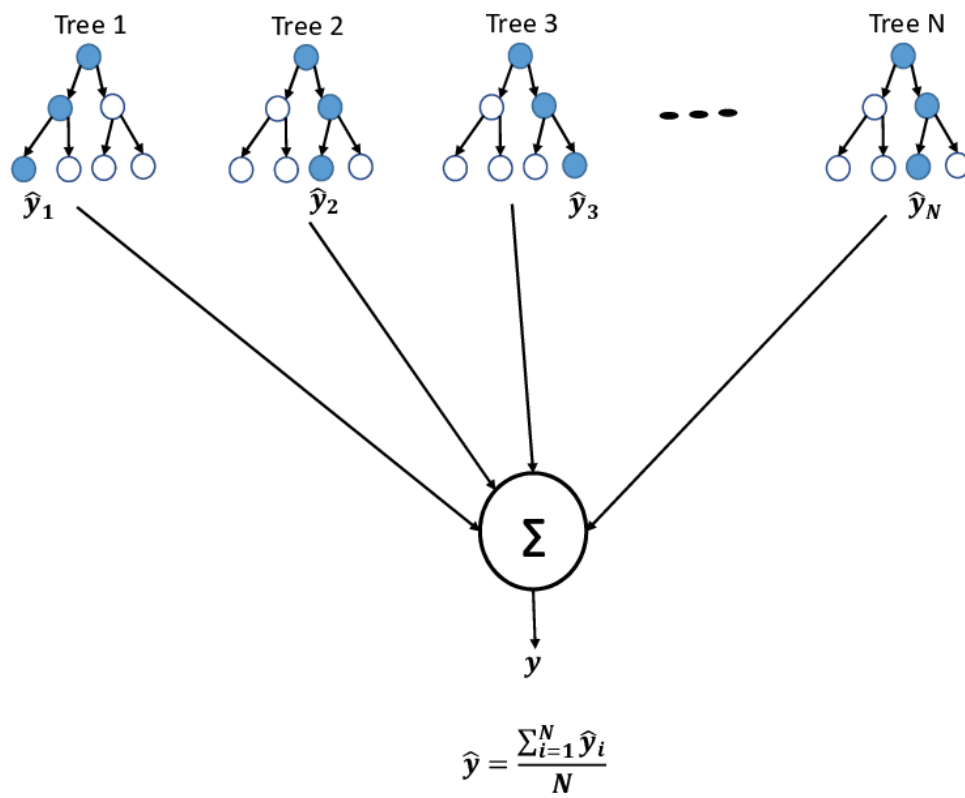
Bootstrap Aggregating or Bagging: The technique of bagging is a cornerstone of Random Forest's training strategy which involves creating multiple bootstrap samples from the original dataset, allowing instances to be sampled with replacement. This results in different subsets of data for each decision 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[33]



CHAPTER 7

SAMPLE CODE

sales_prediction_main.py

```
"""
```

Module for preprocessing and visualizing supermarket sales data.

This script includes functionality to clean, transform, and prepare the dataset for analysis or machine learning tasks, along with visualizations for exploratory data analysis (EDA).

```
"""
```

```
import pandas as pd
```

```
import numpy as np
```

```
import seaborn as sns
```

```
import matplotlib.pyplot as plt
```

```
from sklearn.preprocessing import LabelEncoder, StandardScaler
```

```
from model_training_performance_analysis import
```

```
train_and_predict_model,hyperparameter_tuning
```

```
def preprocess_sales_data(input_path, output_path):
```

```
    """
```

Preprocess the supermarket sales dataset.

Parameters:

input_path (str): Path to the input Excel file.

output_path (str): Path to save the preprocessed Excel file.

Returns:

pd.DataFrame: Preprocessed DataFrame.

dict: Dictionary of LabelEncoders for categorical columns.

```
    """
```

```
# Step 1: Load the Excel file
```

```
df = pd.read_excel(input_path)
```

```
# Step 2: Convert Date and Create DateTime Column
```

```

df['Date'] = pd.to_datetime(df['Date'])
df['Time'] = df['Time'].astype(str) # Convert Time to string format
df['DateTime'] = pd.to_datetime(df['Date'].astype(str) + ' ' + df['Time'])
df.drop(columns=['Date', 'Time'], inplace=True)

# Step 3: Handle Missing Values
df.fillna(0, inplace=True)

# Step 4: Encode Categorical Variables (Exclude 'Product line', 'City', 'Branch')
label_encoders = {}
categorical_columns = ['Customer type', 'Gender', 'Payment']
for column in categorical_columns:
    encoder = LabelEncoder()
    df[column] = encoder.fit_transform(df[column])
    label_encoders[column] = encoder

# Step 5: Feature Engineering: Date-Time Features
df['Year'] = df['DateTime'].dt.year
df['Month'] = df['DateTime'].dt.month
df['Day'] = df['DateTime'].dt.day
df['DayOfWeek'] = df['DateTime'].dt.dayofweek
df['Hour'] = df['DateTime'].dt.hour

# Step 6: Add Additional Columns
df['RevenuePerUnit'] = df['Unit price'] * df['Quantity']
df['TotalTaxAmount'] = df['Total'] * 0.05
df['IsWeekend'] = df['DayOfWeek'].apply(lambda x: 1 if x >= 5 else 0)
df.drop(columns=['DateTime'], inplace=True)

# Step 7: Log Transformation
df['LogQuantity'] = np.log1p(df['Quantity'])
df['LogTotal'] = np.log1p(df['Total'])
df['LogGrossIncome'] = np.log1p(df['gross income'])

# Step 8: Interaction Feature: Quantity * Unit Price
df['QuantityUnitPrice'] = df['Quantity'] * df['Unit price']

```

```

# Outlier Treatment for 'Total'
q1 = df['Total'].quantile(0.25)
q3 = df['Total'].quantile(0.75)
iqr = q3 - q1
df = df[(df['Total'] >= q1 - 1.5 * iqr) & (df['Total'] <= q3 + 1.5 * iqr)]

# Step 9: Scaling Numerical Features (Exclude 'Product line', 'City', 'Branch')
numerical_features = ['Unit price', 'Quantity', 'Tax 5%', 'Total', 'gross income', 'Rating']
scaler = StandardScaler()
df[numerical_features] = scaler.fit_transform(df[numerical_features])

# Step 10: Save Preprocessed Data to Excel
df.to_excel(output_path, index=False)
return df, label_encoders

def visualize_sales_data(input_path):
    """
    Generate visualizations for supermarket sales data.
    Parameters:
        input_path (str): Path to the input Excel file.
    """

# Step 1: Load the Excel file
df = pd.read_excel(input_path)

# Step 2: Ensure 'Total' and 'Product line' columns exist
if 'Total' not in df.columns or 'Product line' not in df.columns:
    raise ValueError("The input dataset must contain 'Total' and 'Product line' columns.")

# Step 3: Sales Distribution by Product Line
plt.figure(figsize=(10, 6))
sns.barplot(x='Product line', y='Total', data=df, estimator=sum, ci=None)
plt.title('Total Sales by Product Line')
plt.xticks(rotation=45)
plt.ylabel('Total Sales')
plt.show()

```

Step 4: Sales Trends Over Time

```
df['Month'] = pd.to_datetime(df['Date']).dt.month # Extract month from the 'Date' column
```

```
sales_trend = df.groupby('Month')['Total'].sum()
```

```
sales_trend.plot(kind='line', figsize=(10, 6), marker='o')
```

```
plt.title('Monthly Sales Trend')
```

```
plt.ylabel('Total Sales')
```

```
plt.xlabel('Month')
```

```
plt.show()
```

Step 5: Branch/City-Wise Sales

```
branch_sales = df.groupby('Branch')['Total'].sum()
```

```
city_sales = df.groupby('City')['Total'].sum()
```

```
plt.figure(figsize=(12, 5))
```

```
plt.subplot(1, 2, 1)
```

```
branch_sales.plot(kind='bar', color=['blue', 'green', 'orange'])
```

```
plt.title('Total Sales by Branch')
```

```
plt.ylabel('Total Sales')
```

```
plt.xlabel('Branch')
```

```
plt.subplot(1, 2, 2)
```

```
city_sales.plot(kind='bar', color=['blue', 'green', 'orange'])
```

```
plt.title('Total Sales by City')
```

```
plt.ylabel('Total Sales')
```

```
plt.xlabel('City')
```

```
plt.show()
```

Step 6: Quantity vs. Total Sales

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

```
sns.scatterplot(x='Quantity', y='Total', data=df, alpha=0.7)
```

```
plt.title('Quantity vs. Total Sales')
```

```
plt.xlabel('Quantity Sold')
```

```
plt.ylabel('Total Sales')
```

```
plt.show()
```

```

# Step 7: Payment Method Distribution
payment_counts = df['Payment'].value_counts()
payment_counts.plot(kind='pie', autopct='%1.1f%%', figsize=(8, 8))
plt.title('Payment Method Distribution')
plt.ylabel("")
plt.show()

# Step 8: Heatmap of Correlations
numeric_df = df.select_dtypes(include=['number'])
plt.figure(figsize=(12, 8))
sns.heatmap(numeric_df.corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap')
plt.show()

# Example usage
if __name__ == "__main__":
    INPUT_FILE = 'supermarket dataset.xlsx'
    OUTPUT_FILE = 'preprocessed_sales_data.xlsx'
    MODEL_FILE = 'random_forest_regressor.pkl'
    preprocessed_data, encoders = preprocess_sales_data(INPUT_FILE, OUTPUT_FILE)
    print("Preprocessing complete. Preprocessed data saved to", OUTPUT_FILE)
    print("Generating visualizations...")
    visualize_sales_data(INPUT_FILE)
    result=train_and_predict_model(INPUT_FILE)
    print(result)
    print("RESULTS AFTER HYPERPARAMETER TUNING")
    hyperparameter_tuning(INPUT_FILE)

```


CHAPTER 8

CONCLUSION AND FUTURE SCOPE

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 [34].

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 neural networks to enhance the model. We can modify the same system to an online learning system that adapts in real-time.[35]

References

- [1]https://www.researchgate.net/publication/338681895_Applied_Machine_Learning_for_Supermarket_Sales_Prediction
- [2] <https://www.prismetric.com/ai-in-demand-forecasting/>
- [3]https://www.researchgate.net/publication/378293870_AI-DRIVEN_PREDICTIVE_ANALYTICS_IN_RETAIL_A_REVIEW_OF_EMERGING_TRENDS_AND_CUSTOMER_ENGAGEMENT_STRATEGIES
- [4]https://www.researchgate.net/publication/370700735_Walmart_Sales_Prediction_Based_on_Machine_Learning
- [5] <https://ijcrt.org/papers/IJCRT2106802.pdf>
- [6]<http://www.ir.juit.ac.in:8080/jspui/bitstream/123456789/3600/1/Big%20Mart%20Sales%20Prediction%20Using%20Machine%20Learning.pdf>
- [7]<https://fastercapital.com/content/Sales-forecast-artificial-intelligence--How-to-Leverage-Artificial-Intelligence-for-Smarter-Sales-Forecasting.html>
- [8]<https://www.tableau.com/analytics/time-series-forecasting#:~:text=Time%20series%20for%20forecasting%20occurs%20when%20you%20make%20scientific,to%20make%20observations%20and%20drive%20future%20strategic%20decision-making.>
- [9]<https://medium.com/analysts-corner/comprehensive-guide-to-time-series-modeling-techniques-applications-and-best-practices-fd330eb0a755#:~:text=Time%20series%20modeling%20is%20a%20powerful%20and%20widely-used,understand%20patterns%2C%20trends%2C%20and%20relationships%20within%20the%20data.>
- [10]<https://pg-p.ctme.caltech.edu/blog/data-analytics/difference-between-classification-clustering-regression>
- [11] <https://www.geeksforgeeks.org/ml-classification-vs-regression/>
- [12]https://github.com/sushantag9/Supermarket-Sales-Data-Analysis/blob/master/supermarket_sales%20-%20Sheet1.csv
- [13] <https://www.geeksforgeeks.org/data-preprocessing-in-data-mining/>
- [14]<https://www.geeksforgeeks.org/how-to-import-an-excel-file-into-python-using-pandas/>
- [15]<https://www.geeksforgeeks.org/how-to-convert-datetime-to-date-in-pandas/>
- [16]<https://www.datacamp.com/tutorial/techniques-to-handle-missing-data-values>

- [17]<https://machinelearningtutorials.org/pandas-encoding-categorical-features-with-examples/>
- [18]<https://www.dataschool.io/introduction-to-feature-engineering/#:~:text=It%27s%20a%20feature%20that%20your%20Machine%20Learning%20model,that%20helps%20your%20model%20to%20make%20better%20predictions%21>
- [19]<https://medium.com/@kyawsawhtoon/log-transformation-purpose-and-interpretation-9444b4b049c9>
- [20]<https://www.mdpi.com/2076-3417/9/23/5191#:~:text=Feature%20interaction%20can%20be%20described%20as%20a%20phenomenon,sum%20of%20the%20performance%20of%20each%20constituent%20feature.>
- [21]<https://www.geeksforgeeks.org/ml-feature-scaling-part-2/>
- [22]<https://www.geeksforgeeks.org/exporting-a-pandas-dataframe-to-an-excel-file/>
- [23]<https://www.analyticsvidhya.com/blog/2022/07/step-by-step-exploratory-data-analysis-ed-a-using-python/>
- [24]<https://www.geeksforgeeks.org/what-is-exploratory-data-analysis/>
- [25]<https://www.geeksforgeeks.org/what-is-feature-engineering/>
- [26] <https://www.javatpoint.com/feature-engineering-for-machine-learning>
- [27]<https://www.learndatasci.com/tutorials/intro-feature-engineering-machine-learning-python/>
- [28]<https://blog.flowworks.ai/ai-in-predictive-sales-forecasting/#:~:text=Data%20is%20the%20foundation%20of%20any%20AI-driven%20system%2C,interaction%20logs%2C%20and%20external%20data%20like%20market%20trends.>
- [29]<https://www.leewayhertz.com/ai-in-demand-forecasting/#AI-for-demand-forecasting-work>
- [30]<https://www.geeksforgeeks.org/how-to-convert-datetime-to-date-in-pandas/>
- [31]<https://medium.datadriveninvestor.com/random-forest-pros-and-cons-clc42fb64f04>
- [32] <https://www.embedded-robotics.com/forecast-sales-using-machine-learning/>
- [33]<https://www.geeksforgeeks.org/random-forest-algorithm-in-machine-learning/>
- [34]https://www.researchgate.net/publication/344099746_SALES_PREDICTION_MODEL_FOR_BIG_MART
- [35]<https://medium.com/@ldicarlo1/sales-time-series-forecasting-using-a-hybrid-prophet-random-forest-machine-learning-solution-fe4bdcbea619>