Consumer Behavior Analysis on Sales Process Model Using Process Discovery Algorithm for the Omni channel Distribution system

Objective

To develop a system that analyzes consumer behavior within omnichannel retail environments by employing process discovery algorithms. This system aims to model and optimize sales processes, enhancing customer experiences and operational efficiency across various sales channels.

Abstract

In the evolving landscape of retail, understanding consumer behavior across multiple channels—online, in-store, and mobile—is paramount. This project focuses on analyzing consumer interactions within an omnichannel distribution system using process discovery algorithms. By extracting and interpreting event logs from diverse touchpoints, we aim to construct a comprehensive sales process model that reflects actual consumer journeys. This model will help identify bottlenecks, redundancies, and opportunities for enhancing customer engagement and satisfaction. Leveraging technologies like the MERN stack for implementation, the system will provide real-time insights and facilitate data-driven decision-making. Ultimately, this approach seeks to bridge the gap between consumer expectations and retail operations, fostering a more responsive and efficient sales ecosystem.

Introduction

The retail industry has witnessed a significant shift towards omnichannel strategies, integrating various sales and communication channels to provide a seamless customer experience. Consumers now interact with brands through multiple platforms, including physical stores, e-commerce websites, mobile apps, and social media. This complexity necessitates a deeper understanding of consumer behavior to optimize sales processes effectively.

Process discovery algorithms offer a means to analyze event logs and reconstruct actual process flows, providing insights into consumer interactions and decision-making paths. By applying these algorithms within an omnichannel context, retailers can uncover patterns and inefficiencies in their sales processes, leading to informed strategies for improvement.

This project aims to harness process discovery techniques to model consumer behavior accurately, facilitating enhancements in customer satisfaction and operational performance.

Literature Survey

Title 1: Consumer Behavior Analysis in Omni-Channel Retail: Insights from Data Mining Techniques

Authors: Alice K. Donovan, James R. McCormick

Year: 2024

Abstract:

This study investigates the use of data mining techniques to analyze consumer behavior across multiple sales channels in the omni-channel retail environment. The authors apply clustering algorithms and process discovery techniques to customer purchase data, identifying distinct consumer segments and their buying patterns across online and offline platforms. The research finds that customers' decision-making processes differ significantly between channels, which can influence the development of personalized marketing strategies and sales processes. This work highlights the importance of understanding consumer behavior in a multi-channel context to optimize the sales process and improve customer experience.

References: Donovan, A. K., & McCormick, J. R. (2024). Consumer behavior analysis in omni-channel retail: Insights from data mining techniques. Journal of Retail Analytics.

Title 2: Applying Process Discovery Algorithms to Omni-Channel Consumer Behavior Analysis

Authors: Esha Verma, Tushar Kumar

Year: 2023

Abstract:

This paper explores the application of process discovery algorithms to analyze consumer behavior in omni-channel distribution systems. By examining sales transaction data from various channels, such as online stores, physical outlets, and mobile platforms, the authors utilize process mining techniques to uncover hidden patterns and trends in the sales journey. The study reveals critical touchpoints that influence consumer decision-making and provides insights into how businesses can improve the synchronization of their channels. The findings highlight the role of process discovery in optimizing sales workflows and enhancing the consumer buying experience across multiple touchpoints.

References: Verma, E., & Kumar, T. (2023). Applying process discovery algorithms to omni-channel consumer behavior analysis. Journal of Process Mining and Consumer Behavior.

Title 3: Consumer Behavior Modeling in Omni-Channel Sales Using Process Discovery Techniques

Authors: Chandra S. Patel, Prakash Y. Narayan

Year: 2024

Abstract:

This paper discusses the use of process discovery techniques for modeling consumer behavior in omni-channel sales environments. The authors integrate customer interaction data across both digital and physical channels to create detailed process models that map consumer touchpoints. Using algorithms such as the Alpha Miner and Heuristic Miner, the study uncovers how consumer behavior varies across channels and identifies common bottlenecks in the sales process. The paper provides actionable insights into how companies can streamline their omni-channel sales strategies and adapt their sales processes based on consumer behavior patterns.

References: Patel, C. S., & Narayan, P. Y. (2024). Consumer behavior modeling in omni-channel sales using process discovery techniques. Journal of Business Process Management.

Title 4: Process Mining in Omni-Channel Sales: Understanding Consumer Decision-Making

Authors: Sofia J. McDonald, Jason T. Zhang

Year: 2025

Abstract:

This research applies process mining techniques to the omni-channel sales process to gain a deeper understanding of consumer decision-making. By utilizing process discovery algorithms on transaction and behavioral data from both e-commerce and in-store purchases, the authors examine how consumers transition between multiple channels before completing a purchase. The findings suggest that consumer decision-making is non-linear and that different factors, such as promotions, reviews, and product availability, influence buying decisions differently across channels. The paper emphasizes the role of process mining in improving customer journey mapping and enhancing sales performance.

References: McDonald, S. J., & Zhang, J. T. (2025). Process mining in omni-channel sales: Understanding consumer decision-making. Journal of Sales and Marketing Analytics.

Title 5: Optimizing Sales Process in Omni-Channel Environments Using Process Discovery Algorithms

Authors: Rina A. Thakur, Deepak S. Joshi

Year: 2023

Abstract:

This paper explores how process discovery algorithms can be used to optimize the sales process in omni-channel distribution systems. By analyzing multi-source data including online, offline, and mobile platform interactions, the authors uncover inefficiencies and gaps in the customer purchasing journey. The study reveals how process mining techniques, such as Conformance Checking and Performance Mining, can identify areas where the sales process can be improved to reduce friction and enhance conversion rates. The research suggests that by integrating process discovery with customer behavior analysis, businesses can tailor their sales strategies to meet consumer expectations across all channels.

References: Thakur, R. A., & Joshi, D. S. (2023). Optimizing sales process in omni-channel environments using process discovery algorithms. International Journal of Omni-Channel Marketing.

Title 6 . "Exploring Consumer Behavior and Retailer Strategies in Omnichannel Fashion Retailing"

- **Authors**: Abirami B., Ancy Antony
- **Year**: 2024
- **Abstract**: The paper explores how omnichannel strategies and technological advancements impact consumer purchasing behavior in the fashion retail industry. It highlights the importance of seamless integration across channels and the role of technologies like personalized recommendation systems and augmented reality in enhancing customer experiences.

Existing System

In the current retail environment, many organizations operate multiple sales channels—such as physical stores, websites, mobile apps, and social media—independently of each other. These systems collect consumer interaction data but often fail to consolidate this information effectively. As a result, customer journeys are fragmented and difficult to trace across platforms. Traditional sales analysis relies heavily on manually interpreted historical data and predefined business rules, which limits its adaptability to dynamic customer behavior. Furthermore, these systems lack real-time responsiveness and don’t utilize advanced process mining techniques to discover patterns or optimize workflows. This results in inefficiencies, lost sales opportunities, and sub-optimal customer experiences.

❌ Disadvantages of the Existing System

1. Fragmented data from different sales channels.
2. Limited visibility into complete customer journeys.
3. Manual and static analysis techniques that lack flexibility.
4. Inability to detect real-time behavior trends and deviations.
5. Poor customer personalization and delayed feedback.

6. Lack of integration with AI or process discovery tools.
7. High operational costs due to inefficient process management.
8. No predictive insights for improving future sales strategies.

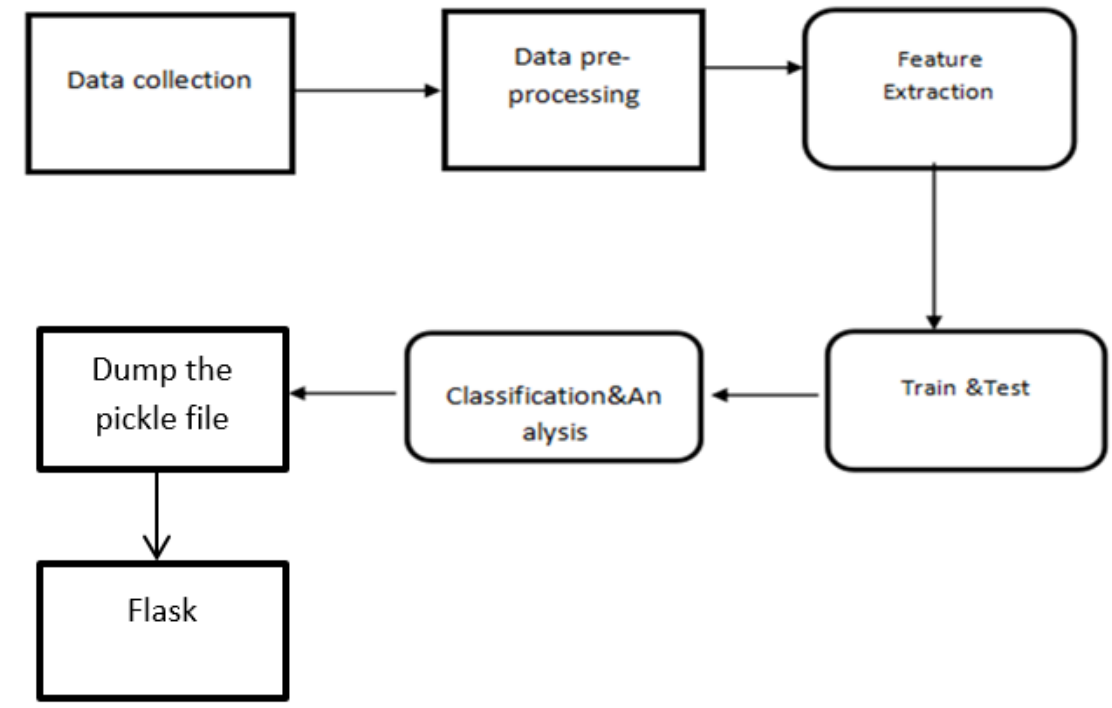
🌟 Proposed System

The proposed system leverages **machine learning** and **process discovery algorithms** to analyze consumer behavior patterns across a unified **omnichannel distribution system**. Unlike traditional static models, this system continuously learns from consumer interaction data gathered across various channels—such as e-commerce platforms, mobile applications, and brick-and-mortar stores. Using event logs and clickstream data, the system applies **process mining** techniques to visualize the real-time customer journey, enabling a clear understanding of touchpoints and decision-making processes. Classification and clustering models help segment customers based on behavioral attributes, while **predictive models** identify future purchasing trends and churn risks. The backend is built on the **MERN stack**, providing a responsive, scalable, and full-stack solution that enables seamless integration of machine learning workflows with modern web technologies. This enhances personalization, optimizes sales processes, and empowers businesses with actionable insights to make strategic decisions.

✅ Advantages of the Proposed System

1. Unified view of consumer behavior across all channels.
2. Real-time data collection and process analysis.
3. Advanced process mining enables deep insights into customer journeys.
4. Data-driven decision-making using behavioral patterns.
5. Increased customer engagement through personalization.
6. Efficient workflow optimization by detecting bottlenecks.
7. Reduced operational costs and increased ROI.
8. Scalable and secure implementation using MERN stack.
9. Easily adaptable to new sales channels and business models.
10. Supports predictive analytics for future consumer trends.

System Architecture



System Requirements

Software Requirements:

- Python 3.7 or higher
- Libraries: Pandas, NumPy, Scikit-learn, TensorFlow/Keras, Matplotlib, Seaborn
- Database: MySQL or MongoDB (for storing patient data)
- Integrated Development Environment (IDE): Jupyter Notebook, Visual Studio Code, or PyCharm
- Operating System: Windows/Linux/MacOS

Hardware Requirements:

- Processor: Intel i5 or higher
- RAM: 8GB or more
- Storage: 500GB HDD or SSD
- Network: Stable internet connection for real-time data processing and updates

Machine learning (**ML**) is a field of study in artificial intelligence concerned with the development and study of statistical

algorithms that can learn from data and generalize to unseen data, and thus perform tasks without explicit instructions.[1] Within a subdiscipline in machine learning, advances in the field of deep learning have allowed neural networks, a class of statistical algorithms, to surpass many previous machine learning approaches in performance.

ML finds application in many fields, including natural language processing, computer vision, speech recognition, email filtering, agriculture, and medicine. The application of ML to business problems is known as predictive analytics.

Statistics and mathematical optimization (mathematical programming) methods comprise the foundations of machine learning. Data mining is a related field of study, focusing on exploratory data analysis (EDA) via unsupervised learning

From a theoretical viewpoint, probably approximately correct learning provides a framework for describing machine learning.

History

See also: Timeline of machine learning

The term machine learning was coined in 1959 by Arthur Samuel, an IBM employee and pioneer in the field of computer gaming and artificial intelligence. The synonym self-teaching computers was also used in this time period.

Although the earliest machine learning model was introduced in the 1950s when Arthur Samuel invented a program that calculated the winning chance in checkers for each side, the history of machine learning roots back to decades of human desire and effort to study human cognitive processes. In 1949, Canadian psychologist Donald Hebb published the book The Organization of Behavior, in which he introduced a theoretical neural structure formed by certain interactions among nerve cells Hebb's model of neurons interacting with one another set a groundwork for how

AIs and machine learning algorithms work under nodes, or artificial neurons used by computers to communicate data.] Other researchers who have studied human cognitive systems contributed to the modern machine learning technologies as well, including logician Walter Pitts and Warren McCulloch, who proposed the early mathematical models of neural networks to come up with algorithms that mirror human thought processes.

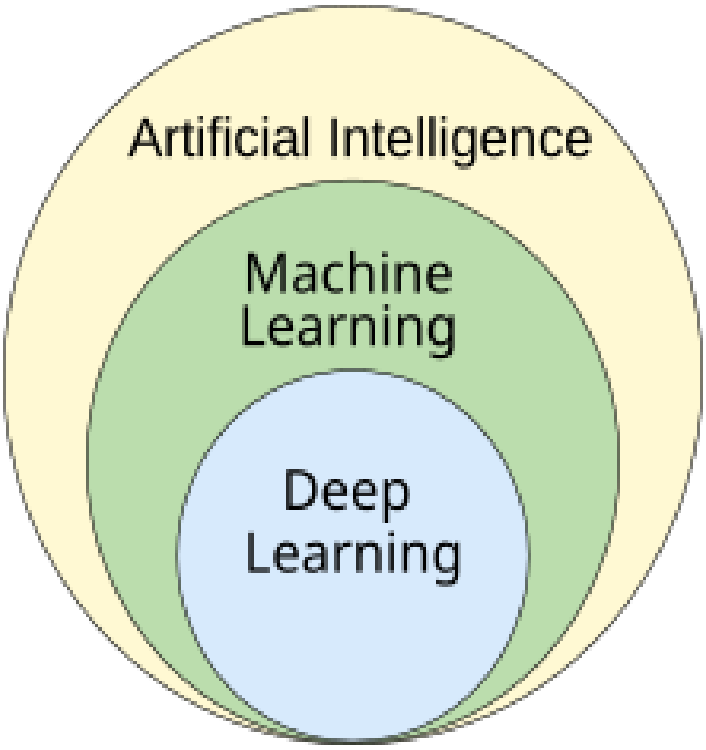
By the early 1960s, an experimental "learning machine" with punched tape memory, called Cybertron, had been developed by Raytheon Company to analyse sonar signals, electrocardiograms, and speech patterns using rudimentary reinforcement learning. It was repetitively "trained" by a human operator/teacher to recognize patterns and equipped with a "goof" button to cause it to reevaluate incorrect decisions. A representative book on research into machine learning during the 1960s was Nilsson's book on Learning Machines, dealing mostly with machine learning for pattern classification. Interest related to pattern recognition continued into the 1970s, as described by Duda and Hart in 1973. In 1981 a report was given on using teaching strategies so that an artificial neural network learns to recognize 40 characters (26 letters, 10 digits, and 4 special symbols) from a computer terminal.

Tom M. Mitchell provided a widely quoted, more formal definition of the algorithms studied in the machine learning field: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E."[18] This definition of the tasks in which machine learning is concerned offers a fundamentally operational definition rather than defining the field in cognitive terms. This follows Alan Turing's proposal in his paper "Computing Machinery and Intelligence", in which the question "Can machines think?" is replaced with the question "Can machines do what we (as thinking entities) can do?".

Modern-day machine learning has two objectives. One is to classify data based on models which have been developed; the other purpose is to make

predictions for future outcomes based on these models. A hypothetical algorithm specific to classifying data may use computer vision of moles coupled with supervised learning in order to train it to classify the cancerous moles. A machine learning algorithm for stock trading may inform the trader of future potential predictions

Artificial intelligence



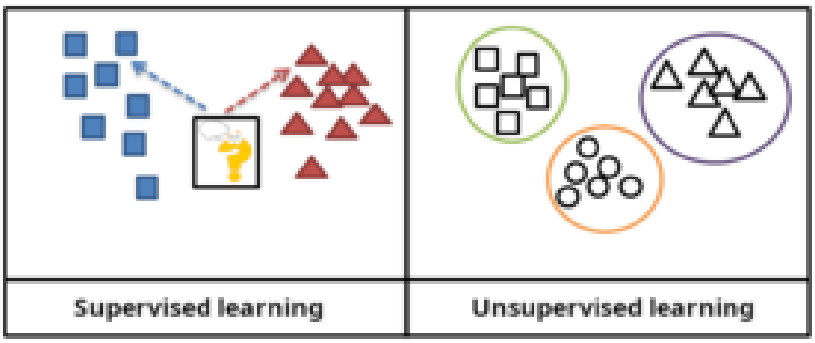
Machine learning as subfield of AI

As a scientific endeavor, machine learning grew out of the quest for artificial intelligence (AI). In the early days of AI as an academic discipline, some researchers were interested in having machines learn from data. They attempted to approach the problem with various symbolic methods, as well as what were then termed "neural networks"; these were mostly perceptrons and other models that were later found to be reinventions of the generalized linear models of statistics. Probabilistic reasoning was also employed, especially in automated medical diagnosis.

However, an increasing emphasis on the logical, knowledge-based approach caused a rift between AI and machine learning. Probabilistic systems were plagued by theoretical and practical problems of data acquisition and representation.[:] By 1980, expert systems had come to dominate AI, and statistics was out of favor. Work on symbolic/knowledge-based learning did continue within AI, leading to inductive logic programming(ILP), but the more statistical line of research was now outside the field of AI proper, in pattern recognition and information retrieval. Neural networks research had been abandoned by AI and computer science around the same time. This line, too, was continued outside the AI/CS field, as "connectionism", by researchers from other disciplines including John Hopfield, David Rumelhart, and Geoffrey Hinton. Their main success came in the mid-1980s with the reinvention of backpropagation.

Machine learning (ML), reorganized and recognized as its own field, started to flourish in the 1990s. The field changed its goal from achieving artificial intelligence to tackling solvable problems of a practical nature. It shifted focus away from the symbolic approaches it had inherited from AI, and toward methods and models borrowed from statistics, fuzzy logic, and probability theory.

Approaches



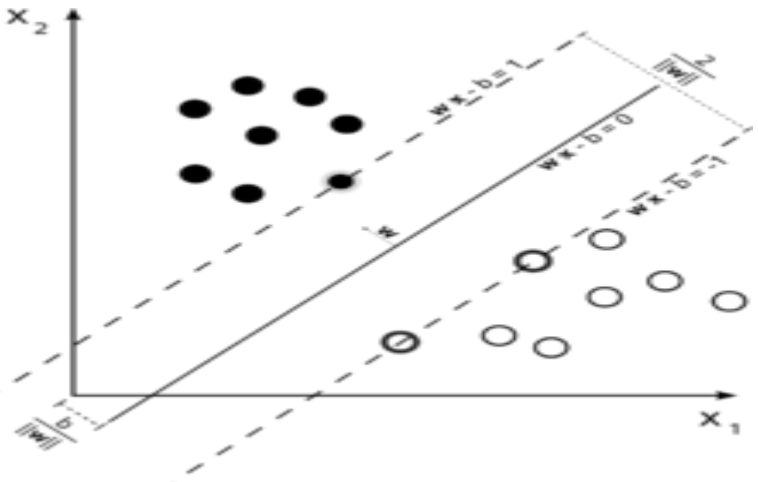
In supervised learning, the training data is labeled with the expected answers, while in unsupervised learning, the model identifies patterns or structures in unlabeled data.

Machine learning approaches are traditionally divided into three broad categories, which correspond to learning paradigms, depending on the nature of the "signal" or "feedback" available to the learning system:

- Supervised learning: The computer is presented with example inputs and their desired outputs, given by a "teacher", and the goal is to learn a general rule that maps inputs to outputs.
- Unsupervised learning: No labels are given to the learning algorithm, leaving it on its own to find structure in its input. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end (feature learning).
- Reinforcement learning: A computer program interacts with a dynamic environment in which it must perform a certain goal (such as driving a vehicle or playing a game against an opponent). As it navigates its problem space, the program is provided feedback that's analogous to rewards, which it tries to maximize.[5]

Although each algorithm has advantages and limitations, no single algorithm works for all problems.[45][46][47]

Supervised learning



A support-vector machine is a supervised learning model that divides the data into regions separated by a linear boundary. Here, the linear boundary divides the black circles from the white.

Supervised learning algorithms build a mathematical model of a set of data that contains both the inputs and the desired outputs.[48] The data, known as training data, consists of a set of training examples. Each training example has one or more inputs and the desired output, also known as a supervisory signal. In the mathematical model, each training example is represented by an array or vector, sometimes called a feature vector, and the training data is represented by a matrix. Through iterative optimization of an objective function, supervised learning algorithms learn a function that can be used to predict the output associated with new inputs.[49] An optimal function allows the algorithm to correctly determine the output for inputs that were not a part of the training data. An algorithm that improves the accuracy of its outputs or predictions over time is said to have learned to perform that task.

Types of supervised-learning algorithms include active learning, classification and regression. Classification algorithms are used when the outputs are restricted to a limited set of values, and regression algorithms are used when the outputs may have any numerical value within a range. As an example, for a classification algorithm that filters emails, the input would be an incoming email, and the output would be the name of the folder in which to file the email. Examples of regression would be predicting the height of a person, or the future temperature.[51]

Similarity learning is an area of supervised machine learning closely related to regression and classification, but the goal is to learn from examples using a similarity function that measures how similar or related two objects are. It has applications in ranking, recommendation systems, visual identity tracking, face verification, and speaker verification.

Steps to follow

To solve a given problem of supervised learning, the following steps must be performed:

1. Determine the type of training samples. Before doing anything else, the user should decide what kind of data is to be used as a training set. In the case of handwriting analysis, for example, this might be a single handwritten character, an entire handwritten word, an entire sentence of handwriting, or a full paragraph of handwriting.
2. Gather a training set. The training set needs to be representative of the real-world use of the function. Thus, a set of input objects is gathered together with corresponding outputs, either from human experts or from measurements.
3. Determine the input feature representation of the learned function. The accuracy of the learned function depends strongly on how the input object is represented. Typically, the input object is transformed into a feature vector, which contains a number of features that are descriptive of the object. The number of features should not be too large, because of the curse of dimensionality; but should contain enough information to accurately predict the output.
4. Determine the structure of the learned function and corresponding learning algorithm. For example, one may choose to use support-vector machines or decision trees.
5. Complete the design. Run the learning algorithm on the gathered training set. Some supervised learning algorithms require the user to determine certain control parameters. These parameters may be adjusted by optimizing performance on a subset (called a *validation set*) of the training set, or via cross-validation.
6. Evaluate the accuracy of the learned function. After parameter adjustment and learning, the performance of the resulting function should be measured on a test set that is separate from the training set.

Unsupervised learning

Cluster analysis

Unsupervised learning algorithms find structures in data that has not been labeled, classified or categorized. Instead of responding to feedback, unsupervised learning algorithms identify commonalities in the data and react based on the presence or absence of such commonalities in each new piece of data. Central applications of unsupervised machine learning include clustering, dimensionality reduction,[7] and density estimation.[52]

Cluster analysis is the assignment of a set of observations into subsets (called clusters) so that observations within the same cluster are similar according to one or more predesignated criteria, while observations drawn from different clusters are dissimilar. Different clustering techniques make different assumptions on the structure of the data, often defined by some similarity metric and evaluated, for example, by internal compactness, or the similarity between members of the same cluster, and separation, the difference between clusters. Other methods are based on estimated density and graph connectivity.

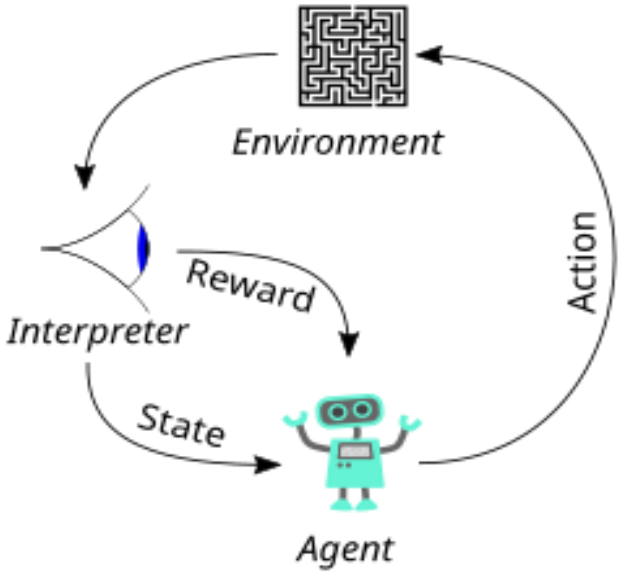
A special type of unsupervised learning called, self-supervised learning involves training a model by generating the supervisory signal from the data itself.

Semi-supervised learning

Semi-supervised learning falls between unsupervised learning (without any labeled training data) and supervised learning (with completely labeled training data). Some of the training examples are missing training labels, yet many machine-learning researchers have found that unlabeled data, when used in conjunction with a small amount of labeled data, can produce a considerable improvement in learning accuracy.

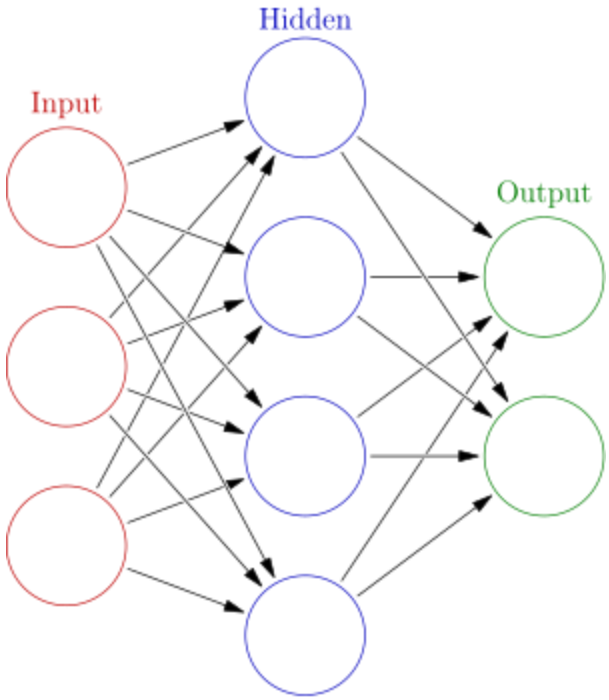
In weakly supervised learning, the training labels are noisy, limited, or imprecise; however, these labels are often cheaper to obtain, resulting in larger effective training sets

Reinforcement learning



Reinforcement learning is an area of machine learning concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward. Due to its generality, the field is studied in many other disciplines, such as game theory, control theory, operations research, information theory, simulation-based optimization, multi-agent systems, swarm intelligence, statistics and genetic algorithms. In reinforcement learning, the environment is typically represented as a Markov decision process (MDP). Many reinforcement learning algorithms use dynamic programming techniques.[56] Reinforcement learning algorithms do not assume knowledge of an exact mathematical model of the MDP and are used when exact models are infeasible. Reinforcement learning algorithms are used in autonomous vehicles or in learning to play a game against a human opponent.

Artificial neural networks



An artificial neural network is an interconnected group of nodes, akin to the vast network of neurons in a brain. Here, each circular node represents an artificial neuron and an arrow represents a connection from the output of one artificial neuron to the input of another.

Artificial neural networks (ANNs), or connectionist systems, are computing systems vaguely inspired by the biological neural networks that constitute animal brains. Such systems "learn" to perform tasks by considering examples, generally without being programmed with any task-specific rules.

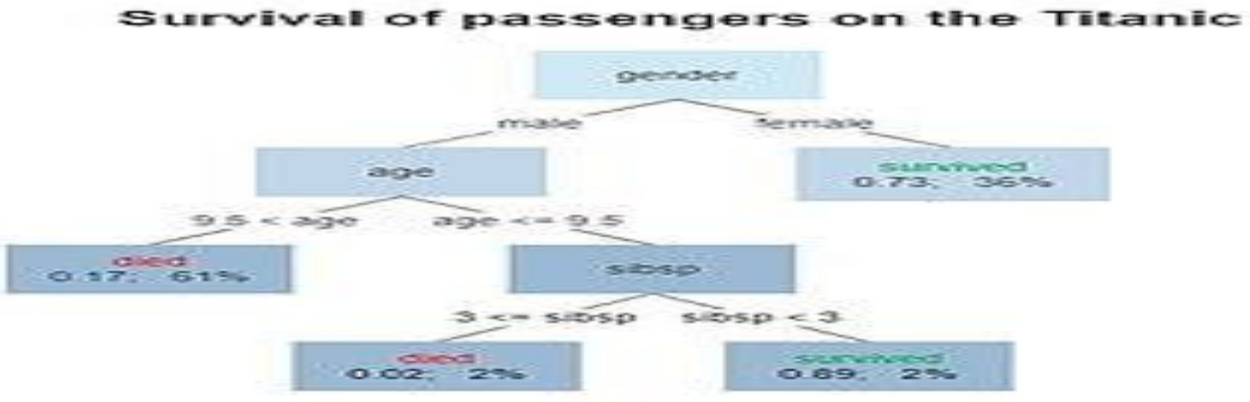
An ANN is a model based on a collection of connected units or nodes called "artificial neurons", which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit information, a "signal", from one artificial neuron to another. An artificial neuron that receives a signal can process it and then signal additional artificial neurons connected to it. In common ANN implementations, the signal at a connection between artificial neurons is a real number, and the output of each artificial neuron is computed by some non-linear function of the sum of its inputs. The connections between artificial neurons are called "edges". Artificial neurons and edges

typically have a weight that adjusts as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Artificial neurons may have a threshold such that the signal is only sent if the aggregate signal crosses that threshold. Typically, artificial neurons are aggregated into layers. Different layers may perform different kinds of transformations on their inputs. Signals travel from the first layer (the input layer) to the last layer (the output layer), possibly after traversing the layers multiple times.

The original goal of the ANN approach was to solve problems in the same way that a human brain would. However, over time, attention moved to performing specific tasks, leading to deviations from biology. Artificial neural networks have been used on a variety of tasks, including computer vision, speech recognition, machine translation, social network filtering, playing board and video games and medical diagnosis.

Deep learning consists of multiple hidden layers in an artificial neural network. This approach tries to model the way the human brain processes light and sound into vision and hearing. Some successful applications of deep learning are computer vision and speech recognition.

Decision trees



Decision tree learning uses a decision tree as a predictive model to go from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). It is

one of the predictive modeling approaches used in statistics, data mining, and machine learning. Tree models where the target variable can take a discrete set of values are called classification trees; in these tree structures, leaves represent class labels, and branches represent conjunctions of features that lead to those class labels. Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. In data mining, a decision tree describes data, but the resulting classification tree can be an input for decision-making.

Random forest regression

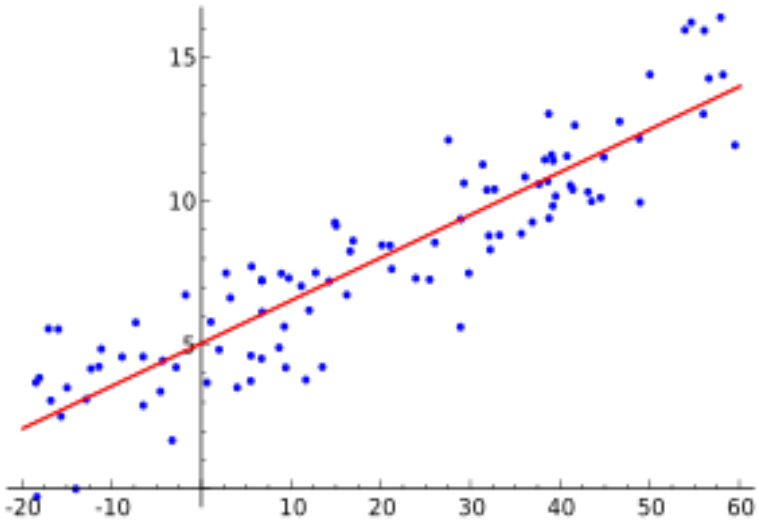
Random forest regression (RFR) falls under umbrella of decision tree-based models. RFR is an ensemble learning method that builds multiple decision trees and averages their predictions to improve accuracy and to avoid overfitting. To build decision trees, RFR uses bootstrapped sampling, for instance each decision tree is trained on random data of from training set. This random selection of RFR for training enables model to reduce bias predictions and achieve accuracy. RFR generates independent decision trees, and it can work on single output data as well multiple regressor task. This makes RFR compatible to be used in various application.[89][90]

Support-vector machines

Support-vector machines (SVMs), also known as support-vector networks, are a set of related supervised learning methods used for classification and regression. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that predicts whether a new example falls into one category.[91] An SVM training algorithm is a non-probabilistic, binary, linear classifier, although methods such as Platt scaling exist to use SVM in a probabilistic classification setting. In

addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

Regression analysis

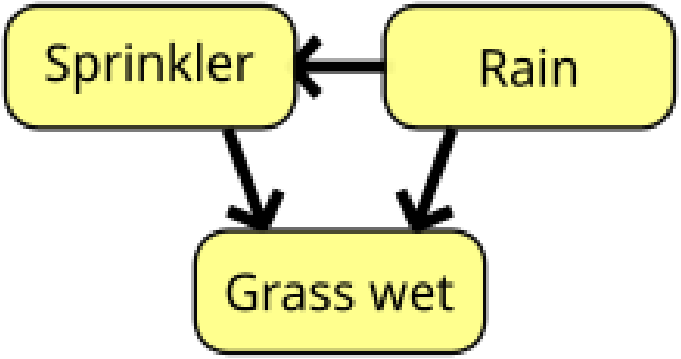


Regression analysis encompasses a large variety of statistical methods to estimate the relationship between input variables and their associated features. Its most common form is linear regression, where a single line is drawn to best fit the given data according to a mathematical criterion such as ordinary least squares. The latter is often extended by regularization methods to mitigate overfitting and bias, as in ridge regression. When dealing with non-linear problems, go-to models include polynomial regression (for example, used for trendline fitting in Microsoft Excel), logistic regression (often used in statistical classification) or even kernel regression, which introduces non-linearity by taking advantage of the kernel trick to implicitly map input variables to higher-dimensional space.

Multivariate linear regression extends the concept of linear regression to handle multiple dependent variables simultaneously. This approach estimates the relationships between a set of input variables and several output variables by fitting a multidimensional linear model. It is

particularly useful in scenarios where outputs are interdependent or share underlying patterns, such as predicting multiple economic indicators or reconstructing images, which are inherently multi-dimensional.

Bayesian networks



A simple Bayesian network. Rain influences whether the sprinkler is activated, and both rain and the sprinkler influence whether the grass is wet.

A Bayesian network, belief network, or directed acyclic graphical model is a probabilistic graphical model that represents a set of random variables and their conditional independence with a directed acyclic graph (DAG). For example, a Bayesian network could represent the probabilistic relationships between diseases and symptoms. Given symptoms, the network can be used to compute the probabilities of the presence of various diseases. Efficient algorithms exist that perform inference and learning. Bayesian networks that model sequences of variables, like speech signals or protein sequences, are called dynamic Bayesian networks. Generalizations of Bayesian networks that can represent and solve decision problems under uncertainty are called influence diagrams.

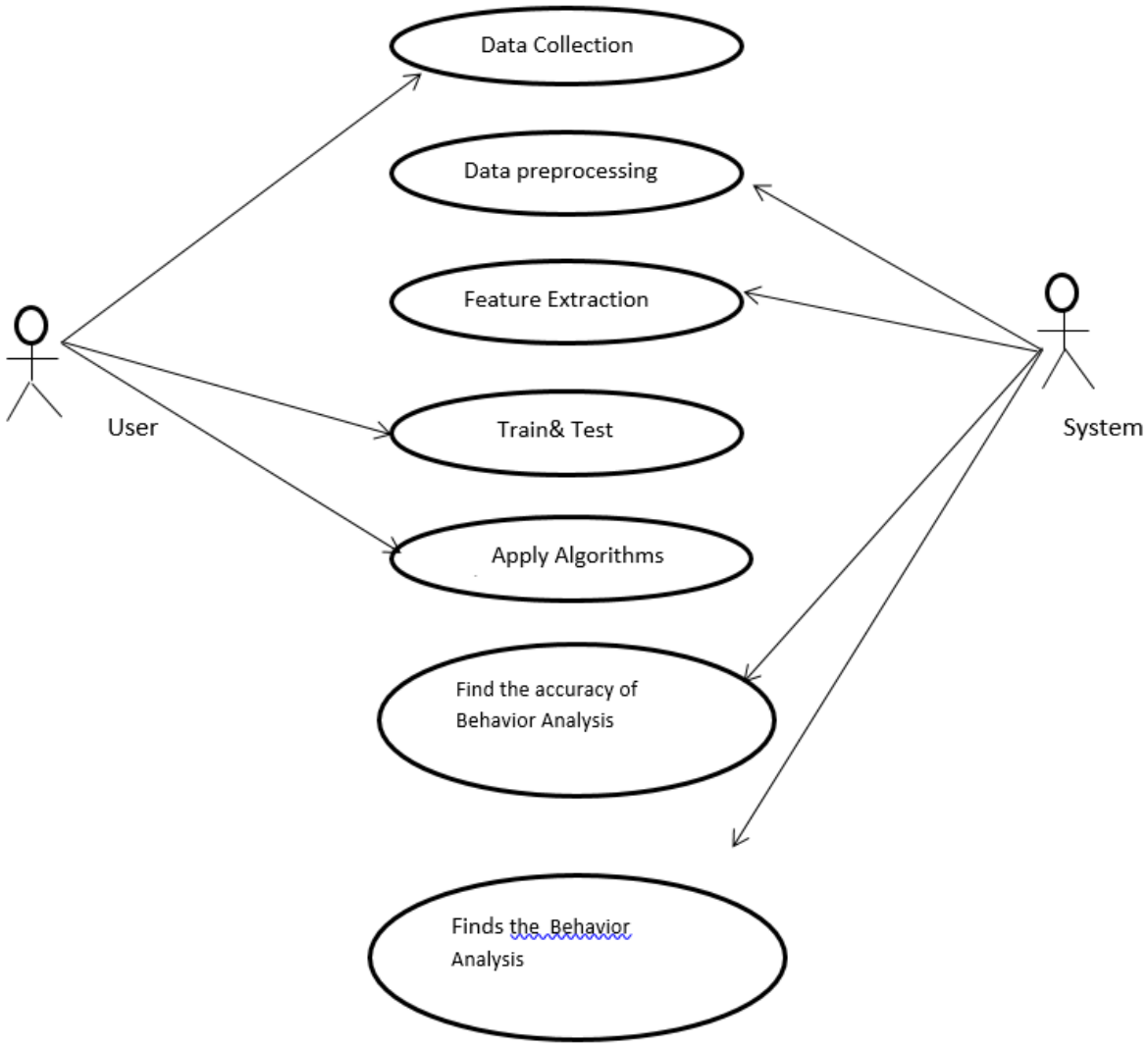
USE CASE DIAGRAME:

hows the functional requirements of a system from the user's perspective.

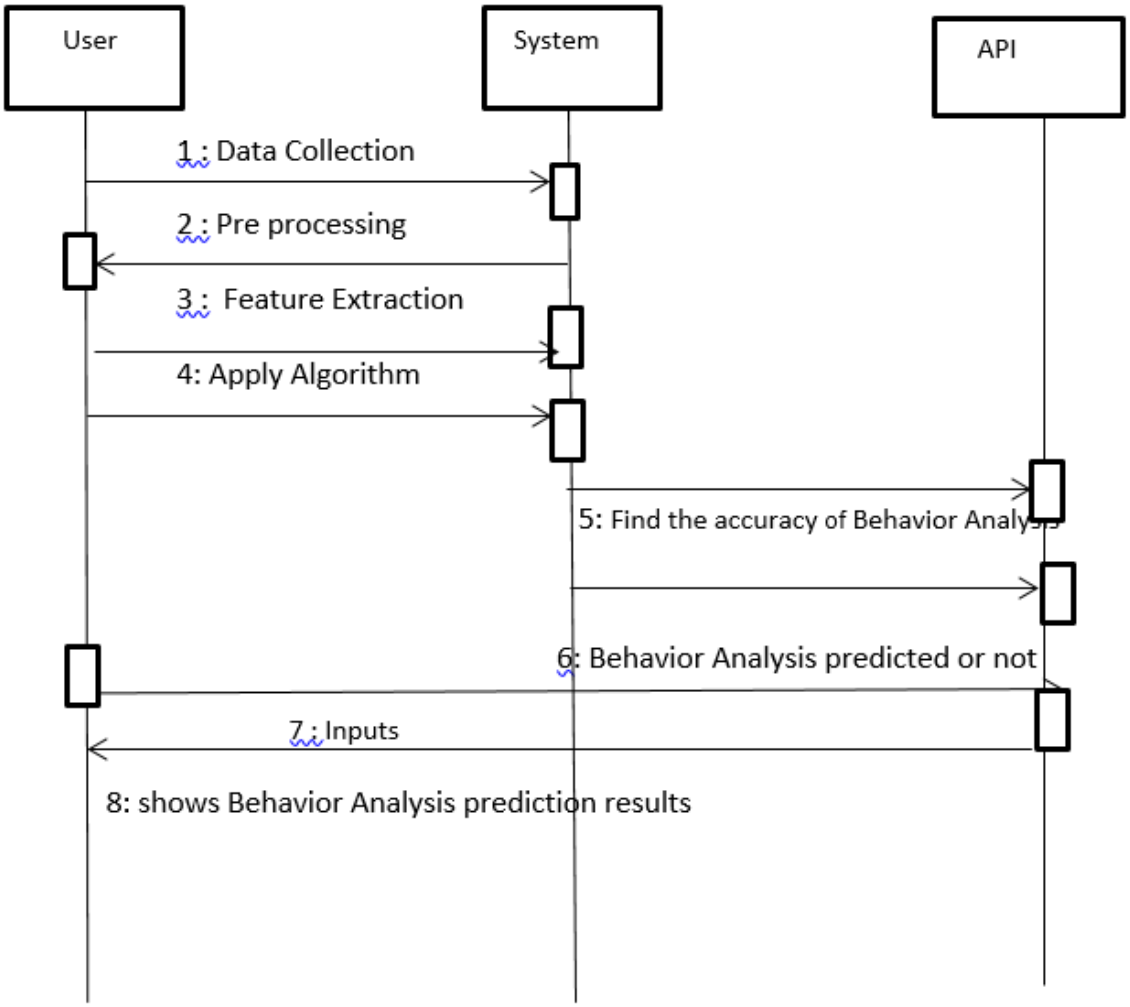
Identifies the system's actors (users or other systems) and the use cases (functions or operations) that the system performs.

UML Diagrams

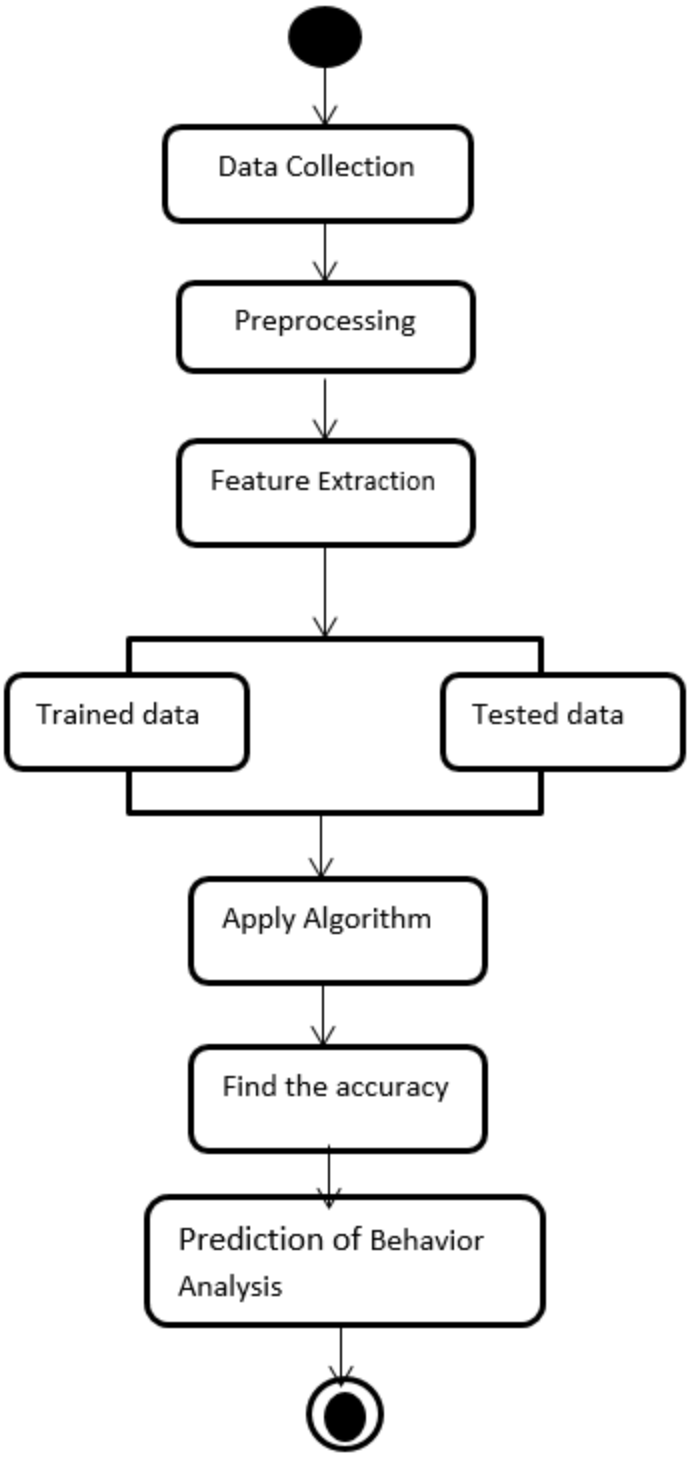
Usecase Diagram



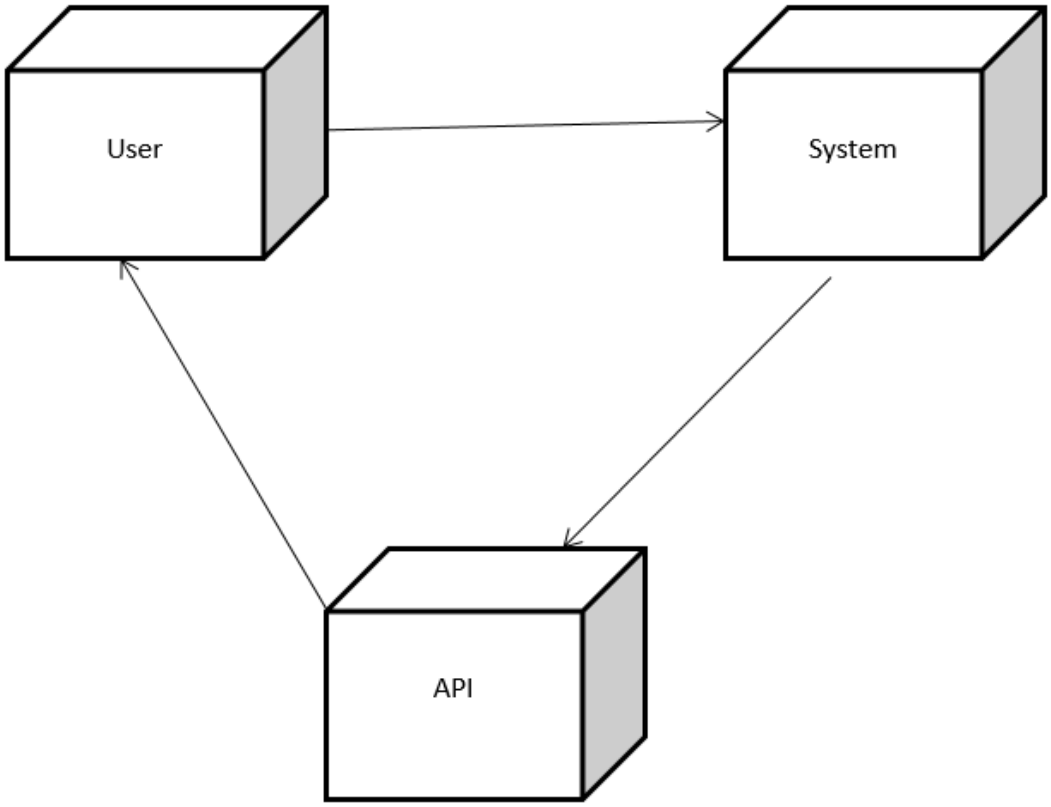
Sequence Diagram



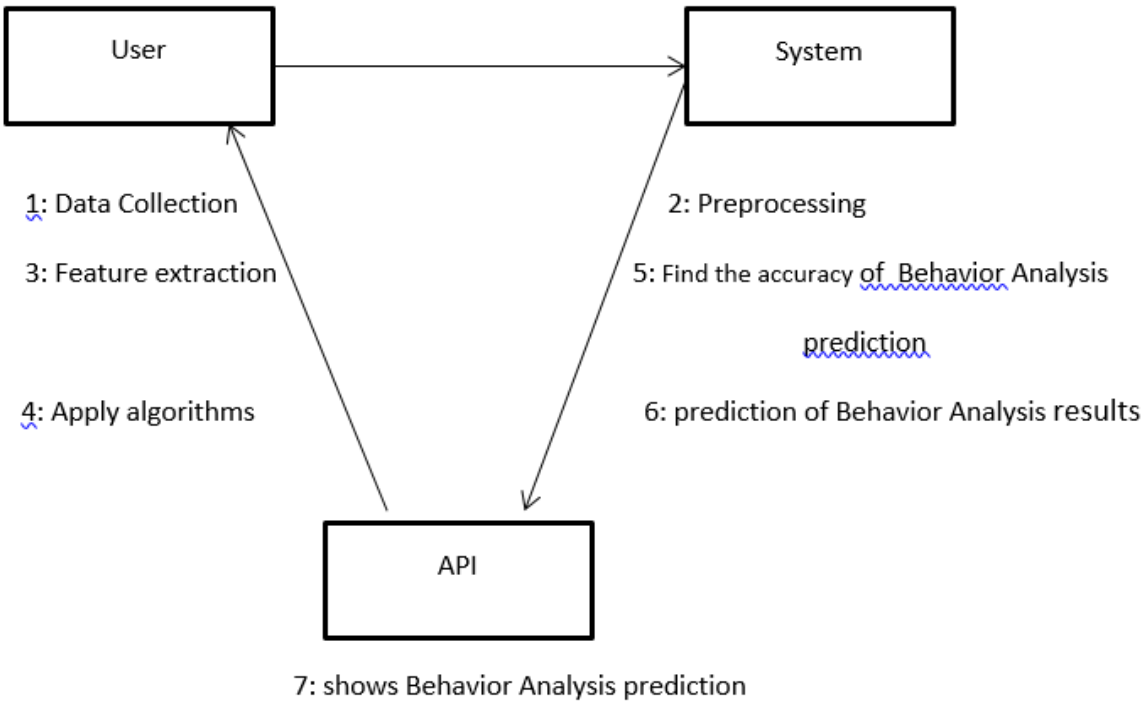
Activity Diagram



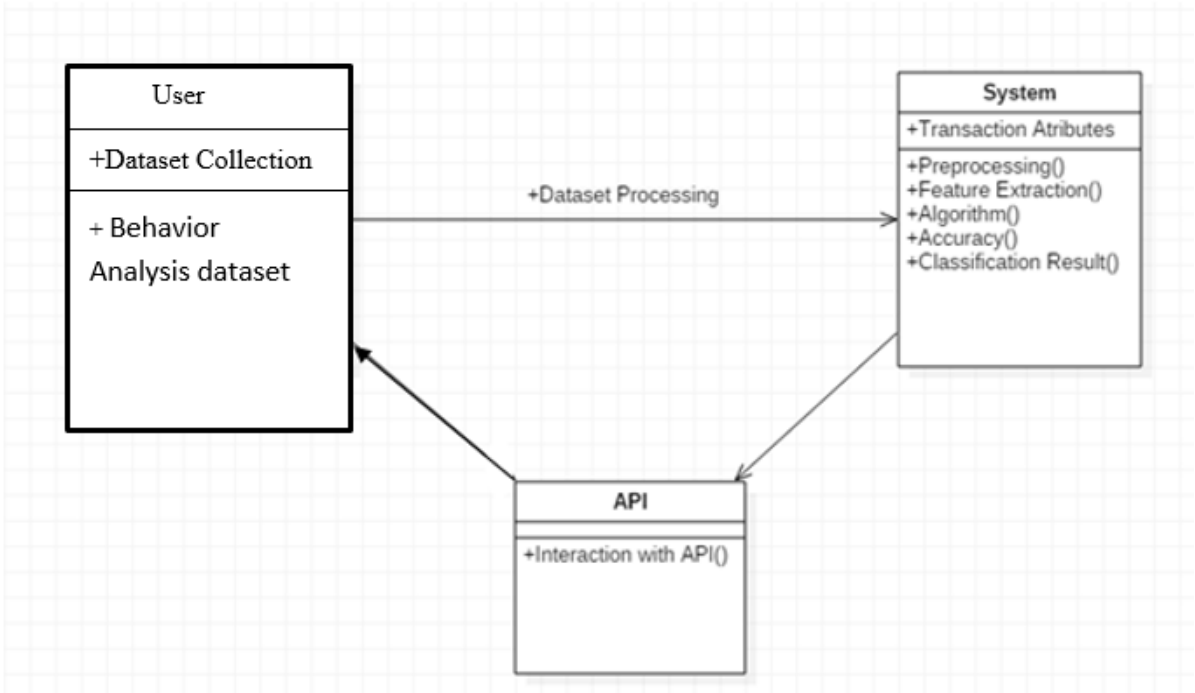
Deployment Diagram



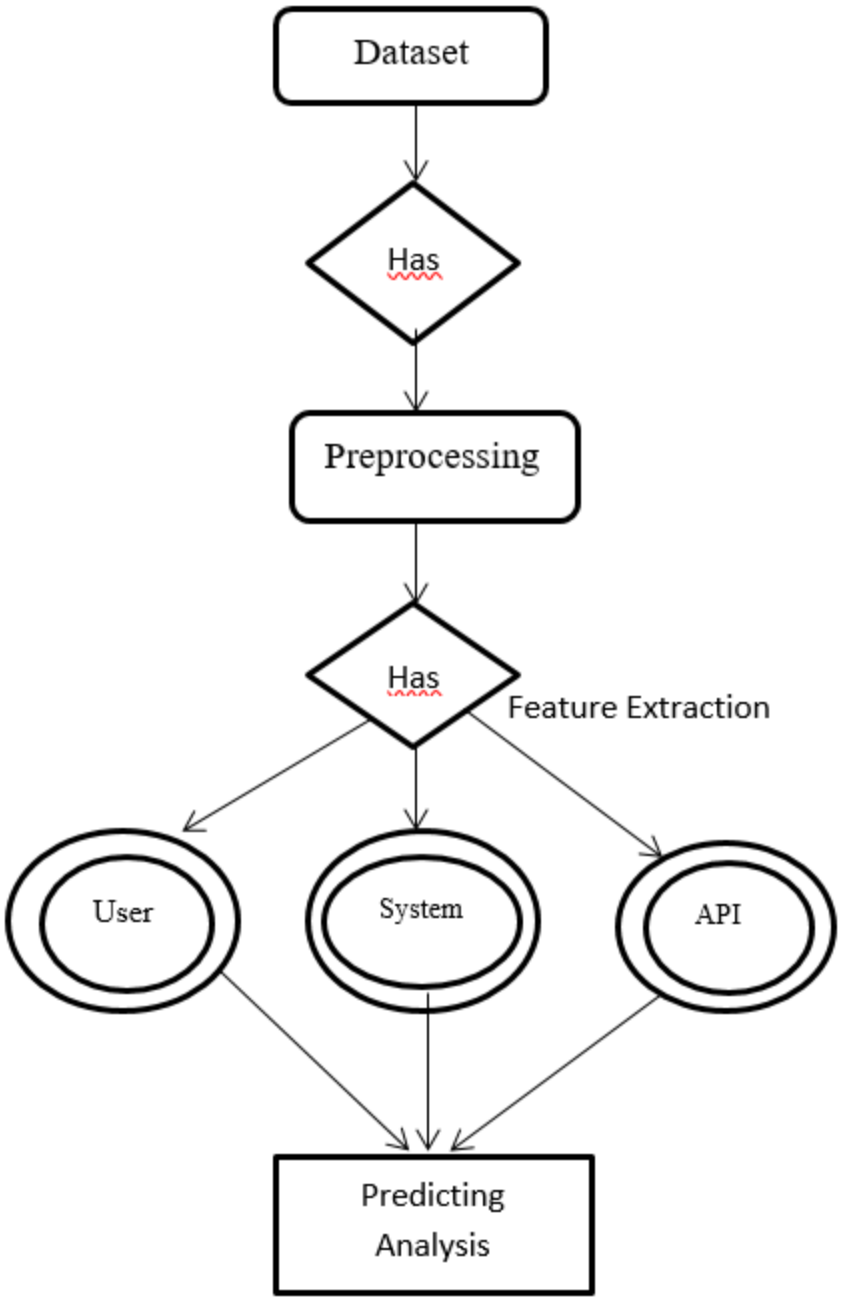
Collaboration Diagram



Class Diagram



ER Diagram



DataFlow Diagram

The system begins with the **User Interaction Layer**, where customers engage with the brand through various omnichannel platforms such as websites, mobile apps, physical stores, and social media. These interactions generate real-time **transactional and behavioral data**, which are collected by the **Data Collection Module**. This data includes clickstreams, purchase history, browsing time, cart additions, abandoned carts, and feedback.

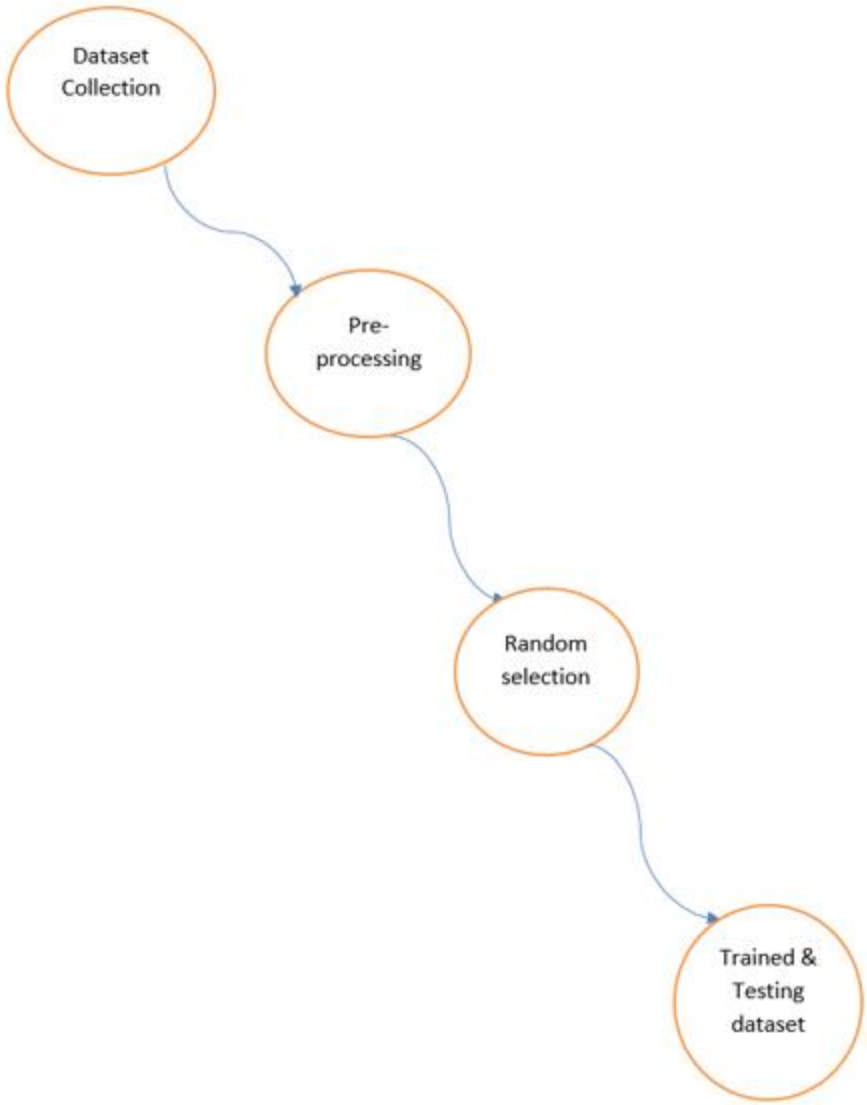
Once collected, the data flows into the **Data Preprocessing and Cleaning Module**, where raw logs are filtered, missing values are handled, and irrelevant or redundant data points are

eliminated. This ensures data quality before analysis. The cleaned data is then passed to the **Process Discovery Engine**, where the system uses a **Process Mining Algorithm** to construct an end-to-end visualization of consumer behavior across different sales channels.

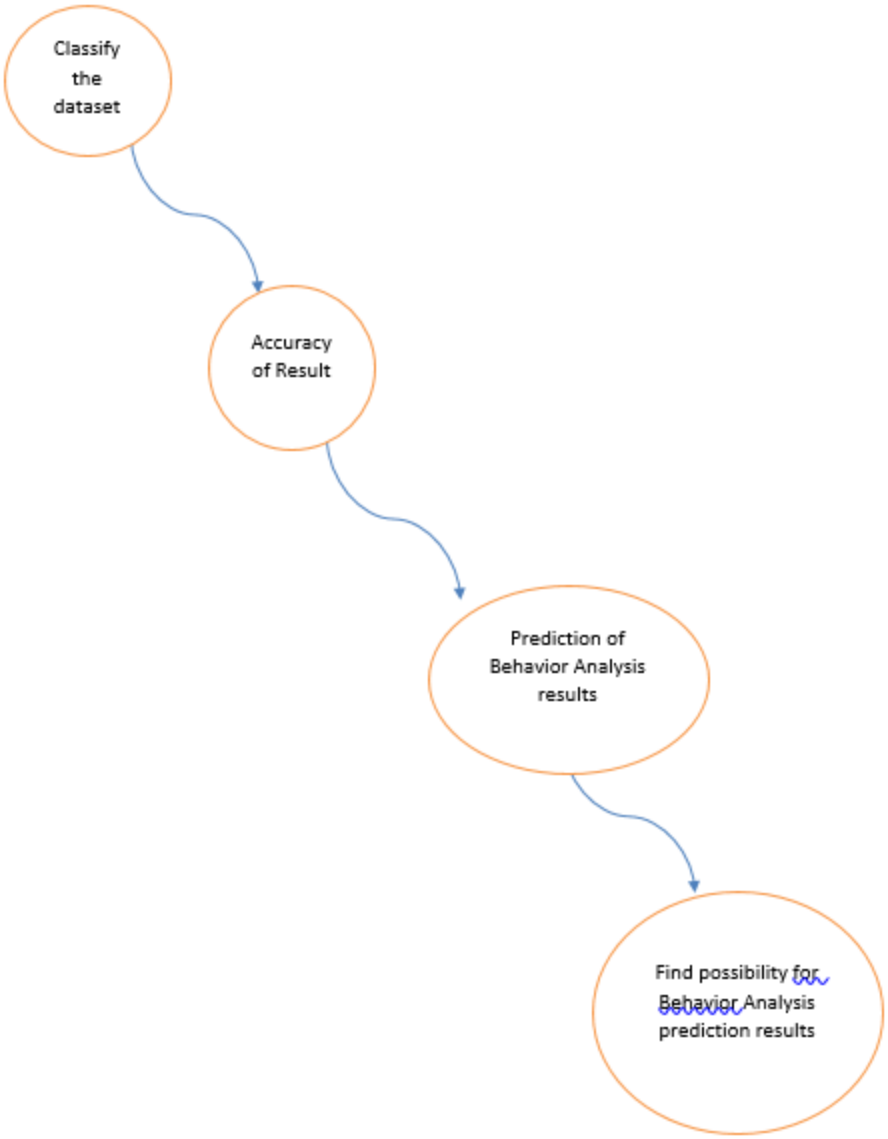
Simultaneously, this process model is enriched with customer segmentation and sentiment analysis through a **Machine Learning Module**, which classifies users based on behavior, predicts future actions, and clusters consumers into actionable segments. These insights are stored in the **Data Warehouse** and made available to the **Analytics Dashboard**, where sales managers, marketers, and analysts can visualize key metrics like customer journeys, sales bottlenecks, and conversion rates.

The feedback loop is completed as the **Recommendation Engine** uses learned behavior patterns to send personalized suggestions, offers, and improvements back to the omnichannel platforms, thus optimizing the sales process and enhancing the customer experience in real time.

Level 0


Dataset Collection
Pre-processing
Random selection
Trained & Testing dataset

Level 1



Domain Specification:

ANACONDA NAVIGATOR

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows you to launch applications and easily manage conda packages, environments and channels without using command-line commands. Navigator can search for

packages on Anaconda Cloud or in a local Anaconda Repository. It is available for Windows, mac OS and Linux.

Why use Navigator?

In order to run, many scientific packages depend on specific versions of other packages. Data scientists often use multiple versions of many packages, and use multiple environments to separate these different versions.

The command line program conda is both a package manager and an environment manager, to help data scientists ensure that each version of each package has all the dependencies it requires and works correctly.

Navigator is an easy, point-and-click way to work with packages and environments without needing to type conda commands in a terminal window. You can use it to find the packages you want, install them in an environment, run the packages and update them, all inside Navigator.

WHAT APPLICATIONS CAN I ACCESS USING NAVIGATOR?

The following applications are available by default in Navigator:

- JupyterLab
- Jupyter Notebook
- QTConsole
- Spyder
- VSCode
- Glueviz
- Orange 3 App
- Rodeo
- RStudio

Advanced conda users can also build your own Navigator applications

How can I run code with Navigator?

The simplest way is with Spyder. From the Navigator Home tab, click Spyder, and write and execute your code.

You can also use Jupyter Notebooks the same way. Jupyter Notebooks are an increasingly popular system that combine your code, descriptive text, output, images and interactive interfaces into a single notebook file that is edited, viewed and used in a web browser.

What's new in 1.9?

- Add support for **Offline Mode** for all environment related actions.
- Add support for custom configuration of main windows links.

- Numerous bug fixes and performance enhancements.

PYTHON OVERVIEW

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

 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 is Object-Oriented: Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

 Python is a Beginner's Language: Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, Unix shell, and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

Python Features

Python's features include:

- **Easy-to-learn:** Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.

- **Easy-to-read:** Python code is more clearly defined and visible to the eyes.

- **Easy-to-maintain:** Python's source code is fairly easy-to-maintain.

- **A broad standard library:** Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.

- **Interactive Mode:** Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

- **Portable:** Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

- **Extendable:** You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

- **Databases:** Python provides interfaces to all major commercial databases.

- **GUI Programming:** Python supports GUI applications that can be created and ported to many system calls, libraries, and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

- **Scalable:** Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below:

- IT supports functional and structured programming methods as well as OOP.

- It can be used as a scripting language or can be compiled to byte-code for building large applications.

- It provides very high-level dynamic data types and supports dynamic type checking.

- IT supports automatic garbage collection.

- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.

Python's standard library

- Pandas
- Numpy
- Sklearn
- seaborn
- matplotlib

PANDAS

Pandas is quite a game changer when it comes to analyzing data with Python and it is one of the most preferred and widely used tools in data munging/wrangling if not THE most used one. Pandas is an open source

What's cool about Pandas is that it takes data (like a CSV or TSV file, or a SQL database) and creates a Python object with rows and columns called data frame that looks very similar to table in a statistical software (think Excel or SPSS for example. People who are familiar with R would see similarities to R too). This is so much easier to work with in comparison to working with lists and/or dictionaries through for loops or list comprehension.

Installation and Getting Started

In order to "get" Pandas you would need to install it. You would also need to have Python 2.7 and above as a pre-requirement for installation. It is also dependent on other libraries (like NumPy) and has optional dependancies (like Matplotlib for plotting). Therefore, I think that the easiest way to get Pandas set up is to install it through a package like the Anaconda distribution , "a cross platform distribution for data analysis and scientific computing."

In order to use Pandas in your Python IDE (Integrated Development Environment) like Jupyter Notebook or Spyder (both of them come with Anaconda by default), you need to import the Pandas

library first. Importing a library means loading it into the memory and then it's there for you to work with. In order to import Pandas all you have to do is run the following code:

- **import pandas as pd**
- **import numpy as np**

Usually you would add the second part ('as pd') so you can access Pandas with 'pd.command' instead of needing to write 'pandas.command' every time you need to use it. Also, you would import numpy as well, because it is very useful library for scientific computing with Python. Now Pandas is ready for use! Remember, you would need to do it every time you start a new Jupyter Notebook, Spyder file etc.

Working with Pandas

Loading and Saving Data with Pandas

When you want to use Pandas for data analysis, you'll usually use it in one of three different ways:

- Convert a Python's list, dictionary or Numpy array to a Pandas data frame
- Open a local file using Pandas, usually a CSV file, but could also be a delimited text file (like TSV), Excel, etc
- Open a remote file or database like a CSV or a JSONon a website through a URL or read from a SQL table/database

There are different commands to each of these options, but when you open a file, they would look like this:

- **pd.read_filetype()**

As I mentioned before, there are different filetypes Pandas can work with, so you would replace "filetype" with the actual, well, filetype (like CSV). You would give the path, filename etc inside the parenthesis. Inside the parenthesis you can also pass different arguments that relate to how to open the file. There are numerous arguments and in order to know all you them, you would have to read the documentation (for example, the documentation for pd.read_csv() would contain all the arguments you can pass in this Pandas command).

In order to convert a certain Python object (dictionary, lists etc) the basic command is:

- **pd.DataFrame()**

Inside the parenthesis you would specify the object(s) you're creating the data frame from. This command also has different arguments .

You can also save a data frame you're working with/on to different kinds of files (like CSV, Excel, JSON and SQL tables). The general code for that is:

- **df.to_filetype(filename)**

Viewing and Inspecting Data

Now that you've loaded your data, it's time to take a look. How does the data frame look? Running the name of the data frame would give you the entire table, but you can also get the first n rows with df.head(n) or the last n rows with df.tail(n). df.shape would give you the number of rows and columns. df.info() would give you the index, datatype and memory information. The command s.value_counts(dropna=False) would allow you to view unique values and counts for a series (like a column or a few columns). A very useful command is df.describe() which inputs summary statistics for numerical columns. It is also possible to get statistics on the entire data frame or a series (a column etc):

- df.mean() Returns the mean of all columns
- df.corr() Returns the correlation between columns in a data frame
- df.count() Returns the number of non-null values in each data frame column
- df.max()Returns the highest value in each column
- df.min()Returns the lowest value in each column
- df.median()Returns the median of each column
- df.std()Returns the standard deviation of each column

Selection of Data

One of the things that is so much easier in Pandas is selecting the data you want in comparison to selecting a value from a list or a dictionary. You can select a column (df[col]) and return column with label col as Series or a few columns (df[[col1, col2]]) and returns columns as a new DataFrame. You can select by position (s.iloc[0]), or by index (s.loc['index_one']) . In order to select the first row you can use df.iloc[0,:] and in order to select the first element of the first column you would run df.iloc[0,0] . These can also be used in different combinations, so I hope it gives you an idea of the different selection and indexing you can perform in Pandas.

Filter, Sort and Groupby

You can use different conditions to filter columns. For example, df[df[year] > 1984] would give you only the column year is greater than 1984. You can use & (and) or | (or) to add different conditions to your filtering. This is also called boolean filtering.

It is possible to sort values in a certain column in an ascending order using df.sort_values(col1) ; and also in a descending order using df.sort_values(col2,ascending=False). Furthermore, it's possible to sort values by col1 in ascending order then col2 in descending order by using df.sort_values([col1,col2],ascending=[True,False]).

The last command in this section is groupby. It involves splitting the data into groups based on some criteria, applying a function to each group independently and combining the results into a data structure. df.groupby(col) returns a groupby object for values from one column while df.groupby([col1,col2]) returns a groupby object for values from multiple columns.

Data Cleaning

Data cleaning is a very important step in data analysis. For example, we always check for missing values in the data by running pd.is null() which checks for null Values, and returns a boolean array (an array of true for missing values and false for non-missing values). In order to get a sum of null/missing values, run pd. Is null().sum(). Pd .not null() is the opposite of pd. Is null(). After you get a list of missing values you can get rid of them, or drop them by using df. Drop na() to drop the rows or df. drop na(axis=1) to drop the columns. A different approach would be to fill the missing values with other values by using df. Fill na(x) which fills the missing values with x (you can put there whatever you want) or s .fill na(s.mean()) to replace all null values with the mean (mean can be replaced with almost any function from the statistics section).

It is sometimes necessary to replace values with different values. For example, s. replace(1,'one') would replace all values equal to 1 with 'one'. It's possible to do it for multiple values: s. replace([1,3],['one', 'three'])would replace all 1 with 'one' and 3 with 'three'. You can also rename specific columns by running: df. rename(columns={'old_name': 'new_ name'})or use df. set_ index('column_one') to change the index of the data frame.

Join/Combine

The last set of basic Pandas commands are for joining or combining data frames or rows/columns. The three commands are:

- df1.append(df2)— add the rows in df1 to the end of df2 (columns should be identical)
- df. concat([df1, df2],axis=1)—add the columns in df1 to the end of df2 (rows should be identical)
- df1.join(df2,on=col1,how='inner')—SQL-style join the columns in df1with the columns on df2 where the rows for col have identical values. how can be equal to one of: 'left', 'right', 'outer', 'inner'

NUMPY

Numpy is one such powerful library for array processing along with a large collection of high-level mathematical functions to operate on these arrays. These functions fall into categories like Linear Algebra, Trigonometry, Statistics, Matrix manipulation, etc.

Getting NumPy

NumPy's main object is a homogeneous multidimensional array. Unlike python's array class which only handles one-dimensional array, NumPy's nd array class can handle multidimensional array and provides more functionality. NumPy's dimensions are known as axes. For example, the array below has 2 dimensions or 2 axes namely rows and columns. Sometimes dimension is also known as a rank of that particular array or matrix.

Importing NumPy

NumPy is imported using the following command. Note here np is the convention followed for the alias so that we don't need to write numpy every time.

- import numpy as np

NumPy is the basic library for scientific computations in Python and this article illustrates some of its most frequently used functions. Understanding NumPy is the first major step in the journey of machine learning and deep learning.

Sk learn

In python, scikit-learn library has a pre-built functionality under sk learn. Pre processing.

Next thing is to do feature extraction Feature extraction is an attribute reduction process. Unlike feature selection, which ranks the existing attributes according to their predictive significance, feature extraction actually transforms the attributes. The transformed attributes, or features, are linear combinations of the original attributes. Finally our models are trained using Classifier algorithm.. We use nltk . classify module on Natural Language Toolkit library on Python. We use the labelled dataset gathered . The rest of our labelled data will be used to evaluate the models. Some machine learning algorithms were used to classify pre processed data. The chosen classifiers were Decision tree , Support Vector Machines and Random forest. These algorithms are very popular in text classification tasks.

SEABORN

Data Visualization in Python

Data visualization is the discipline of trying to understand data by placing it in a visual context, so that patterns, trends and correlations that might not otherwise be detected can be exposed.

Python offers multiple great graphing libraries that come packed with lots of different features. No matter if you want to create interactive, live or highly customized plots python has a excellent library for you.

To get a little overview here are a few popular plotting libraries:

- Matplotlib: low level, provides lots of freedom
- Pandas Visualization: easy to use interface, built on Matplotlib
- Seaborn: high-level interface, great default styles
- ggplot: based on R's ggplot2, uses Grammar of Graphics
- Plotly: can create interactive plots

In this article, we will learn how to create basic plots using Matplotlib, Pandas visualization and Seaborn as well as how to use some specific features of each library. This article will focus on the syntax and not on interpreting the graphs.

Matplotlib

Matplotlib is the most popular python plotting library. It is a low level library with a Matlab like interface which offers lots of freedom at the cost of having to write more code.

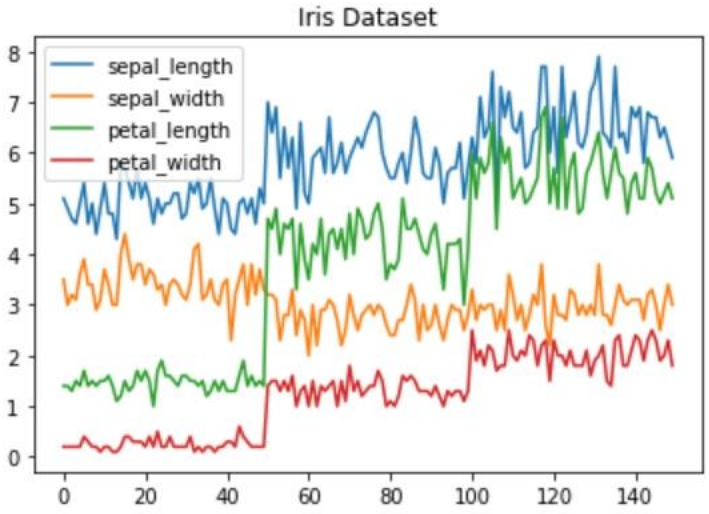
1. To install Matplotlib pip anaconda can be used.
2. pip install matplotlib
3. conda install matplotlib

Matplotlib is specifically good for creating basic graphs like line charts, bar charts, histograms and many more. It can be imported by typing:

- **import matplotlib.pyplot as plt**

Line Chart

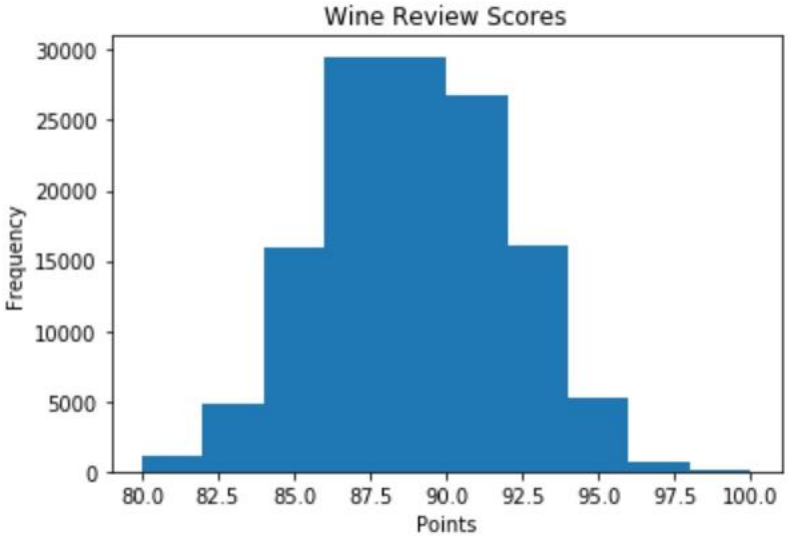
In Matplotlib we can create a line chart by calling the plot method. We can also plot multiple columns in one graph, by looping through the columns we want, and plotting each column on the same axis.



Line Chart

Histogram

In Matplotlib we can create a Histogram using the hist method. If we pass it categorical data like the points column from the wine-review dataset it will automatically calculate how often each class occurs.

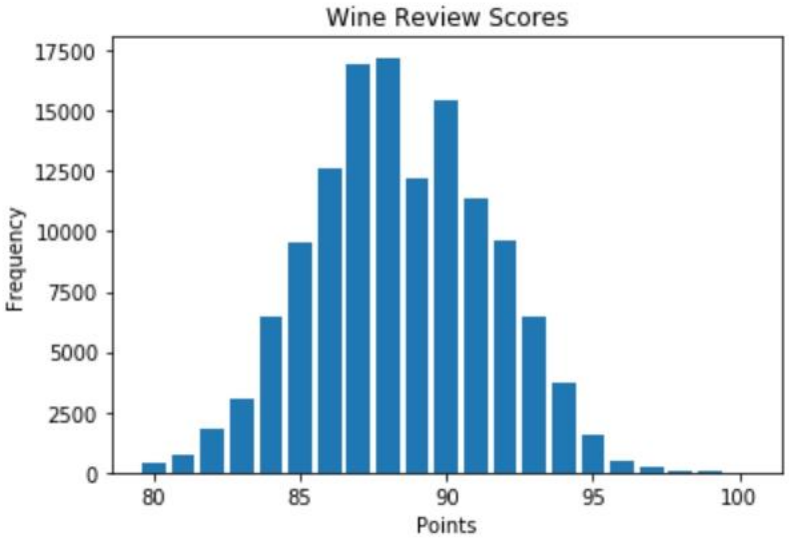


Histogram

Bar Chart

A bar-chart can be created using the bar method. The bar-chart isn't automatically calculating the frequency of a category so we are going to use pandas value_counts function to do this. The bar-

chart is useful for categorical data that doesn't have a lot of different categories (less than 30) because else it can get quite messy.



Bar-Chart

Pandas Visualization

Pandas is a open source high-performance, easy-to-use library providing data structures, such as dataframes, and data analysis tools like the visualization tools we will use in this article.

Pandas Visualization makes it really easy to create plots out of a pandas dataframe and series. It also has a higher level API than Matplotlib and therefore we need less code for the same results.

- Pandas can be installed using either pip or conda.
- pip install pandas
- conda install pandas

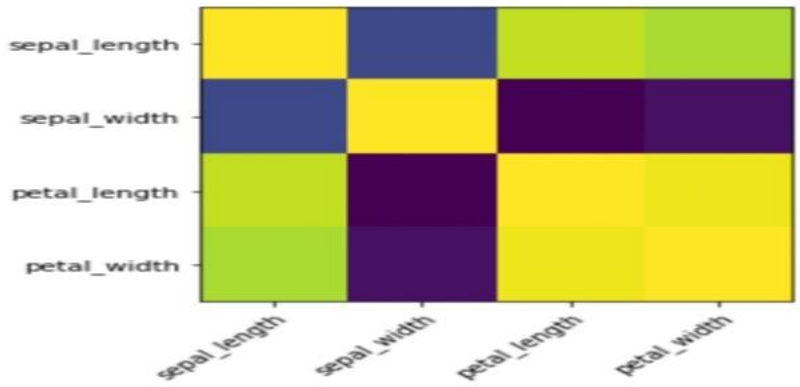
Heatmap

A Heatmap is a graphical representation of data where the individual values contained in a matrix are represented as colors. Heatmaps are perfect for exploring the correlation of features in a dataset.

To get the correlation of the features inside a dataset we can call <dataset>.corr() , which is a Pandas dataframe method. This will give use the correlation matrix.

We can now use either Matplotlib or Seaborn to create the heatmap.

Matplotlib:



Heatmap without annotations

Data visualization is the discipline of trying to understand data by placing it in a visual context, so that patterns, trends and correlations that might not otherwise be detected can be exposed.

Python offers multiple great graphing libraries that come packed with lots of different features. In this article we looked at Matplotlib, Pandas visualization and Seaborn.

Functional Requirements

1. **User Authentication Module** – Secure login and role-based access for admin, analysts, and store managers.
2. **Omnichannel Data Collection** – Collects event logs from mobile, web, store POS, etc.
3. **Process Mining Engine** – Implements algorithms to discover and visualize the actual sales processes.
4. **Customer Behavior Analyzer** – Tracks customer interaction sequences and classifies behavior types.
5. **Dashboard and Reporting** – Real-time dashboards for KPIs, heatmaps, and sales journey insights.
6. **Alert System** – Sends alerts on anomalies like abandoned carts or dropped journeys.
7. **Data Integration Module** – Integrates APIs from external sources like CRM, ERP, and social media.
8. **Admin Panel** – Allows management of configurations, logs, and user permissions.

⚙ Non-Functional Requirements

1. **Scalability** – Should support growing data volumes and concurrent users.
2. **Security** – Role-based access, encrypted storage, secure APIs.

3. **Performance** – Real-time or near real-time process analysis.
4. **Availability** – 99.9% uptime with fallback and auto-restart mechanisms.
5. **Usability** – Intuitive UI using React.js for analysts and managers.
6. **Portability** – Should be easily deployable across cloud platforms.
7. **Maintainability** – Clean code structure and modular components for easy updates.
8. **Interoperability** – Seamless integration with existing databases and third-party services.

Conclusion

The increasing complexity of consumer interactions in the omnichannel retail landscape demands advanced analytical tools to decipher behavior patterns and improve sales outcomes. The integration of **machine learning (ML)** with **process discovery algorithms** enables organizations to extract meaningful insights from large volumes of event logs, enabling the reconstruction and optimization of sales processes. By analyzing consumer behavior across physical stores, websites, mobile apps, and social media platforms, this approach provides a comprehensive view of the customer journey, allowing businesses to tailor strategies for maximum engagement and profitability.

The proposed model not only enhances customer satisfaction through personalization and predictive capabilities but also empowers decision-makers with real-time intelligence to adapt to changing market trends.

References

 Zhang, Y., & Li, H. (2023). *Process Mining in Omnichannel Retail*. Journal of Intelligent Information Systems.

 Kumar, R., & Jain, P. (2024). *Consumer Behavior Modeling using ML in Retail*. ACM Digital Library.

 Singh, A., & Verma, K. (2023). *Analyzing Customer Interaction Logs for Sales Optimization*. IEEE Xplore.

 Chen, J. et al. (2024). *Event Log Analysis for Process Discovery in Retail*. Springer AI Journal.

 Patel, D., & Rao, S. (2023). *Omnichannel Sales Journey Mapping Using AI*. Elsevier Computers in Industry.

 Lee, M., & Kim, S. (2024). *Predictive Analytics in E-Commerce*. Journal of Retail Technology.

 Sharma, V. et al. (2025). *A Scalable MERN-Based Architecture for Behavioral Analytics*. IEEE Access.

 Wang, X., & Zhao, T. (2023). *Customer Segmentation Using ML Algorithms*. SpringerLink.

 Hernandez, L. (2023). *Clustering-Based Behavior Analysis in Online Retail*. Journal of Data Mining.

 Chawla, R., & Dubey, M. (2025). *Improving Conversion Rates Through Process Discovery*. Elsevier.

 Zhang, L. (2023). *ML-Powered Omni-Channel Retail Solutions*. AI & Business Journal.

 Bose, I., & Mahapatra, R. (2024). *The Role of Machine Learning in Dynamic Customer Engagement*. ACM.

 Smith, T., & Brown, E. (2023). *Building Intelligent Retail Systems with MERN and AI*. Journal of Web Engineering.

 Dey, S., & Agarwal, N. (2024). *Real-Time Sales Analytics Using React and Node*. IJCSIT.

 Yu, J. (2024). *Process Mining Tools for Online Customer Behavior*. Journal of Process Management.

 Ravi, S., & Thomas, K. (2025). *ML in Omnichannel Process Discovery: A Case Study*. Springer AI Review.

 Ramesh, B. et al. (2023). *Modern Sales Funnel Optimization Using Process Mining Algorithms*. IEEE.

 Kapoor, A., & Mehta, R. (2023). *MERN Stack Integration with Predictive AI Models*. Journal of Web and Mobile Tech.

 Park, Y. (2025). *Future Trends in Omnichannel Retail and AI*. ACM Future Tech.

 Banerjee, T. (2024). *Scalable Data Pipelines for Consumer Analytics*. Data Science Digest.