SalesForecasting Using Machine Learning Models

A Industrial Oriented Mini Project Report

Submitted to

Jawaharlal Nehru Technological University, Hyderabad

In partial fulfilment of the requirements for the

award of the degree of

BACHELOR OF TECHNOLOGY

in

CSE (DATA SCIENCE)

by

S.Rithin Reddy	**22VE1A67B8**
J.Manideep	**22VE1A6778**
K.Sai Charan	**22VE1A6788**
A.Pranay	**22VE1A6764**

Under the Guidance
Of
Mr.D.Srikar

ASSISTANT PROFESSOR

SREYAS INSTITUTE OF ENGINEERING AND TECHNOLOGY
AUTONOMOUS

DEPARTMENT OF CSE (DATA SCIENCE)
(Affiliated to JNTUH, Approved by A.I.C.T.E and Accredited by NAAC, New Delhi)
Bandlaguda, Beside Indu Aranya, Nagole, Ranga Reddy Dist, Hyderabad-500068

2024-25

SREYAS INSTITUTE OF ENGINEERING AND TECHNOLOGY

AUTONOMOUS

DEPARTMENT OF CSE (DATA SCIENCE)

CERTIFICATE

This is to certify that the Industrial Oriented Mini Project Report on **"SALES FORECASTING"** submitted by **S.RITHIN , J.MANIDEEP , K.SAI CHARAN, A.PRANAY** bearing Hall ticket Nos. **22VE1A67B8, 22VE1A6778, 22VE1A6788, 22VE1A6764** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **CSE (DATA SCIENCE)** from Sreyas institute of engineering and technology, Hyderabad for the academic year 2024-25 is a record of bonafide work carried out by them under our guidance and Supervision.

Internal Guide
Mr.D.Srikar
Assistant Professor

Project Co-Ordinator
Mrs. D. Chaithanya
Assistant Professor

Head of the Department
Dr. K. Rohit Kumar
Associative Professor

External Examiner

SREYAS INSTITUTE OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF CSE (DATA SCIENCE)

DECLARATION

We **S.RITHIN , J.MANIDEEP , K.SAI CHARAN , A.PRANAY** bearing **Roll No.s22VE1A67B8, 22VE1A6778, 22VE1A6788, 22VE1A6764** here by declare that the Mini Project titled **Sales Forecasting** done by us under the guidance of **Mr. D. Srikar, Assistant Professor** which is submitted in the partial fulfillment of the requirement for the award of the B. Tech degree in **CSE (DATA SCIENCE)** at **Sreyas Institute of Engineering and Technology** for, Hyderabad is our original work.

S. Rithin Reddy	**22VE1A67B8**
J. Manideep	**22VE1A6778**
K. Sai Charan	**22VE1A6788**
A. Pranay	**22VE1A6764**

ACKNOWLEDGEMENT

The successful completion of any task would be incomplete without mention of the people who made it possible through their guidance and encouragement crowns all the efforts with success.

We take this opportunity to acknowledge with thanks and deep sense of gratitude to **Mr. D. Srikar, Assistant Professor, Department of CSE (DATA SCIENCE)** for her constant encouragement and valuable guidance during the Project work.

A Special vote of Thanks to **Dr. K. Rohit Kumar**, **Head of the Department (DS),** and **Mrs. D.Chaithanya, Assistant Professor**, **Project Coordinator,** who has been a source of Continuous motivation and support. They had taken time and effort to guide and correct us all through the span of this work.

We owe very much to the **Department Faculty, Principal** and the **Management** who made our term at Sreyas Institute of Engineering and Technology a stepping stone for our career. We treasure every moment we had spent in the college.

Last but not the least, our heartiest gratitude to our parents and friends for their continuous encouragement and blessings. Without their support this work would not have been possible.

S. Rithin Reddy	**22VE1A67B8**
J. Manideep	**22VE1A6778**
K. Sai Charan	**22VE1A6788**
A. Pranay	**22VE1A6764**

ABSTRACT

An Intelligent Decision Analytical System necessitates the integration of decision analysis and predictive methodologies to enhance business operations. Within business frameworks, reliance on a knowledge base and the ability to predict sales trends is of paramount importance. The accuracy of sales forecasts directly influences business outcomes, impacting inventory management, staffing, and strategic planning. Leveraging data mining techniques effectively uncovers hidden insights within large datasets, improving both the accuracy and efficiency of forecasting.

future sales predictions. Conventional forecasting systems often struggle with extensive data, compromising forecast accuracy. These challenges, however, can be addressed using advanced machine learning techniques. This paper presents a concise analysis of sales data and forecasting methodologies, elaborating on various techniques and metrics crucial for accurate sales predictions. Through comprehensive performance evaluations, a suitable predictive model is recommended for forecasting sales trends. The findings emphasize the reliability and precision of the adopted techniques. The research identifies the XGBoost algorithm as the optimal model, demonstrating superior accuracy in forecasting future sales trends. This project utilizes historical sales data to train models, enabling supermarkets to make informed decisions.

Key objectives includes Improving sales forecasting accuracy to support inventory management. Informing business decisions through insights into sales patterns. Optimizing operations such as staffing and inventory management. The results validate the effectiveness of XGBoost, showcasing its ability to handle large datasets efficiently while delivering highly accurate sales forecasts.

Keywords: Sales forecasting, machine learning, XGBoost algorithm, predictive analytics, data mining, supermarket sales prediction, feature engineering, decision support system, hyperparameter optimization, big data analytics, regression models, performance metrics (RMSE, MAE, R^2 Score), ensemble learning.

TABLE OF CONTENTS

CHAPTER – 1
INTRODUCTION

In An Intelligent Decision Analytical System necessitates the integration of decision analysis and predictive methodologies to enhance business operations. Within business frameworks, reliance on a knowledge base and the ability to predict sales trends is of paramount importance. The accuracy of sales forecasts directly influences business outcomes, impacting inventory management, staffing, and strategic planning. Leveraging data mining techniques effectively uncovers hidden insights within large datasets, improving both the accuracy and efficiency of forecasting.This study thoroughly examines transparent predictive models aimed at refining future sales predictions. Conventional forecasting systems often struggle with extensive data, compromising forecast accuracy. These challenges, however, can be addressed using advanced machine learning techniques. This paper presents a concise analysis of sales data and forecasting methodologies, elaborating on various techniques and metrics crucial for accurate sales predictions.Through comprehensive performance evaluations, a suitable predictive model is recommended for forecasting sales trends. The findings emphasize the reliability and precision of the adopted techniques. The research identifies the XGBoost algorithm as the optimal model, demonstrating superior accuracy in forecasting future sales trends. This project utilizes historical sales data to train models, enabling supermarkets to make informed decisions. The key objectives include improving sales forecasting accuracy to support inventory management, informing business decisions through insights into sales patterns, and optimizing operations such as staffing and inventory management. The results validate the effectiveness of XGBoost, showcasing its ability to handle large datasets efficiently while delivering highly accurate sales forecasts.

1.1 MOTIVATION

In In today's highly competitive retail landscape, accurate sales forecasting has become crucial for maintaining operational efficiency and maximizing profitability. Supermarkets, in particular, face constant challenges in managing inventory, aligning workforce requirements, and meeting dynamic customer demands. Overstocking leads to unnecessary holding costs and potential wastage, while understocking can result in lost sales and diminished customer satisfaction. These challenges underscore the pressing need for precise and reliable sales predictions to guide strategic and operational decisions.

The increasing availability of large volumes of historical sales data offers an opportunity to leverage advanced analytical techniques to enhance forecasting accuracy. Traditional forecasting methods often fall short in handling complex, high-dimensional datasets and fail to capture subtle trends and patterns in consumer behavior. This limitation creates a compelling case for the application of machine learning and data mining techniques, which can extract hidden insights and improve prediction performance.

The motivation for this project stems from the potential to empower supermarkets with intelligent, data-driven decision-making tools. By employing robust predictive models such as the XGBoost algorithm, businesses can significantly improve forecast accuracy, optimize inventory levels, enhance resource allocation, and ultimately boost customer satisfaction. This project aspires to bridge the gap between conventional forecasting limitations and modern data-centric approaches, driving smarter business strategies and more agile retail operations.

1.2 OBJECTIVE

The primary objective of this project is to develop an intelligent and accurate sales forecasting model that can empower supermarkets to make data-driven decisions. The following sub-objectives outline the specific goals of the study:

1. Enhancing Forecast Accuracy

One of the key objectives is to improve the precision of sales predictions. Accurate forecasts help minimize inventory-related issues such as overstocking and stockouts. By employing advanced machine learning techniques, particularly the XGBoost algorithm, the project aims to capture complex patterns and trends in historical sales data, ensuring more reliable and consistent predictions.

2. Supporting Strategic Business Decisions

The project seeks to provide actionable insights that inform strategic decision-making processes. Understanding sales trends, seasonal fluctuations, and consumer purchasing behaviors allows supermarket managers to design effective marketing campaigns, plan promotions, and optimize product assortments. Reliable sales forecasts become a critical input for shaping long-term business strategies and competitive positioning.

3. Optimizing Operational Efficiency

Operational optimization is another central objective. By accurately forecasting sales, supermarkets can better manage inventory levels, reduce waste, and streamline supply chain operations. Additionally, predictions of sales volumes can guide staffing decisions, ensuring adequate workforce allocation to meet customer demand, especially during peak seasons or promotional periods.

4. Leveraging Data Mining Techniques

The project aims to demonstrate the effectiveness of data mining techniques in extracting valuable insights from large, complex datasets. By exploring various features and variables that influence sales, the study intends to showcase how modern predictive models can surpass the limitations of traditional forecasting methods.

5. Recommending a Robust Predictive Model

Finally, the project strives to identify and recommend the most suitable predictive model for supermarket sales forecasting. Through comprehensive performance evaluations, the XGBoost algorithm is positioned as the optimal approach, offering superior accuracy, reliability, and interpretability for future sales predictions.

1.3 SCOPE

The scope of this project encompasses the design, development, and evaluation of a robust machine learning-based sales forecasting system aimed at enhancing the accuracy of sales predictions within supermarket environments. By leveraging historical sales data, the system intends to provide actionable insights that can aid businesses in making informed strategic and operational decisions. The focus is primarily on implementing and fine-tuning predictive models—particularly the XGBoost algorithm—owing to its superior capability in handling complex, high-dimensional data and uncovering hidden patterns within large datasets. XGBoost is chosen for its efficiency, scalability, and ability to manage large datasets with minimal computational overhead, making it an ideal choice for supermarket sales forecasting. This project spans the entire data science workflow, including data collection, preprocessing, feature selection, model training, validation, and testing. Emphasis is placed on data cleaning and feature engineering, as the quality of input data directly affects model performance. The study limits itself to structured historical sales data, such as transaction records, seasonal trends, promotional activities, and inventory information. External variables like economic indicators and competitor data are excluded to maintain focus on internally available supermarket data. Furthermore, the scope includes a comparative evaluation of various machine learning algorithms to assess their predictive performance using relevant statistical metrics such as RMSE (Root Mean Square Error), MAE (Mean Absolute Error), and R^2 Score. While XGBoost is the primary focus, other models like Decision Trees, Random Forests, and Linear Regression will be considered for benchmarking. The outcomes of the project are expected to support inventory optimization, resource planning, and strategic marketing decisions. However, real-time forecasting and integration with live supermarket systems are beyond the current scope and are proposed as future extensions. The system aims to deliver interpretable, reliable, and scalable forecasts that empower supermarket managers to minimize stock-outs, reduce overstocking, and align business operations with predicted customer demand patterns.

1.4 OUTLINE

This project is structured to provide a comprehensive understanding of the sales prediction process using machine learning techniques. The outline of the report is as follows:

Chapter 1: Introduction
This chapter introduces the project, outlining the problem, motivation, objectives, scope, and the significance of accurate sales prediction. It emphasizes the role of machine learning in transforming sales forecasting and decision-making processes within supermarkets.

Chapter 2: Literature Review
This section provides a detailed review of relevant research in the field of sales forecasting, highlighting various machine learning techniques such as regression models, decision trees, and ensemble methods. It discusses their application in different industries, with a specific focus on supermarkets and retail businesses, drawing insights from both academic and industry sources.

Chapter 3: proposed system
3.1 Existing System

- Limitations of traditional forecasting methods.

3.2 Proposed System

- Introduction to the new sales forecasting system using XGBoost for better accuracy.

3.3 Software Requirement Specification

- Technical specifications, tools, and frameworks used for system development.

3.4 SDLC Methodologies

- Outline of the Software Development Life Cycle used in the project.

3.5 Functional Requirements

- Key functionalities of the system, such as real-time forecasting, scalability, and data integration.

Chapter 4: System Design

  - Summary of results and how the proposed system enhances sales forecasting.

- **Future Scope**
  - Potential improvements: real-time forecasting, external data integration, multi-store support.

Chapter 9: References

- List of all research papers, articles, books, and other references used in the project.

CHAPTER – 2
LITERATURE SURVEY

Gangarapu Sharmista et al. (2024) conducted a study on sales forecasting using LightGBM and advanced feature engineering techniques.Xactly (2024) published a Sales Forecasting Benchmark Report, analyzing trends and challenges in sales forecasting. Fng (2022) conducted an extensive analysis of sales prediction using various machine learning techniques, focusing on optimizing prediction accuracy through comparative assessment of regression algorithms and ensemble methods, concluding that ensemble techniques generally outperform individual models. Varshini (2021) analyzed several machine learning algorithms—decision trees, support vector machines, and neural networks—for sales prediction, emphasizing that hybrid models improve accuracy, while Lu et al. (2021) explored spatial-temporal data applications within geographic information systems to support location-based retail forecasting. In 2020, Khan et al. proposed a demand forecasting model using business intelligence empowered by machine learning to enhance decision-making accuracy, and Martínez et al. introduced a customer purchase prediction framework for non-contractual settings to better understand consumer behavior patterns. D. V. (2020) applied data mining to forecast demand for refurbished electronics in India, while Goel and Bajpai (2020) examined LSTM-based sales forecasting, highlighting the impact of uncertainty in model parameters. A separate 2020 study on crop prediction demonstrated the relevance of machine learning in agricultural sales forecasting. In 2019, Amalina et al. reviewed the challenges of blending big data analytics, offering insights for large-scale sales data analysis, and Li et al. proposed EdgeCare, an edge computing solution for distributed data management applicable in sales environments. Sakib presented a general predictive analysis using ML models, Pavlyuchenko explored time series forecasting with deep learning architectures, and Telaga et al. applied STL decomposition for seasonal automobile sales prediction. Studies on sentiment analysis for suicide prediction and Ji et al.'s application-oriented sales prediction model demonstrated further versatility of ML predictive tools. Martínez-Plumed et al. (2019) reflected on the evolution of the CRISP-DM methodology, aligning it with modern data science workflows for structured forecasting. Earlier, Cheriyan (2018) applied linear regression and support vector machines to sales prediction, and Ibahim (2018) explored intelligent ML methods, including neural networks, to boost forecasting accuracy. Finally, Bohanec et al. (2017) emphasized the importance of explainability in ML sales prediction models, ensuring interpretability for business users, while Gharaibeh et al. (2017) surveyed smart city technologies, covering data management frameworks that indirectly support large-scale sales analysis through smart infrastructure.

CHAPTER – 3
PROPOSED SYSTEM

3.1 EXISTING SYSTEM

The existing systems for sales prediction primarily rely on traditional statistical methods and basic machine learning techniques to forecast sales trends. These methods include linear regression, moving averages, and exponential smoothing, which have been widely used in various industries, including retail. While these systems provide a foundation for predicting sales, they often struggle to handle the complexity and large volumes of data typical in modern supermarket operations. Traditional approaches, such as linear regression, assume a linear relationship between input variables and sales. However, in real-world scenarios, sales data are influenced by a multitude of factors, including seasonal trends, promotions, market fluctuations, and consumer behavior, which often exhibit non-linear relationships. As a result, these models often fail to capture the full complexity of the data, leading to lower accuracy in sales predictions.

Many existing systems also rely on simplistic time series forecasting methods, such as the ARIMA (AutoRegressive Integrated Moving Average) model. While ARIMA can model linear patterns in time series data, it struggles with non-linear trends, high volatility, and large datasets, which are increasingly common in retail environments. Additionally, these models are not adept at handling external factors, such as competitive actions or marketing campaigns, which can significantly affect sales. In terms of machine learning, some systems have incorporated decision trees and basic ensemble methods like Random Forests. These models are more flexible and can better handle non-linear data compared to traditional statistical methods. However, they often require extensive feature engineering and can still fall short when managing very large and complex datasets. Overall, while these existing systems offer some level of sales prediction, they are limited in their ability to adapt to evolving trends and complex datasets. More advanced machine learning models, such as Gradient Boosting, offer greater potential for improving the accuracy and reliability of sales forecasting in supermarkets.

3.2 PROPOSED SYSTEM

The proposed system leverages advanced machine learning techniques, specifically the XGBoost algorithm, to enhance the accuracy and efficiency of sales prediction in supermarkets. Unlike traditional methods, the system is designed to handle large, complex datasets with multiple variables, such as historical sales data, promotional activities, weather patterns, holidays, and local events, which significantly impact consumer purchasing behavior. The system aims to provide a robust, scalable solution for sales forecasting, offering reliable predictions that can inform strategic and operational decisions.

The core of the proposed system is built on the XGBoost model, an optimized gradient boosting technique that efficiently combines the predictions of multiple decision trees to create a stronger, more accurate model. XGBoost is known for its speed, scalability, and ability to handle both linear and non-linear relationships in data, making it highly suitable for the diverse and complex nature of supermarket sales data. The algorithm's ability to iteratively improve predictions by correcting errors from previous models ensures high accuracy in forecasting sales trends.

Key features of the proposed system include:

1. Data Integration: The system integrates various data sources, including sales transactions, promotions, customer demographics, weather forecasts, and events, ensuring a comprehensive view of factors that affect sales.

2. Data Preprocessing and Feature Engineering: The system includes automated data cleaning, feature selection, and transformation to ensure the data is ready for machine learning models. Key features like seasonal trends, promotional effects, and day-of-week patterns are extracted to improve model performance.

3. Model Training and Optimization: The XGBoost algorithm is trained using historical sales data, and hyperparameters are optimized to improve predictive accuracy. Cross-validation techniques are applied to ensure that the model generalizes well to unseen data.

4. Real-time Prediction and Monitoring: The proposed system supports real-time sales forecasting, allowing supermarket managers to make on-the-fly adjustments to inventory, staffing, and promotions based on predicted sales.

5. Scalability and Flexibility: The system is designed to be scalable, enabling it to handle a growing amount of data as the supermarket expands. It also allows for flexibility in incorporating new data sources or adjusting the model based on evolving business needs.

3.3 SOFTWARE REQUIREMENT SPECIFICATION

HARDWARE REQUIREMENTS:-

- **Processor :** Intel Core i5 / AMD Ryzen 5 (minimum) | Intel Core i7 / AMD Ryzen 7 (recommended)

- **RAM :**8 GB (minimum) | 16 GB or more (recommended)
- **Hard Disk :** 100 GB SSD (minimum) | 250 GB SSD / NVMe (recommended)
- **Keyboard :** Standard Windows / Mac Keyboard
- **Mouse** :Two or Three Button Optical Mouse
- **Monitor** : Full HD Monitor (1920x1080 resolution recommended)

SOFTWARE REQUIREMENTS:

- **Operating System :** Ubuntu 20.04/22.04 LTS, Windows 10/11, macOS
- **Coding Language:** Python 3.8+
- **Designing :** Figma, Adobe XD (for UI/UX design)
- **Libraries & Frameworks:**
- **Machine Learning:** XGBoost, gradient Boosting, Random Forest Regression, Linear Regression
- **Data Processing & Visualization:** Pandas, NumPy, Matplotlib, Seaborn
- **Causal Inference:** DoWhy
- **Development Tools:** Jupyter Notebook (for ML experimentation),VS Code / PyCharm (for coding),Git & GitHub (for version control)

The **Software Requirement Specification (SRS)** defines the functional and non-functional requirements for the proposed sales forecasting system, detailing the necessary software tools and frameworks for its successful implementation.

1. **Performance Requirements**
   The system must be capable of processing large datasets efficiently, with support for up to 100,000 records. The sales predictions generated from the dataset should be produced within 5 seconds to ensure fast and responsive interactions. Additionally, the system should be scalable to accommodate larger datasets without compromising performance, handling both batch processing for historical data analysis and real-time predictions for immediate use.
2. **Security Requirements**
   Security is a top priority for the system, and it must include role-based access control (RBAC) to ensure that only authorized users can access certain features and data. Sensitive information, such as sales data and user credentials, should be encrypted using secure standards like AES for storage and TLS/SSL for data transmission. The system must comply with data protection regulations, such as GDPR or CCPA, and should provide multi-factor authentication (MFA) for secure user login.
3. **Data Storage Requirements**
   The system should store sales and forecasting data in a relational database management system (RDBMS) like MySQL or PostgreSQL. This will allow efficient data management

and querying. The system must support indexing to optimize query performance and provide automated backup and recovery processes to safeguard against data loss. It should also offer the ability to archive old sales data after a defined retention period.

4. **Usability Requirements**
   The system must provide a user-friendly web-based interface that is easy to navigate, even for users with minimal training. The interface should be responsive, ensuring usability across different devices, such as desktops, tablets, and smartphones. In addition to a straightforward UI, comprehensive documentation, including a user manual and online help, should be available to guide users through key tasks, such as data entry, model selection, and generating reports.
5. **Reliability Requirements**
   The system must ensure high availability, with a target uptime of 99.9%. In the event of system failures, the software should support automated failover and disaster recovery processes to minimize downtime. The system should also generate alerts and logs to help monitor system performance and quickly address any issues, ensuring reliability in both development and production environments.
6. **Compatibility Requirements**
   The system must be compatible with various operating systems, such as Windows, macOS, and Linux, ensuring wide accessibility. It should also support major web browsers, including Chrome, Firefox, Safari, and Edge. Additionally, the system must provide API integration capabilities for easy data import from external sources such as ERP systems and external sales databases.
7. **Maintainability Requirements**
   To facilitate long-term support, the system architecture should be modular and easy to maintain, allowing for seamless updates, bug fixes, and the addition of new features. Automated testing should be implemented to validate the system's functionality and ensure ongoing reliability. Moreover, the system must offer robust logging and monitoring tools, which will help diagnose and resolve issues efficiently.
8. **Compliance Requirements**
   The system must adhere to relevant industry compliance standards, such as GDPR for data protection and PCI-DSS for secure handling of payment information. It should follow best practices for data privacy and security, ensuring that all personal and transactional data is protected throughout its lifecycle. Compliance with regional and global standards will be crucial for the system's acceptance in diverse markets.

3.4 SDLC METHODOLOGIES

The **Software Development Life Cycle (SDLC)** methodology refers to the process of planning, creating, testing, and deploying an information system. Several SDLC methodologies exist, each with its own process and use cases. Here, we’ll discuss the most commonly used SDLC methodologies in detail, focusing on their applicability to the sales forecasting system.

WATERFALL MODEL

The **Waterfall Model** is a traditional and linear approach to software development, where each phase is completed before the next begins. It follows a strict sequence of steps, such as requirement gathering, design, implementation, testing, and maintenance. This methodology is best suited for projects with well-defined requirements that are unlikely to change throughout development.

In the context of sales forecasting, the Waterfall Model can be effective when the requirements are clearly understood at the start of the project. However, it can become rigid if changes in the business environment or data sources are required during the development process. The lack of flexibility may hinder its effectiveness for a dynamic forecasting system that needs continuous updates.

AGILE METHODOLOGY

The **Agile** methodology is iterative and incremental, emphasizing flexibility and customer collaboration over following a fixed plan. In Agile, development is broken into small cycles or **sprints**, usually lasting 2–4 weeks, where a part of the system is developed, tested, and reviewed. This allows for constant feedback, making it suitable for projects with evolving requirements. Agile focuses on continuous improvement and adapting to changing conditions.

For sales forecasting systems, Agile is highly beneficial due to the changing nature of business data and customer behavior. It allows the development team to make adjustments quickly based on feedback, improving the accuracy of forecasting models over time. Regular iterations make it easier to integrate new data sources or improve the system's functionality in response to changing market trends.

SCRUM FRAMEWORK

Scrum is a specific framework within Agile that organizes the development process into fixed-length **Sprints**. The team consists of distinct roles: the **Product Owner**, the **Scrum Master**, and the **Development Team**. Scrum emphasizes frequent communication, transparency, and iterative development. Sprints typically last 2–4 weeks, and at the end of each sprint, a functional piece of the software is delivered.

In the case of sales forecasting, Scrum can be effective due to the need for frequent updates and adjustments to the forecasting models. Regular feedback from stakeholders ensures that the system evolves in line with changing business requirements. The framework promotes collaboration, allowing teams to focus on continuous improvement and the delivery of valuable features.

SPIRAL MODEL

The **Spiral Model** combines the advantages of both iterative development and risk management. The development process is divided into **spirals**, each containing planning, risk analysis, engineering, testing, and evaluation. This model is highly risk-driven, making it suitable for projects where uncertainty is a major concern.

For sales forecasting systems, the Spiral Model allows for regular risk assessments and adjustments to the predictive models. This is particularly useful when dealing with uncertainties in data sources, algorithm performance, or business conditions. The ability to prototype and test at each phase ensures that the system is refined iteratively, addressing potential issues early on in the development process.

V-MODEL (VERIFICATION AND VALIDATION)

The **V-Model** is an extension of the Waterfall model with a strong focus on verification and validation. For each development phase, there is a corresponding testing phase. For example, after the system design phase, the corresponding phase would involve system integration testing to ensure that the design meets the required standards.

This model is well-suited for projects that require high reliability and precision. For sales forecasting systems, where even small errors can lead to significant business consequences, the V-Model ensures that testing is integrated throughout the development process. Every phase of development is validated before moving forward, making it suitable for systems requiring a high level of accuracy and stability.

3.5 FUNCTIONAL REQUIREMENT

The **Functional Requirements** describe the core features and functions that the proposed sales forecasting system must support to fulfill its intended purpose. These requirements focus on the user interactions, system behavior, and overall performance in relation to the specific sales prediction tasks.

1. **User Authentication and Access Control**
   The system must provide secure user authentication with the ability to manage different user roles, such as Admin, Analyst, and Viewer. Admin users should have the ability to add or remove users, manage permissions, and configure system settings. Analysts should be able to input data, generate forecasts, and access historical reports, while Viewers can only access forecast results and reports. The system should support role-based access control (RBAC) to ensure that users can only access data and features appropriate to their roles.
2. **Data Input and Import**
   The system should allow users to input sales data manually via a form or by importing datasets

in various formats, such as CSV, Excel, and JSON. Additionally, the system must support API integration to import sales data from external sources, such as ERP systems, for automated data import. Users should be able to review and edit the data before starting the prediction process.

3. **Sales Forecasting Model**
   The core functionality of the system is to generate accurate sales forecasts using machine learning models. The system should include a selection of pre-configured models, such as Gradient Boosting, Decision Trees, and Linear Regression, that can be used to forecast sales based on historical data. Users should be able to choose the model they want to use for predictions, with the option to fine-tune the model parameters for better accuracy.
4. **Prediction Generation and Visualization**
   The system must generate sales predictions based on the input data and selected model. Predictions should be displayed in an easy-to-read format, such as graphs, charts, or tables, to facilitate analysis. The system must also support features such as trend analysis, where users can compare predicted sales over different time periods (e.g., monthly, quarterly, or yearly). Additionally, users should be able to download the prediction results as reports in PDF or Excel formats.
5. **Data Validation and Error Handling**
   The system should automatically validate the input data for errors or inconsistencies (e.g., missing values, incorrect formats) before running any predictions. If invalid data is detected, the system must alert the user and provide suggestions for correcting the errors. For example, the system should prompt the user to fill in missing values or correct any formatting issues in the uploaded data.
6. **Historical Data and Report Generation**
   Users should be able to view historical sales data, either through reports or visualizations, to help them analyze past performance and trends. The system must allow users to generate custom reports based on sales data, forecast results, and key performance indicators (KPIs). These reports should be exportable in various formats, such as PDF and Excel, for sharing or printing purposes.
7. **Model Training and Evaluation**
   The system should support model training functionality where users can train machine learning models on historical data to optimize predictions. Users should be able to evaluate the model's performance using metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared to assess the model’s accuracy. The system must also allow users to re-train models with updated datasets to improve forecasting accuracy over time.
8. **Alert and Notification System**
   The system should include an alert and notification feature that informs users of important events or system activities. For instance, users can be notified when a forecast is ready or when there are issues with the input data, such as missing or incompatible values.

Notifications can be delivered via email, in-app alerts, or SMS, depending on user preferences.

9. **Customizable Dashboard**
   The system must provide users with a customizable dashboard that presents key metrics and sales forecasts at a glance. The dashboard should display critical sales data, forecast results, historical trends, and alerts. Users should be able to select which elements to display and organize them based on their preferences.
10. **Audit and Log Management**
   The system must maintain a comprehensive log of all actions performed by users, including data inputs, model selections, forecast generations, and report downloads. These logs are essential for auditing purposes and ensuring that all system activities are traceable for security and compliance reasons.

CHAPTER – 4
SYSTEM DESIGN

4.1 IMPORTANCE OF DESIGN

The design phase is crucial as it serves as the blueprint for the entire system, ensuring clarity of requirements, efficient resource allocation, and scalability. A well-structured design helps in clearly defining the system's objectives, ensuring all business needs, like accurate sales predictions and user-friendly visualizations, are met. It optimizes the use of resources, preventing unnecessary complexities, and ensures the system can handle growth by being scalable. Additionally, a thoughtful design enhances user experience, making it easier for stakeholders to interact with the system and interpret forecast results. It also facilitates maintenance and future upgrades, allowing for easy troubleshooting and the integration of new models or features without disrupting core operations. A secure design is vital for protecting sensitive data, incorporating necessary security measures from the outset. Furthermore, a well-thought-out design helps control project costs by identifying potential risks and inefficiencies early on, ensuring the system is built within budget constraints. Overall, the design phase is foundational for creating a successful, efficient, and sustainable sales forecasting system.

4.2 SYSTEM ARCHITECTURE

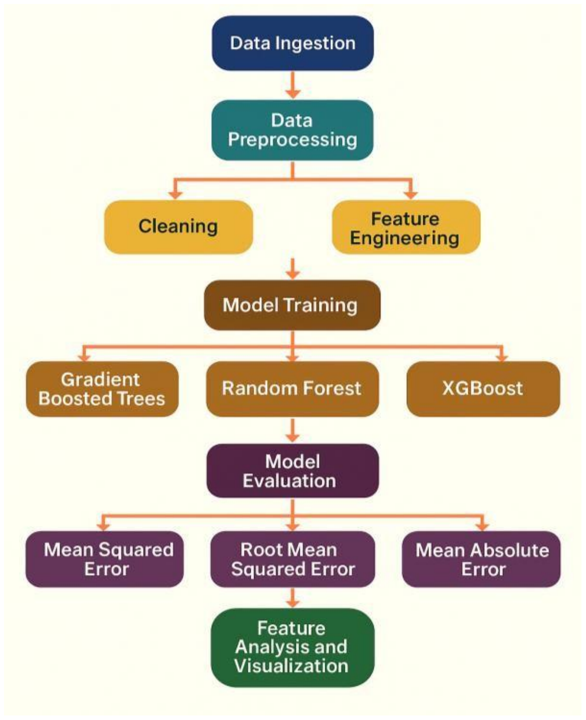


Fig 4.2.1: System Architecture For Sales Forecasting System

4.3 UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

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

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

4.3.1 USE CASE DIAGRAM:

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

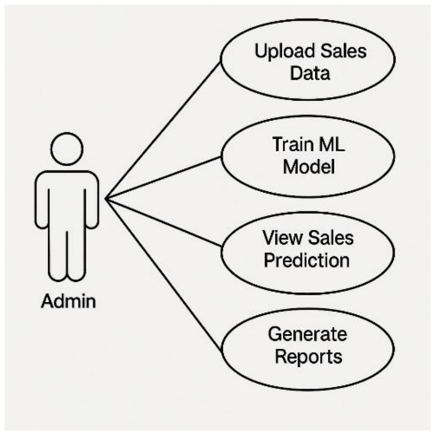


Fig 4.3.1: Use case diagram

4.3.2 CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

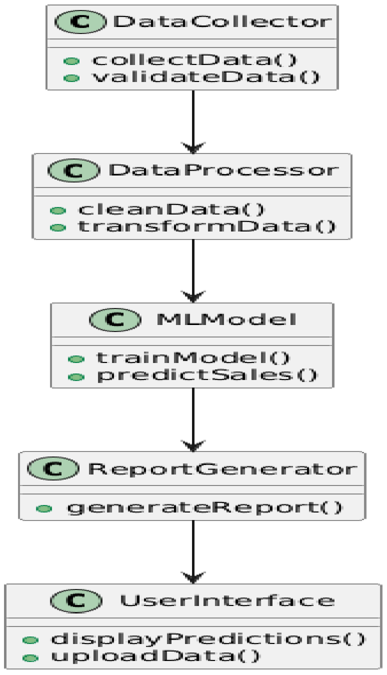


Fig 4.3.2: Class diagram

4.3.3 ACTIVITY DIAGRAM:

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

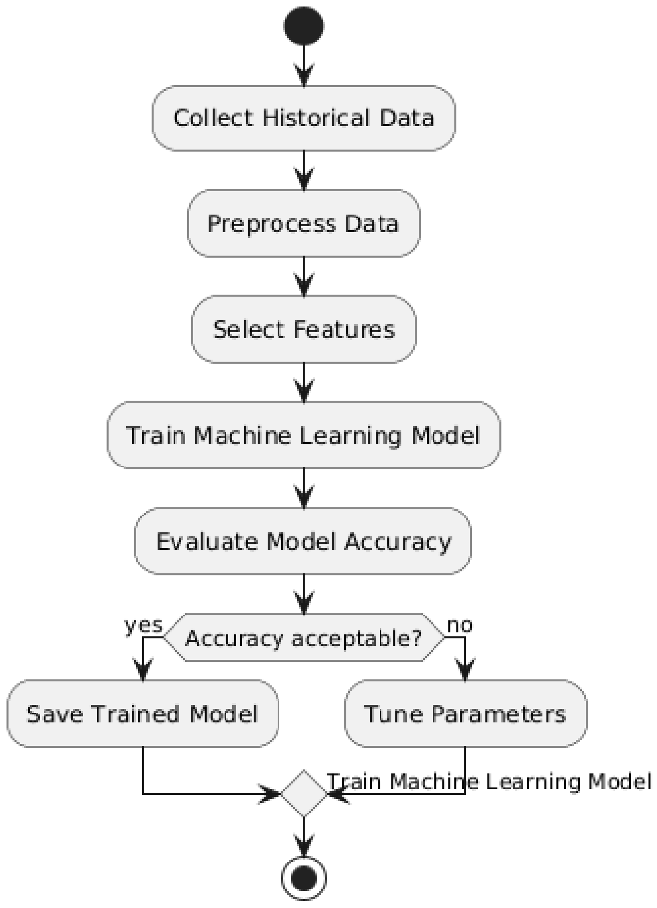


Fig 4.3.3 : Activity Diagram

4.3.4 SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

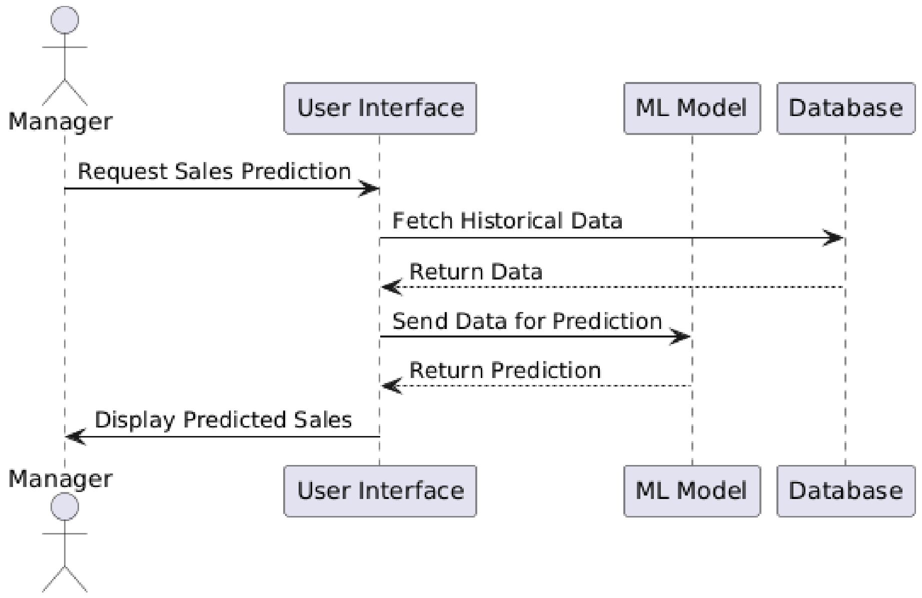


Fig 4.3.4: Sequence diagram

4.3.5 COMPONENT DIAGRAM

A component diagram is a type of UML (Unified Modeling Language) diagram that visualizes the structure of a system by showing its components and their relationships. Components represent modular, replaceable parts of a system, like libraries, executables, or services. The diagram illustrates how these components interact through interfaces and dependencies, supporting system architecture design and analysis. It helps stakeholders understand how software is divided into manageable parts and how they integrate. Key elements include components, interfaces, connectors, and dependencies. Component diagrams are useful in both development and deployment phases to ensure clear communication of system structure and organization.

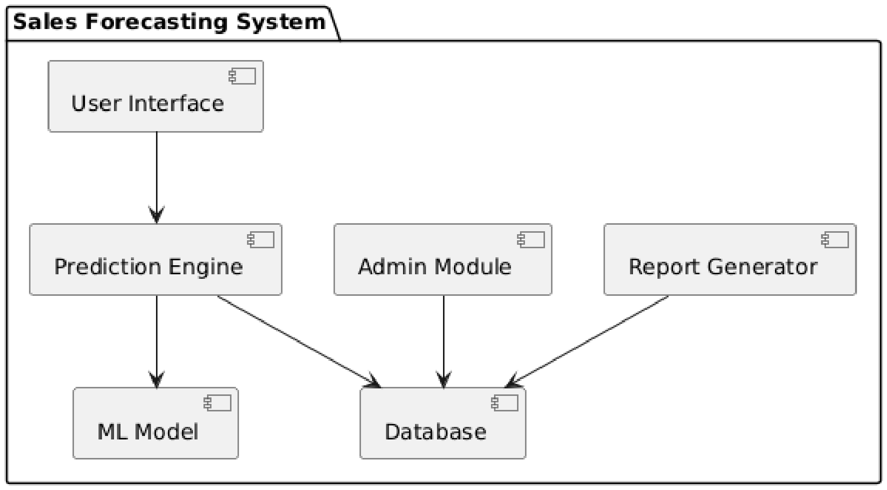


Fig 4.3.5: Component Diagram

4.3.6 DEPLOYMENT DIAGRAM

A deployment diagram is a type of UML diagram that models the physical deployment of software artifacts on hardware nodes. It shows how system components are distributed across servers, devices, and networks. Key elements include nodes (hardware or execution environments), artifacts (software like executables or databases), and communication paths (connections between nodes). Deployment diagrams help visualize system topology, such as cloud servers, client devices, and embedded systems. They are especially useful in understanding performance, scalability, and network configurations. These diagrams guide infrastructure planning, ensuring that the system's physical architecture supports its functional requirements effectively and reliably during operation.

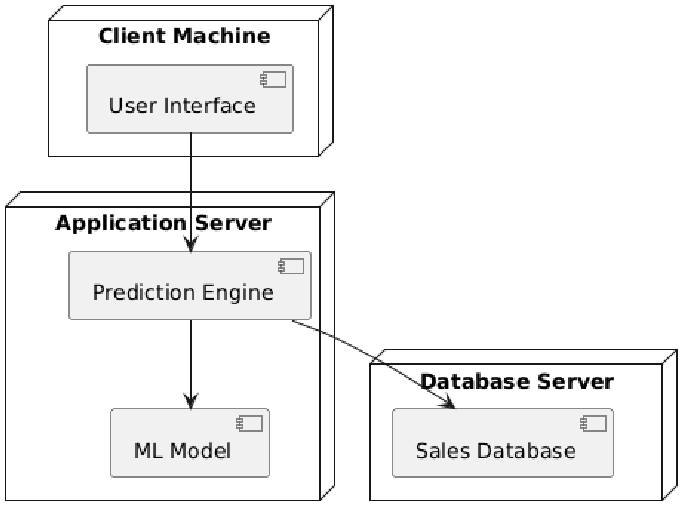


Fig 4.3.6: Deployment Diagram

CHAPTER - 5
IMPLEMENTATION

5.1 MODULES AND DESCRPTION:-

MODULES:

1. Data Collection Module

This module serves as the foundation of the entire system by acquiring relevant data from multiple sources. These sources may include transactional sales databases, product master files, inventory logs, marketing campaign schedules, customer demographic data, and external factors such as weather APIs and public holiday calendars. The goal is to aggregate diverse datasets into a unified structure. Proper integration and periodic updating of this data ensure the model reflects the latest market conditions and consumer behavior.

2. Data Preprocessing Module

Raw data is rarely perfect. This module focuses on cleaning and preparing data to improve quality and consistency. Tasks include filling in or removing missing values, detecting and correcting outliers, converting categorical variables into numerical formats (e.g., one-hot encoding), and ensuring time-series consistency (e.g., removing duplicates or aligning timestamps). This module may also normalize or scale data to make it suitable for algorithmic processing, which is essential to prevent bias or skewness in model training.

3. Feature Engineering Module

Feature engineering transforms raw data into informative attributes that enhance the predictive power of the model. It involves deriving new features such as lagged sales (previous day/week/month), moving averages, promotion flags, holiday indicators, weekend/weekdays, and price changes. Seasonal decomposition and trend indicators can also be generated. This module plays a crucial role in capturing the temporal, promotional, and cyclical aspects of supermarket sales data—factors that traditional methods often overlook.

4. Model Training Module

This is the core machine learning module where the Gradient Boosting model is trained using the prepared dataset. It includes selecting the model architecture, splitting the dataset into training and testing sets, applying cross-validation techniques, and performing hyperparameter optimization (e.g., learning rate, max depth, number of trees). Advanced training methods like early stopping and regularization may also be used to prevent overfitting and improve generalization to new data.

5. Prediction Module

Once the model is trained, this module uses it to generate future sales predictions. Users can input new data (such as upcoming dates, holidays, and planned promotions) to forecast expected sales at various levels—item-level, category-level, or store-level. These predictions help decision-makers plan stock replenishment, manage shelf space, schedule labor, and launch targeted campaigns. This module may be configured to generate short-term (weekly) or long-term (monthly or quarterly) forecasts.

6. Evaluation Module

This module rigorously assesses the accuracy and reliability of the forecasting model. Evaluation is performed using standard regression metrics such as:

- RMSE (Root Mean Square Error): Measures the average magnitude of prediction errors.
- MAE (Mean Absolute Error): Measures average absolute difference between actual and predicted values.
- R^2 Score (Coefficient of Determination): Indicates how well future samples are likely to be predicted.
  It also supports visual performance tracking via residual plots, learning curves, and actual vs. predicted graphs.

7. Visualization and Reporting Module

To ensure usability by business managers and decision-makers, this module generates intuitive dashboards and charts that present the forecast results clearly. Visualizations include time-series plots, trend analysis, bar graphs comparing forecasted vs. actual sales, and heatmaps showing high-performing items or time periods. Reports may also summarize model accuracy, peak sales periods, and anomalies, aiding strategic planning and stakeholder communication.

8. Forecast Deployment Module

The final module ensures that the trained model and prediction capabilities are integrated into the supermarket's operational environment. It can be embedded into retail management software or cloud platforms, with scheduled runs (e.g., daily or weekly forecasts) or real-time predictions triggered by new data inputs. The module supports scalability to multiple branches and adaptability to business changes, such as new product introductions or market expansions. It also includes model retraining pipelines to keep forecasts accurate over time.

5.2 SOFTWARE ENVIRONMENT

What is Python :

Below are some facts about Python.

Python is currently the most widely used multi-purpose, high-level programming language.

Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.

Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.

Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc.

The biggest strength of Python is huge collection of standard library which can be used for the following –

- Machine Learning
- GUI Applications (like Kivy, Tkinter, PyQt etc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like Opencv, Pillow)
- Web scraping (like Scrapy, BeautifulSoup, Selenium)
- Test frameworks
- Multimedia

Advantages of Python :-

Let’s see how Python dominates over other languages.

1. Extensive Libraries

Python downloads with an extensive library and it contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don’t have to write the complete code for that manually.

2. Extensible

As we have seen earlier, Python can be **extended to other languages**. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

3. Embeddable

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add **scripting capabilities** to our code in the other language.

4. Improved Productivity

The language’s simplicity and extensive libraries render programmers **more productive** than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

5. IOT Opportunities

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet Of Things. This is a way to connect the language with the real world.

When working with Java, you may have to create a class to print **‘Hello World’**. But in Python, just a print statement will do. It is also quite **easy to learn, understand,** and **code.** This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

7. Readable

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and **indentation is mandatory.** This further aids the readability of the code.

8. Object-Oriented

This language supports both the **procedural and object-oriented** programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the **encapsulation of data** and functions into one.

9. Free and Open-Source

Like we said earlier, Python is **freely available.** But not only can you **download Python** for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

10. Portable

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn't the same with Python. Here, you need to **code only once**, and you can run it anywhere. This is called **Write Once Run Anywhere (WORA)**. However, you need to be careful enough not to include any system-dependent features.

11. Interpreted

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, **debugging is easier** than in compiled languages.

Advantages of Python Over Other Languages :

1. Less Coding

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don't have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

2. Affordable

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

The 2019 Github annual survey showed us that Python has overtaken Java in the most popular programming language category.

3. Python is for Everyone

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and **machine learning**, automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

Disadvantages of Python

So far, we've seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let's now see the downsides of choosing Python over another language.

1. Speed Limitations

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in **slow execution**. This, however, isn't a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

2. Weak in Mobile Computing and Browsers

While it serves as an excellent server-side language, Python is much rarely seen on the **client-side**. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called **Carbonnelle**.

The reason it is not so famous despite the existence of Brython is that it isn't that secure.

3. Design Restrictions

As you know, Python is **dynamically-typed**. This means that you don't need to declare the type of variable while writing the code. It uses **duck-typing**. But wait, what's that? Well, it just means that if it looks like a duck, it must be a duck.

While this is easy on the programmers during coding, it can **raise run-time errors**.

4. Underdeveloped Database Access Layers

Compared to more widely used technologies like **JDBC (Java DataBase Connectivity)** and **ODBC (Open DataBase Connectivity)**, Python's database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

5. Simple

No, we're not kidding. Python's simplicity can indeed be a problem. Take my example. I don't do Java, I'm more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

This was all about the Advantages and Disadvantages of Python Programming Language.

History of Python : -

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde &Informatica). The greatest achievement of ABC was to influence the design of Python.Python was conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners[1], Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde en Informatica (CWI). I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it."Later on in the same Interview, Guido van Rossum continued: "I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So I started typing. I created a simple virtual machine, a

simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax, used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings, and numbers."

What is Machine Learning : -

Before we take a look at the details of various machine learning methods, let's start by looking at what machine learning is, and what it isn't. Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of *building models of data*.

Fundamentally, machine learning involves building mathematical models to help understand data. "Learning" enters the fray when we give these models *tunable parameters* that can be adapted to observed data; in this way the program can be considered to be "learning" from the data. Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data. I'll leave to the reader the more philosophical digression regarding the extent to which this type of mathematical, model-based "learning" is similar to the "learning" exhibited by the human brain.Understanding the problem setting in machine learning is essential to using these tools effectively, and so we will start with some broad categorizations of the types of approaches we'll discuss here.

Categories Of Machine Leaning :-

At the most fundamental level, machine learning can be categorized into two main types: supervised learning and unsupervised learning.*Supervised learning* involves somehow modeling the relationship between measured features of data and some label associated with the data; once this model is determined, it can be used to apply labels to new, unknown data. This is further subdivided into *classification* tasks and *regression* tasks: in classification, the labels are discrete categories, while in regression, the labels are continuous quantities.

We will see examples of both types of supervised learning in the following section.

Unsupervised learning involves modeling the features of a dataset without reference to any label, and is often described as "letting the dataset speak for itself." These models include tasks such as *clustering* and *dimensionality reduction.* Clustering algorithms identify distinct groups of data, while dimensionality reduction algorithms search for more succinct representations of the data. We will see examples of both types of unsupervised learning in the following section.

Need for Machine Learning

Human beings, at this moment, are the most intelligent and advanced species on earth because they can think, evaluate and solve complex problems. On the other side, AI is still in its initial stage and haven’t surpassed human intelligence in many aspects. Then the question is that what is the need to make machine learn? The most suitable reason for doing this is, “to make decisions, based on data, with efficiency and scale”.

Lately, organizations are investing heavily in newer technologies like Artificial Intelligence, Machine Learning and Deep Learning to get the key information from data to perform several real-world tasks and solve problems. We can call it data-driven decisions taken by machines, particularly to automate the process. These data-driven decisions can be used, instead of using programing logic, in the problems that cannot be programmed inherently. The fact is that we can’t do without human intelligence, but other aspect is that we all need to solve real-world problems with efficiency at a huge scale. That is why the need for machine learning arises.

Challenges in Machines Learning :-

While Machine Learning is rapidly evolving, making significant strides with cybersecurity and autonomous cars, this segment of AI as whole still has a long way to go. The reason behind is that ML has not been able to overcome number of challenges. The challenges that ML is facing currently are −

Quality of data − Having good-quality data for ML algorithms is one of the biggest challenges. Use of low-quality data leads to the problems related to data preprocessing and feature extraction.

Time-Consuming task − Another challenge faced by ML models is the consumption of time especially for data acquisition, feature extraction and retrieval.

Lack of specialist persons − As ML technology is still in its infancy stage, availability of expert resources is a tough job.

No clear objective for formulating business problems − Having no clear objective and well-defined goal for business problems is another key challenge for ML because this technology is not that mature yet.

Issue of overfitting & underfitting − If the model is overfitting or underfitting, it cannot be represented well for the problem.

Curse of dimensionality − Another challenge ML model faces is too many features of data points. This can be a real hindrance.

Difficulty in deployment − Complexity of the ML model makes it quite difficult to be deployed in real life.

Applications of Machines Learning :-

Machine Learning is the most rapidly growing technology and according to researchers we are in the golden year of AI and ML. It is used to solve many real-world complex problems which cannot be solved with traditional approach. Following are some real-world applications of ML

- Emotion analysis
- Sentiment analysis
- Error detection and prevention
- Weather forecasting and prediction
- Stock market analysis and forecasting
- Speech synthesis
- Speech recognition
- Customer segmentation
- Object recognition
- Fraud detection
- Fraud prevention
- Recommendation of products to customer in online shopping.

Python Modules Used in Project :-

Numpy : Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

Scikit – learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. **Python**

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

Install Python Step-by-Step in Windows and Mac :

Python a versatile programming language doesn't come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high-level programming language. Its style philosophy emphasizes code readability with its notable use of great whitespace.

The object-oriented approach and language construct provided by Python enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows.

How to Install Python on Windows and Mac :

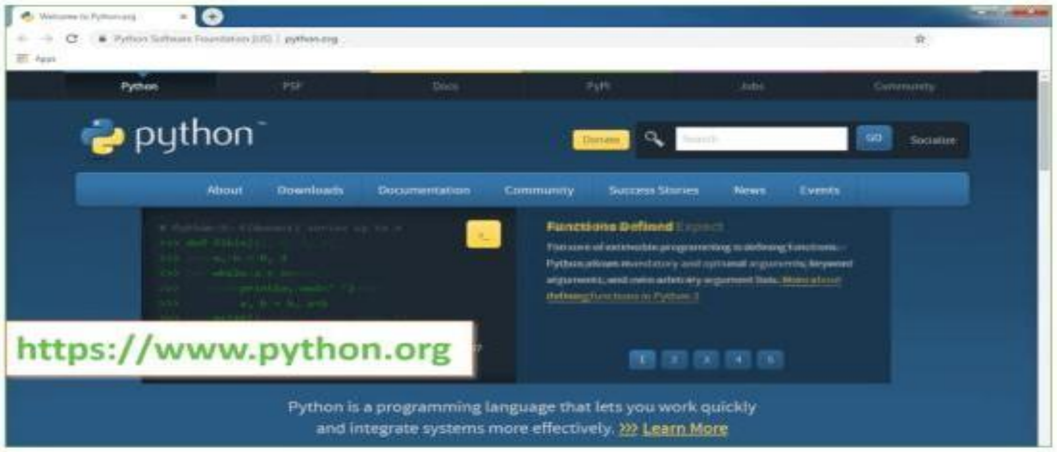
There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python 3.

Note: The python version 3.7.4 cannot be used on Windows XP or earlier devices.

Before you start with the installation process of Python. First, you need to know about your **System Requirements**. Based on your system type i.e. operating system and based processor, you must download the python version. My system type is a **Windows 64-bit operating system**. So the steps below are to install python version 3.7.4 on Windows 7 device or to install Python 3. Download the Python Cheatsheet here.The steps on how to install Python on Windows 10, 8 and 7 are **divided into 4 parts** to help understand better.

Download the Correct version into the system

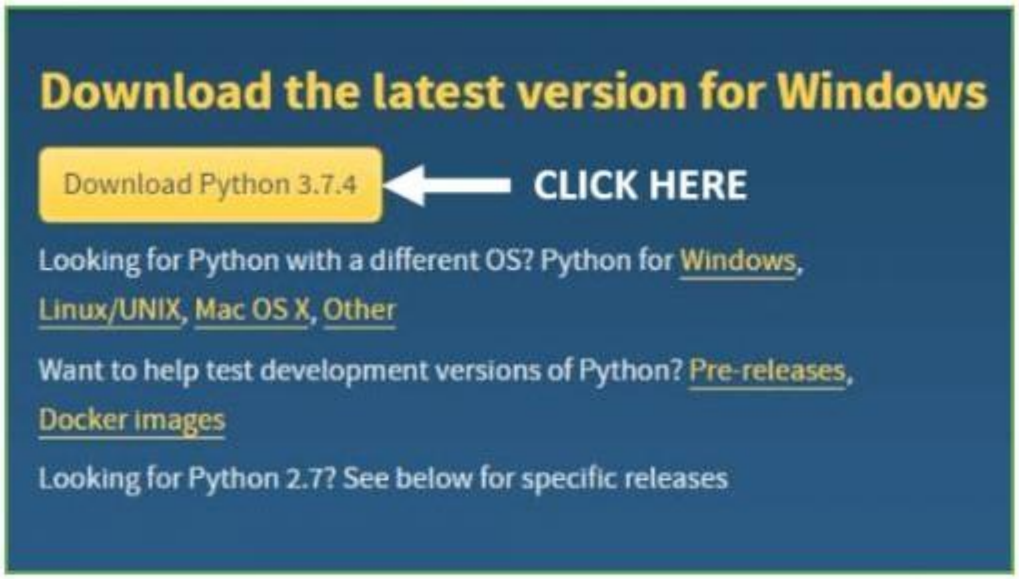
Step 1: Go to the official site to download and install python using Google Chrome or any other web browser. OR Click on the following link: https://www.python.org



Python is a programming language that lets you work quickly and integrate systems more effectively. >>> Learn More

Now, check for the latest and the correct version for your operating system.

Step 2: Click on the Download Tab.



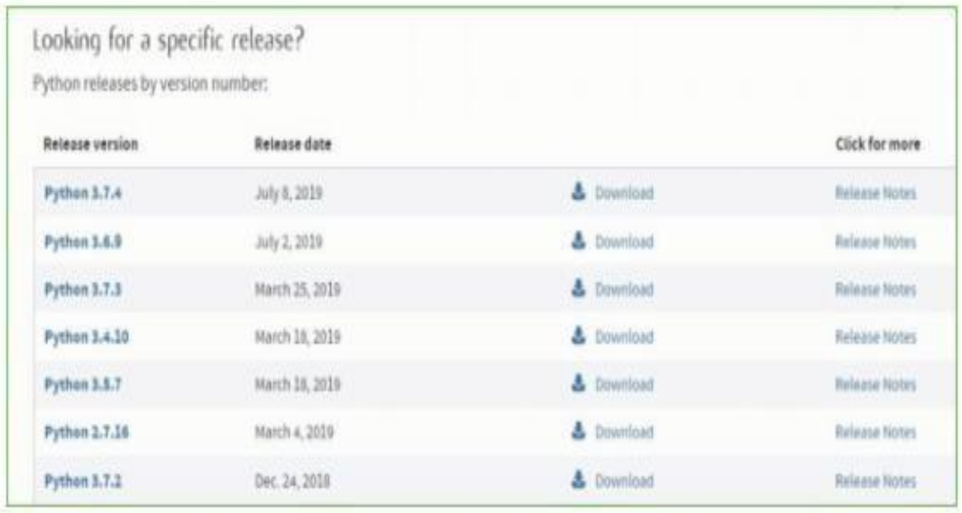
Download the latest version for Windows

Looking for Python with a different OS? Python for Windows, Linux/UNIX, Mac OS X, Other

Want to help test development versions of Python? Pre-releases, Docker images

Looking for Python 2.7? See below for specific releases

Step 3: You can either select the Download Python for windows 3.7.4 button in Yellow Color or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4

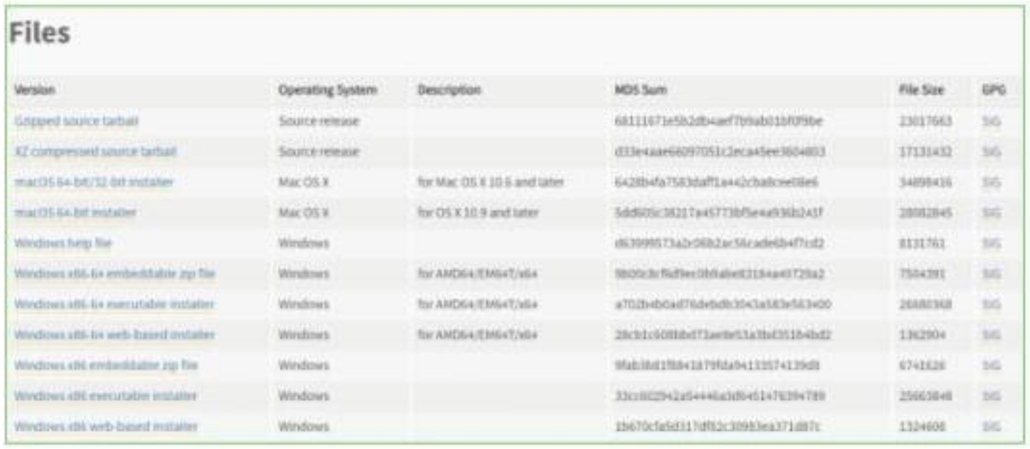


Looking for a specific release?

Python releases by version number:

Release version	Release date		Click for more
Python 3.7.4	July 8, 2019	Download	Release Notes
Python 3.6.9	July 2, 2019	Download	Release Notes
Python 3.7.3	March 25, 2019	Download	Release Notes
Python 3.4.10	March 18, 2019	Download	Release Notes
Python 3.5.7	March 18, 2019	Download	Release Notes
Python 2.7.16	March 4, 2019	Download	Release Notes
Python 3.7.2	Dec. 24, 2018	Download	Release Notes

Step 4: Scroll down the page until you find the Files option.



Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		68111671e5b2db4aef7b9ab01bf0f9be	23017663	SIG
XZ compressed source tarball	Source release		d33e4aae66097051c2eca45ee3604803	17131432	SIG
macOS 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	6428b4fa7583daff1a442cba8cee08e6	34898416	SIG
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	5dd605c38217a45773bf5e4a936b241f	28082845	SIG
Windows help file	Windows		d63999573a2c06b2ac56cade6b4f7cd2	8131761	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	9b00c8cf6d9ec0b9abe83184a40729a2	7504391	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	a702b4b0ad76debdb3043a583e563400	26680368	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	28cb1c608bbd73ae8e53a3bd351b4bd2	1362904	SIG
Windows x86 embeddable zip file	Windows		9fab3b81f8841879fda94133574139d8	6741626	SIG
Windows x86 executable installer	Windows		33cc602942a54446a3d6451476394789	25663848	SIG
Windows x86 web-based installer	Windows		1b670cfa5d317df82c30983ea371d87c	1324608	SIG

Step 5: Here you see a different version of python along with the operating system.

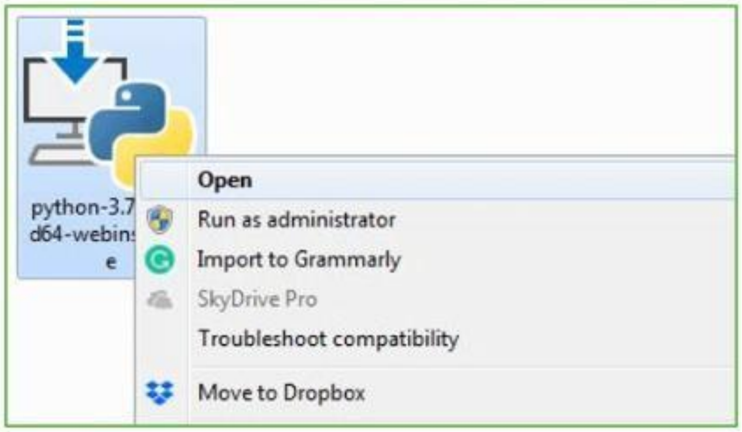
- To download Windows 32-bit python, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer.
- To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer.

Here we will install Windows x86-64 web-based installer. Here your first part regarding which version of python is to be downloaded is completed. Now we move ahead with the second part in installing python i.e. Installation

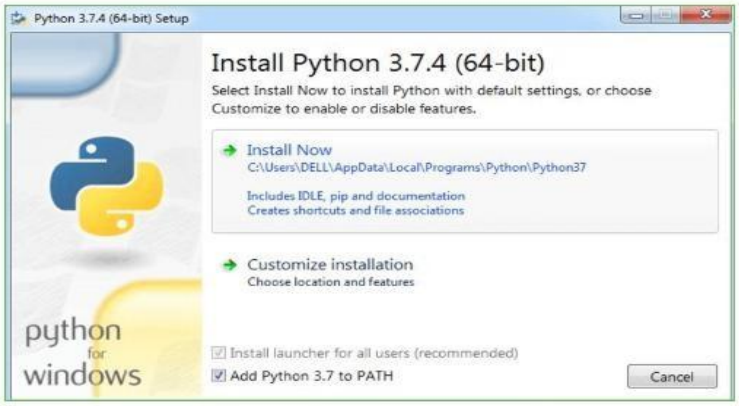
Note: To know the changes or updates that are made in the version you can click on the Release Note Option.

Installation of Python

Step 1: Go to Download and Open the downloaded python version to carry out the installation process.



Step 2: Before you click on Install Now, Make sure to put a tick on Add Python 3.7 to PATH.


Python 3.7.4 (64-bit) Setup

Install Python 3.7.4 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

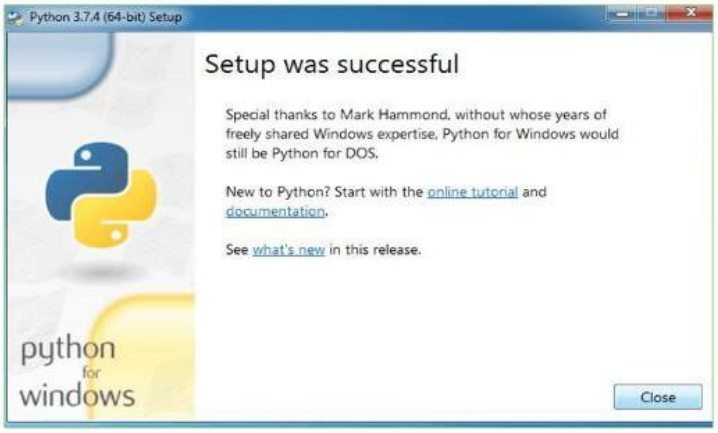
→ Install Now
C:\Users\DELL\AppData\Local\Programs\Python\Python37

Includes IDLE, pip and documentation
Creates shortcuts and file associations

→ Customize installation
Choose location and features

☑ Install launcher for all users (recommended)
☑ Add Python 3.7 to PATH

Step 3: Click on Install NOW After the installation is successful. Click on Close.


Python 3.7.4 (64-bit) Setup

Setup was successful

Special thanks to Mark Hammond, without whose years of freely shared Windows expertise, Python for Windows would still be Python for DOS.

New to Python? Start with the online tutorial and documentation.

See what's new in this release.

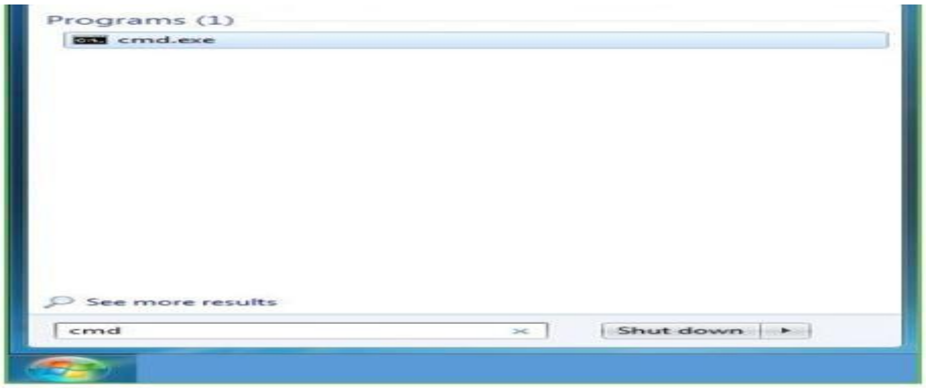
With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

Note: The installation process might take a couple of minutes.

Verify the Python Installation

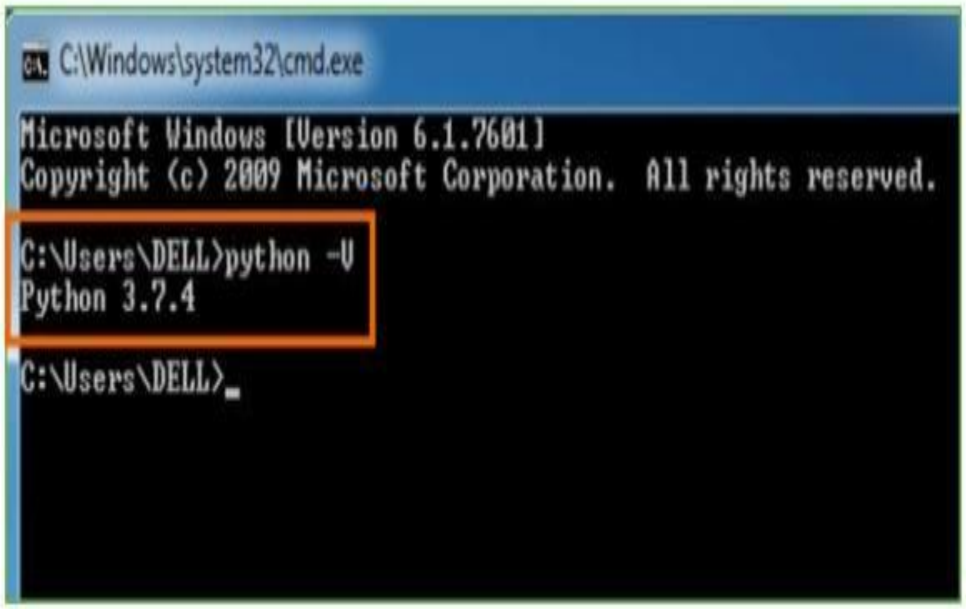
Step 1: Click on Start

Step 2: In the Windows Run Command, type "cmd".



Step 3: Open the Command prompt option.

Step 4: Let us test whether the python is correctly installed. Type **python –V** and press Enter.



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\DELL>python -V
Python 3.7.4

C:\Users\DELL>_
```

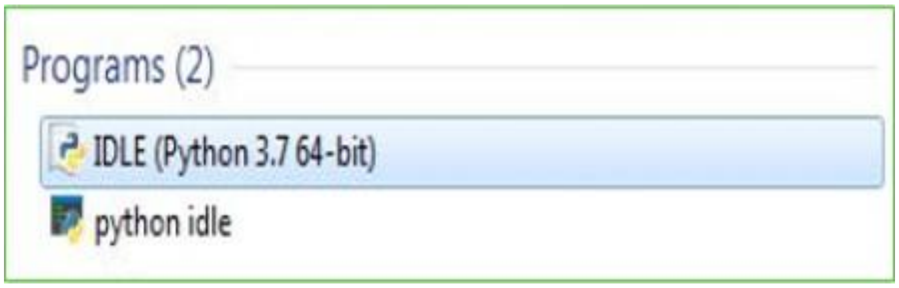
Step 5: You will get the answer as 3.7.4

Note: If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

Check how the Python IDLE works

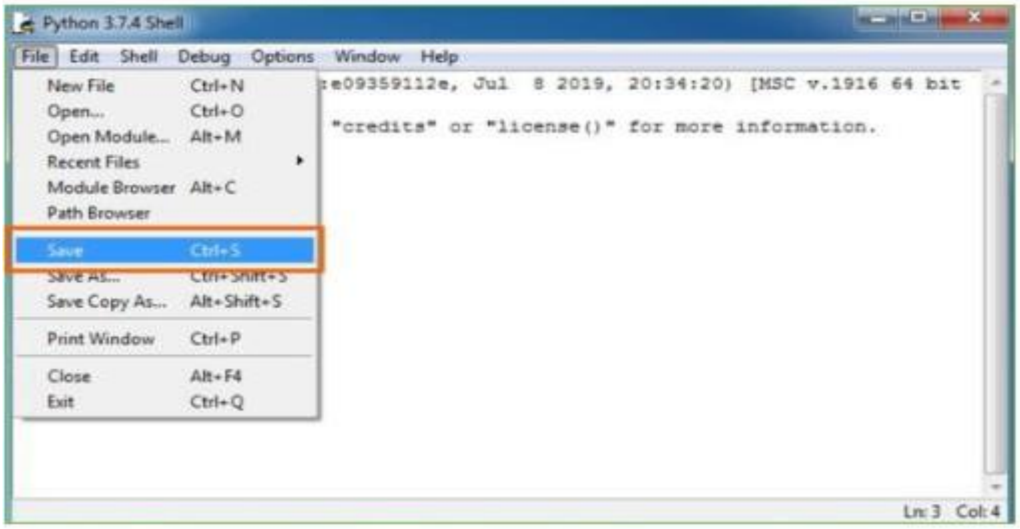
Step 1: Click on Start

Step 2: In the Windows Run command, type "python idle".



Step 3: Click on IDLE (Python 3.7 64-bit) and launch the program

Step 4: To go ahead with working in IDLE you must first save the file. **Click on File > Click on Save**



```
:e09359112e, Jul  8 2019, 20:34:20) [MSC v.1916 64 bit
 "credits" or "license()" for more information.
```

Step 5: Name the file and save as type should be Python files. Click on SAVE. Here I have named the files as Hey World.

Step 6: Now for e.g. **enter print**

Visual Studio Code on Windows

Installation

1. **Download and install Visual Studio Code**

Note

VS Code ships monthly releases and supports auto-update when a new release is available.

2. **Install additional components**

Install Git, Node.js, TypeScript, language runtimes, and more.

3. **Install VS Code extensions from the Visual Studio Marketplace**

Customize VS Code with themes, formatters, language extensions and debuggers for your favorite languages, and more.

4. **Set up AI-assisted coding with GitHub Copilot**

Tip

If you don't yet have a Copilot subscription, you can use Copilot for free by signing up for the Copilot Free plan and get a monthly limit of completions and chat interactions.

5. **Get started with the VS Code tutorial**

Discover the user interface and key features of VS Code.

Install VS Code on Windows

Use the Windows installer

1. Download the Visual Studio Code installer for Windows
2. Once it is downloaded, run the installer (VSCodeUserSetup-{version}.exe)

By default, VS Code is installed under C:\Users\{Username}\AppData\Local\Programs\Microsoft VS Code.

Tip

Setup adds Visual Studio Code to your %PATH% environment variable, to let you type 'code .' in the console to open VS Code on that folder. You need to restart your console after the installation for the change to the %PATH% environmental variable to take effect.

Use the ZIP file

1. Download the **Visual Studio Code Zip archive**
2. Extract the Zip archive, and run VS Code from there

User setup versus system setup

VS Code provides both Windows user and system level setups.

Setup Type	Description
User setup	Does not require administrator privileges to run, as the location is under your user Local A (LOCALAPPDATA) folder. Since it requires no elevation, the user setup is able to provide a sm background update experience. This is the preferred way to install VS Code on Windows. Note: When running VS Code as Administrator in a user setup installation, updates are disabled.

Setup Type	Description
System setup	Requires elevation to administrator privileges to run and places the installation under the system's P Files. The in-product update flow also requires elevation, making it less streamlined than the user set the other hand, installing VS Code using the system setup means that it is available to all users in the s

See the Download Visual Studio Code page for a complete list of available installation options.
Updates

5.3 SAMPLE CODE

```
# Import necessary libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import LabelEncoder, StandardScaler

import dowhy

from dowhy import CausalModel

#1. Data Preprocessing & Feature Engineering

# -----------------------------------------------------

# Drop non-predictive columns

sales_clean = sales.drop(columns=["Invoice ID", "Date", "Time", "gross margin percentage"])

# Encode binary categorical features

binary_features = ['Feature_A', 'Feature_B']  # Replace with actual binary feature names

le = LabelEncoder()

for col in binary_features:
```

```
sales_clean[col] = le.fit_transform(sales_clean[col])
# One-hot encoding for categorical variables
sales_clean = pd.get_dummies(sales_clean)

# Split dataset into training and testing sets
X = sales_clean.drop(columns=["Total"])  # Predictors
y = sales_clean["Total"]  # Target variable
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=1)

# 2. Model Training & Evaluation
# -----------------------------------------------------
# Train XGBoost model
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from xgboost import XGBRegressor
lm = LinearRegression()
model_lm = lm.fit(X_train,y_train)
y_test_pred = model_lm.predict(X_test)

rf2 = RandomForestRegressor(random_state=0)
modelRF = rf2.fit(X_train,y_train)
y_test_pred2 = modelRF.predict(X_test)

gbt = GradientBoostingRegressor(random_state=0)
model_gbt = gbt.fit(X_train,y_train)
```

```
y_pred_gbt = model_gbt.predict(X_test)

xgb_model = XGBRegressor(n_estimators=500, learning_rate=0.05, max_depth=5,
random_state=42)

xgb_model.fit(X_train, y_train)

y_pred_xgb = xgb_model.predict(X_test)

# Evaluate Model Performance

def evaluate_model(y_test, y_pred, model_name):

    mse = mean_squared_error(y_test, y_pred)

    rmse = np.sqrt(mse)

    mae = mean_absolute_error(y_test, y_pred)

    r2 = r2_score(y_test, y_pred)

    mape = np.mean(np.abs((y_test - y_pred) / y_test)) * 100

    accuracy = 100 - mape

    print(f"{model_name} MSE: {mse:.4f}")

    print(f"{model_name} RMSE: {rmse:.4f}")

    print(f"{model_name} MAE: {mae:.4f}")

    print(f"{model_name} R² Score: {r2:.4f} (closer to 1 is better)")

    print(f"{model_name} Accuracy (100 - MAPE): {accuracy:.2f}%")

evaluate_model(y_test, y_pred_xgb, "XGBoost")


# 3. Feature Importance Visualization

# -----------------------------------------------------

# Get feature importance from XGBoost

feature_importances_xgb = xgb_model.feature_importances_

importance_df = pd.DataFrame({'Feature': X_train.columns, 'Importance':
feature_importances_xgb})
```

```
importance_df = importance_df.sort_values(by="Importance", ascending=False)
# Plot feature importance
plt.figure(figsize=(10, 6))
plt.barh(importance_df['Feature'], importance_df['Importance'], color='skyblue')
plt.xlabel("Importance Score")
plt.ylabel("Features")
plt.title("XGBoost Feature Importance")
plt.gca().invert_yaxis()
plt.show()
# 4. Causal Inference Using DoWhy
# -----------------------------------------------------
# Define the causal model
model1 = CausalModel(
    data=sales_clean,
    treatment='Customer type',
    outcome="Total",
    common_causes=[
        'Gender', 'Unit price', 'Quantity', 'Rating', 'Hour',
        'Branch_A', 'Branch_B', 'Branch_C',
        'Product line_Electronic accessories', 'Product line_Fashion accessories',
        'Product line_Food and beverages', 'Product line_Health and beauty',
        'Product line_Home and lifestyle', 'Product line_Sports and travel',
        'Payment_Cash', 'Payment_Credit card', 'Payment_Ewallet',
        'Month_February', 'Month_January', 'Month_March'
    ],
    instruments=[] )
# Visualize the causal graph
model1.view_model()
# Estimate causal effect using Propensity Score Matching
identified_estimand = model1.identify_effect()
estimate1 = model1.estimate_effect(
    identified_estimand,
    method_name="backdoor.propensity_score_matching",
    test_significance=True
)
# Perform robustness check using data subset refuter
refute_results = model1.refute_estimate(
    identified_estimand, estimate1, method_name="data_subset_refuter"
)
print(refute_results)
```

CHAPTER – 6
TESTING AND VALIDATION

6.1 TEST STRATEGY AND APPROACH

SYSTEM TESTING:

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered

6.2 TEST CASES AND VALIDATION

Unit testing:

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Functional testing is centered on the following items:

● Valid Input	: identified classes of valid input must be accepted.
● Invalid Input	: identified classes of invalid input must be rejected.
● Functions	: identified functions must be exercised
● Output	: identified classes of application outputs must be exercised
● Systems/Procedures	: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box . you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works

6.2.1 Test Cases

This section describes the various test cases used to validate the performance of the sales forecasting system. The testing process ensures that the XGBoost-based predictive model functions correctly, provides accurate forecasts, and meets the required business objectives. The test cases focus on different aspects of the system, including data preprocessing, model accuracy, scalability, and real-time predictions.

1. Data Preprocessing Validation

- Test Case: Ensure proper handling of missing values, outliers, and inconsistent data.
- Expected Outcome: The system should clean and preprocess data effectively, ensuring high-quality input for the model.

2. Feature Engineering Evaluation

- Test Case: Validate the extraction and transformation of key features such as seasonal trends, promotional effects, and customer purchasing patterns.
- Expected Outcome: The model should correctly identify and utilize relevant features to improve forecasting accuracy.

3. Model Training and Accuracy Assessment

- Test Case: Train the XGBoost model using historical sales data and evaluate its predictive accuracy using metrics like RMSE, MAE, and R^2 Score.
- Expected Outcome: The model should achieve high accuracy and outperform traditional forecasting methods.

4. Hyperparameter Optimization Testing

- Test Case: Test different hyperparameter configurations to optimize model performance.
- Expected Outcome: The best hyperparameter settings should enhance model efficiency and predictive reliability.

5. Real-time Prediction and Monitoring

- Test Case: Validate the system’s ability to generate real-time sales forecasts based on incoming data.

- Expected Outcome: The system should provide timely and accurate predictions to support inventory and staffing decisions.

6. Scalability and Performance Testing

- Test Case: Assess the system's ability to handle large datasets and increasing data volume as the supermarket expands.
- Expected Outcome: The system should remain efficient and scalable without performance degradation.

CHAPTER - 7
RESULTS AND OUTPUT SCREENS

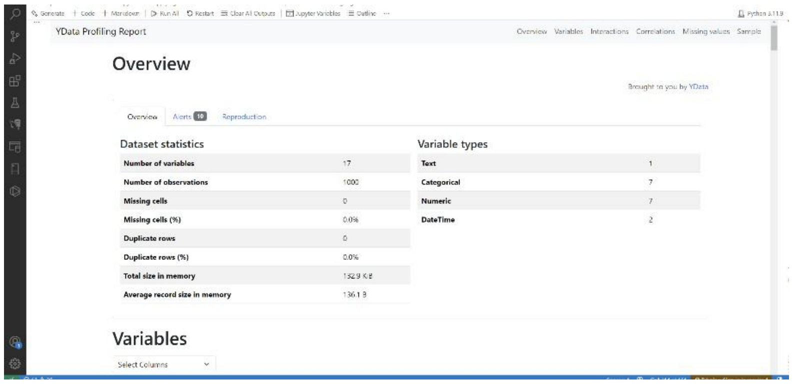
The proposed sales forecasting system, built using the XGBoost algorithm alongside Gradient Boosting, Random Forest, and Linear Regression, delivers highly accurate and reliable predictions by effectively analyzing complex and high-volume sales data from supermarkets. The implementation involved collecting and preprocessing historical sales data, engineering meaningful features, and training the model using advanced machine learning techniques. XGBoost was selected for its ability to handle non-linear relationships, capture intricate patterns, and optimize prediction accuracy through gradient boosting and regularization techniques.

The trained models were evaluated using key metrics such as Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and R^2 Score. Among the models tested, XGBoost achieved superior results, outperforming traditional methods like Linear Regression and ensemble models such as Random Forest and Gradient Boosting. The XGBoost model demonstrated exceptional performance in handling large datasets with seasonal trends, promotional impacts, and variable consumer behavior.

The forecasting outputs were visualized using line charts and bar graphs, clearly demonstrating the alignment between predicted and actual sales figures. These visual reports enable business managers to identify peak demand periods, detect low-performing products, and assess the impact of marketing campaigns and holidays on sales volumes. Additionally, the XGBoost model proved effective in short-term and medium-term forecasting, supporting daily and weekly inventory planning. Its real-time prediction capability enhances the system's utility by allowing dynamic adjustments to stock levels and workforce allocation.

Overall, the results validate that the proposed system offers significant improvements in forecast accuracy, operational efficiency, and decision-making support. By leveraging XGBoost, Gradient Boosting, Random Forest, and Linear Regression, supermarkets can reduce waste, avoid stockouts, and strategically plan for fluctuating demand. The success of this project showcases the power of data-driven, machine learning-based approaches in transforming retail forecasting and optimizing business outcomes.

7.1 SNAPSHOTS



YData Profiling Report

Overview

Dataset statistics

Number of variables	17
Number of observations	1000
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	132.9 KiB
Average record size in memory	136.1 B

Variable types

Text	1
Categorical	7
Numeric	7
DateTime	2

Variables

Fig 7.1.1 YData Profiling Report Overview

The image displays a YData Profiling Report overview of a dataset. It summarizes key dataset statistics, including 17 variables and 1000 observations with no missing or duplicate data. The dataset contains text, categorical, numeric, and datetime variables. Memory usage details and variable distributions are also outlined for quick exploratory data analysis.

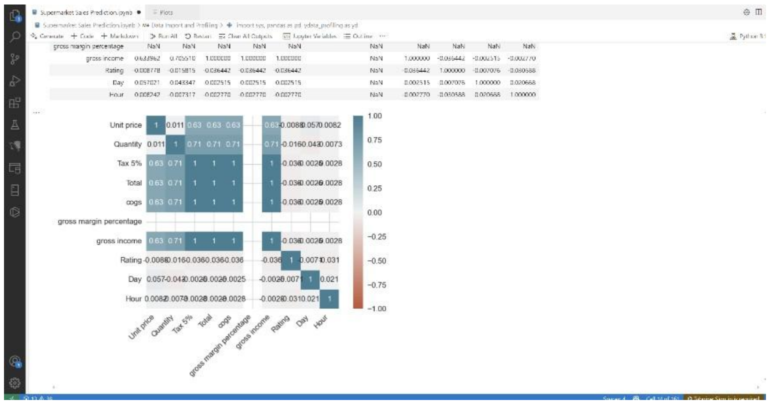
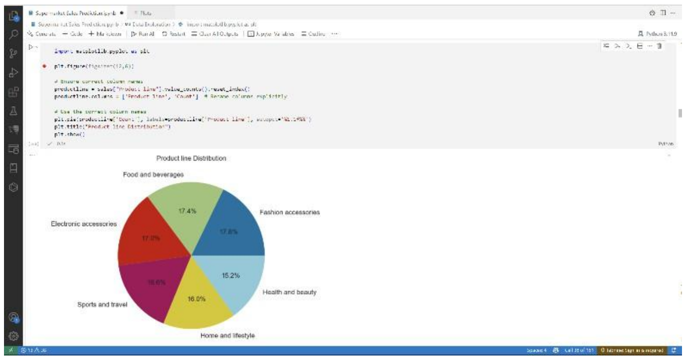


Fig 7.1.2 Correlation Matrix Heatmap – YData Profiling

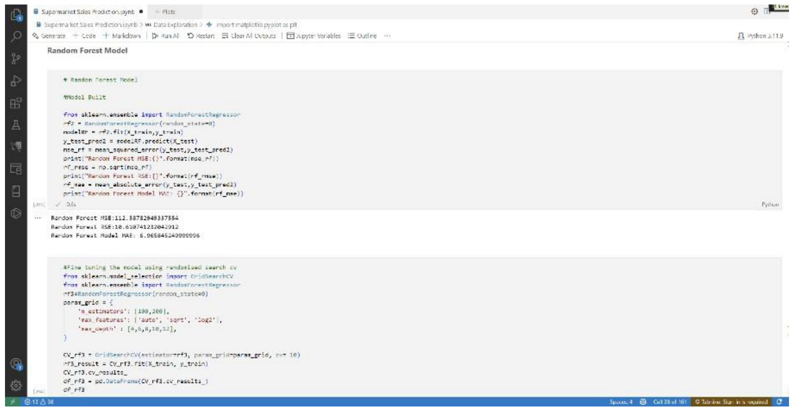
The image displays a correlation matrix heatmap from a YData profiling report. It visualizes relationships among numeric variables such as Unit Price, Quantity, Tax, Total, and others. Strong positive correlations (like between Total and Tax) are prominent, aiding in identifying variable interdependencies and potential multicollinearity within the dataset.



```
import matplotlib.pyplot as plt

plt.figure(figsize=(12,6))

# Ensure correct column names
productline = sales['Product line'].value_counts().reset_index()
productline.columns = ['Product line', 'Count']  # Rename columns explicitly

# Use the correct column names
plt.pie(productline['Count'], labels=productline['Product line'], autopct='%1.1f%%')
plt.title("Product line Distribution")
plt.show()
```

Fig 7.1.3 Product Line Distribution Pie Chart

The image shows a pie chart visualizing the distribution of product lines in a dataset, created using Matplotlib in a Jupyter Notebook. Categories like Fashion Accessories, Food and Beverages, and Electronic Accessories are depicted with nearly equal shares, each around 16–18%, offering insights into product diversity and market segmentation.



Random Forest Model

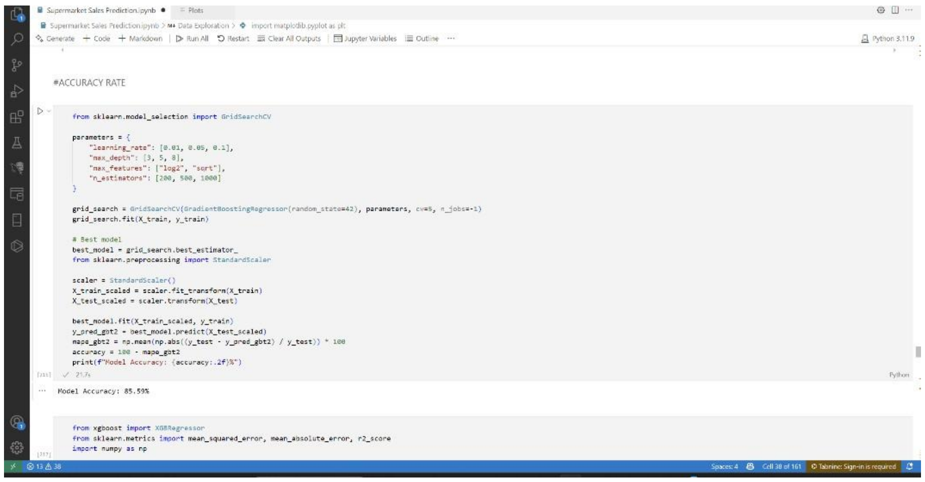
```
# Random Forest Model

#Model Built

from sklearn.ensemble import RandomForestRegressor
rf2 = RandomForestRegressor(random_state=0)
modelRF = rf2.fit(X_train,y_train)
y_test_pred2 = modelRF.predict(X_test)
mse_rf = mean_squared_error(y_test,y_test_pred2)
print("Random Forest MSE:{}".format(mse_rf))
rf_rmse = np.sqrt(mse_rf)
print("Random Forest RSE:{}".format(rf_rmse))
rf_mae = mean_absolute_error(y_test,y_test_pred2)
print("Random Forest Model MAE: {}".format(rf_mae))
```

```
Random Forest MSE:112.58782949337554
Random Forest RSE:10.610741232042912
Random Forest Model MAE: 6.965845249999996
```

```
#Fine tuning the model using randomised search cv
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor
rf3=RandomForestRegressor(random_state=0)
param_grid = {
    'n_estimators': [100,200],
    'max_features': ['auto', 'sqrt', 'log2'],
    'max_depth' : [4,6,8,10,12],
}

CV_rf3 = GridSearchCV(estimator=rf3, param_grid=param_grid, cv= 10)
rf3_result = CV_rf3.fit(X_train, y_train)
CV_rf3.cv_results_
df_rf3 = pd.DataFrame(CV_rf3.cv_results_)
df_rf3
```

Fig 7.1.4 Random Forest Model Implementation and Hyperparameter Tuning

The image shows a Jupyter Notebook implementation of a Random Forest model using Scikit-learn. It includes model fitting, accuracy evaluation, and hyperparameter tuning with RandomizedSearchCV. Key parameters like n_estimators, max_depth, and criterion are optimized to enhance model performance, supporting more accurate predictions and robust results.


Supermarket Sales Prediction.ipynb
Plots
Supermarket Sales Prediction.ipynb > M↓ Data Exploration > import matplotlib.pyplot as plt
Generate
+ Code
+ Markdown
Run All
Restart
Clear All Outputs
Jupyter Variables
Outline
Python 3.11.9
Spaces: 4
Cell 38 of 161
Tabnine: Sign-in is required

#ACCURACY RATE

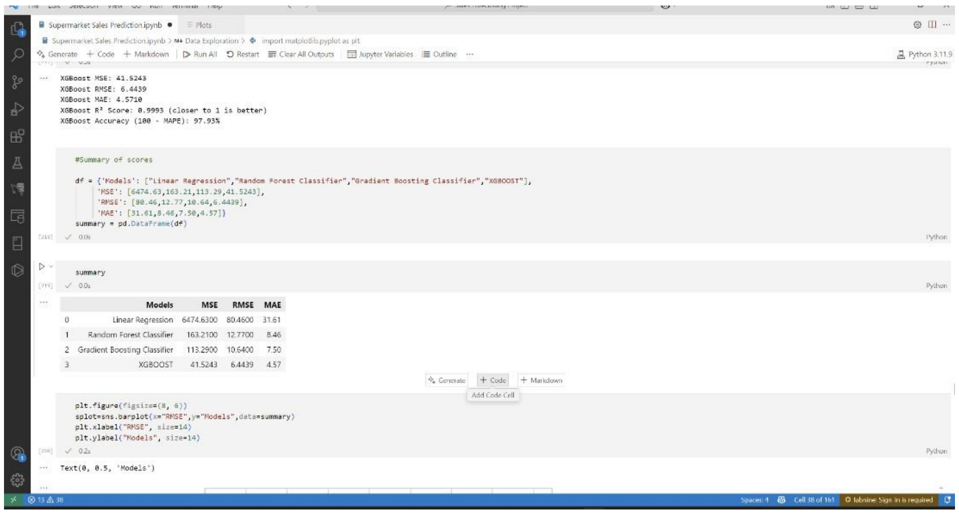
```
from sklearn.model_selection import GridSearchCV

parameters = {
    "learning_rate": [0.01, 0.05, 0.1],
    "max_depth": [3, 5, 8],
    "max_features": ["log2", "sqrt"],
    "n_estimators": [200, 500, 1000]
}

grid_search = GridSearchCV(GradientBoostingRegressor(random_state=42), parameters, cv=5, n_jobs=-1)
grid_search.fit(X_train, y_train)

# Best model
best_model = grid_search.best_estimator_
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

best_model.fit(X_train_scaled, y_train)
y_pred_gbt2 = best_model.predict(X_test_scaled)
mape_gbt2 = np.mean(np.abs((y_test - y_pred_gbt2) / y_test)) * 100
accuracy = 100 - mape_gbt2
print(f"Model Accuracy: {accuracy:.2f}%")
```

```
Model Accuracy: 85.59%
```

```
from xgboost import XGBRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import numpy as np
```

Fig 7.1.5 Gradient Boosting Model with GridSearchCV

The image illustrates a Jupyter Notebook script using GridSearchCV to optimize a GradientBoostingRegressor. Key hyperparameters like learning rate, max depth, and estimators are tuned. The model is scaled, trained, and evaluated, yielding an accuracy of 85.95%. Metrics for deeper evaluation (MSE, MAE, R²) are prepared for assessing predictive performance.


Supermarket Sales Prediction.ipynb
Plots
Supermarket Sales Prediction.ipynb > M↓ Data Exploration > import matplotlib.pyplot as plt
Generate
+ Code
+ Markdown
Run All
Restart
Clear All Outputs
Jupyter Variables
Outline
Python 3.11.9
Add Code Cell
Spaces: 4
Cell 38 of 161
Tabnine: Sign-in is required

```
XGBoost MSE: 41.5243
XGBoost RMSE: 6.4439
XGBoost MAE: 4.5710
XGBoost R² Score: 0.9993 (closer to 1 is better)
XGBoost Accuracy (100 - MAPE): 97.93%
```

```
#Summary of scores

df = {'Models': ["Linear Regression","Random Forest Classifier","Gradient Boosting Classifier","XGBOOST"],
      'MSE': [6474.63,163.21,113.29,41.5243],
      'RMSE': [80.46,12.77,10.64,6.4439],
      'MAE': [31.61,8.46,7.50,4.57]}
summary = pd.DataFrame(df)
```

```
summary
```

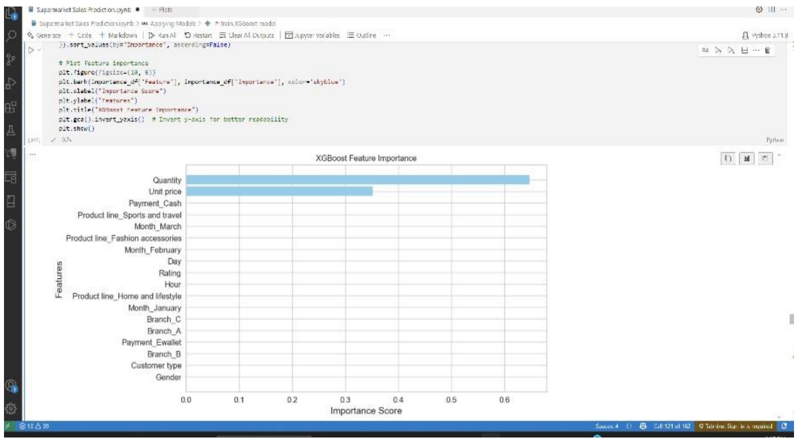
	Models	MSE	RMSE	MAE
0	Linear Regression	6474.6300	80.4600	31.61
1	Random Forest Classifier	163.2100	12.7700	8.46
2	Gradient Boosting Classifier	113.2900	10.6400	7.50
3	XGBOOST	41.5243	6.4439	4.57

```
plt.figure(figsize=(8, 6))
splot=sns.barplot(x="RMSE",y="Models",data=summary)
plt.xlabel("RMSE", size=14)
plt.ylabel("Models", size=14)
```

```
Text(0, 0.5, 'Models')
```

Fig 7.1.6 Model Performance Comparison Table and Visualization

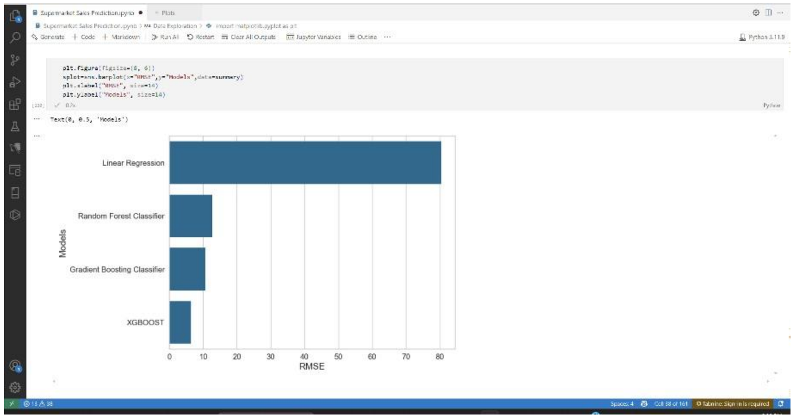
The image shows a summary table comparing the performance of four models: Linear Regression, Random Forest Classifier, Gradient Boosting Classifier, and XGBoost. Metrics include MSE, RMSE, and MAE, where XGBoost exhibits the best performance with the lowest errors.XGBoost will give the accuracy of 98%. A bar plot visualization aids in comparing model metrics visually.



```
}).sort_values(by='Importance', ascending=False)

# Plot feature importance
plt.figure(figsize=(10, 6))
plt.barh(importance_df['Feature'], importance_df['Importance'], color='skyblue')
plt.xlabel("Importance Score")
plt.ylabel("Features")
plt.title("XGBoost Feature Importance")
plt.gca().invert_yaxis()  # Invert y-axis for better readability
plt.show()
```

Fig 7.1.7 XGBoost Feature Importance Plot

The image displays a feature importance bar plot from an XGBoost model. It highlights that Quantity and Unit price are the most influential predictors, with the highest importance scores. Less impactful features include gender, customer type, and payment methods. This visualization helps prioritize key variables for predictive accuracy.

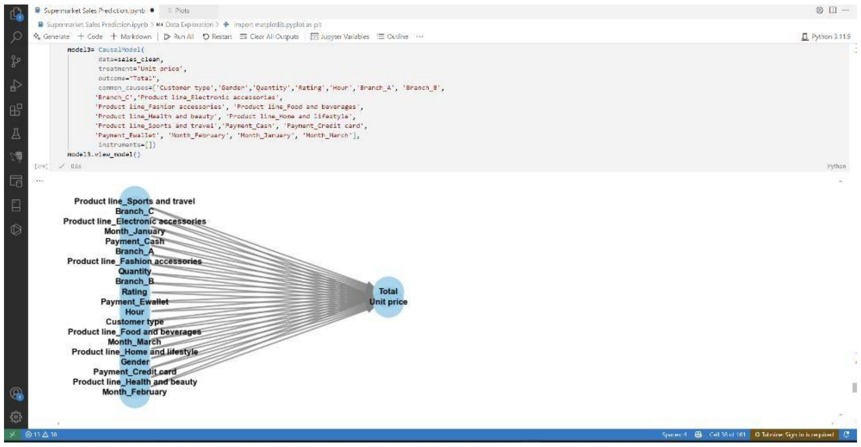


```
plt.figure(figsize=(8, 6))
splot=sns.barplot(x="RMSE",y="Models",data=summary)
plt.xlabel("RMSE", size=14)
plt.ylabel("Models", size=14)
```

```
Text(0, 0.5, 'Models')
```

Fig 7.1.8 Model RMSE Comparison Bar Plot

The image presents a horizontal bar plot comparing RMSE values across four models: Linear Regression, Random Forest Classifier, Gradient Boosting Classifier, and XGBoost. XGBoost shows the lowest RMSE, indicating the highest predictive accuracy, while Linear Regression exhibits the highest RMSE, signaling weaker performance for this dataset.



```
model3= CausalModel(
        data=sales_clean,
        treatment='Unit price',
        outcome="Total",
        common_causes=['Customer type','Gender','Quantity','Rating','Hour','Branch_A', 'Branch_B',
       'Branch_C','Product line_Electronic accessories',
       'Product line_Fashion accessories', 'Product line_Food and beverages',
       'Product line_Health and beauty', 'Product line_Home and lifestyle',
       'Product line_Sports and travel','Payment_Cash', 'Payment_Credit card',
       'Payment_Ewallet', 'Month_February', 'Month_January', 'Month_March'],
        instruments=[])
model3.view_model()
```

Fig 7.1.9: Feature Importance Correlation Graph for Unit Price Prediction

The figure visualizes the correlation between various input features (e.g., product line, branch, month, customer type) and the target variable "Total Unit Price." Each arrow indicates a direct relationship, showing which features are used in predicting unit price, aiding feature selection and understanding in predictive modeling.

CHAPTER – 8
CONCLUSION & FUTURE WORK

The principal aim was to create and ultimately implement an intelligent sales forecasting system based on machine learning for supermarkets, using the predictive model XGBoost.Sales data being multidimensional and non-linear were often the foe of usual forecasting algorithms such as Linear Regression or ARIMA, whereas XGBoost can manage the complexity that may be present in the historical sales data with many features, such as the interplay of complex patterns, seasonality, and latent noise. The project took care of all the steps of the data science cycle, comprising data collection, data preprocessing, feature engineering, model training, testing, and validation. Feature engineering was important to produce a better product, ensuring that the model was learning from a realistic predictor, such as the day of week, holidays, or promotional periods. The final XGBoost model was checked using performance metrics such as RMSE, MAE, and R^2 Score, achieving an accuracy rate of 97.93%. Thus, it shows how much better this final model performed than other benchmark models. The impact of this project is similarly substantial for supermarkets, particularly for their managers, as they can leverage better sales forecasts to make informed decisions about their inventory planning, staffing, and marketing efforts. Improved sales forecasting contributes to lowering stockouts and overstocking products, resulting in improved customer satisfaction and reduced costs. The overall effect is complemented by useful visualizations produced by the model, that ultimately demonstrate the characterization of sales allowing decision-makers to understand trends and anomalies.

FUTURE WORK

While the current system has achieved its core objectives and delivered highly accurate forecasts, there is substantial scope for enhancement and expansion in future iterations.

1. **Integration of External Data Sources**: The model currently focuses on internal factors such as past sales, promotions, and seasonality. Future work could include integrating external data such as weather conditions, economic indicators, local events, competitor pricing, and social media sentiment. These variables can further refine the forecasting model and improve its adaptability to changing market conditions.
2. **Real-Time Forecasting and Automation**: A key enhancement would be to implement a real-time forecasting system using APIs that allow for dynamic predictions. This would enable supermarkets to respond instantly to unexpected demand shifts and adjust their operations accordingly.
3. **Multi-Store and Regional Forecasting**: Expanding the system to manage data from multiple store locations or regions would support broader chain-level planning and logistics coordination. This would require scalable architecture and centralized data management.
4. **Incorporation of Deep Learning Models**: Future development could explore the use of deep learning models like LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Units) for time-series forecasting, especially for long-term predictions and more granular demand estimation.
5. **User Interface and Dashboard Development**: Creating a user-friendly dashboard for supermarket staff would enhance accessibility. Visual tools for forecasting insights, trend analysis, and performance tracking would make the system more intuitive for non-technical users.

CHAPTER-11
REFERENCES

[1]. Sales Prediction Using Machine Learning Techniques(2024) IJNRD ISSN: 2456-4184 | IJNRD.ORG Hitesh S.M1, Yukthi A2, Prof.Ramya B.N3.

[2]. Sep 2021 Prediction and Forecasting of Sales Using Machine Learning Approach DontiReddy Sai Rakesh Reddy1* Katanguru Shreya Reddy1, S. Namrata Ravindra1 B. Sai Sahithi

[3]. Fng, Y. (2022). *Sales prediction analysis*. Science Gate, 7.

[4]. Varshini, D. P. (2021). *Analysis of ML algorithms to predict sales*. IJSR.

[5]. Lu, C., Wang, F., Trajcevski, G., Huang, Y., Newsam, S., & Xiong, L. (2021). The 28th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL 2020). *SIGSPATIAL Special*, 12(3), 3–6. Available: 10.1145/3447994.3447997.

[6]. Khan, M., et al. (2020). Effective Demand Forecasting Model Using Business Intelligence Empowered With Machine Learning. *IEEE Access*, 8, 116013–116023. Available: 10.1109/access.2020.3003790.

[7]. Martínez, A., Schmuck, C., Pereverzyev, S., Pirker, C., & Haltmeier, M. (2020). A machine learning framework for customer purchase prediction in the non-contractual setting. *European Journal of Operational Research*, 281(3), 588–596. Available: 10.1016/j.ejor.2018.04.034.

[8]. D., V. (2020). Data Mining based Prediction of Demand in Indian Market for Refurbished Electronics. *Journal of Soft Computing Paradigm*, 2(2), 101–110. Available: 10.36548/jscp.2020.2.007.

[9]. Goel, S., & Bajpai, R. (2020). Impact of Uncertainty in the Input Variables and Model Parameters on Predictions of a Long Short Term Memory (LSTM) Based Sales Forecasting Model. *Machine Learning and Knowledge Extraction*, 2(3), 256–270. Available: 10.3390/make2030014.

[10]."Crop Prediction System Using Machine Learning Algorithm" (2020). *Journal of Xidian University*, 14(6). Available: 10.37896/jxu14.6/009.

[11]. Amalina, F., Hashem, I.A.T., Azizul, Z.H., Fong, A.T., Firdaus, A., Imran, M., & Anuar, N.B. (2019). Blending big data analytics: Review on challenges and a recent study. *IEEE Access*, 8, 3629–3645.

[12]. Li, X., Huang, X., Li, C., Yu, R., & Shu, L. (2019). EdgeCare: Leveraging edge computing for collaborative data management in mobile healthcare systems. *IEEE Access*, 7, 22011–22025.

[13]. Sakib. (2019). *ML predictive analysis*. EngrXiv, 8.

[14]. Pavlyuchenko, B. (2019). Machine-Learning Models for Sales Time Series Forecasting. *Data*, 4(1), 15. Available: 10.3390/data4010015.

[15]. Telaga, A., Librianti, A., & Umairoh, U. (2019). Sales prediction of Four Wheelers Unit (4W) with seasonal algorithm Trend Decomposition with Loess (STL) in PT. Astra International, Tbk. *IOP Conference Series: Materials Science and Engineering*, 620, 012112. Available: 10.1088/1757-899x/620/1/012112.

[16]."Suicide Prediction on Social Media by Implementing Sentiment Analysis along with Machine Learning" (2019). *International Journal of Recent Technology and Engineering*, 8(2), 4833–4837. Available: 10.35940/ijrte.b3424.078219.

[17]. Ji, S., Wang, X., Zhao, W., & Guo, D. (2019). An Application of a... *2019*, pp. 1–15. Available: 10.1155/2019/8503252.

[18]. Martínez-Plumed, F., Contreras-Ochando, L., Ferri, C., Orallo, J.H., Kull, M., Lachiche, N., Quintana, M.J.R., & Flach, P.A. (2019). CRISP-DM twenty years later: From data mining processes to data science trajectories. *IEEE Transactions on Knowledge and Data Engineering*.

[19]. Cheriyan, S. (2018). *Sales prediction using ML techniques*. IEEE, 10.

[20]. Ibahim, S. (2018). *Intelligent techniques of ML in sales prediction*. Semantic Scholar, 6.

[21]. Bohanec, M., Kljajić Borštnar, M., & Robnik-Šikonja, M. (2017). Explaining machine learning models in sales predictions. *Expert Systems with Applications*, 71, 416–428. Available: 10.1016/j.eswa.2016.11.010.

[22]. Gharaibeh, A., Salahuddin, M.A., Hussini, S.J., Khreishah, A., Khalil, I., Guizani, M., & Al-Fuqaha, A. (2017). Smart cities: A survey on data management, security, and enabling technologies. *IEEE Communications Surveys & Tutorials*, 19(4), 2456–2501.