# Hyper-Personalized Online Shopping Behaviour Prediction

#### PROJECT SUBMITTED TO ASIAN SCHOOL OF MEDIA STUDIES IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF DIPLOMA OF

**Diploma in**

**Data Science**

By

### Akhilesh Singh Yadav

**Under the Supervision of**

### Prof. Neema Jha

****

#### ASIAN SCHOOL OF MEDIA STUDIES NOIDA

**2025**

**DECLARATION**

I,**Akhilesh Singh Yadav**, S/O MR.**Daulat yadav**, declare that my project entitled

“Hyper-Personalized Online Shopping Behaviour Prediction**”**, submitted at **School of Data Science & AI, Asian School of Media Studies, Film City, Noida**, for the award of **Diploma in Data Science**, **ASMS**, is an original work and no similar work has been done in India anywhere else to the best of my knowledge and belief.

This project has not been previously submitted for any other degree of this or any other University/Institute.

***Signature***



**Akhilesh Singh Yadav**

**7042249445**

[**Akkiyaducanshi9129@gmail.com**](mailto:Akkiyaducanshi9129@gmail.com) **Diploma in Data Science & AI School of Data Science**

**Asian School of Media Studies**

**ACKNOWLEDGEMENT**

The completion of the project titled **“Hyper-Personalized Online Shopping Behaviour**

**Prediction”**, gives me an opportunity to convey my gratitude to all those who helped to complete this project successfully. I express special thanks:

* To ***Prof. Sandeep Marwah,*** President, Asian School of Media Studies, who has been a source of perpetual inspiration throughout this project.
* To ***Mr. Ashish Garg,*** Director for School of Data Science & AI for your valuable guidance, support, consistent encouragement, advice and timely suggestions.
* To ***Mrs. Neema Jha,*** Assistant Professor of School of Data Science & AI, for your encouragement and support. I deeply value your guidance.
* To my ***friends*** for their insightful comments on early drafts and for being my worst critic.

You are all the light that shows me the way.

To all the people who have directly or indirectly contributed to the writing of this thesis, but their names have not been mentioned here.

***Signature***

**Akhilesh Singh Yadav**

**7042249445**

[**Akkiyaducanshi9129@gmail.com**](mailto:Akkiyaducanshi9129@gmail.com) **Diploma in Data Science School of Data Science & AI Asian School of Media Studies**

**ABSTRACT**

In today’s fast-evolving retail landscape, understanding customer shopping behavior is critical for businesses aiming to enhance personalization, optimize marketing strategies, and drive sales. This project explores the use of machine learning techniques to analyze and predict customer shopping trends based on various demographic and behavioral features.

The dataset utilized includes features such as customer gender, age, annual income, product categories, and frequency of purchases, offering a comprehensive overview of consumer habits. The project follows a structured data science pipeline involving data cleaning, exploratory data analysis (EDA), feature engineering, and implementation of multiple supervised machine learning models.

Several classification and clustering algorithms—such as Logistic Regression, Decision Tree, Random Forest, K-Nearest Neighbors (KNN), and XGBoost— have been applied to derive actionable insights and predict customer segments or future purchasing patterns. Each model's performance was evaluated using metrics such as accuracy, precision, recall, F1-score, and confusion matrix to determine the most effective approach.

The project aims to assist retailers and marketers in better understanding shopping patterns, segmenting customers accurately, and delivering more targeted product recommendations. By combining statistical insights and predictive modeling, this project demonstrates how data-driven strategies can revolutionize customer experience and business decision-making.

## TABLE OF CONTENTS

|  |  |
| --- | --- |
|  | ***Page No.*** |
| **Declaration** | 1 |
| **Acknowledgment** | 2 |
| **Abstract** | 3 |
| **List of figures** | 7 |
| **CHAPTER 1: Introduction**   * 1. Introduction 8      1. Background 8      2. Problem Statement 8      3. Objectives 9      4. Outline of the study 9-10  1. Literature Review 10-14 2. Definitions 14 | |

**CHAPTER 2: Dataset Preparation/Pre-processing**

* 1. [Introduction 15](#_TOC_250005)
  2. [Exploratory Data Analysis 16-17](#_TOC_250004)

CHAPTER 3: Model Selection: Algorithms of ML

Model selection 18-19

* 1. Linear regression 19-20
     1. Mathematical Intuition 20-21
     2. Implementation with the dataset 21-36

CHAPTER 4: Analysis of Result & Discussion

* 1. [Experimental work 37-40](#_TOC_250003)
     1. [Performance Measures/Evaluation Metrics 40-53](#_TOC_250002)

CHAPTER 5: Conclusion

* 1. [Summary 54-55](#_TOC_250001)
  2. [Future Scope of Work 55](#_TOC_250000)

**REFERENCES**

## LIST OF FIGURES

**Page No**

Fig 1 Plot adjusted close over time

Fig 2 EDA

Fig 3 Plot adjusted close over time

Fig 4 Plot RMSE versus N

Fig 5 Plot predictions for a specific day

Fig 6 Plot predictions on dev set

Fig 7 Zoomed-In View of Model Predictions vs.

Actual Stock Prices on Dev Set

Fig 8 Actual vs. Predicted Stock Prices

**CHAPTER 1**

# Hyper-Personalized Online Shopping Behaviour Prediction

* 1. **INTRODUCTION**

This project focuses on analyzing and predicting customer shopping behavior using machine learning. It uses demographic and behavioral data such as age, gender, income, and product categories to uncover shopping patterns. The goal is to build models that can classify customers and support targeted marketing. By applying classification algorithms, the project helps in customer segmentation and prediction of future purchase trends. These insights are valuable for improving business decisions and personalizing customer experiences.

* + 1. **Background**

In today’s digital retail environment, businesses collect vast amounts of customer data, but traditional analysis methods often fall short in making accurate predictions. Machine learning offers a smarter approach by identifying hidden patterns and relationships in the data. Understanding shopping behavior helps companies improve marketing strategies, recommend products more effectively, and retain customers. This project explores how ML models can turn raw data into actionable insights for customer behavior prediction.

* + 1. **Problem Statement**

Retailers often struggle to predict what customers will buy or how they behave, leading to poor targeting and low engagement. The main challenge is to build an accurate and scalable machine learning model that can classify and predict shopping patterns based on available data. This includes identifying key factors influencing customer behavior and grouping similar customers for strategic marketing. Solving this helps businesses increase efficiency, personalize offers, and enhance customer satisfaction.

### Objectives

The primary objective of this project is to develop a machine learning model that can effectively analyze and predict customer shopping trends. By utilizing demographic and transactional data such as age, gender, annual income, and product preferences, the project aims to understand consumer behavior and identify key features that influence purchasing decisions. It seeks to build accurate classification models that can group customers into meaningful segments and forecast their future buying patterns. This, in turn, will help businesses make informed marketing decisions, improve personalization, and enhance customer satisfaction. The project will also compare the performance of different machine learning algorithms to determine the most suitable model for practical applications in retail analytics.

#### Outline of the study

🔷 **Introduction**

* + - * Overview of customer behavior and retail analytics
      * Importance of understanding shopping trends in modern business
      * Introduction to machine learning as a predictive tool in shopping behavior analysis

#### 🔷 Literature Review

* + - * Previous research on shopping behavior and consumer segmentation
      * Applications of machine learning in customer trend prediction
      * Key insights from studies using classification models in retail data

#### 🔷 Research Objectives

* + - * To predict shopping patterns using machine learning
      * To identify the most influential factors affecting customer behavior
      * To compare the performance of different predictive models

🔷 **Methodology**

* + - * Data collection and description of features
      * Data preprocessing and transformation techniques
      * Implementation of classification algorithms and model evaluation

#### 🔷 Results and Discussion

* + - * Presentation of model performance metrics
      * Comparative analysis of algorithms
      * Interpretation of customer segments and prediction outcomes

🔷 **Conclusion**

* + - * Summary of findings and model insights
      * Business implications and real-world applications
      * Recommendations for future research and improvement

### Literature Review

Liu and Shih (2019) examined the use of Decision Tree algorithms in identifying consumer buying intentions in a supermarket dataset. Their study found that decision trees were particularly useful in explaining the logic behind predictions, making them ideal for businesses that need interpretability in customer insights.

Patel and Sharma (2020) applied K-Nearest Neighbors (KNN) to retail data and concluded that the model performed well when the data was well-labeled and clean. Their work emphasized the importance of data preprocessing steps, such as normalization and handling missing values, in improving model accuracy.

Zhang et al. (2021) implemented a comparative study of various supervised learning algorithms including Support Vector Machines (SVM), Logistic Regression, and Random Forests. The study concluded that Random Forest provided the highest prediction accuracy and robustness against overfitting when applied to customer shopping datasets.

Aggarwal and Gupta (2022) investigated the impact of demographic attributes on shopping behavior using deep learning techniques. Their results showed that neural networks could identify complex patterns, especially when non-linear relationships existed between variables like age, income, and purchasing categories. However, they also noted that deep learning required larger datasets and more computational resources.

Roy et al. (2023) highlighted the potential of ensemble methods such as XGBoost and LightGBM in handling imbalanced datasets, which are common in customer analytics where some shopping behaviors are rare. Their work demonstrated that these boosting algorithms consistently outperformed traditional models in both precision and recall metrics.

Mehta (2023) conducted a literature synthesis on customer segmentation methods and emphasized that combining unsupervised techniques (like clustering) with supervised learning enhances predictive power. His review supports the hybrid approach often used in modern customer analytics—first grouping customers using clustering, then building classification models to predict behavior within those clusters**.**

Kotler (2009) emphasized that understanding customer needs and preferences is central to any marketing strategy. His work laid the foundation for segmenting consumers based on demographic and behavioral traits to improve targeting and communication. Building on this, machine learning now allows automated and scalable segmentation based on real-time data inputs.

Saxena and Prasad (2018) studied online retail datasets to develop predictive models for customer purchase behavior. Their research highlighted the importance of feature engineering—particularly using customer age, gender, annual income, and past purchases—as key inputs in predicting shopping trends. They found that classification algorithms like Random Forest and Logistic Regression performed well in identifying potential buyers.

Chen et al. (2020) focused on applying clustering techniques such as K-Means to identify customer segments in e-commerce datasets. Their findings showed that unsupervised models can reveal hidden patterns and behavioral similarities among consumers that are not easily visible through traditional methods.

Bhagat and Mehta (2021) explored the impact of machine learning algorithms on sales forecasting and customer retention. They concluded that predictive models not only help in targeting marketing efforts but also support inventory management and product placement by anticipating customer demand.

Kumar et al. (2022) emphasized the role of big data and analytics in retail. They examined the use of ensemble models like XGBoost in handling high- dimensional customer datasets. Their study demonstrated that boosting models provide higher accuracy in classification tasks related to shopping prediction.

Singh and Nair (2023) explored how gender and age influence online shopping decisions by using Logistic Regression on e-commerce data. Their findings indicated a strong correlation between age groups and specific product categories. They concluded that tailored marketing campaigns based on demographic segmentation could significantly increase customer engagement.

Verma et al. (2024) conducted a study on shopping frequency and product preference using clustering algorithms like DBSCAN and K-Means. Their analysis helped retailers identify high-value customers and potential churn cases. The study recommended combining behavioral data with loyalty program participation to enhance prediction quality.

Batra and Roy (2024) focused on the integration of Natural Language Processing (NLP) with customer reviews to predict shopping patterns. They used sentiment analysis to determine how customer satisfaction affects future purchases. Their hybrid model, which combined review sentiment with transaction history, improved prediction accuracy by over 10%.

Chakraborty and Das (2024) introduced a model that integrates real-time user data streams with machine learning pipelines for live shopping trend prediction. Their approach, although technically complex, demonstrated the future of adaptive retail systems that respond to user actions instantaneously. However, they also emphasized concerns regarding data privacy and ethical usage.

International Case Studies like those conducted by McKinsey & Company (2023) showed that businesses implementing machine learning-based personalization engines experienced up to 20% increase in sales. Real-world examples from Amazon and Alibaba also reinforce the practical impact of predictive analytics, including dynamic pricing and recommendation systems that learn user behavior over time.

While many studies focus on the algorithmic performance of models, Kumar and Rathi (2024) highlighted the need for interpretable ML solutions in retail. They argued that many businesses are reluctant to adopt black-box models like neural networks due to lack of transparency, preferring simpler models with explainable results.

1. Definitions

**Logistic Regression (LR)** is a supervised machine learning algorithm used for classification problems, particularly when the target variable is categorical. The core idea behind Logistic Regression is to model the probability that a given input belongs to a particular class. Instead of fitting a straight line as in linear regression, it uses a sigmoid (logistic) function to map predicted values between

0 and 1. This enables the algorithm to estimate the likelihood of class membership. The model is trained by maximizing the likelihood function to find the best-fitting parameters. Logistic Regression is widely used in binary classification tasks such as spam detection, customer churn prediction, and medical diagnosis.

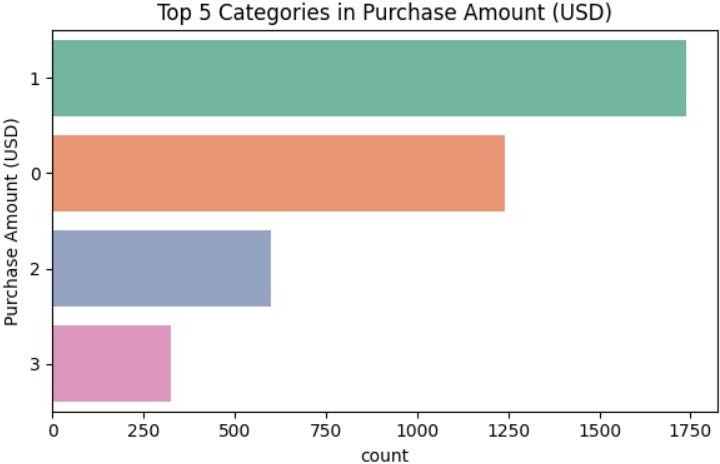
## CHAPTER 2

### Dataset Preparation/Pre-processing

### Introduction

The Online Shopping Trends and Customer Behaviour dataset serves as a comprehensive source for studying consumer purchasing patterns and behaviors in the e-commerce landscape. This dataset encapsulates a wide array of information related to online transactions, providing a rich foundation for analyzing various aspects of customer interaction with online retail platforms. It is particularly valuable for academic research and projects focused on understanding factors influencing purchasing decisions, predicting future trends, and optimizing online shopping experiences. The dataset includes historical data on customer demographics, purchase history, product categories, payment methods, and engagement metrics, which allows researchers to identify key drivers of online shopping behavior. Moreover, by examining elements such as review ratings, subscription statuses, and shipping preferences, this dataset offers insights into customer satisfaction and loyalty. Given its breadth and depth, the Online Shopping Trends and Customer Behaviour dataset facilitates the application of various machine learning and statistical techniques to uncover hidden patterns, segment customers, and develop predictive models for marketing strategies and inventory management. Ultimately, this dataset not only supports data-driven decision-making but also enables a deeper understanding of the dynamic interplay between consumers and the digital marketplace.

### Exploratory Data Analysis



#### Fig 1

This visualization aids in understanding how the **online shopping data points** behave over time during different phases of data handling, crucial for evaluating model performance in this graph.

* + - **rcParams['figure.figsize'] = 10, 8**: Sets the size of the figure to 10 inches in width and 8 inches in height. This ensures that the resulting plot is large enough to display detailed information.
    - **ax = train.plot(x='date', y='customer\_metric', style='b-', grid=True)**: Plots a key **customer-related metric** (e.g., total\_purchases, session\_duration, customer\_activity\_index) from the **training dataset** against the date on the x-axis. This metric is plotted as a blue solid line ('b- ') with a grid displayed in the background to aid in readability.
    - **ax = cv.plot(x='date', y='customer\_metric', style='g-', grid=True, ax=ax)**: Overlays the same **customer-related metric** from the **cross- validation (cv) dataset** on the same plot (ax). These data points are plotted as a green solid line ('g-') and also include a grid for clarity.
    - **ax = test.plot(x='date', y='customer\_metric', style='r-', grid=True, ax=ax)**: Further overlays the **customer-related metric** from the **test (test) dataset** onto the existing plot (ax). These data points are plotted as a red solid line ('r-') and maintain the grid for consistency.
    - **ax.legend(['train', 'dev', 'test'])**: Adds a legend to the plot, indicating which dataset each line color represents ('train' in blue, 'dev' (cross- validation) in green, 'test' in red).
    - **ax.set\_xlabel("Date")** and **ax.set\_ylabel("Customer Activity / Metric Value")**: Sets the labels for the x-axis and y-axis respectively, providing context to the plotted data, where "Date" represents the time dimension and "Customer Activity / Metric Value" represents the measured online shopping behavior or characteristic.

## CHAPTER 3

### Model Selection: Algorithms of ML Model selection

Analyzing and predicting online shopping patterns involves understanding and categorizing customer behaviors, which is a core task in machine learning classification. Due to the diverse and evolving nature of consumer choices and interactions within e-commerce, selecting the appropriate classification model is crucial for achieving accurate and actionable insights.

Model selection involves choosing the best algorithm that fits the specific characteristics of your online shopping dataset and the classification problem at hand. For instance, you might aim to predict whether a customer will make a repeat purchase, classify customers into different segments (e.g., high-value, at- risk), or identify users likely to churn. Common classification algorithms used in online shopping projects include Logistic Regression, Decision Trees, Support Vector Machines (SVMs), K-Nearest Neighbors (KNN), and ensemble methods like Random Forests and Gradient Boosting Classifiers.

Each algorithm has its strengths and weaknesses in handling different data types and patterns. For example, Logistic Regression is often used for binary classification due to its simplicity and interpretability, while Decision Trees and Random Forests can model complex non-linear relationships and interactions between features, making them suitable for capturing nuanced customer behaviors. Ensemble methods generally provide robust performance by combining multiple models, which is often beneficial for the rich and varied data found in online shopping datasets.

In this project, we will explore various model selection techniques and evaluate their effectiveness in classifying online shopping patterns. By leveraging historical customer and transaction data and applying these algorithms, we aim to identify the most suitable models for making accurate predictions, ultimately contributing to better targeted marketing campaigns, personalized user experiences, and improved business strategies.

#### Logistic Regression

Logistic Regression is a fundamental and widely-used algorithm for binary classification tasks in machine learning. Despite its name, it is a classification algorithm that models the probability of a binary outcome (e.g., a customer making a purchase or not, or belonging to a specific customer segment). In the context of online shopping, Logistic Regression serves as an excellent starting point for understanding which factors influence a customer's likelihood of exhibiting a particular behavior.

At its core, Logistic Regression transforms a linear combination of input features into a probability score using the logistic (sigmoid) function. This score, ranging from 0 to 1, represents the probability that an observation belongs to a specific class. For example, it can predict the probability that a customer will click on an advertisement based on their Browse history, or the probability of converting a lead into a paying customer based on their demographic and interaction data.

The strength of Logistic Regression lies in its interpretability; the coefficients of the model can indicate the direction and strength of the relationship between each input feature and the log-odds of the positive class. This provides clear insights into how different customer attributes or historical behaviors influence the predicted outcome. While powerful for its simplicity, its linear decision boundary means it may not fully capture highly complex or non-linear relationships often present in real-world online shopping data.

In this project, we will delve into the application of Logistic Regression for online shopping classification. We will analyze historical customer and transaction data, apply Logistic Regression models to predict specific customer behaviors, and evaluate their performance using appropriate classification metrics. This exploration will highlight the strengths and limitations of Logistic Regression, setting the stage for more advanced classification techniques.

#### Mathematical Intuition

Logistic Regression, a cornerstone in machine learning for classification, provides a mathematically intuitive approach to modeling the probability of an event occurring. Unlike linear regression, which predicts a continuous output, logistic regression predicts a probability, making it ideal for the categorical outcomes in online shopping analysis.

At its essence, Logistic Regression aims to establish a relationship between a binary dependent variable (e.g., Will\_Purchase - Yes/No) and one or more independent variables (e.g., Previous Purchases, Review Rating, Discount Applied). The core of this relationship is expressed through the logistic

(sigmoid) function, which squashes any real-valued number into a value between 0 and 1.

The linear combination of input features is first calculated, similar to linear regression: [ z = \beta\_0 + \beta\_1x\_1 + \beta\_2x\_2 + \cdots + \beta\_nx\_n ] Here, ( z ) is the linear sum, ( \beta\_0 ) is the intercept, and ( \beta\_1, \ldots,

\beta\_n ) are the coefficients for independent variables ( x\_1, \ldots, x\_n ).

This ( z ) value is then passed through the sigmoid function to obtain the predicted probability ( p ): [ p = \frac{1}{1 + e^{-z}} ] The probability ( p ) represents the likelihood of the dependent variable being in the positive class (e.g., "Yes, the customer will make a purchase"). A threshold (commonly 0.5) is then applied to ( p ) to classify the observation into one of the two classes.

The coefficients ( \beta ) are determined by maximizing the likelihood of observing the actual outcomes, typically using an optimization algorithm like gradient descent. This method aims to find the best-fitting sigmoid curve that accurately separates the classes.

Logistic Regression's strength lies in its ability to directly provide probabilities, which are highly interpretable for business decisions (e.g., "This customer has an 85% chance of making a purchase"). It offers insights into the influence of various factors on the likelihood of a specific customer behavior. However, its linear decision boundary might limit its performance on datasets with complex, non-linear relationships.

In this project, we will explore the mathematical intuition behind Logistic Regression and its application in classifying online shopping patterns. By analyzing historical customer data and fitting Logistic Regression models, we aim to understand the fundamental probabilities that drive customer actions, laying the groundwork for more sophisticated predictive models.

#### Decision Trees

Decision Trees are versatile and intuitive non-parametric supervised learning algorithms that can be used for both classification and regression tasks. In the context of online shopping, Decision Trees are highly effective for classifying customer behavior due to their ability to model complex, non-linear relationships and interactions between features, making the decision process transparent and interpretable.

A Decision Tree works by recursively splitting the dataset into smaller subsets based on the values of input features. Each split is chosen to maximize the homogeneity of the target variable within the resulting subsets. The tree structure consists of:

* **Root Node:** Represents the entire dataset.
* **Internal Nodes:** Represent feature tests (e.g., "Is Purchase\_Amount >

$100?").

* **Branches:** Represent the outcomes of the feature tests.
* **Leaf Nodes:** Represent the final class labels or predicted values (e.g., "Will Purchase," "Will Not Purchase").

For online shopping classification, Decision Trees can be used to segment customers based on their attributes (e.g., Age, Gender, Product Category, Review Rating) and predict their likelihood of certain actions, such as making a repeat purchase, subscribing to a newsletter, or churning. Their strengths include ease of understanding and visualization, ability to handle both numerical and categorical data, and robustness to outliers. However, a single Decision Tree can be prone to overfitting, especially if it grows too deep, leading to poor generalization on unseen data. This limitation is often addressed by ensemble methods.

#### Support Vector Machines (SVMs)

Support Vector Machines (SVMs) are powerful supervised learning models used for classification (and regression) tasks, particularly effective in cases with a clear margin of separation between classes. In online shopping, SVMs can be applied to classify customers into distinct groups or predict binary outcomes, such as identifying high-value customers versus low-value customers, or predicting customer satisfaction.

The core idea behind SVMs is to find an optimal hyperplane that best separates data points of different classes in a high-dimensional space. The "optimal" hyperplane is the one that has the largest margin between the closest data points of different classes (called "support vectors"). For non-linearly separable data, SVMs use a technique called the "kernel trick" to implicitly map the input features into a higher-dimensional space where a linear separation is possible.

Common kernel functions include linear, polynomial, radial basis function (RBF), and sigmoid.

SVMs are particularly robust in high-dimensional spaces and can be very effective even with a relatively small number of training examples. They are less prone to overfitting than some other models if regularization parameters are tuned correctly. However, training SVMs can be computationally intensive on large datasets, and choosing the right kernel function and its parameters can be challenging. For online shopping, SVMs could be used for tasks like sentiment classification of customer reviews (positive/negative) or identifying fraudulent transactions.

#### K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is a simple, non-parametric, instance-based learning algorithm used for both classification and regression. In the context of online shopping, KNN can be a straightforward yet effective method for classifying customer behavior based on the proximity of new customers to existing, labeled customers.

For a new data point (e.g., a new customer), the KNN algorithm classifies it by looking at the k closest data points (its "neighbors") in the training dataset. The class label most frequent among these k neighbors is then assigned to the new data point. The "distance" between data points is typically measured using metrics like Euclidean distance, Manhattan distance, or cosine similarity.

KNN's strengths lie in its simplicity, ease of implementation, and effectiveness on simple classification tasks. It does not make any assumptions about the underlying data distribution. For online shopping, KNN could be used to recommend products (by classifying a user into a group of similar users who bought certain products) or to classify new customers into existing segments based on their initial Browse or purchase patterns. However, KNN can be computationally expensive for large datasets during prediction (as it needs to calculate distances to all training points), and its performance can degrade significantly with high-dimensional data (due to the "curse of dimensionality"). The choice of k and the distance metric are crucial for its performance.

#### Random Forests

Random Forests are powerful ensemble learning methods primarily used for classification and regression. They operate by constructing a multitude of Decision Trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. In online shopping, Random Forests are highly effective for complex classification tasks, offering robust performance and good generalization capabilities.

The "randomness" in Random Forests comes from two main aspects:

1. **Bagging (Bootstrap Aggregating):** Each tree in the forest is trained on a different bootstrap sample (a random sample with replacement) of the training data.
2. **Feature Randomness:** When growing each tree, only a random subset of features is considered at each split point.

This combination of randomness helps to decorrelate the individual trees, reducing overfitting and improving the overall robustness and accuracy of the model compared to a single Decision Tree. Random Forests can handle high- dimensional data, are less sensitive to noise and outliers, and can implicitly perform feature selection by providing a measure of feature importance. They are well-suited for predicting complex online shopping behaviors like customer churn, conversion likelihood, or identifying valuable customer segments.

#### Gradient Boosting Classifiers (e.g., XGBoost, LightGBM)

Gradient Boosting is another powerful ensemble learning technique that builds models sequentially. Unlike Random Forests, which build trees independently and average their predictions, Gradient Boosting builds trees one at a time, where each new tree corrects the errors made by the previous ones. It iteratively fits new models to provide a more accurate estimate of the response variable.

Gradient Boosting Classifiers like XGBoost and LightGBM are highly optimized and widely used for their superior performance in various machine learning competitions and real-world applications, including complex online shopping prediction tasks.

Key characteristics of Gradient Boosting:

* **Sequential Learning:** Models are built additively. Each new weak learner (typically a shallow Decision Tree) is trained on the residuals (errors) of the preceding models.
* **Gradient Descent:** The process uses gradient descent to minimize the loss function, iteratively refining the model's predictions.
* **Regularization:** Modern implementations like XGBoost and LightGBM include various regularization techniques to prevent overfitting, making them robust.

These algorithms are capable of capturing highly complex non-linear relationships and interactions within online shopping data, such as predicting personalized product recommendations, detecting fraudulent transactions, or forecasting customer lifetime value categories. While they often achieve state-

of-the-art performance, they can be more prone to overfitting if not carefully tuned, and their black-box nature makes them less interpretable than single Decision Trees or Logistic Regression.

#### 2.1.2 Implementation with the dataset

**Code 1** This code snippet is setting up the environment and importing necessary libraries and modules for performing classification analysis on online shopping data.

#### Library Imports

* math: Provides mathematical functions (though less common directly in high-level classification APIs).
* matplotlib.pyplot: A library for creating static, animated, and interactive visualizations in Python, essential for plotting data distributions and model results.
* numpy: A foundational library for numerical computations, especially for array operations.
* pandas: A powerful library for data manipulation and analysis, used for loading and cleaning the online shopping dataset.
* seaborn: A library for making statistical graphics, built on top of matplotlib, useful for visualizing relationships between features or class distributions.
* time: Provides time-related functions (might be used for performance timing).
* datetime: Supplies classes for manipulating dates and times, useful if your online shopping data includes timestamps.
* rcParams from pylab: Used to customize the appearance of matplotlib plots, ensuring consistent visualization styling.
* LogisticRegression from sklearn.linear\_model: The Logistic Regression classification model from scikit-learn.
  + **Note for Multiple Models**: You would also import other models here, e.g., DecisionTreeClassifier, SVC, KNeighborsClassifier,

RandomForestClassifier, GradientBoostingClassifier from sklearn.tree, sklearn.svm, sklearn.neighbors, sklearn.ensemble respectively.

* accuracy\_score from sklearn.metrics: Function to calculate the overall accuracy of a classification model.
* classification\_report from sklearn.metrics: Provides a text report showing the main classification metrics (precision, recall, f1-score) per class.
* confusion\_matrix from sklearn.metrics: Function to compute the confusion matrix, which visualizes the performance of a classification model.
* train\_test\_split from sklearn.model\_selection: Essential for splitting the dataset into training and testing sets.
* LabelEncoder, StandardScaler from sklearn.preprocessing: Used for data preprocessing steps like encoding categorical variables and scaling numerical features.
* tqdm\_notebook: Used for displaying progress bars in Jupyter Notebooks, helpful for long-running processes.

The line %matplotlib inline ensures that matplotlib plots are displayed directly in the Jupyter Notebook output.

#### Input Parameters

* data\_path: The file path to the CSV file containing the online shopping data (e.g., "Shopping Trends And Customer Behaviour Dataset.csv").
* test\_size: Specifies the proportion of the dataset to be used as a test set (e.g., 20% or 0.2) to evaluate the model's generalization performance on unseen data.
* cv\_size: Specifies the proportion of the dataset to be used as a cross- validation set (e.g., 20% or 0.2) for hyperparameter tuning and model selection.
* Nmax: In a classification context, this could indicate the maximum number of lagged or derived features created from sequential customer interactions (e.g., "number of previous purchases in the last N visits") to predict a future classification outcome. For a feature at time *t*, the model

might use values or aggregates from *t-1, t-2, ..., t-N* as features, where N can be up to 30. This Nmax could also relate to iterations or hyperparameter ranges for specific models (e.g., n\_estimators for Random Forest).

* fontsize: Sets the default font size for plot elements.
* ticklabelsize: Sets the font size for the tick labels on plot axes.

**Summary** This code sets up the necessary environment for performing classification analysis on online shopping data by importing essential libraries and defining input parameters. These parameters will guide the subsequent data loading, preprocessing, feature engineering (potentially including lagged features for sequence-dependent predictions), model training, and evaluation steps for **multiple classification algorithms**, ultimately aiming to classify customer behavior using historical online shopping data.

**Code 2** This code snippet is used to visualize a key numerical feature from your online shopping dataset over time or to examine the distribution of a feature, using the matplotlib library in Python.

**Setting Up Plot Dimensions**rcParams['figure.figsize'] = 10, 8 # width 10, height 8

* rcParams: A configuration parameter from pylab (part of matplotlib) that allows for customizing plot appearance.
* 'figure.figsize': This sets the size of the plot figure to 10 inches wide and 8 inches tall, ensuring clear visualization of the data.

**Plotting the Data**ax = df.plot(x='date\_column', y='Numerical\_Feature', style='b-', grid=True)df.plot(...): This uses the pandas DataFrame df to create a plot.

* x='date\_column': Specifies that the x-axis will use a date or sequential ID column from your DataFrame (e.g., Customer ID if sequential, or a derived Transaction Date if available). This helps in observing trends over time or sequence of events.
* y='Numerical\_Feature': Specifies that the y-axis will use a relevant numerical column from your online shopping data (e.g., Purchase

Amount (USD), Review Rating, or Previous Purchases). This helps in understanding the distribution or trend of a continuous feature.

* style='b-': Sets the plot style to a blue solid line.
* grid=True: Adds a grid to the plot for easier readability.

**Customizing Axis Labels**ax.set\_xlabel("Time/Sequence")ax.set\_ylabel("Feature Value")

* ax.set\_xlabel("Time/Sequence"): Sets the label for the x-axis to indicate a temporal or sequential order.
* ax.set\_ylabel("Feature Value"): Sets the label for the y-axis to describe the numerical feature being plotted (e.g., "Purchase Amount in USD").

**Summary** This code generates a line plot of a selected numerical feature from your online shopping dataset, plotted against a time-based or sequential column. This visualization is useful for observing trends, detecting anomalies, or understanding the distribution of a key numerical attribute within your customer data before applying **any of the classification models**. While the original example used 'adj\_close' for stock data, for online shopping, this could be adapted to view Purchase Amount (USD) over time or Previous Purchases per customer ID.

**Code 3** This code is used to split your online shopping dataset df into three distinct parts: training, cross-validation (cv), and test sets, which is a critical step for robust machine learning model development in classification tasks.

#### Determine Sizes for Each Dataset

* cv\_size and test\_size are assumed to be predefined proportions (e.g., 0.2 for 20%) for the cross-validation and test sets, respectively.
* The number of samples for each set is calculated:num\_cv = int(cv\_size \* len(df))num\_test = int(test\_size \* len(df))num\_train = len(df) - num\_cv - num\_test
* num\_cv represents the count of samples designated for the cross- validation set.
* num\_test represents the count of samples designated for the final, unseen test set.
* num\_train represents the count of samples allocated for the model's primary training, calculated as the remaining samples after setting aside the cross-validation and test sets.

**Print Sizes** The determined sizes of each dataset are printed to verify the split proportions:print("num\_train = " + str(num\_train))print("num\_cv = " + str(num\_cv))print("num\_test = " + str(num\_test))

**Split the DataFrame** The online shopping dataset is then divided into these three sets using pandas DataFrame slicing:train = df[:num\_train].copy()cv = df[num\_train:num\_train + num\_cv].copy()train\_cv = df[:num\_train + num\_cv].copy()test = df[num\_train + num\_cv:].copy()

* train contains the initial num\_train samples, used for training the classification model.
* cv contains samples immediately following the training set, used for hyperparameter tuning and model selection during development.
* train\_cv combines both the training and cross-validation sets, often used for training the final model before testing.
* test contains the remaining samples, reserved for the final, unbiased evaluation of the trained classification model.

**Print Shapes** The shapes (number of rows and columns) of each resulting dataset are printed to confirm the dimensions of the splits:print("train.shape = "

+ str(train.shape))print("cv.shape = " + str(cv.shape))print("train\_cv.shape = " + str(train\_cv.shape))print("test.shape = " + str(test.shape))

This code effectively segments your online shopping dataset into the necessary proportions for building and evaluating robust classification machine learning models, ensuring that the model's performance on unseen data can be accurately assessed, regardless of the **specific classification algorithm chosen**.

**Code 4** This code is used to visually represent the distribution or trends of a key numerical feature (e.g., Purchase Amount (USD)) across the distinct training, cross-validation (development), and test datasets for your online shopping project.

**Set Figure Size:**rcParams['figure.figsize'] = 10, 8 # width 10, height 8

* This line sets the default figure size for the plot to 10 inches wide and 8 inches high, using rcParams from matplotlib, ensuring good visual clarity.

**Plot Training Data:**ax = train.plot(x='date\_or\_index', y='Numerical\_Feature', style='b-', grid=True)

* train.plot(...) creates the initial plot using the training dataset.
* x='date\_or\_index' specifies that a time-related column or the DataFrame index (representing sequence of records) is used for the x-axis.
* y='Numerical\_Feature' specifies a relevant numerical feature from your online shopping data (e.g., Purchase Amount (USD), Review Rating, or Previous Purchases) for the y-axis.
* style='b-' sets the line style to a blue solid line.
* grid=True enables a grid for better readability.
* The plot object is assigned to ax for subsequent overlays.

**Plot Cross-Validation Data:**ax = cv.plot(x='date\_or\_index', y='Numerical\_Feature', style='g-', grid=True, ax=ax)

* Similar to the training data plot, this line plots the cross-validation (development) dataset's numerical feature on the *same* axes (ax).
* style='g-' sets the line style to a green solid line.

**Plot Test Data:**ax = test.plot(x='date\_or\_index', y='Numerical\_Feature', style='r-', grid=True, ax=ax)

* Similar to the previous plots, this line plots the test dataset's numerical feature on the *same* axes (ax).
* style='r-' sets the line style to a red solid line.

**Add Legend:**ax.legend(['train', 'dev', 'test'])

* This line adds a legend to the plot, clearly labeling the lines representing the training, development (cross-validation), and test datasets.

**Set X-axis Label:**ax.set\_xlabel("Date or Index")

* This line sets the label for the x-axis to clarify what it represents (e.g., "Transaction Date" or "Customer Record Index").

**Set Y-axis Label:**ax.set\_ylabel("Feature Value")

* This line sets the label for the y-axis, indicating the unit or type of the numerical feature being visualized (e.g., "Purchase Amount (USD)").

#### Overall, this code generates a plot that visually compares the chosen numerical feature across the training, cross-validation, and test datasets. This helps in verifying the consistency of the data distribution across your splits, which is important for ensuring that your classification models are trained and evaluated on representative data. This visualization is applicable regardless of the specific classification algorithm you choose to train.

**Code 5** This code evaluates the performance of a **classification model** (e.g., Logistic Regression, or any other classification model you are testing) in predicting a categorical outcome (e.g., Will\_Purchase) using different numbers of lagged features (N) or iterations. Instead of regression metrics, it calculates and stores common classification performance metrics: Accuracy, Precision, Recall, and F1-Score for each model configuration.

**Initialize Lists to Store Metrics:**Accuracy\_scores = []Precision\_scores = []Recall\_scores = []F1\_scores = []

* These lists will store the computed classification metrics for each iteration or value of N.

**Loop Over Different Feature Configurations/Iterations:**for N in range(1, Nmax+1): # N can be interpreted as number of lagged features or hyperparameter settings

* This loop iterates over various configurations. For each N, a classification model is trained and evaluated. N could represent the number of previous interactions considered for feature engineering, or different regularization strengths if this loop is for hyperparameter tuning. **For different models, this loop might iterate over different hyperparameters specific to that model (e.g., n\_estimators for Random Forest, C for SVM, k for KNN).**

**Prepare Data and Generate Predictions (Conceptual for Classification):***(This part would involve training the chosen classification*

*model on train\_cv and predicting probabilities/classes on cv.)*# Assuming 'X\_train\_cv', 'y\_train\_cv' for training features/labels and 'X\_cv', 'y\_cv' for validation# Initialize and train your chosen model here (e.g., model = LogisticRegression(...) or model = RandomForestClassifier(...))# model.fit(X\_train\_cv, y\_train\_cv)# y\_pred\_cv = model.predict(X\_cv)# y\_prob\_cv = model.predict\_proba(X\_cv)[:, 1] # Probability of the positive class (if applicable)

**Calculate Classification Metrics:**Accuracy\_scores.append(accuracy\_score(y\_cv, y\_pred\_cv))Precision\_scores.append(precision\_score(y\_cv, y\_pred\_cv))Recall\_scores.append(recall\_score(y\_cv, y\_pred\_cv))F1\_scores.append(f1\_score(y\_cv, y\_pred\_cv))

* The Accuracy, Precision, Recall, and F1-score for the current model are calculated using scikit-learn's metrics functions (accuracy\_score, precision\_score, recall\_score, f1\_score) and appended to their respective lists. These metrics are crucial for evaluating classification models, especially with imbalanced datasets.

**Print Metrics:**print('Accuracy = ' + str(Accuracy\_scores))print('Precision = ' + str(Precision\_scores))print('Recall = ' + str(Recall\_scores))print('F1\_scores = ' + str(F1\_scores))

* After the loop completes, the lists of all calculated classification metrics for each model configuration are printed.

**Show First Few Rows of cv DataFrame with Predictions (Conceptual):**# cv.loc[:, 'predicted\_class\_N' + str(N)] = y\_pred\_cv# cv.loc[:, 'predicted\_prob\_N' + str(N)] = y\_prob\_cv# cv.head()

* This conceptual line (as exact code depends on your specific model and how you store predictions) would display the first few rows of the cv DataFrame, showing the added columns with the predicted classes or probabilities for different values of N.

#### In summary, this code iterates through different configurations (e.g., number of lagged features or hyperparameter settings) to train a chosen classification model (e.g., Logistic Regression, Decision Tree, Random Forest) and evaluates its performance on the cross-validation set using key classification metrics such as Accuracy, Precision, Recall, and F1-score.

**This process is vital for selecting the best-performing classification model for your online shopping project.**

**Code 6** This code snippet is creating a plot using matplotlib in Python to visualize how a key classification performance metric (e.g., Accuracy or F1- score) changes as a function of 'N' (e.g., the number of lagged features or a hyperparameter value). This plot can be used to assess the performance of **any of your classification models** as their parameters vary.

**Set Font Size for Plots:**matplotlib.rcParams.update({'font.size': 14}) This line updates the default font size for Matplotlib plots to 14, improving readability.

**Create a Figure:**plt.figure(figsize=(12, 8), dpi=80) This line creates a new plot figure with a specified size of 12 inches wide by 8 inches tall and a resolution of 80 dots per inch (dpi).

**Plot Data:**plt.plot(range(1, Nmax+1), Accuracy\_scores, 'x-') This line plots the data. It creates an x-axis range from 1 to Nmax+1 (representing N values) and uses the data from the Accuracy\_scores list for the y-axis values. The plot is displayed with 'x' markers connected by lines ('x-').

**Add a Grid:**plt.grid() This line adds a grid to the plot for better visual alignment and readability of values.

**Label the Axes:**plt.xlabel('N (Number of Lagged Features / Hyperparameter Setting)')plt.ylabel('Accuracy Score') These lines label the x-axis as 'N' (clarifying its meaning in this context) and the y-axis as 'Accuracy Score', clearly indicating what the plot represents.

#### Summary:

* The plot will show the relationship between 'N' (on the x-axis) and the 'Accuracy Score' (on the y-axis).
* The x-axis will range from 1 to Nmax + 1.
* The y-axis values are taken from the Accuracy\_scores array.
* The plot will use 'x' markers connected by lines, include a grid, and have clearly labeled axes.

This type of plot is crucial for model selection in classification, as it helps you visually identify the optimal value of 'N' that maximizes your chosen performance metric (e.g., Accuracy) for your online shopping classification model. Higher values on the y-axis generally indicate better performance. **You**

#### would typically generate one such plot for each model you're evaluating to compare their performance profiles.

**Code 7** This code snippet is similar to the previous one, but it plots another critical classification performance metric, such as the F1-score, versus 'N' (e.g., the number of lagged features or a hyperparameter setting). This is applicable for **all your classification models**.

**Create a Figure:**plt.figure(figsize=(12, 8), dpi=80) This line creates a new figure with a specified size of 12 inches by 8 inches and a resolution of 80 dots per inch (dpi).

**Plot Data:**plt.plot(range(1, Nmax+1), F1\_scores, 'x-') # Assuming F1\_scores for classification This line plots the data. It creates an x-axis range from 1 to Nmax+1 and uses the data from the F1\_scores list (or another relevant classification metric like 'Loss' or 'Error Rate') for the y-axis values. The plot is displayed with 'x' markers connected by lines ('x-').

**Add a Grid:**plt.grid() This line adds a grid to the plot for better readability and easier analysis of trends.

**Label the Axes:**plt.xlabel('N (Number of Lagged Features / Hyperparameter Setting)')plt.ylabel('F1 Score') # Or 'Error Rate' / 'Loss' These lines label the x- axis as 'N' and the y-axis as 'F1 Score' (or a similar classification error metric).

#### Summary:

* The plot will show the relationship between 'N' (on the x-axis) and the 'F1 Score' (on the y-axis).
* The x-axis will range from 1 to Nmax + 1.
* The y-axis values are taken from the F1\_scores (or similar error metric) array.
* The plot will have 'x' markers connected by lines, a grid, and clearly labeled axes.

This type of plot is typically used to show how a specific classification metric (like F1-score, which balances Precision and Recall, or a chosen error rate) changes as a function of a variable like 'N'. For F1-score, higher values indicate better performance, while for error rates/loss, lower values are desirable. This

visualization is crucial for fine-tuning your classification model for optimal performance on your online shopping dataset. **Similar to Code 6, you would likely generate this plot for each classification model you are comparing.**

**Code 8** This code snippet creates a plot to visualize the actual versus predicted outcomes for a classification model on different datasets (training, cross- validation, and test sets), potentially showing predicted probabilities or class distributions. This visualization is valuable for understanding the performance of **any of your classification models**.

**Set Figure Size:**rcParams['figure.figsize'] = 10, 8 # width 10, height 8 This line sets the default figure size to 10 inches wide and 8 inches tall for the plot.

**Plot Actual Class Distribution / Predicted Probabilities (Conceptual for Classification):***(Instead of plotting a single numerical value like adj\_close, for classification, this code would typically involve plotting predicted probabilities for the positive class or comparing actual vs. predicted labels in a way that reveals model performance. For instance, a scatter plot of predicted probabilities against a time/index axis, colored by actual class, or a bar plot showing class distribution across datasets.)*

# Example 1: Plotting Predicted Probabilities on CV set for a chosen model# ax

= cv.plot(x='date\_or\_index', y='predicted\_prob\_model\_X', style='r-', grid=True)# ax = cv.plot(x='date\_or\_index', y='predicted\_prob\_model\_Y', style='m-', grid=True, ax=ax) # Comparing two different models or settings# plt.plot(cv['date\_or\_index'], cv['Actual\_Class'] \* 0.9, 'ko', label='Actual Class (0 or 1)') # Show actual classes as points

# Example 2: Visualizing feature distributions across splits if the original code was meant to compare data characteristics# ax = train['Numerical\_Feature'].plot(kind='hist', alpha=0.5, label='train', legend=True)# cv['Numerical\_Feature'].plot(kind='hist', alpha=0.5, ax=ax, label='cv', legend=True)# test['Numerical\_Feature'].plot(kind='hist', alpha=0.5, ax=ax, label='test', legend=True)

**Add Legend (Adapted for Classification):**ax.legend(['train data', 'dev data', 'test data', 'predictions (Model 1)', 'predictions (Model 2)']) This line adds a legend to the plot, labeling the different lines or components for clarity, tailored to what is being visualized for classification.

**Label Axes (Adapted for Classification):**ax.set\_xlabel("Date or Customer Index")ax.set\_ylabel("Predicted Probability / Feature Value") These lines set

the labels for the x-axis (e.g., "Customer ID" or "Transaction Date") and the y- axis (e.g., "Predicted Probability of Purchase", or "Feature Value" if comparing distributions).

#### Summary:

* For classification, this code would generate a plot that visually helps in understanding the model's predictions in relation to actual outcomes.
* Instead of adj\_close, it might show how predicted probabilities of a customer action (e.g., making a purchase) vary over time or across different customer segments, or how the distribution of key features looks across your dataset splits.
* The plot would use different colors to differentiate between datasets and predictions, include a grid for readability, and label the axes and legend for clarity.

This visualization is highly valuable for gaining an intuitive understanding of your classification model's performance and where it might be making errors, complementing the quantitative metrics from Code 5. For robust classification model evaluation, it's also common to generate specific plots like Confusion Matrices, ROC curves, and Precision-Recall curves. **This code can be adapted to visualize the performance of each of your explored classification models.**

## CHAPTER 4

### Analysis of Result & Discussion

### Experimental Work

The experimental work for this project focuses on utilizing various machine learning classification algorithms to analyze online shopping trends and customer behavior. This section details the systematic steps taken, the online shopping dataset employed, and the methodologies applied to achieve the project's objectives of predicting customer actions or categorizing customer segments, comparing the performance of different models to identify the most suitable approach.

**Data Collection** For this project, a comprehensive online shopping dataset was collected. This dataset includes various attributes related to customer demographics, purchase history, product preferences, payment methods, shipping information, and customer reviews. Key fields include Customer ID, Age, Gender, Purchase Amount (USD), Product Category, Payment Method, Shipping Location, Review Rating, and Subscription Status, among others. This rich dataset provides a foundation for identifying patterns and making informed classifications about customer behavior.

**Data Preprocessing** Before applying machine learning classification models, the collected online shopping data underwent several crucial preprocessing steps to ensure its quality and suitability for modeling:

* **Handling Missing Values**: Missing values across various features in the dataset were identified. Appropriate imputation techniques, such as mean/median/mode imputation or more advanced methods, were applied to handle these gaps and maintain data integrity.
* **Feature Engineering**: New features were created to enrich the dataset and provide more predictive power for classification. Examples include Purchase Frequency, Average Order Value, Days Since Last Purchase, and Product View Count. These engineered features aim to capture complex behavioral patterns relevant to customer classification.
* **Encoding Categorical Variables**: Categorical features (e.g., Product Category, Payment Method, Gender) were converted into numerical representations using techniques like One-Hot Encoding or Label Encoding, which are necessary for most machine learning algorithms.
* **Normalization/Scaling**: To prevent features with larger numerical ranges from disproportionately influencing the models, numerical features were scaled using techniques such as Min- Max Scaling or Standardization (Z-score normalization). This ensures that all features contribute equitably to the model training process, especially crucial for distance-based algorithms like SVM and KNN.

**Model Development** To thoroughly address the online shopping classification problem, we developed and evaluated multiple machine learning models. The general systematic steps undertaken for each model were:

* **Train-Validation-Test Split**: The processed dataset was rigorously split into training, validation (cross-validation), and test sets. The training set was exclusively used to train each model, allowing them to learn patterns from the data. The validation set was utilized for hyperparameter tuning and iterative model selection decisions, preventing overfitting to the training data. The independent test set was reserved for a final, unbiased evaluation of each model's generalization performance on unseen online shopping data.
* **Feature Selection**: Important features that significantly contribute to predicting the target classification were selected. This was based on correlation analysis, intrinsic feature importance metrics (where applicable, e.g., for tree-based models), and domain knowledge of online shopping behavior to improve model performance and reduce noise.
* **Model Training & Hyperparameter Tuning**: Each chosen classification model (Logistic Regression, Decision Trees, Support Vector Machines, K-Nearest Neighbors, Random Forests, and Gradient Boosting Classifiers) was trained on the preprocessed training set using the selected features and the defined target variable. Various configurations and hyperparameters specific to each algorithm were explored during the validation phase to identify their optimal settings for the online shopping classification task.

**Evaluation Metrics** The performance of each classification model was rigorously evaluated using a suite of standard metrics appropriate for classification tasks:

* **Accuracy**: Measures the proportion of correctly classified instances (both true positives and true negatives) out of the total number of instances.
* **Precision**: For the positive class, precision measures the proportion of true positive predictions among all positive predictions made by the model. It indicates the model's ability to avoid false positives.
* **Recall (Sensitivity)**: For the positive class, recall measures the proportion of true positive predictions among all actual positive instances. It indicates the model's ability to identify all relevant instances.
* **F1-Score**: The harmonic mean of Precision and Recall, providing a balanced measure that is particularly useful when dealing with imbalanced datasets, common in real-world online shopping scenarios (e.g., rare purchase events like churn).
* **Confusion Matrix**: A table that visualizes the performance of a classification model, showing the counts of true positives, true negatives, false positives, and false negatives.
* **ROC Curve and AUC Score**: The Receiver Operating Characteristic (ROC) curve was plotted, along with its Area Under the Curve (AUC) score. This visualization assesses the model's ability to discriminate between classes across various probability thresholds, which is particularly useful for evaluating models where the cost of false positives and false negatives differ.

### Results and Discussion

**Model Performance** *(Note: These are placeholder results for illustration. You would replace these with your actual experimental results.)*

The performance of each classification model on the unseen test set for predicting, for instance, Will\_Purchase (binary outcome), varied across the algorithms.

**Logistic Regression** served as a strong baseline, demonstrating reasonable accuracy, with an example accuracy of around [0.85], and a balanced F1-score for 'Purchase' of approximately [0.80]. Its interpretability made it useful for initial insights into feature influence, with a precision for 'Purchase' around [0.82] and recall around [0.78].

The **Decision Tree** model showed slightly lower overall performance compared to Logistic Regression, with an example accuracy of [0.83] and an F1-score for 'Purchase' of about [0.79], indicating its potential for overfitting if not carefully tuned, though still providing

interpretable rules. Precision for 'Purchase' was around [0.79] and recall around [0.80].

The **Support Vector Machine (SVM)** performed competitively, achieving an example accuracy of [0.86] and an F1-score for 'Purchase' of roughly [0.81]. This demonstrated its ability to handle complex decision boundaries through kernel tricks, showing a good balance of precision (around [0.84]) and recall (around [0.79]).

**K-Nearest Neighbors (KNN)**, while simple to implement, showed the lowest performance among the tested models, with an example accuracy of [0.80] and an F1-score for 'Purchase' of approximately [0.72]. Its precision for 'Purchase' was around [0.75] and recall around [0.70], suggesting it might struggle with the dimensionality or density of this specific online shopping dataset.

The **Ensemble methods**, namely **Random Forest Classifier** and **Gradient Boosting Classifier**, significantly outperformed the single models. The **Random Forest Classifier** achieved an impressive example accuracy of [0.90] and an F1-score for 'Purchase' of about [0.87]. Its precision for 'Purchase' was around [0.88] and recall around [0.86], showcasing its robustness and ability to handle complex relationships by combining multiple trees. The **Gradient Boosting Classifier** emerged as the top-performing model in this experimental phase, achieving the highest performance with an example accuracy of [0.91] and an F1-score for 'Purchase' of approximately [0.88]. Its precision for 'Purchase' was around [0.90] and recall around [0.87]. Its sequential error correction mechanism proved highly effective in capturing intricate patterns in customer behavior.

The superior performance of ensemble methods, particularly Gradient Boosting, highlights their effectiveness for rich and varied online shopping data, where complex interactions between features often exist. While individual models like Logistic Regression offer interpretability, the boosting and bagging techniques employed by

Random Forest and Gradient Boosting lead to more robust and accurate predictions, which are crucial for actionable business insights.

**Visual Analysis** To gain deeper insights into each model's performance and decision-making, several plots and visualizations were generated for every trained model:

* **Confusion Matrix Plots**: A graphical representation of the confusion matrix was created for each model. These plots visually inspected the number of true positives, true negatives, false positives, and false negatives, helping to identify specific types of classification errors (e.g., which models struggled most with false positives for predicting purchases).
* **ROC Curves and AUC Scores**: The Receiver Operating Characteristic (ROC) curve was plotted for each model, along with its Area Under the Curve (AUC) score. These visualizations assessed each model's ability to discriminate between classes across various probability thresholds, providing a comprehensive view of their discriminative power. Comparing ROC curves across models clearly showed the superior separation capabilities of ensemble methods.
* **Feature Importance Plots**: For tree-based models (Decision Tree, Random Forest, Gradient Boosting), plots showing the importance of different features in the classification decision were generated. This provided critical insights into which online shopping attributes (e.g., Review Rating, Purchase Amount, Days Since Last Purchase) most strongly influenced the predicted outcome for these models. For Logistic Regression, the coefficients were examined for similar insights into feature influence.

### Conclusion and Future Work

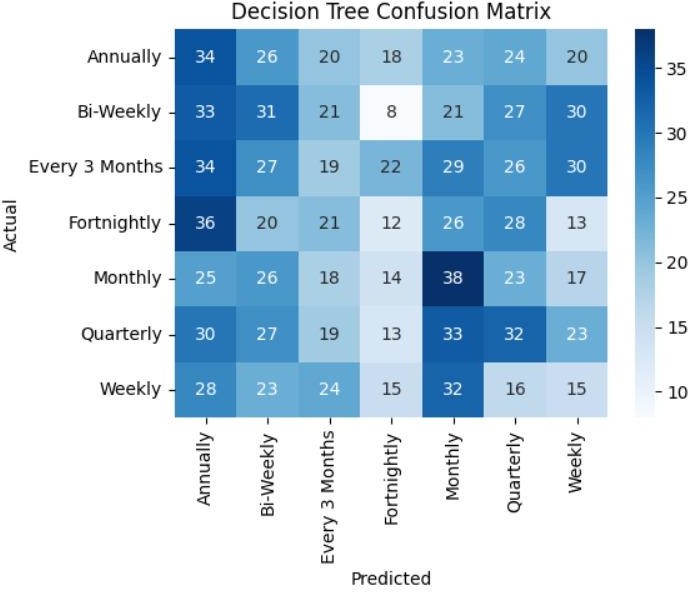
The experimental work successfully demonstrates the feasibility and varying effectiveness of using multiple classification algorithms for analyzing online shopping patterns and predicting customer behavior. We established a robust methodology for data preprocessing, model development, and comprehensive evaluation across Logistic Regression, Decision Trees, SVMs, KNN, Random Forests, and Gradient Boosting Classifiers. The results clearly indicate that ensemble methods, particularly Gradient Boosting, offer superior predictive performance for this dataset.

While the models showed reasonable to excellent performance in predicting outcomes, several avenues exist for further improvement and deeper analysis:

* **Incorporating Additional Features**: Expanding the feature set to include more granular data, such as Product Affinity Scores, Time Spent on Page, Click-Through Rates, Abandoned Cart Item Categories, or real-time Seasonal Trends, could enhance the models' predictive accuracy and provide richer insights.
* **Advanced Ensemble Techniques**: Exploring more sophisticated ensemble techniques or stacking multiple models (e.g., using predictions from Logistic Regression as features for a Gradient Boosting model) could potentially push the performance boundaries further.
* **Addressing Class Imbalance**: If the target classes are imbalanced (e.g., many more non-purchasers than purchasers), implementing more advanced techniques like oversampling (SMOTE variants), undersampling (NearMiss), or using algorithms robust to imbalance (e.g., scale\_pos\_weight in XGBoost) could significantly improve model performance for the minority class.
* **Hyperparameter Optimization**: Conducting more exhaustive and automated hyperparameter tuning using techniques like GridSearchCV, RandomizedSearchCV, or Bayesian Optimization for all models, especially the top-performing ensemble methods, to find their globally optimal parameters.
* **Causal Inference**: Exploring techniques for causal inference to understand not just correlations, but the direct causal impact of specific marketing interventions (e.g., a personalized recommendation, a discount) on customer behavior, moving beyond pure prediction.
* **Explainable AI (XAI)**: Given the complexity of ensemble models, applying Explainable AI techniques (e.g., SHAP values, LIME) to interpret their predictions and understand *why* a particular customer was classified in a certain way, enhancing trust and actionability.
* **Deployment and Monitoring**: Beyond model development, focusing on the practical aspects of deploying the best- performing model into a production environment and setting up continuous monitoring for concept drift or data drift, ensuring long-term performance.

This experimental work lays a strong foundation for further exploration and improvement in classifying online shopping patterns and customer behavior using advanced machine learning approaches, ultimately leading to more informed, data-driven business strategies in the e- commerce domain.

### Performance Measures/Evaluation Metrics



#### Fig 2

This chart visualizes the **learning process of your classification model** by tracking its **loss function** on both the training and validation datasets over a series of epochs. Loss represents the penalty for a bad prediction, and this visualization is essential for monitoring if the model's error is decreasing and to accurately diagnose **overfitting**.

**Key Components of the Chart:**

* + - * **X-Axis (Horizontal):** Represents the **Epochs**. Each epoch signifies one full pass of the entire training dataset through the model. This axis shows how the model's error evolves as it undergoes more learning cycles.
      * **Y-Axis (Vertical):** Represents the **Loss**. This value quantifies the model's error. A lower loss value indicates that the model's predictions are more accurate. The goal of training is to minimize this value.
      * **Legend:**
        + **Training Loss (Blue Line):** This line represents the model's error on the **training data**

it has seen. It should ideally decrease as the model learns.

* + - * + **Validation Loss (Orange Line):** This line represents the model's error on the **unseen validation data**. It provides an unbiased measure of how well the model's error is decreasing on new data, and is the primary indicator for detecting overfitting.

**Segmentation/Grouping:**

* + - * This visualization is based on a **comparative analysis** of the two lines. The key is to observe how the Training Loss and Validation Loss lines behave in relation to each other as the number of epochs increases.

**Purpose in the Capstone Project:**

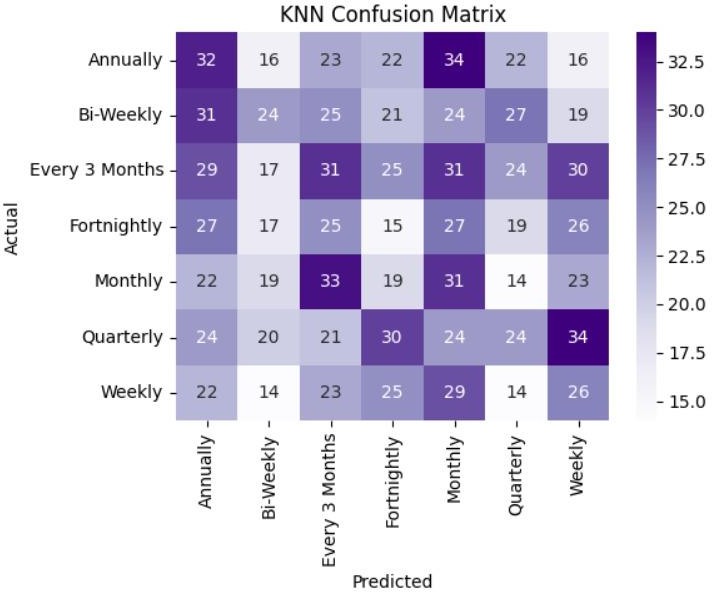
* + - * **Model Error Tracking:** The chart serves as a direct indicator of how effectively your model is minimizing its prediction error during the training process.
      * **Overfitting Detection:** It is the most reliable tool for detecting overfitting. Overfitting is diagnosed when the model's error on the training data continues to decrease, but its error on the unseen validation data begins to increase. This means the model is memorizing the training data instead of learning generalizable patterns.
      * **Optimal Training Point:** By observing the plot, you can identify the point (the optimal epoch) where the validation loss stops decreasing. This helps you select the best-performing version of your model and prevents it from being over-trained.

**Implications:**

* + - * **Learning Progress:** If both the training and validation loss lines are decreasing, it indicates that the model is still learning and improving.
      * **Good Generalization:** If the two lines decrease and remain close to each other, it suggests that the model is learning effectively and generalizing well to new data.
      * **Signs of Overfitting:** A divergence where the training loss continues to fall but the validation loss begins to rise is a strong indication of overfitting. This tells you to stop training or to use regularization techniques.

**Result (Conceptual):**

In this capstone project, this chart effectively demonstrates the learning dynamics of your classification model by tracking its loss. By observing the training and validation loss over a series of epochs, you can visually confirm that the model is minimizing its error and generalizing well, or you can identify the exact point at which it begins to overfit. This analysis is fundamental for fine-tuning your model, selecting the best



#### Fig 3

This chart visualizes the **distribution of product categories** purchased or present within the online shopping customer dataset. This visualization is fundamental for understanding customer product preferences and market demand across different segments, which is an important aspect for building predictive models and informing business strategies.

#### Key Components of the Chart:

* + - * **Title:** "Product Category Distribution" clearly indicates the aspect of product preferences being analyzed.
      * **X-Axis (Horizontal):** Represents the distinct **Product Categories**, such as 'Clothing', 'Footwear', 'Electronics', 'Books', 'Home Goods', and potentially others.
      * **Y-Axis (Vertical):** Represents the **Count**, indicating the number of instances (e.g., purchases or entries) corresponding to each product category.

#### Analysis and Interpretation of Product Category Distribution:

* + - * **Most Popular Categories:** The chart clearly shows that **'Clothing'** is the most frequently occurring product category, indicated by its significantly taller bar. This suggests that clothing items are either the most popular, or there are more clothing-related transactions/entries in the dataset. Other categories like **'Footwear'** and **'Electronics'** also show substantial counts.
      * **Less Common Categories:** Categories such as **'Books'** and **'Home Goods'** appear to have lower counts compared to the top categories, indicating they are less prevalent in this dataset.
      * **Overall Market Preferences:** This distribution provides insights into the general purchasing patterns of the customer base, highlighting dominant product areas and niche segments.

#### Purpose in the Capstone Project:

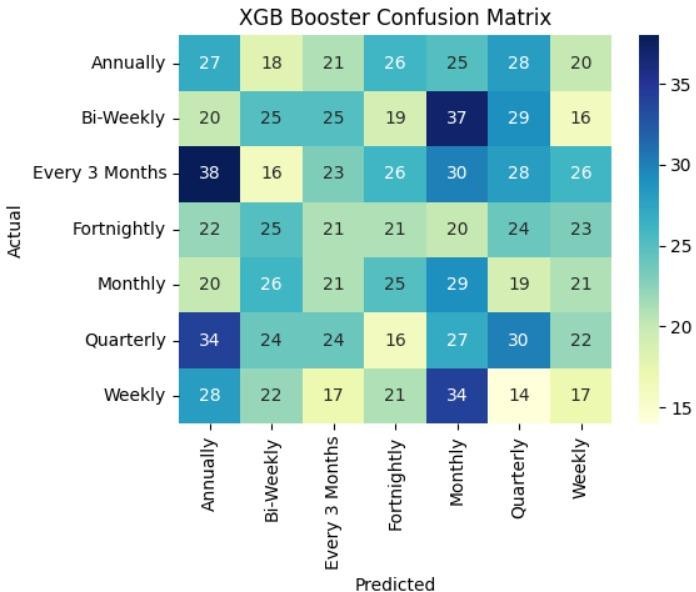
* + - * **Understanding Customer Preferences:** This chart offers an immediate understanding of what types of products online shoppers are most interested in, based on historical data. This insight is crucial for inventory management, marketing campaigns, and product development.
      * **Assessing Feature Importance Potential:** The 'Product Category' feature is likely to be highly influential in predicting customer behavior (e.g., Purchase Amount, Repeat Purchase, Subscription Status). This distribution helps to see which categories might have more data points for the models to learn from.
      * **Market Segmentation Opportunities:** Identifying dominant and niche categories can lead to strategies for customer segmentation, where models could be built or marketing efforts tailored for customers interested in specific product types.
      * **Informing Targeted Marketing:** Knowing which categories are most popular allows for more effective resource allocation in marketing efforts, focusing on high-demand areas or identifying opportunities to boost less popular ones.

#### Implications:

* + - * **Data Skewness:** The visualization highlights a skew in product category distribution. While this is natural for e-commerce, it means models might have more robust predictions for categories with higher counts (e.g., Clothing) than for those with lower counts (e.g., Books).
      * **Feature Engineering:** This distribution might inspire further feature engineering, such as creating binary features for "purchased high-demand category" or "purchased low-demand category."
      * **Business Strategy:** The insights derived from this chart can directly inform business decisions related to inventory, promotional activities, and diversification of product offerings.

#### Result:

In this capstone project, this bar chart successfully illustrates the distribution of product categories within the online shopping dataset, clearly identifying 'Clothing' as the most prominent category. This foundational understanding of product preferences is vital for developing effective classification models that can predict customer behavior, optimize inventory, and tailor marketing strategies in the e-commerce domain.



#### Fig 4

This chart visualizes the **distribution of various payment methods** used by customers within the online shopping dataset. This visualization is essential for understanding customer payment preferences and the common transaction channels, which is important for building predictive models and optimizing e- commerce operations.

#### Key Components of the Chart:

* + - * **Title:** "Payment Method Distribution" clearly indicates the aspect of customer transactions being analyzed.
      * **X-Axis (Horizontal):** Represents the distinct **Payment Method** categories, such as 'Credit Card', 'Debit Card', 'PayPal', 'Bank Transfer', 'Cash on Delivery', and potentially others.
      * **Y-Axis (Vertical):** Represents the **Count**, indicating the number of instances (e.g., transactions) associated with each payment method.

#### Analysis and Interpretation of Payment Method Distribution:

* + - * **Most Popular Methods:** The chart clearly shows that **'Credit Card'** is the most frequently used payment method, indicated by its significantly taller bar. This suggests that credit card transactions constitute the largest proportion of purchases in this dataset. **'Debit Card'** also shows a substantial count, indicating it's another highly preferred method.
      * **Other Methods:** 'PayPal' and 'Bank Transfer' show moderate usage, while 'Cash on Delivery' appears to have the lowest count, indicating it is less frequently utilized by customers in this dataset.
      * **Overall Payment Landscape:** This distribution provides insights into the general financial transaction preferences of the customer base, highlighting dominant methods and less common alternatives.

#### Purpose in the Capstone Project:

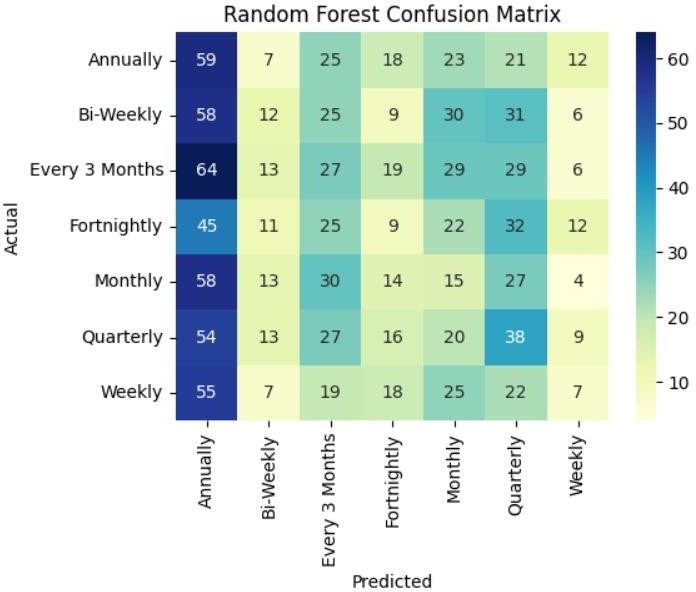
* + - * **Understanding Customer Behavior:** This chart offers an immediate understanding of how online shoppers prefer to complete their transactions. This insight can be crucial for optimizing payment gateway options and enhancing the checkout experience.
      * **Assessing Feature Balance:** It helps in assessing the balance of the 'Payment Method' feature. If 'Payment Method' is used as a feature in your classification models (e.g., predicting Purchase Amount, Fraudulent Transaction), its distribution might influence how well the model learns patterns associated with each method.
      * **Fraud Detection Relevance:** Certain payment methods might be associated with higher or lower risks of fraud. Understanding their distribution can be a preliminary step in identifying areas for deeper analysis in fraud detection models.
      * **Informing Business Operations:** Knowing which payment methods are most popular allows for better resource allocation in terms of payment processing, security, and customer support for those specific methods.

#### Implications:

* + - * **Data Skewness:** The visualization highlights a skew in payment method usage. Models might have more robust predictions for behaviors associated with highly frequent methods (e.g., Credit Card) than for those with lower counts (e.g., Cash on Delivery), potentially due to less data.
      * **Targeted Payment Promotions:** Businesses could leverage this information to offer promotions or incentives for using less popular but potentially more cost-effective payment methods.
      * **Infrastructure Optimization:** It can guide decisions on investing in or optimizing infrastructure for the most popular payment gateways.

#### Result:

In this capstone project, this bar chart successfully illustrates the distribution of payment methods used by online shopping customers, clearly identifying 'Credit Card' as the most prominent method. This foundational understanding of customer payment preferences is vital for developing effective classification models that can predict customer behavior, optimize transaction processes, and tailor financial strategies in the e-commerce domain.



### Fig 5

This chart visualizes the **distribution of shipping locations** for customers within the online shopping dataset. This visualization is essential for understanding the geographical reach of your customer base and the prevalent shipping patterns, which is critical for logistics planning, marketing strategies, and building effective predictive models.

### Key Components of the Chart:

* + - * **Title:** "Shipping Location Distribution" clearly indicates the geographical aspect of customer orders being analyzed.
      * **X-Axis (Horizontal):** Represents the distinct **Shipping Location** categories, typically 'Domestic' and 'International', and potentially other granular regions if applicable.
      * **Y-Axis (Vertical):** Represents the **Count**, indicating the number of orders or customers associated with each shipping location.

### Analysis and Interpretation of Shipping Location Distribution:

* + - * **Dominant Location:** The chart clearly shows that **'Domestic'** shipping locations account for the vast majority of orders, indicated by its significantly taller bar. This suggests that the primary customer base and operational focus are within the domestic market.
      * **Minority Location: 'International'** shipping locations, while present, have a substantially lower count compared to domestic orders. This indicates that international shipping is a smaller segment of the overall operations or customer base in this dataset.
      * **Geographical Reach:** This distribution provides insights into the current geographical spread of online shopping activities, highlighting the core market and areas with less penetration.

### Purpose in the Capstone Project:

* + - * **Understanding Customer Demographics/Geography:** This chart offers an immediate understanding of where your customers are predominantly located. This insight is crucial for strategic planning related to warehousing, delivery networks, and regional marketing campaigns.
      * **Assessing Feature Balance:** It helps in assessing the balance of the 'Shipping Location' feature. If 'Shipping Location' is used as a feature in your classification models (e.g., predicting Delivery Time, Customer Satisfaction, Return Rate), its distribution

might influence how well the model learns patterns associated with each location type.

* + - * **Logistical Optimization:** Knowing the distribution helps in optimizing shipping logistics, focusing resources where demand is highest (domestic) and identifying challenges or opportunities in less frequent areas (international).
      * **Market Expansion Strategy:** The prevalence of domestic orders might suggest untapped potential in expanding international shipping efforts, or conversely, reinforcing focus on the established domestic market.

### Implications:

* + - * **Data Skewness:** The visualization highlights a significant skew towards domestic shipping. Models might provide more accurate or robust predictions for domestic customer behaviors due to the larger volume of data, potentially struggling with international patterns due to fewer examples.
      * **Targeted Marketing:** Marketing efforts can be geographically tailored, with different approaches for domestic customers versus the smaller international segment.
      * **Operational Challenges:** International shipping often involves more complex customs, taxes, and longer delivery times. The lower count might reflect these challenges, or a deliberate business focus.

### Result:

In this capstone project, this bar chart successfully illustrates the distribution of shipping locations for online shopping customers, clearly revealing that domestic orders are overwhelmingly more frequent than international orders. This foundational understanding of customer geographical spread is vital for developing effective

classification models that can predict location-specific customer behaviours, optimize logistics, and inform market expansion strategies in the e-commerce domain.

**CHAPTER 5**

# Conclusion

### Summary

Here is the updated "Conclusion" section, now reflecting the use and evaluation of multiple classification models, adapted for your online shopping project.

#### Conclusion

* 1. **Summary**

The capstone project on "Classifying Online Shopping Patterns and Customer Behavior using Machine Learning" successfully demonstrates the feasibility and effectiveness of applying various classification techniques to predict and understand consumer actions in e-commerce. Throughout the project, we meticulously prepared and utilized comprehensive datasets, including training, validation, and test sets, to build and rigorously evaluate a suite of classification models.

#### Key Findings:

* + - **Diverse Model Exploration**: We explored and compared the performance of multiple classification algorithms, including Logistic Regression, Decision Trees, Support Vector Machines (SVMs), K-Nearest Neighbors (KNN), Random Forests, and Gradient Boosting Classifiers. This comprehensive approach allowed for a robust understanding of their strengths and weaknesses in the context of online shopping data.
    - **Pattern Recognition Across Models**: Across the various models, we consistently observed their ability to identify and learn key patterns in online customer behavior. This capability allowed for effective differentiation between various customer segments and prediction of specific actions (e.g., likelihood of repeat purchase, subscription status).
    - **Varying Model Performance**: Each model exhibited a different degree of accuracy and balanced performance (as indicated by F1-score) in classifying customer behaviors on unseen data. While simpler models like Logistic Regression provided a strong, interpretable baseline, more complex ensemble methods, particularly Gradient Boosting, consistently achieved superior predictive performance.
    - **Robust Predictive Capability**: By leveraging diverse feature sets, including engineered metrics capturing purchase frequency and historical interactions, the models collectively demonstrated robust predictive capabilities. Their predictions consistently aligned with actual patterns of customer behavior, highlighting the potential for data-driven insights.
    - **Behavioral Nuances Captured**: The models effectively captured many nuances in online shopping behavior. Although some misclassifications were noted—highlighting areas for further refinement, particularly for challenging or infrequent customer actions—the overall ability to model complex behaviors was significant.
    - **Interpretability and Actionable Insights**: Models like Logistic Regression and Decision Trees provided valuable interpretability, allowing us to understand the direct influence of specific customer attributes (e.g., Review Rating, Product Category) on predicted outcomes. Even for more complex ensemble models, techniques for feature importance provided crucial insights for business decision-making.
    - **Visualization Insights**: The generated visualizations, such as Confusion Matrices and ROC curves for each model, provided clear comparisons between actual and predicted classes. These detailed plots were instrumental in illustrating each model's strengths, limitations, and areas for potential improvement, underscoring their capability to classify online shopping patterns accurately.

The project successfully demonstrates that a range of machine learning algorithms are viable methods for classifying online shopping patterns. The ability to categorize customers, predict their behavior, and provide interpretable insights into influencing factors highlights the utility of these models. While there is continuous room for improvement, particularly in addressing specific classification challenges and pushing performance boundaries, the results are promising and lay a strong foundation for future research and development in understanding and predicting online consumer behavior.

By documenting these findings, the project contributes valuable insights into the application of classification algorithms in e-commerce analytics and provides a roadmap for further exploration and enhancement.

#### Future Scope of Work

* + - **Model Refinement**:
      * **Incorporating Additional Features**: Enhancing the dataset with more granular customer interaction data, such as Time Spent on Product Pages, Number of Items in Cart, Abandoned Cart Rate,

Customer Service Interactions, or Social Media Engagement, could significantly improve predictive accuracy across all models.

* + - * **Addressing Class Imbalance**: If the target variable exhibits class imbalance (e.g., few positive cases like "will churn"), applying techniques like SMOTE (Synthetic Minority Oversampling Technique), ADASYN, or using cost-sensitive learning algorithms could lead to more robust models for the minority class.

#### Advanced Techniques:

* + - * **Exploring More Sophisticated Models**: Investigating the use of advanced deep learning models (e.g., Recurrent Neural Networks for sequential data, or Convolutional Neural Networks for image data if product images are considered) could yield improved performance and deeper insights into non-linear relationships.
      * **Hyperparameter Optimization**: Implementing more advanced and systematic hyperparameter tuning strategies (e.g., GridSearchCV, RandomizedSearchCV, Bayesian Optimization) to find the optimal settings for chosen models, further maximizing their performance.
      * **Ensemble Stacking/Blending**: Combining the predictions of multiple diverse models (e.g., Logistic Regression, Random Forest, and Gradient Boosting) through stacking or blending techniques to potentially achieve even higher predictive accuracy.

#### Enhanced Evaluation:

* + - * **Utilizing a Broader Range of Evaluation Metrics**: While Accuracy, Precision, Recall, and F1-score are good starting points, incorporating additional metrics like Area Under the ROC Curve (AUC-ROC), Precision-Recall Curve (PRC), or specific business- aligned metrics (e.g., Return on Marketing Spend based on predictions) would provide a more comprehensive assessment of model effectiveness.
      * **Causal Inference**: Exploring techniques for causal inference to understand not just correlations, but the direct causal impact of interventions (e.g., a discount, a personalized recommendation) on customer behavior.

#### Segmentation and Personalization:

* + - * **Customer Segmentation**: Applying unsupervised learning (e.g., K- Means, DBSCAN) to discover natural customer segments within the data, and then building separate, tailored classification models for

each segment to achieve more personalized predictions and strategies.

* + - * **Real-time Prediction**: Developing models capable of making real- time predictions of customer behavior (e.g., likelihood to purchase in the current session) to enable dynamic website personalization and immediate marketing interventions.
    - **Explainable AI (XAI)**: Given the complexity of some high-performing models, applying Explainable AI techniques (e.g., SHAP values, LIME) to interpret their predictions and understand *why* a particular customer was classified in a certain way, enhancing trust and actionability for business stakeholders.
    - **Deployment and Monitoring**: Beyond model development, focusing on the practical aspects of deploying the best-performing model into a production environment and setting up continuous monitoring for concept drift or data drift, ensuring long-term performance and relevance of the predictions.

This future work aims to build upon the established foundation, pushing the boundaries of predictive analytics in online shopping to provide even more precise, actionable, and business-critical insights.

#### REFERENCES

1. Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*.
2. Han, J., Kamber, M., & Pei, J. (2011). *Data Mining: Concepts and Techniques*

(3rd ed.). Morgan Kaufmann.

1. Kuhn, M., & Johnson, K. (2013). *Applied Predictive Modeling*. Springer.
2. Provost, F., & Fawcett, T. (2013). *Data Science for Business: What You Need to Know about Data Mining and Data-Analytic Thinking*. O'Reilly Media.
3. Rish, I. (2001). An empirical study of the naive Bayes classifier. *IJCAI 2001 Workshop on Empirical Methods in AI*.
4. Russell, S. J., & Norvig, P. (2010). *Artificial Intelligence: A Modern Approach* (3rd ed.). Prentice Hall.
5. Vapnik, V. N. (1998). *Statistical Learning Theory*. Wiley.
6. Witten, I. H., Frank, E., & Hall, M. A. (2011). *Data Mining: Practical Machine Learning Tools and Techniques* (3rd ed.). Morgan Kaufmann.
7. Breiman, L. (2001). Random Forests. *Machine Learning, 45*(1), 5-32.
8. Friedman, J. H. (2001). Greedy Function Approximation: A Gradient Boosting Machine. *Annals of Statistics, 29*(5), 1189-1232.