Enhancing Ad Relevance using Data Mining and Ad Profiling

A major project report submitted in partial fulfilment of therequirementfor the award of degree of

Bachelor of Technology

in

Computer Science & Engineering / Information Technology

Submitted by

Akshit Kumar (201569)

Under the guidance & supervision of

Dr. Pankaj Dhiman

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology, Waknaghat, Solan - 173234 (India)

Candidate's Declaration

We hereby declare that the work presented in this report entitled **'Enhancing Ad Relevance using Data Mining and Ad Profiling'** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering/Information Technology** submitted in the Department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology, Waknaghat is an authentic record of my own work under the supervision of **Dr. Pankaj Dhiman** (Assistant Professor(SG), Department of Computer Science & Engineering and Information Technology).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

(Student Signature with Date)
Student Name: Akshit Kumar
Roll No.: 201569

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

(Supervisor Signature with Date)
Supervisor Name: Dr. Pankaj Dhiman
Designation: Assistant Professor(SG)
Department: Computer Science & Engineering and Information Technology
Dated:

Acknowledgment

First and foremost, I want to express my profound thanks and admiration to the all-powerful God for the heavenly gift that has allowed us to continue the project work successfully. My sincere appreciation and responsibilities are owed to Dr.Shubham Goel, my Computer Science and Engineering Department supervisor at Jaypee University of Information Technology in Wakhnaghat. My supervisor has extensive expertise and a strong interest in deep learning, which will be invaluable as we conduct this research. We owe the completion of this project to his boundless patience, intellectual direction, encouragement, vigorous supervision, constructive criticism, helpful counsel, reading of several mediocre draughts and corrections at every level, and so on. In addition, I express my deepest gratitude to everyone who has helped me in any way, whether directly or indirectly, to ensure the success of our project. Considering the specifics of the case, I want to express my gratitude to the numerous members of the staff, both teaching and non-teaching, who have provided me with helpful assistance and made my pursuit possible. Lastly, thank our parents for their ongoing assistance and patience.

Akshit Kumar(201596)

Table of Content

List of Tables

List of Figures

Abstract

Digital advertising is a dynamic landscape, and despite this fact, ensuring ad relevance to the user remains a challenge. The online web industry relies on the keywords the advertisers assign to find the target audience for the ads; however, while efficient enough, this technique has limitations, particularly concerning ad relevance to the content and the user targeted.

This project deals with the problem of keyword mismatching, exploring the impact of broad match keywords, negative keywords and keyword synonyms on the overall relevance of ads on the internet. The project aimed to address these challenges with the help of web content mining, Optical Character Recognition (OCR) and image classification. Using these techniques, the project aims to get ads related to the content on the web page, generating tags for ads and assigning them to the web page according to the content's relevance to the topic presented .

For ads and digital marketing leaders, this project aims to improve the average click rate (CR) on the ads, leading towards growth of the digital marketing industry and free content available on the internet. For those in the ad world, we wrap things up with practical tips to make ads more spot-on. The goal isn't just quick fixes but also encouraging more research and progress in the digital advertising game. Advertisers can create a stronger bond with the right audience in the always-changing online scene by trying these methods and refining strategies.

Chapter 1:Introduction

1.1INTRODUCTION

The online landscape is an ever-expansive realm of information and connectivity in the digital age. As the internet continues evolving, so does how advertisers strive to reach their audiences effectively. In this era of ubiquitous online content and digital advertising, the challenge remains: how can advertisers ensure their messages resonate with and engage their intended viewers in a meaningful and relevant manner? Advertising can be broadly categorized into two primary types: textual advertising, which features text snippets that resemble the content of a web page, and graphical advertising, where ads are visually presented in various formats and sizes. The classification of graphic ads often relies on keyword targeting, where advertisers choose relevant keywords for their image ads. The ad is presented when a user's search query or web page content aligns with these keywords.

However, this approach can be problematic as a single keyword mismatch can lead to the display of irrelevant ads. Several common errors that can lead to such situations include Overly Broad Keywords. As Mayer et al. (2018) highlighted, comprehensive keywords can lead to ads appearing in contexts that are only loosely related to the advertiser's intended topic. The classic example is the keyword "apple," which, when used broadly, might result in ad placements in fruit and technology-related searches. This practice risks ad irrelevance and inefficient resource allocation. Lack of Negative Keywords, Chen and Gao (2015) emphasized the significance of specifying negative keywords to exclude terms or phrases that shouldn't trigger an ad. The absence of negative keywords can result in ad displays in unrelated searches or content, diminishing the precision of ad targeting and relevance to the audience. Match Type Variations, the choice of match types explored by Smith and Johnson (2019), play a critical role in determining the relevance of ad displays. The difference between exact match and phrase match can lead to overly narrow or overly broad ad placements, impacting the accuracy of ad relevance. Keyword synonyms, as discussed by the same authors, introduce complexities in ad targeting. Ad networks may interpret synonymous or closely related terms as matching keywords.

This can lead to ad displays in unexpected contexts, potentially causing confusion and ad irrelevance. Misspellings and Typos In the case of misspellingsand typos, Kovacs and Smith (2017) demonstrated that failing to include common variations of keywords may lead to missed opportunities for relevant ad displays. Conversely, ads may be shown for unrelated searches due to incorrectly spelt keywords. Homographs and Polysemous Words Homographs and polysemous words pose another challenge, as different meanings in various contexts can trigger ad displays that lack relevance due to their failure to consider the context.

The user's intent is often ambiguous, making the interpretation of keywords difficult. Keyword Overload Keyword overload, characterized by an excess of keywords in a single ad group, can overwhelm ad platforms, as highlighted by Zhang et al. (2018). When there are too many keywords, it becomes challenging to match them effectively with relevant content, leading to ad displays in contexts unrelated to any of the keywords. Lack of Location Targeting, failure to include location-specific modifiers or targeting, as discussed by Li and Wang (2020), can result in ad displays in locations that are not relevant to the advertiser's target audience. Location-specific ad targeting is crucial for ensuring ad relevance and efficiency. Language Mismatches: using keywords in a language different from the targeted audience or content language can lead to irrelevant ad displays.

The alignment of keywords with the audience's language and content is vital for ad relevance and engagement. Irrelevant Broad Keywords, using overly general or irrelevant keywords to capture a broad audience, as observed by Chen and Gao (2015), can result in ad displays in contexts that are not directly related to the advertiser's offerings, diminishing the quality of ad relevance.

1.2 PROBLEM STATEMENT

1.2.1 Online advertisements often lack relevance due to limited integration of user behavior and content features, resulting in inefficient targeting and reduced user engagement.

1.2.2 Existing models for ad personalization are limited in capturing complex relationships between user preferences and ad content, leading to suboptimal recommendations.

1.2.3 Irrelevant advertisements affect users by causing frustration and disengagement, while advertisers face reduced click-through rates and diminished returns on investment.

1.2.4 Poor ad relevance leads to decreased user trust, reduced brand visibility, and wasted advertising resources, ultimately impacting digital marketing effectiveness.

1.2.5 We propose leveraging Web Content Mining and User Profiling with machine learning techniques to build a robust predictive model for improving ad relevance and user engagement.

1.3 OBJECTIVES

The successful implementation of this project is anticipated to yield the following outcomes:

1.3.1 Improved User Experience:

Users will be presented with ads that resonate with their preferences and the content they are engaging with, resulting in a more positive browsing experience.

1.3.2 Enhanced Brand Perception:

Brands benefit from increased relevance and alignment, avoiding uncomfortable situations and potential user backlash.

1.3.3 Higher Engagement:

Contextually relevant Advertisements are more likely to be clicked on, leading to higher engagement rates and improved campaign performance.

1.3.4 To Develop a Robust Web Page Classification Model

1.3.5 To Implement Advertisement Profiling using Machine Learning

1.3.5 Devise Advanced Matching Algorithms The primary objective of this project is to enhance the relevance of digital advertisements by developing a comprehensive system that leverages advanced web page classification techniques and ad profiling.

1.4 SIGNIFICANCE AND MOTIVATION OF THE PROJECT WORK

In this ever-evolving digital marketing landscape, the primary goal is clear: crafting ads that authentically resonate with users. Success in the online arena often hinges on advertisers strategically utilizing keywords to target their desired audience. However, this method faces obstacles despite its efficiency, particularly in seamlessly aligning ads with content and user preferences.

Enter our project—a purpose-driven journey to unravel the intricacies of keyword mismatches. We delve into the repercussions of employing broad match keywords, capitalizing on negative keywords, and exploring synonymous alternatives. For the leaders in advertising and the architects of digital marketing, imagine a rise in the average click rate (CR).

A surge that not only propels the digital marketing industry forward but also strengthens the foundation of free content available on the internet. This project serves as a catalyst for change, a spark igniting transformation in the vast online cosmos. To our colleagues in the advertising world, we offer not just quick fixes but an invitation to embark on an innovative journey. Envision a landscape where advertisers forge unbreakable connections with precisely the right audience, a realm where strategies evolve and refine with each passing moment in the online narrative.

Chapter 2: Literature Survey

S. No.	Paper Title	Journal/ Conference Year	Tools/ Techniques/ Dataset	Results	Limitations
1.	Comparing Image Captioning Techniques Using Deep Learning Models,	Mat Journals 2023	CNN, RNN, ResNet, LSTM	Work more Efficiently when running the model using GPU, CNN for encoding and RNN for decoding.	It requires a lot of storage for download.
2.	Effective Web Scraping for Data Science by Victor Ashioya	Data Science andArtificial Intelligence Conference 2023	Beautiful Soup, Scrappy, and Selenium	Dataextracted: country, name, energy source, population	No other methodologies were compared, direct approach.

3.	Scrapping Relevant Images from WebPages Without download, by Erdin c Uzun,1559 1131/2023/88 ART, https://doi.org /10.1145/361 6849	ACM 2023	TC Approach, SVM, KNN, Decision Trees, Random Forest, Ada Boost.	A semi-automatic approach for web data extraction.	The complex algorithm requires a bit extra run time
4.	Machine learning and artificial intelligence use in marketing: a general taxonomy 2022:439457 https://doi.org/10.1 007/s43039022 00057 w	Italian Journal of Marketing (2022)	Various ML and AI algorithms, Description of the effect of the use of ML and AI in marketing	Improve shopping fundamentals, improve consumption experience, Improved decision making	No practical data, all theoretical knowledge.
5.	Lotfi, Chaimaa Sr Srinivasan,Swetha Ertz,Myriam Latr ousImen.(2021) Web Scraping Techniques and Applications. 10.52458/978 93 91842 086 38	SCRS CONFERENC E PROCEEDIN GS ON INTELLIGEN T SYSTEMS 2021	Web Crawlers, Web scraping parsers, Hidden crawlers, Simple HTML parsers, in-built parsers like inChrome and Firefox	By comparing the performance, Scrappy provided better results.	Cannot by pass Security measures that prevent web scrapping.

6.	Website categorization: A formal approach and robustness analysis in the case of e-commerce detection, https://doi.org/10.1016/j.eswa.2019.113001	ELSEVIER 2020	Decision trees, Support vector, neural networks, and more advanced techniques like convolution neural networks (CNNs)or Recurrent neural networks (RNNs).	Scalability, Deployment and ethical considerations are also vital aspects of a website categorization system.	Categorization only done fore-commerce websites, which are limited in number.
7.	Phishing Website Classification and Detection Using Machine Learning, 10.1109/ICCCI48352.2020.9104161	IEEE Xplore 2020	Lexical analysis of URL malicious URL classification and detection Phishing website Classification.	Created a model that detects phishing websites accurately.	Need access to hardware settings, which is considered a security breach
8.	Identifying machine earning Techniques for Classification of target advertising, https://doi.org/10.1016/j.icte.012	ICT Express journal 2020	Behavioral targeting, User profiling, Contextual advertising, Real-time bidding, Click fraud	Targeted online advertising strategies are identified and classified into two broad categories, user-centric and content-centric approaches.	No Data to justify the statement.

Chapter 3: System Development

3.EXISTING SYSTEM:

The existing system for ad relevance prediction primarily relies on traditional machine learning algorithms, such as Logistic Regression, Decision Tree, K-Nearest Neighbors (KNN), Support Vector Machine (SVM), and Gaussian Naive Bayes (NB). These algorithms work by analyzing user behavior data, such as demographics, browsing patterns, and past interactions, to predict the likelihood of ad clicks. However, these models are often limited in their ability to capture complex, non-linear relationships in large and diverse datasets, and they may struggle to provide highly accurate predictions for dynamic user behavior. Furthermore, the lack of ensemble methods in the existing system results in less robust performance and reduced accuracy. While these algorithms can still offer basic predictions, the existing system lacks advanced techniques like boosting and voting classifiers, which could significantly improve model performance by combining multiple weak learners for more accurate and generalized predictions, leading to better ad relevance and user engagement.

3.1.1 DISADVANTAGES OF EXISTING SYSTEM:

1. The existing system primarily relies on basic machine learning algorithms, which struggle to model complex, non-linear relationships within large and diverse datasets, limiting its ability to provide highly accurate predictions for dynamic user behavior.
2. Traditional algorithms like Logistic Regression and Decision Trees may fail to capture intricate patterns in user interactions with ads, resulting in reduced ad relevance and engagement, especially in cases with complex or evolving user behavior.
3. The absence of ensemble techniques in the existing system means it lacks the robustness of combined model predictions, which can lead to reduced accuracy and lower reliability when predicting user preferences for ad clicks.
4. Existing methods often underperform when handling real-world data with diverse features, making it difficult to accurately target ads, impacting user satisfaction and diminishing the effectiveness of ad delivery strategies.

3.2 Proposed System:

proposed system aims to enhance advertisement relevance by leveraging Web Content Mining and User Profiling to predict user preferences accurately. The Clicked Ads Dataset is utilized to analyze user behavior and ad interaction patterns. The system implements multiple machine learning algorithms, including Logistic Regression, Decision Tree, Multi-Layer Perceptron (MLP), Random Forest, K-Nearest Neighbors (KNN), Support Vector Machine (SVM), and Gaussian Naïve Bayes (NB). To further improve performance, an ensemble approach is applied, combining the strengths of individual models. Specifically, a Voting Classifier is employed, integrating Adaboost and ExtraTree classifiers to produce a robust and accurate final prediction. By capturing complex relationships between user behavior and ad content, the system ensures better personalization, higher engagement, and improved targeting accuracy. This comprehensive approach leverages advanced predictive techniques to optimize ad delivery, ultimately enhancing user satisfaction and maximizing advertising effectiveness.

3.2.1 Advantages of proposed system:

1. The proposed system utilizes Web Content Mining and User Profiling to capture complex patterns, improving prediction accuracy by analyzing user behavior and ad interaction, thus providing more personalized and relevant ad targeting.
2. By implementing multiple advanced machine learning algorithms and ensemble techniques, the system is capable of generating more robust, generalized predictions, enhancing the accuracy and effectiveness of ad relevance predictions.
3. The use of a Voting Classifier combining Adaboost and ExtraTree classifiers improves model performance by leveraging the strengths of individual models, leading to a more precise and reliable final prediction.
4. The system’s ability to capture intricate relationships between user behavior and ad content ensures better targeting, higher user engagement, and ultimately enhances the efficiency of ad delivery, maximizing overall advertising effectiveness.

3.3 FUNCTIONAL REQUIREMENTS

1. Data Collection
2. Pre-processing
3. Training and Testing
4. Modelling
5. Predicting

3.4 NON FUNCTIONALREQUIREMENTS

Scalability

The system should be capable of handling large-scale datasets, ensuring seamless operation and quick processing times as the volume of data, such as user interactions and ad clicks, increases over time.

Performance

The system must provide fast prediction times for ad relevance, ensuring minimal latency when delivering personalized ads, even with complex models and large datasets, to enhance the user experience.

Accuracy

The system should consistently produce highly accurate predictions regarding ad relevance, ensuring that personalized ads align closely with user preferences, improving user engagement and satisfaction.

Usability

The system should be user-friendly, with an intuitive interface for stakeholders to easily interact with the ad personalization engine, monitor model performance, and adjust parameters without technical expertise.

Security

The system must ensure the confidentiality and integrity of user data, including behavioral and demographic information, adhering to privacy regulations and best practices to protect sensitive information from unauthorized access.

4. SYSTEM ARCHITECTURE

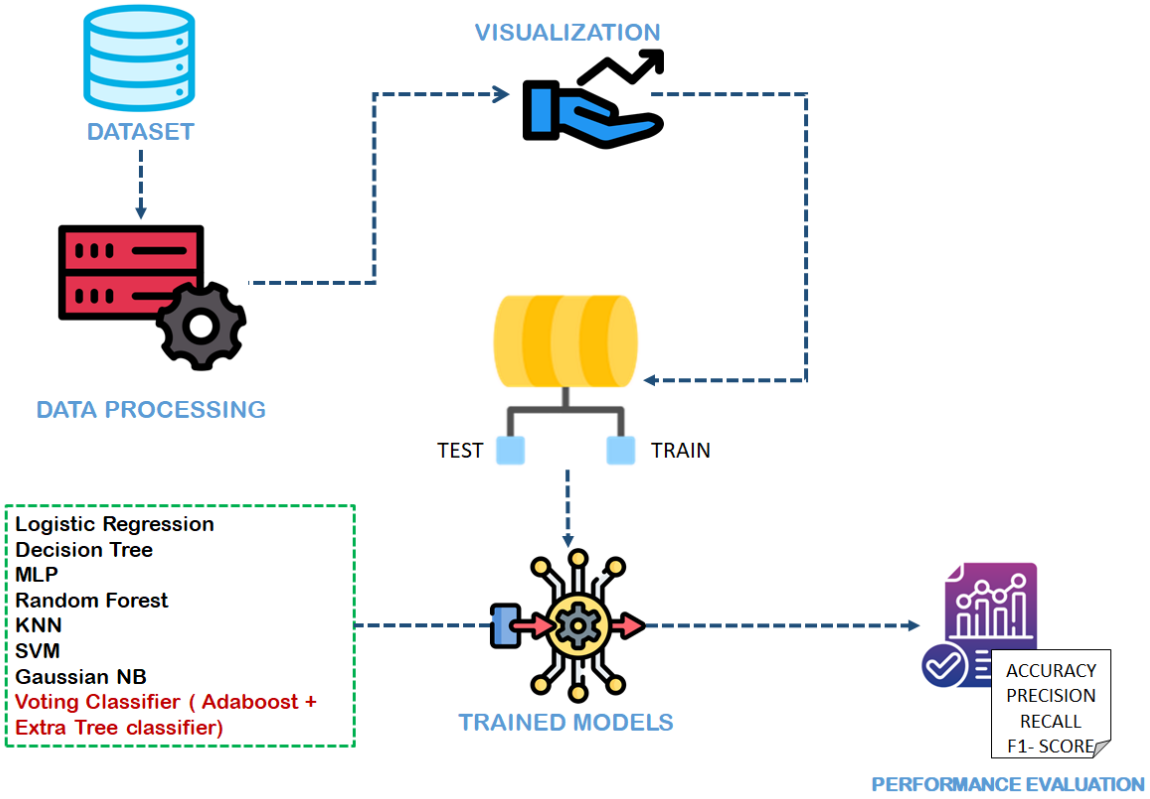


Fig.4.1.1 System architecture

DATA FLOW DIAGRAM:

The Data Flow Diagram (DFD) for the ad relevance system begins with the input of user data, including behavioral patterns, demographics, and ad interaction details. This data is processed by the user profiling module, which analyzes and categorizes user preferences. Simultaneously, content-based features from advertisements are extracted and mapped to the user profiles to identify potential relevance. The system then employs a set of predictive models that process this integrated data to generate ad relevance predictions. These predictions are sent to the ad delivery engine, which selects the most relevant advertisements for the user based on the output. The feedback loop collects user interaction data (e.g., clicks, views) to continuously refine and optimize the models. This entire process is monitored and managed by the control module, ensuring the smooth flow of data and system operations. Data storage handles the collected datasets and user profiles, supporting continuous learning and model improvements over time.

Goals:

5. To collect and preprocess user data, including behavioral patterns and demographic information, ensuring the system has accurate and up-to-date information for ad relevance prediction.
6. To integrate content-based features of advertisements with user profiles, enabling the system to match ads to individual user preferences effectively.
7. To generate accurate ad relevance predictions using advanced machine learning models, ensuring that personalized ads are aligned with user interests.
8. To deliver the most relevant ads to users in real-time, ensuring high engagement rates and a seamless user experience with minimal latency.
9. To continuously improve the ad relevance prediction models by collecting feedback data from user interactions, enabling the system to adapt and optimize over time.

4.2 UML DIAGRAMS

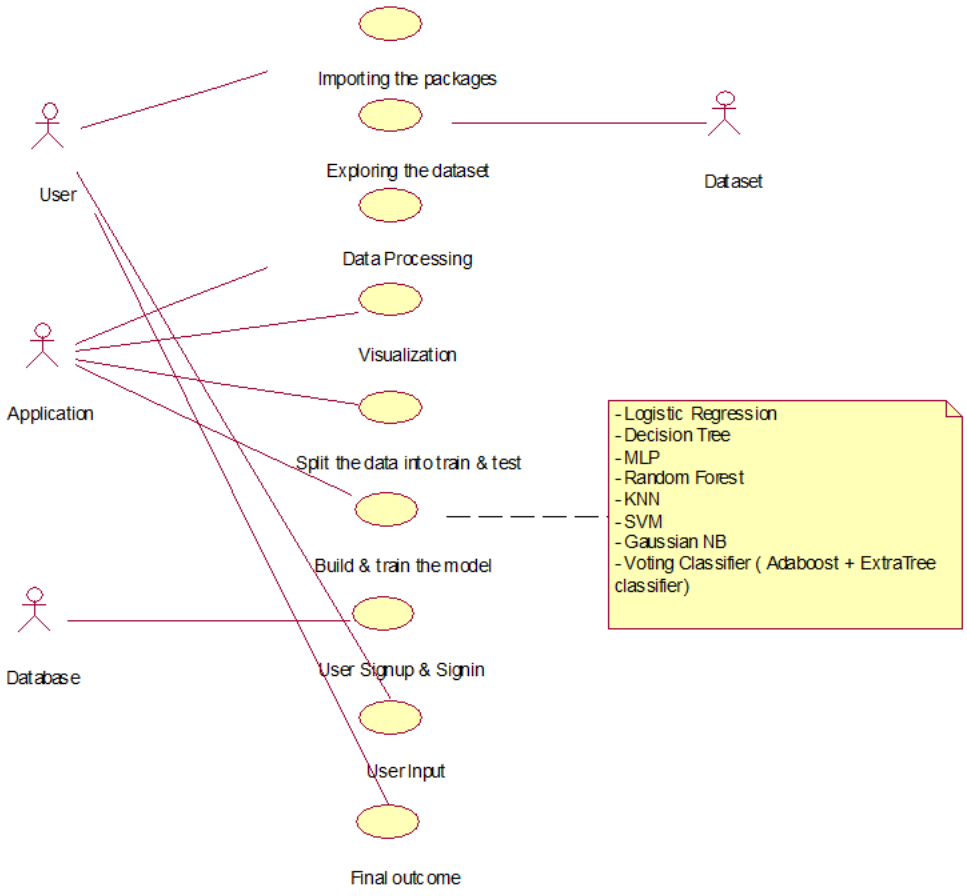
The UML diagram for the ad relevance system represents the interactions between key components. The user interacts with the system, providing behavioral and demographic data, which is processed by the User Profiling module. This module analyzes the user data and creates a profile. Simultaneously, advertisements' content features are extracted by the Ad Content module and mapped to the user profile. The Ad Prediction Engine then processes this integrated data and generates ad relevance scores. These scores are passed to the Ad Delivery module, which selects and delivers the most relevant ads to the user. Feedback from user interactions, such as clicks or views, is sent back to the system, allowing the prediction models to adapt and improve. All data is stored in a centralized database for continuous learning and optimization. The system is controlled and monitored by a central Management module, ensuring smooth operations and data flow throughout the process.

Goals of UML:

To visually represent the system's components, including user profiles, ad content, and prediction modules, providing a clear understanding of their interactions and relationships.
To model the flow of data between modules, ensuring accurate and efficient ad relevance prediction and delivery to users.
To define the role of the feedback loop, enabling continuous learning and model optimization based on user interactions, improving ad relevance over time.
To ensure system scalability by outlining how new modules or features can be integrated into the existing architecture without disrupting overall functionality.
To provide a blueprint for developers and stakeholders to understand the structure of the system, facilitating collaboration and ensuring proper implementation.
To aid in the identification of potential bottlenecks or inefficiencies in the system's data flow, helping improve performance and user experience.

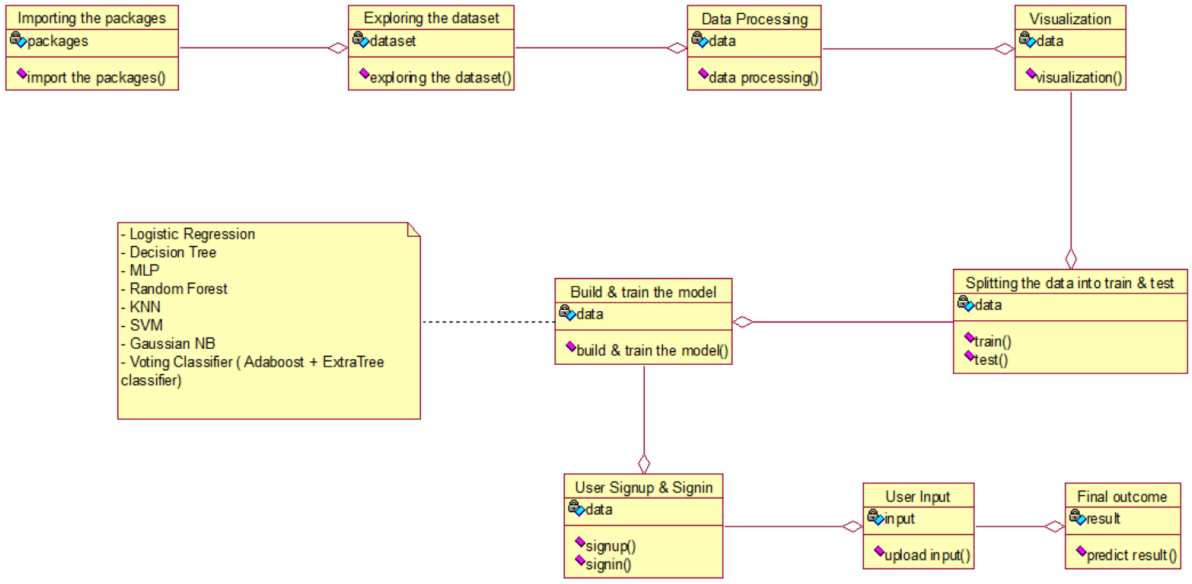
Use Case Diagram

The use case diagram illustrates the interactions between users and the system components. Key actors, such as the user and the system administrator, are depicted, with use cases such as providing data, receiving ad recommendations, and updating profiles. These use cases reflect how users interact with the ad relevance system, including data input, feedback, and ad delivery, ensuring seamless personalization of advertisements.



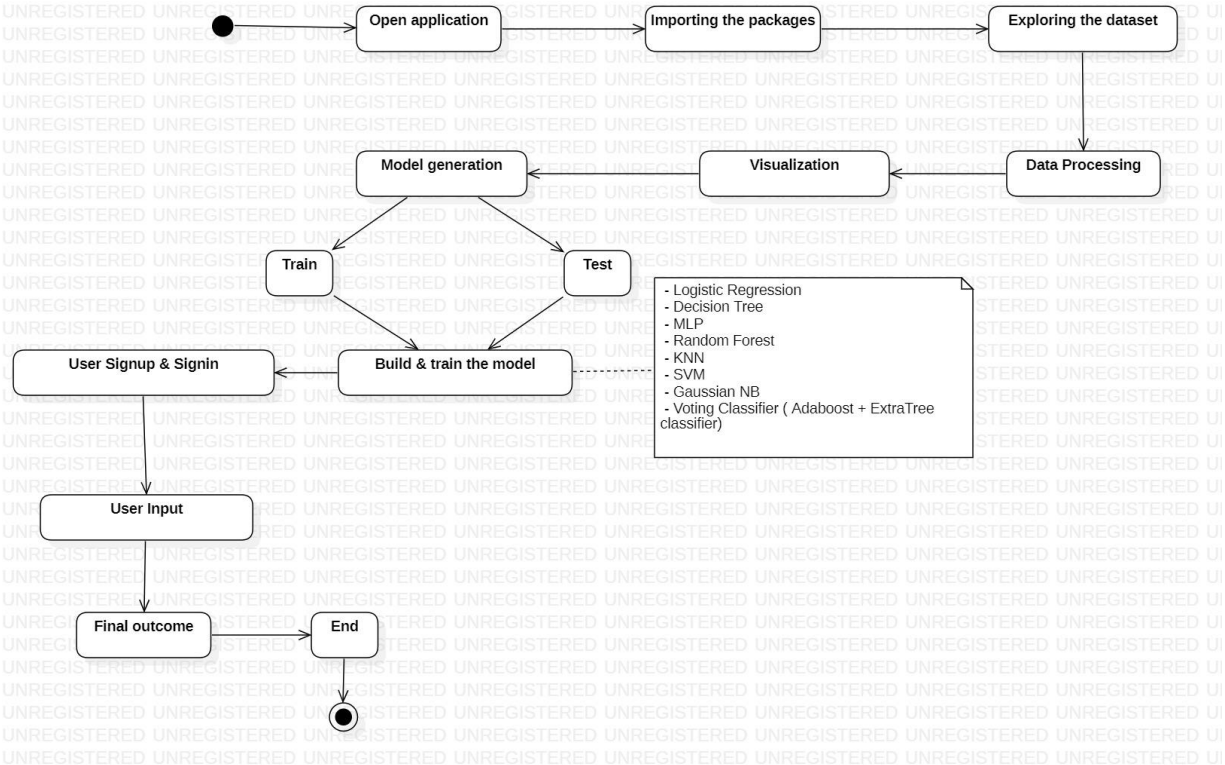
Class Diagram

The class diagram defines the structure of the system by showing the classes and their relationships. Core classes include User Profile, Ad Content, Prediction Engine, and Feedback Loop. Each class has attributes and methods relevant to its function, with associations indicating data flow and interactions between them. The diagram helps visualize how data is organized and processed within the system.



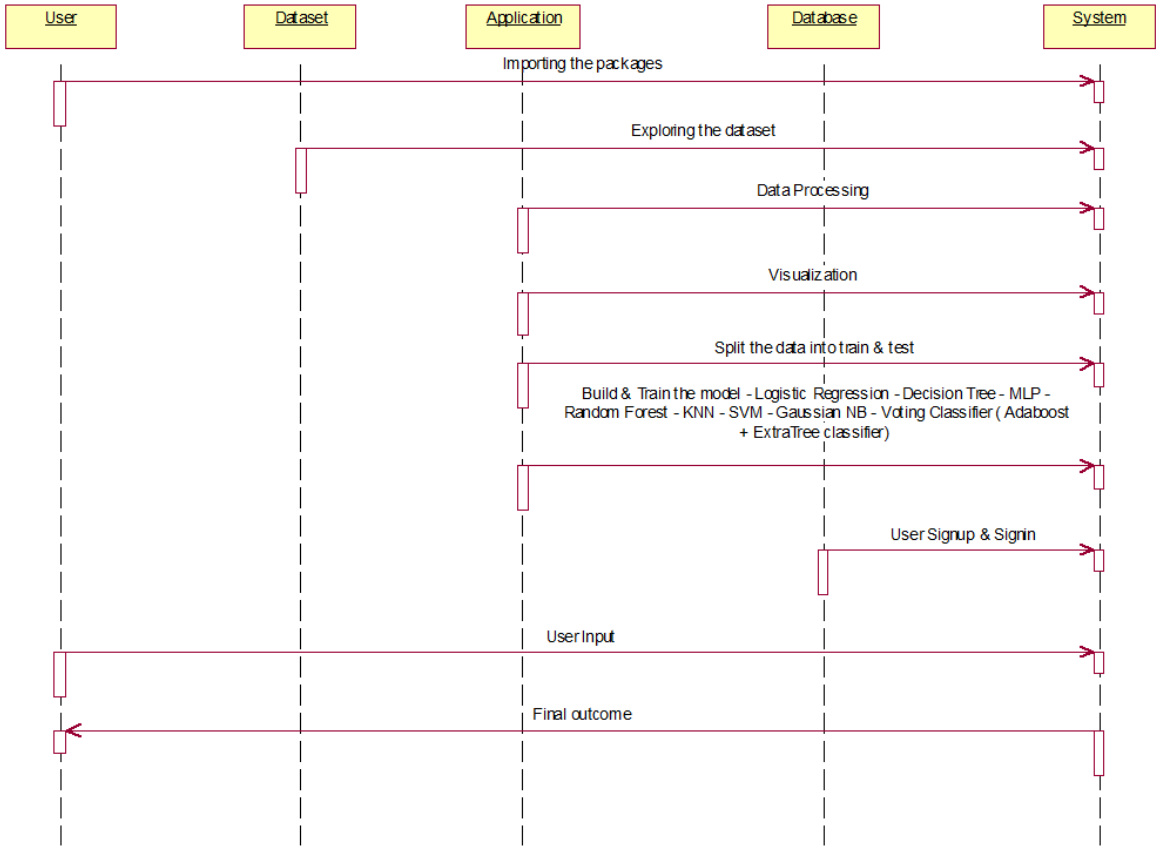
Activity Diagram

The activity diagram illustrates the flow of actions from data collection to ad delivery. It shows the process from gathering user data, integrating it with ad content, predicting relevance, and delivering personalized ads. Feedback is collected for system improvement. Each activity is represented sequentially, highlighting decision points and flow control.



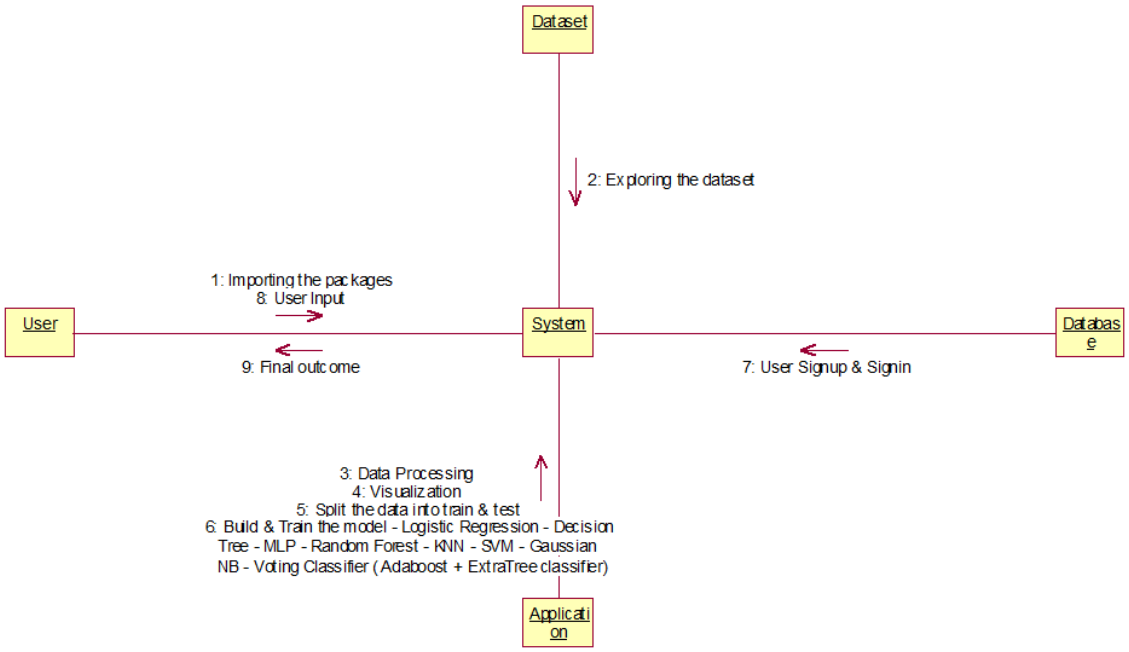
Sequence Diagram

The sequence diagram outlines the order of interactions between system components during ad delivery. It tracks the sequence from user data input, profile creation, ad relevance prediction, to the final delivery of ads. The diagram shows the time-based flow and message exchanges between modules, ensuring an efficient process.



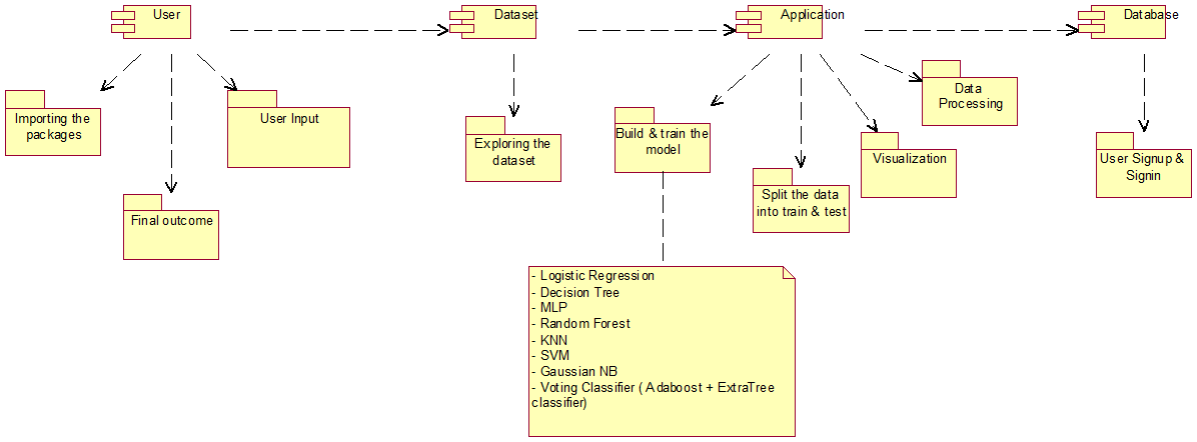
Collaboration Diagram

The collaboration diagram represents the system's components and their interactions in terms of messages. It highlights how the User Profile, Ad Content, and Prediction Engine modules collaborate to process data and deliver personalized ads. The diagram visualizes how components work together in an efficient, coordinated manner to achieve the system's objectives.



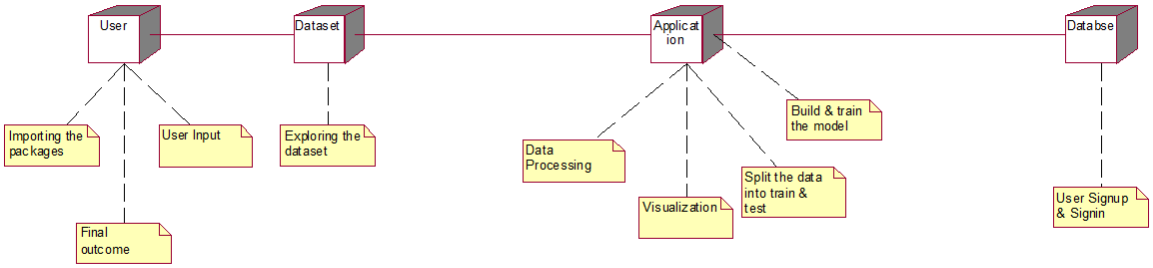
Component Diagram

The component diagram depicts the system's modular structure, showing the major components and their dependencies. Components such as User Profile Manager, Ad Prediction Module, and Feedback Handler are displayed along with their interfaces, indicating how they interact to produce the final ad relevance predictions and ensure continuous model optimization.



Deployment Diagram

The deployment diagram illustrates the physical distribution of system components across servers and devices. It shows how the database, application server, and user interface interact in the cloud or on-premise environment. This diagram ensures proper configuration of hardware resources to support efficient ad relevance processing and delivery.



5. IMPLEMENTATION MODULES:

Dataset loading: The Clicked Ads Dataset is loaded into the system, ensuring that all necessary features such as user interactions, demographics, and ad content are properly imported for further analysis and processing.

Data Processing: Duplicate entries in the dataset are identified and removed to ensure data integrity, maintaining only unique user interactions and ad clicks, which helps in improving model accuracy and performance.

Visualization: The dataset is visualized using various charts and graphs to better understand patterns in user behavior, ad interactions, and key features. This helps identify trends and relationships for model development.

Split the Data into Train & Test: The dataset is split into training and testing sets, with a predefined ratio (e.g., 80/20), ensuring that the model is trained on one portion while being evaluated on another to assess performance.

Model generation: Model building - Logistic Regression - Decision Tree - MLP - Random Forest - KNN - SVM - Gaussian NB - Voting Classifier (Adaboost + Extra Tree classifier). Performance evaluation metrics for each algorithm is calculated.

User signup & login: Using this module will get registration and login

User input: Using this module will give input for prediction

Prediction: final predicted displayed

Extension:
Extension involves applying ensemble methods like Voting Classifier (Adaboost + Extra Tree) for improved accuracy, and building a front-end using Flask for user testing and authentication.

Advantages:

1. The ensemble method combines multiple models, leveraging their strengths to produce a more robust and accurate prediction, enhancing the overall ad relevance performance.
2. Voting Classifier improves model accuracy by integrating the outputs of different classifiers, ensuring better generalization across diverse user behavior patterns and ad content types.
3. Building a front-end using Flask allows real-time user testing, providing an intuitive interface for interaction with the system, enabling immediate feedback and engagement for continuous improvement.
4. User authentication ensures secure access to the system, protecting sensitive user data and enhancing the trustworthiness of the ad relevance platform, ensuring privacy compliance and user confidence.

Algorithms:

Logistic Regression:

Logistic Regression is a statistical method used for binary classification. It models the probability of a binary outcome based on input features using a logistic function. In this context, it predicts whether a user will click on an ad or not based on their profile and behavior. It’s efficient for linear relationships and provides probability scores, making it useful for classification tasks where a decision threshold can be applied.

Decision Tree:

A Decision Tree is a tree-like model used for classification and regression tasks. It splits data into subsets based on feature values, creating a series of decision rules. In this context, it classifies user interactions by evaluating features such as demographics and ad type. The model's interpretability is its main advantage, as it visualizes decision-making steps, making it easy to understand and debug the ad relevance prediction process.

MLP (Multilayer Perceptron):

Multilayer Perceptron (MLP) is a type of neural network consisting of multiple layers of neurons, each layer fully connected to the next. It can capture non-linear relationships and complex patterns in data. In this context, MLP is used to predict ad relevance by processing features through its layers, learning intricate patterns from user interaction data. Its ability to handle non-linear data relationships enhances the accuracy of the ad relevance predictions.

Random Forest:

Random Forest is an ensemble learning method that constructs multiple decision trees and merges their outputs to improve prediction accuracy. It reduces overfitting by averaging the results from many trees. In this context, Random Forest is used to predict ad relevance by aggregating decisions from several trees, each trained on different data subsets. It handles both numerical and categorical data effectively, providing high accuracy and robustness in real-time ad delivery.

KNN (K-Nearest Neighbors):

K-Nearest Neighbors (KNN) is a simple, instance-based learning algorithm that classifies a data point based on the majority class of its k-nearest neighbors in the feature space. In this context, KNN is used to classify ad relevance by comparing a user's behavior and preferences with similar users. It is effective for high-dimensional data and provides intuitive predictions, but its performance can degrade with large datasets due to its reliance on distance calculations.

SVM (Support Vector Machine):

Support Vector Machine (SVM) is a supervised machine learning algorithm used for classification and regression tasks. It constructs hyperplanes in a high-dimensional space to separate classes of data. In this context, SVM is employed to classify whether a user will interact with a given ad based on features like demographics and behavior. SVM is effective in handling high-dimensional spaces and is particularly useful for complex decision boundaries between classes, such as ad relevance.

Gaussian NB (Naive Bayes):

Gaussian Naive Bayes (Gaussian NB) is a probabilistic classifier based on Bayes' theorem and assumes features are conditionally independent given the class. It uses Gaussian distributions for continuous features. In this context, Gaussian NB is applied to predict the probability of ad relevance, where each feature (like user behavior or ad type) follows a Gaussian distribution. It is fast and efficient, especially with large datasets, though it may not capture complex relationships between features.

Voting Classifier (Adaboost + Extra Tree):

Voting Classifier is an ensemble method that combines the predictions of multiple classifiers to make a final decision based on majority voting or weighted votes. In this case, Adaboost and Extra Tree classifiers are combined. Adaboost increases weak classifier performance by focusing on misclassified instances, while Extra Tree is an unpruned decision tree model. Together, they provide a robust prediction for ad relevance by leveraging their diverse strengths, improving accuracy and reducing bias.

6.2 SAMPLE CODE:

```
# Data Processing
import numpy as np
import pandas as pd
import datetime as dt

# Data Visualizing
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import matplotlib.gridspec as gridspec
from matplotlib.ticker import MaxNLocator
from IPython.display import display, HTML
import plotly.express as px
import plotly.graph_objs as go
from IPython.display import display, HTML
from IPython.display import Image

# Data Clustering
from mlxtend.frequent_patterns import apriori# Data pattern exploration
from mlxtend.frequent_patterns import association_rules# Association rules conversion

# Data Modeling
```

```
from sklearn.ensemble import RandomForestRegressor

# Math
from scipy import stats  # Computing the t and p values using scipy
from statsmodels.stats import weightstats

# Warning Removal
import warnings
def ignore_warn(*args, **kwargs):
    pass
warnings.warn = ignore_warn#ignore annoying warning (from sklearn and seaborn)
# https://stackoverflow.com/questions/22216076/unicodedecodeerror-utf8-codec-cant-decode-byte-0xa5-in-position-0-invalid-s/50538501#50538501
df = pd.read_csv('../input/ecommerce-data/data.csv', encoding= 'unicode_escape')
In [100]:
linkcode
df
df.describe()
df.info()
df.columns
print(df.duplicated().sum())
df.drop_duplicates(inplace = True)
# https://stackoverflow.com/questions/574730/python-how-to-ignore-an-exception-and-proceed/575711#575711
# https://stackoverflow.com/questions/59127458/pandas-fillna-using-groupby-and-mode
def cleaning_description(df):
    try:
        return df.mode()[0] # df.mode().iloc[0]
    except Exception:
        return 'unknown'

df[['StockCode', 'Description']] = df[['StockCode', 'Description']].fillna(df[['StockCode',
'Description']].groupby('StockCode').transform(cleaning_description))

# Cleaning Description field for proper aggregation
df['Description'] = df['Description'].str.strip().copy()
def clean_InvoiceNo(InvoiceNo):
```

```
    if InvoiceNo[0] == 'C':
        return InvoiceNo.replace(InvoiceNo[0], '')
    else:
        return InvoiceNo
df['InvoiceNo'] = df['InvoiceNo'].apply(clean_InvoiceNo)
# Plot Quantity
plt.figure(constrained_layout=True, figsize=(12, 5))
sns.boxplot(df['Quantity'])

# remove outliers for Quantity
df = df[(df['Quantity'] < 15000) & (df['Quantity'] > -15000)]
# Change datatype of InvoiceDate as datetime type
df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'])
# df['date'] = pd.to_datetime(df['InvoiceDate'], utc=False)
# df['date'].dtypes

# Create new features
df['date'] = df['InvoiceDate'].dt.date# df['date'].dt.normalize()  # Show only date
df['day'] = df['InvoiceDate'].dt.day
df['month'] = df['InvoiceDate'].dt.month
df['year'] = df['InvoiceDate'].dt.year
df['hour'] = df['InvoiceDate'].dt.hour
df['dayofweek'] = df['InvoiceDate'].dt.dayofweek
df['dayofweek'] = df['dayofweek'].map( {0: '1_Mon', 1: '2_Tue', 2: '3_Wed', 3:
'4_Thur', 4: '5_Fri', 5: '6_Sat', 6: '7_Sun'})
# Clean UnitPrice
'''
Steps to clean Unit Price
df['UnitPrice'].describe()
df[df['UnitPrice'] < 0]
sns.boxplot(df['UnitPrice'])
sns.distplot(df['UnitPrice'])
df[df['StockCode'] == 'M']
df[df['UnitPrice'] > 15000]
'''
df = df[df['UnitPrice'] >= 0]
# Fill CustomerID with unknown
df['CustomerID'].dropna(inplace=True)
# Create a new feature Revenue
df['Revenue'] = df['UnitPrice'] * df['Quantity']
```

```
CustomerID_Rev = df.groupby('CustomerID')[['Revenue',
                    'Quantity',
                    'UnitPrice']].agg(['sum',
                                       'mean',
                                       'median']).sort_values(by=[('Revenue',          'sum')],
ascending=False)
display(CustomerID_Rev.reset_index())

display(pd.DataFrame(CustomerID_Rev.iloc[1:][('Revenue','sum')].describe()))

# Remove the unknown CustomerID
sns.distplot(CustomerID_Rev.iloc[1:][('Revenue','sum')], kde=False)
Item_retured = df[df['Quantity'] < 0].groupby('CustomerID')[['Revenue',
                                    'Quantity']].agg(['sum']).sort_values(by=[('Quantity',
'sum')], ascending=True).head(10)

sns.barplot(x=Item_retured.index, y=abs(Item_retured[('Quantity','sum')]))
plt.ylabel('A number of Quantity returned')
plt.xticks(rotation=90)
plt.show()

Item_retured
most_prefered_items                    =                    df.groupby(['StockCode',
'UnitPrice'])[['Quantity']].sum().sort_values(by=['Quantity'],ascending=False).he
ad(10)

most_prefered_items
most_prefered_items1                                                                =
df.groupby(['StockCode'])[['Quantity']].sum().sort_values(by=['Quantity'],ascend
ing=False).head(10)

most_prefered_items2                    =                    df.groupby(['StockCode',
'UnitPrice'])[['Quantity']].sum().sort_values(by=['Quantity'],ascending=False).he
ad(10)

sns.barplot(x=most_prefered_items1.index, y=most_prefered_items1['Quantity'])
plt.ylabel('A number of Quantity returned')
plt.xticks(rotation=90)
plt.show()
```

```
display(most_prefered_items1)
display(most_prefered_items2)
least_prefered_items =
df.groupby(['StockCode'])[['Quantity']].sum().sort_values(by=['Quantity'],ascend
ing=False)
least_prefered_items =
least_prefered_items[least_prefered_items['Quantity']==0]
print('A list of least preferred items: ', len(least_prefered_items))
least_prefered_items
InvoiceNumber_Country =
pd.DataFrame(df.groupby(['Country'])['InvoiceNo'].count())

fig = go.Figure(data=go.Choropleth(
        locations=InvoiceNumber_Country.index, # Spatial coordinates
        z = InvoiceNumber_Country['InvoiceNo'].astype(float), # Data to be
color-coded
locationmode = 'country names', # set of locations match entries in `locations`
colorscale = 'Reds',
colorbar_title = "Order number",
      ))

fig.update_layout(
title_text = 'Order number per country',
   geo = dict(showframe = True, projection={'type':'mercator'})
)
fig.layout.template = None
fig.show()
# Source: https://stackoverflow.com/questions/36220829/fine-control-over-the-
font-size-in-seaborn-plots-for-academic-papers/36222162#36222162
country_revenue = df.groupby('Country')[['Revenue']].agg(['sum',
                        'mean',
                        'median']).sort_values(by=[('Revenue', 'sum')],
ascending=False)
display(country_revenue)

fig = plt.figure(constrained_layout=True, figsize=(20, 6))
a = sns.barplot(y=country_revenue.index, x=country_revenue[('Revenue',
'sum')])
plt.xlabel('Total Revenue from all country', fontsize=18)
plt.ylabel('Country', fontsize=18)
```

```
fig = plt.figure(constrained_layout=True, figsize=(20, 6))
country_revenue = country_revenue.drop('United Kingdom')
sns.barplot(y=country_revenue.index, x=country_revenue[('Revenue', 'sum')])
plt.xlabel('Total Revenue from all country but UK', fontsize=18)
plt.ylabel('Country', fontsize=18)
plt.show()
country_quantity = df.groupby('Country')[['Quantity']].agg(['sum',
                                'mean',
                                'median']).sort_values(by=[('Quantity',                'sum')],
ascending=False)

display(country_quantity)

fig = plt.figure(constrained_layout=True, figsize=(20, 6))
a    =    sns.barplot(y=country_quantity.index,    x=country_quantity[('Quantity',
'sum')])
plt.xlabel('Total Quantity from all country', fontsize=18)
plt.ylabel('Country', fontsize=18)


fig = plt.figure(constrained_layout=True, figsize=(20, 6))
country_quantity = country_quantity.drop('United Kingdom')
sns.barplot(y=country_quantity.index, x=country_quantity[('Quantity', 'sum')])
plt.xlabel('Total Quantity from all country but UK', fontsize=18)
plt.ylabel('Country', fontsize=18)
plt.show()
unitprice_average = df.groupby('Country')[['UnitPrice']].agg(['sum',
                                'mean']).sort_values(by=[('UnitPrice',                'mean')],
ascending=False)
display(unitprice_average)

fig = plt.figure(constrained_layout=True, figsize=(20, 6))
a    =    sns.barplot(y=unitprice_average.index,    x=unitprice_average[('UnitPrice',
'mean')])
plt.xlabel('Total Quantity from all country', fontsize=18)
plt.ylabel('Country', fontsize=18)
```

```
fig = plt.figure(constrained_layout=True, figsize=(20, 6))
unitprice_average = unitprice_average.drop('United Kingdom')
sns.barplot(y=country_quantity.index, x=unitprice_average[('UnitPrice',
'mean')])
plt.xlabel('Total Quantity from all country but UK', fontsize=18)
plt.ylabel('Country', fontsize=18)
plt.show()
month_sales = df.groupby(['month'])['Revenue'].agg(['sum','mean'])

fig, axes = plt.subplots(1, 2, figsize=(18, 5))
axes = axes.flatten()

sns.barplot(x=month_sales.index, y=month_sales['sum'],
ax=axes[0]).set_title("Total Revenue over a year")
plt.ylabel('a')
plt.xticks(rotation=90)

sns.barplot(x=month_sales.index, y=month_sales['mean'],
ax=axes[1]).set_title("Average Revenue over a year")
plt.xticks(rotation=90)
plt.show()

month_sales
hour_sales = df.groupby(['hour'])['Revenue'].agg(['sum','mean'])

fig, axes = plt.subplots(1, 2, figsize=(18, 5))
axes = axes.flatten()

sns.barplot(x=hour_sales.index, y=hour_sales['sum'], ax=axes[0]).set_title("Total
Revenue in a day")
plt.ylabel('a')
plt.xticks(rotation=90)

sns.barplot(x=hour_sales.index, y=hour_sales['mean'],
ax=axes[1]).set_title("Average Revenue per Invoice in a day")
plt.xticks(rotation=90)
plt.show()

hour_sales
dayofweek_sales = df.groupby(['dayofweek'])['Revenue'].agg(['sum','mean',])
```

```
fig, axes = plt.subplots(1, 2, figsize=(18, 5))
axes = axes.flatten()

sns.barplot(x=dayofweek_sales.index, y=dayofweek_sales['sum'], ax=axes[0]).set_title("Total Revenue over a week")
plt.ylabel('a')
plt.xticks(rotation=90)

sns.barplot(x=dayofweek_sales.index, y=dayofweek_sales['mean'], ax=axes[1]).set_title("Average Revenue over a week")
plt.xticks(rotation=90)
plt.show()

dayofweek_sales
# Get our date range for our data
print('Date Range: %s to %s' % (df['InvoiceDate'].min(), df['InvoiceDate'].max()))

# We're taking all of the transactions that occurred before December 01, 2011
df = df[df['InvoiceDate'] < '2011-12-01']
# Get total amount spent per invoice and associate it with CustomerID and Country
invoice_customer_df = df.groupby(by=['InvoiceNo', 'InvoiceDate']).agg({'Revenue': sum,'CustomerID': max,'Country': max,}).reset_index()
invoice_customer_df
# Source: https://pandas.pydata.org/pandas-docs/stable/user_guide/timeseries.html#dateoffset-objects
# We set our index to our invoice date
# And use Grouper(freq='M') groups data by the index 'InvoiceDate' by Month
# We then group this data by CustomerID and count the number of unique repeat customers for that month (data is the month end date)
# The filter fucntion allows us to subselect data by the rule in our lambda function i.e. those greater than 1 (repeat customers)

monthly_repeat_customers_df = invoice_customer_df.set_index('InvoiceDate').groupby([
pd.Grouper(freq='M'), 'CustomerID']).filter(lambda x: len(x) > 1).resample('M').nunique()['CustomerID']
```

```
monthly_repeat_customers_df
# Number of Unique customers per month
monthly_unique_customers_df =
df.set_index('InvoiceDate')['CustomerID'].resample('M').nunique()
monthly_unique_customers_df
# Ratio of Repeat to Unique customers
monthly_repeat_percentage =
monthly_repeat_customers_df/monthly_unique_customers_df*100.0
monthly_repeat_percentage
fig = plt.figure(constrained_layout=True, figsize=(20, 6))
grid = gridspec.GridSpec(nrows=1, ncols=1,  figure=fig)

ax = fig.add_subplot(grid[0, 0])

pd.DataFrame(monthly_repeat_customers_df.values).plot(ax=ax, figsize=(12,8))

pd.DataFrame(monthly_unique_customers_df.values).plot(ax=ax,grid=True)

ax.set_xlabel('Date')
ax.set_ylabel('Number of Customers')
ax.set_title('Number of Unique vs. Repeat Customers Over Time')
plt.xticks(range(len(monthly_repeat_customers_df.index)), [x.strftime('%m.%Y')
for x in monthly_repeat_customers_df.index], rotation=45)
ax.legend(['Repeat Customers', 'All Customers'])
# Let's investigate the relationship between revenue and repeat customers
monthly_revenue_df =
df.set_index('InvoiceDate')['Revenue'].resample('M').sum()

monthly_rev_repeat_customers_df =
invoice_customer_df.set_index('InvoiceDate').groupby([
pd.Grouper(freq='M'), 'CustomerID']).filter(lambda x: len(x) >
1).resample('M').sum()['Revenue']

# Let's get a percentage of the revenue from repeat customers to the overall
monthly revenue
monthly_rev_perc_repeat_customers_df =
monthly_rev_repeat_customers_df/monthly_revenue_df * 100.0
monthly_rev_perc_repeat_customers_df
fig = plt.figure(constrained_layout=True, figsize=(20, 6))
```

```
grid = gridspec.GridSpec(nrows=1, ncols=1,  figure=fig)

ax = fig.add_subplot(grid[0, 0])
pd.DataFrame(monthly_rev_repeat_customers_df.values).plot(ax=ax,
figsize=(12,8))

pd.DataFrame(monthly_revenue_df.values).plot(ax=ax,grid=True)

ax.set_xlabel('Date')
ax.set_ylabel('Number of Customers')
ax.set_title('Number of Unique vs. Repeat Customers Over Time')
plt.xticks(range(len(monthly_repeat_customers_df.index)), [x.strftime('%m.%Y')
for x inmonthly_repeat_customers_df.index], rotation=45)
ax.legend(['Repeat Customers', 'All Customers'])
# Now let's get quantity of each item sold per month
date_item_df = df.set_index('InvoiceDate').groupby([pd.Grouper(freq='M'),
'StockCode'])['Quantity'].sum()
date_item_df.head(15)
# Rank items by the last month's sales
last_month_sorted_df = date_item_df.loc['2011-11-30']
last_month_sorted_df = last_month_sorted_df.reset_index()
last_month_sorted_df.sort_values(by='Quantity', ascending=False).head(10)
# Let's look at the top 5 items sale over a year
date_item_df = df.loc[df['StockCode'].isin(['23084', '84826', '22197', '22086',
'85099B'])].set_index('InvoiceDate').groupby([
pd.Grouper(freq='M'), 'StockCode','Description'])['Quantity'].sum().reset_index()

date_item_df
date_item_df = date_item_df.reset_index()

sns.set(style='whitegrid')
plt.figure(constrained_layout=True, figsize=(12, 5))
sns.lineplot(x=date_item_df['InvoiceDate'], y=date_item_df['Quantity'],
hue=date_item_df['StockCode'])
df.groupby(['StockCode',
'Description'])['InvoiceNo'].count().sort_values(ascending = False).head(10)
Num_Canceled_Orders = df[df['Quantity']<0]['InvoiceNo'].nunique()
Total_Orders = df['InvoiceNo'].nunique()
print('Cancellation Rate:
{:.2f}%'.format(Num_Canceled_Orders/Total_Orders*100 ))
```

```
Monthly_Reorder_Items_Revenue = df.set_index('InvoiceDate').groupby([
pd.Grouper(freq='M'), 'StockCode']).filter(lambda x: len(x) >
1).resample('M').sum()['Revenue']
Monthly_One_Items_Revenue = df.set_index('InvoiceDate').groupby([
pd.Grouper(freq='M'), 'StockCode']).filter(lambda x: len(x) ==
1).resample('M').sum()['Revenue']
#Monthly_Revenue = df.groupby(['year','month']).sum()['Revenue']  # Generate
the same Result
Monthly_Revenue =
df.set_index('InvoiceDate').groupby([pd.Grouper(freq='M')]).sum()['Revenue']
fig = plt.figure(constrained_layout=True, figsize=(20, 6))

ax = fig.add_subplot()
pd.DataFrame(Monthly_Reorder_Items_Revenue.values).plot(ax=ax,
figsize=(12,8))
pd.DataFrame(Monthly_Revenue.values).plot(ax=ax,grid=True)
pd.DataFrame(Monthly_One_Items_Revenue.values).plot(ax=ax,grid=True)

ax.set_xlabel('Date')
ax.set_ylabel('Number of Customers')
ax.set_title('Number of Unique vs. Repeat vs Total Items Over Time')
plt.xticks(range(len(monthly_repeat_customers_df.index)), [x.strftime('%m.%Y')
for x in monthly_repeat_customers_df.index], rotation=45)
ax.legend(['Repeat Items', 'All Items', 'One Item'])
Sample_df = df[:50]
Sample_df = Sample_df[['InvoiceNo', 'Description']]
In [141]:
Sample_df.set_index('InvoiceNo', inplace=True)
In [142]:
linkcode
# Note that the quantity bought is not considered, only if the item was present or
not in the basket
basket = pd.get_dummies(Sample_df)
basket_sets = pd.pivot_table(basket, index='InvoiceNo', aggfunc='sum')
basket_sets
# Aprioriaplication: frequent_itemsets
# Note that min_support parameter was set to a very low value, this is the
Spurious limitation, more on conclusion section
frequent_itemsets = apriori(basket_sets, min_support=0.22, use_colnames=True)
```

```
frequent_itemsets['length'] = frequent_itemsets['itemsets'].apply(lambda x: len(x))
frequent_itemsets
# Advanced and strategical data frequent set selection
frequent_itemsets[ (frequent_itemsets['length'] > 1) &
                   (frequent_itemsets['support'] >= 0.02)]
# Generating the association_rules: rules
# Selecting the important parameters for analysis
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)
rules
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)
rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']].sort_values('support', ascending=False).head()
# Visualizing the rules distribution color mapped by Lift
plt.figure(figsize=(14, 8))
plt.scatter(rules['support'], rules['confidence'], c=rules['lift'], alpha=0.9, cmap='YlOrRd');
plt.title('Rules distribution color mapped by lift');
plt.xlabel('Support')
plt.ylabel('Confidence')
plt.colorbar();
# df.InvoiceDate = pd.to_datetime(df.InvoiceDate, format="%m/%d/%Y %H:%M")
df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'])

df['Revenue'] = df['Quantity']*df['UnitPrice']
linkcode
invoice_ct = df.groupby(by='CustomerID', as_index=False)['InvoiceNo'].count()
invoice_ct.columns = ['CustomerID', 'NumberOrders']
invoice_ct
unitprice = df.groupby(by='CustomerID', as_index=False)['UnitPrice'].mean()
unitprice.columns = ['CustomerID', 'Unitprice']
unitprice
revenue = df.groupby(by='CustomerID', as_index=False)['Revenue'].sum()
revenue.columns = ['CustomerID', 'Revenue']
revenue
total_items = df.groupby(by='CustomerID', as_index=False)['Quantity'].sum()
total_items.columns = ['CustomerID', 'NumberItems']
total_items
```

```
earliest_order = df.groupby(by='CustomerID', as_index=False)['InvoiceDate'].min()
earliest_order
earliest_order.columns = ['CustomerID', 'EarliestInvoice']
earliest_order['now'] = pd.to_datetime((df['InvoiceDate']).max())
linkcode
earliest_order
# == earliest_order['days_as_customer'] = 1 + (earliest_order.now-earliest_order.EarliestInvoice).dt.days
# Source: https://kite.com/python/docs/pandas.core.indexes.accessors.TimedeltaProperties
earliest_order['days_as_customer'] = 1 + (earliest_order['now']-earliest_order['EarliestInvoice']).dt.days
linkcode
earliest_order.drop('now', axis=1, inplace=True)
earliest_order
# when was their last order and how long ago was that from the last date in file (presumably
# when the data were pulled)
last_order = df.groupby(by='CustomerID', as_index=False)['InvoiceDate'].max()
last_order.columns = ['CustomerID', 'last_purchase']
last_order['now'] = pd.to_datetime((df['InvoiceDate']).max())
last_order['days_since_last_purchase'] = 1 + (last_order.now-last_order.last_purchase).astype('timedelta64[D]')
last_order.drop('now', axis=1, inplace=True)
last_order
#combine all the dataframes into one
import functools
dfs = [invoice_ct,unitprice,revenue,earliest_order,last_order,total_items]
CustomerTable = functools.reduce(lambda left,right: pd.merge(left,right,on='CustomerID', how='outer'), dfs)
CustomerTable['OrderFrequency'] = CustomerTable['NumberOrders']/CustomerTable['days_as_customer']
CustomerTable
CustomerTable.corr()['Revenue'].sort_values(ascending = False)
x = CustomerTable[['NumberOrders','Unitprice', 'days_as_customer', 'days_since_last_purchase', 'NumberItems', 'OrderFrequency']]
y = CustomerTable['Revenue']
reg = RandomForestRegressor()
reg.fit(x.values, y)
```

```
#list(zip(x, reg.feature_importances_))
coef = pd.Series(reg.feature_importances_, index = x.columns)

imp_coef = coef.sort_values()
imp_coef.plot(kind = "barh")
plt.title("Feature importance using Linear Model")
recency = df.groupby(by='CustomerID', as_index=False)['InvoiceDate'].max()
recency.columns = ['CustomerID', 'last_purchase']
recency['now'] = pd.to_datetime((df['InvoiceDate']).max())
recency['Recency'] = 1 + (recency.now-
recency['last_purchase']).astype('timedelta64[D]')
recency.drop(['now','last_purchase'], axis=1, inplace=True)
recency.head()
#check frequency of customer means how many transaction has been done..

frequency = df.copy()
frequency.drop_duplicates(subset=['CustomerID','InvoiceNo'], keep="first",
inplace=True)
frequency =
frequency.groupby('CustomerID',as_index=False)['InvoiceNo'].count()
frequency.columns = ['CustomerID','Frequency']
frequency.head()
monetary=df.groupby('CustomerID',as_index=False)['Revenue'].sum()
monetary.columns = ['CustomerID','Monetary']
monetary.head()
dfs = [recency, frequency, monetary]
rfm = functools.reduce(lambda left,right: pd.merge(left,right,on='CustomerID',
how='outer'), dfs)
rfm
#bring all the quartile value in a single dataframe
rfm_segmentation = rfm.copy()
rfm_segmentation
from sklearn.cluster import KMeans
SSE_to_nearest_centroid = []

for kinrange(1,15):
kmeans = KMeans(n_clusters=k)
kmeans.fit(rfm_segmentation)
SSE_to_nearest_centroid.append(kmeans.inertia_)
```

```
plt.figure(figsize=(20,8))
plt.plot(range(1,15),SSE_to_nearest_centroid,"-o")
plt.title("SSE / K Chart", fontsize=18)
plt.xlabel("Amount of Clusters",fontsize=14)
plt.ylabel("Inertia (Mean Distance)",fontsize=14)
plt.xticks(range(1,20))
plt.grid(True)
plt.show()
#fitting data in Kmeans theorem.
kmeans = KMeans(n_clusters=3, random_state=0).fit(rfm_segmentation)

# this creates a new column called cluster which has cluster number for each row respectively.
rfm_segmentation['cluster'] = kmeans.labels_
rfm_segmentation.head()
plt.figure(figsize=(8,5))
sns.boxplot(rfm_segmentation['cluster'],rfm_segmentation.Recency)

plt.figure(figsize=(8,5))
sns.boxplot(rfm_segmentation['cluster'],rfm_segmentation.Frequency)

plt.figure(figsize=(8,5))
sns.boxplot(rfm_segmentation['cluster'],rfm_segmentation.Frequency)
quantile = rfm.quantile(q=[0.25,0.5,0.75])
quantile
# lower the recency, good for store..
def RScore(x):
   if x <= quantile['Recency'][0.25]:
      return 1
elif x <= quantile['Recency'][0.50]:
      return 2
elif x <= quantile['Recency'][0.75]:
      return 3
   else:
      return 4

# higher value of frequency and monetary lead to a good consumer.
def FScore(x):
   if x <= quantile['Frequency'][0.25]:
```

```
        return 4
elif x <= quantile['Frequency'][0.50]:
        return 3
elif x <= quantile['Frequency'][0.75]:
        return 2
    else:
        return 1

def MScore(x):
    if x <= quantile['Monetary'][0.25]:
        return 4
elif x <= quantile['Monetary'][0.50]:
        return 3
elif x <= quantile['Monetary'][0.75]:
        return 2
    else:
        return 1
rfm_segmentation
rfm_segmentation['R_quartile'] = rfm_segmentation['Recency'].apply(RScore)
rfm_segmentation['F_quartile'] = rfm_segmentation['Frequency'].apply(FScore)
rfm_segmentation['M_quartile'] = rfm_segmentation['Monetary'].apply(MScore)
rfm_segmentation
# Approach 1: group customer's attributes, leading to detail customer's profile
# for example 121 and 112 are different.
rfm_segmentation['RFMScore'] = rfm_segmentation['R_quartile'].astype(str) \
                        + rfm_segmentation['F_quartile'].astype(str) \
                        + rfm_segmentation['M_quartile'].astype(str)
# Approach 2: group customer's attributes, leading to more general customers' profile
# for example 121 and 112 are the same.
rfm_segmentation['TotalScore'] = rfm_segmentation['R_quartile'] \
                        + rfm_segmentation['F_quartile'] \
                        + rfm_segmentation['M_quartile']
print("Best                                                    Customers:
",len(rfm_segmentation[rfm_segmentation['RFMScore']=='111']))
print('Loyal                                                   Customers:
',len(rfm_segmentation[rfm_segmentation['F_quartile']==1]))
print("Big                                                      Spenders:
",len(rfm_segmentation[rfm_segmentation['M_quartile']==1]))
```

```
print('Almost Lost: ', len(rfm_segmentation[rfm_segmentation['RFMScore']=='134']))
print('Lost Customers: ',len(rfm_segmentation[rfm_segmentation['RFMScore']=='344']))
print('Lost Cheap Customers: ',len(rfm_segmentation[rfm_segmentation['RFMScore']=='444']))

Image(url= "https://i.imgur.com/YmItbbm.png?")
rfm_segmentation.sort_values(by=['RFMScore', 'Monetary'], ascending=[True, False])
rfm_segmentation.groupby('RFMScore')['Monetary'].mean()
Score_Recency = rfm_segmentation.groupby('TotalScore')['Recency'].mean().reset_index()
Score_Monetatry = rfm_segmentation.groupby('TotalScore')['Monetary'].mean().reset_index()
Score_Frequency = rfm_segmentation.groupby('TotalScore')['Frequency'].mean().reset_index()
sns.barplot(x=Score_Recency['TotalScore'],y=Score_Recency['Recency'])

plt.figure(constrained_layout=True, figsize=(12, 4))

plt.subplot(1,2,1)
sns.barplot(x=Score_Frequency['TotalScore'],y=Score_Frequency['Frequency'])

plt.subplot(1,2,2)
sns.barplot(x=Score_Monetatry['TotalScore'],y=Score_Monetatry['Monetary'])
plt.subplots_adjust(wspace = 0.2)
def get_month(x):
   return dt.datetime(x.year, x.month, 1)
df['InvoiceMonth'] = df['InvoiceDate'].apply(get_month)
# https://stackoverflow.com/questions/27517425/apply-vs-transform-on-a-group-object
# explain the difference between  apply - transform. In this case, use transform for CohortMonth.
# CohortMonth: the first time a customer came to our retail store.
df['CohortMonth'] = df.groupby('CustomerID')['InvoiceMonth'].transform('min')
def get_date_int(df, column):
   year = df[column].dt.year
   month = df[column].dt.month
   day = df[column].dt.day
```

```
    return year, month, day
invoice_year, invoice_month, _ = get_date_int(df, 'InvoiceMonth')
cohort_year, cohort_month, _ = get_date_int(df, 'CohortMonth')

years_diff = invoice_year - cohort_year
months_diff = invoice_month - cohort_month

df['CohortIndex'] = years_diff * 12 + months_diff + 1

df.head()
## grouping customer berdasarkan masing masing cohort
cohort_data = df.groupby(['CohortMonth', 'CohortIndex'])['CustomerID'].nunique().reset_index()
# To solve the problem when ploting heatmap diagram below.
cohort_data['CohortMonth'] = cohort_data['CohortMonth'].dt.date
cohort_counts = cohort_data.pivot(index='CohortMonth', columns='CohortIndex', values='CustomerID')
cohort_counts
cohort_sizes = cohort_counts.iloc[:,0]
retention = cohort_counts.divide(cohort_sizes, axis=0)
retention.round(2) * 100
plt.figure(figsize=(15, 8))
plt.title('Retention rates')
sns.heatmap(data = retention,
annot = True,
fmt = '.0%',
vmin = 0.0, vmax = 0.5,
cmap = 'BuGn')
plt.show()
```

Chapter 4: Testing

The software system testing is done to check every feature and performance of the software after all its components have been integrated. It tests if the software is working properly as per the requirements and it is able to solve the customers' needs.

The software system testing is performed to detect any issues occurring after integration *of* multiple units of the software. It finds defects in the integrated software as well as in the complete software as a whole.

The software system testing is conducted on the complete software with the perspective of the system requirements, functional requirements, or both. It verifies the design and characteristics of the software, and how well it satisfies the end user requirements.

Sometimes the system testing validates the software beyond the requirements mentioned in System Requirement Specification(SRS). It is conducted by a testing team who is not a part of the development process, and hence have an unbiased testing mindset. It is a part of both functional and non-functional testing and is performed with the help of the black box testing techniques.

Process of Software System Testing

The process of the the software system testing are listed below –

Step 1 − Configure the test environment where the software system testing is to be performed.

Step 2 − Develop the software system test cases.

Step 3 − Generate the test data for running the software system test cases.

Step 4 − Execute the software system test cases and analyze the results.

Step 5 − In case of failure of the software system test cases, the defects are reported.

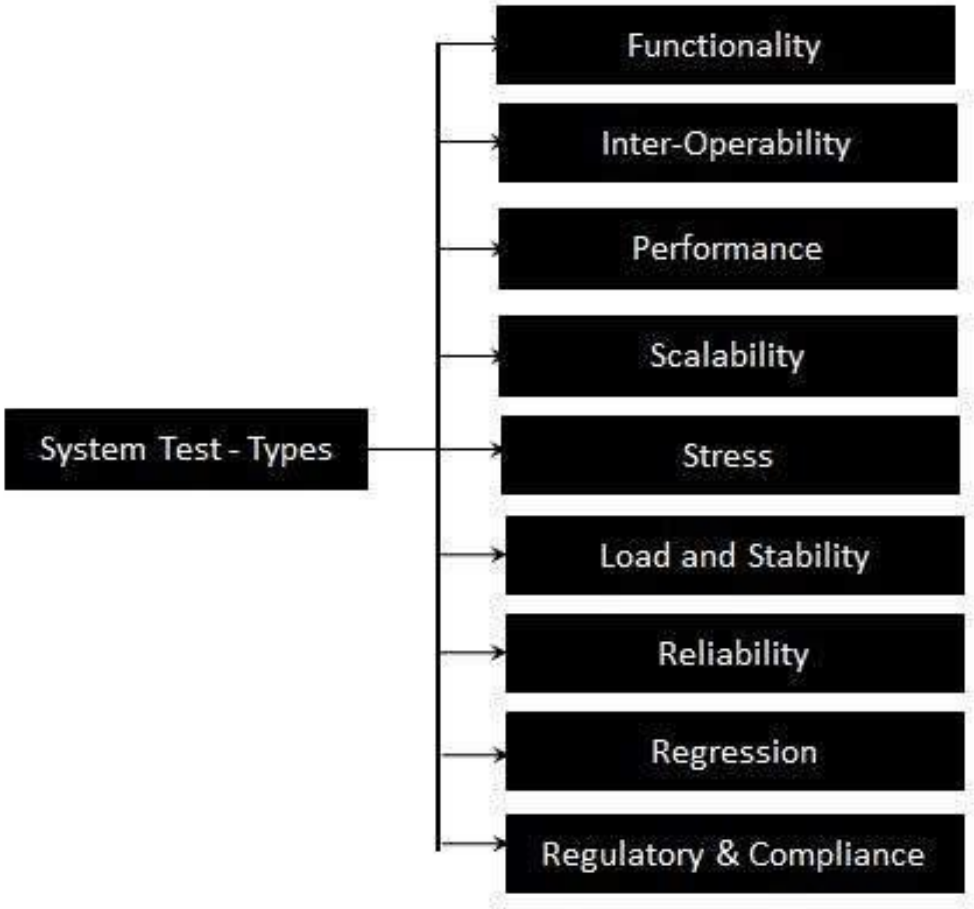
Step 6 − The entire regression test cases are executed to check if the existing functionalities of the software are working as expected.

Step 7 − Report all the regression related defects.

Step 8 − Retest all the fixed defects.

4.1 Testing Strategy:

The types of the software system testing are listed below –



Performance Testing

Performance Testing is performed to verify the performance, stability, reliability etc of the software.

Load Testing

Load Testing is performed to verify the amount of load or traffic that the software can accommodate before it crashes. It finds the threshold limit of the maximum count of users that the software can bear at a time after its software undergoes a breakdown.

Stress Testing

Stress Testing is performed to verify if there exists any security problems leading to the potential scope of hacking, and other vulnerabilities. It ensures that the safety of data is maintained while it is being exchanged between multiple units of the software. The stress testing is done along with the Penetration Testing and user access control testing techniques.

Scalability Testing

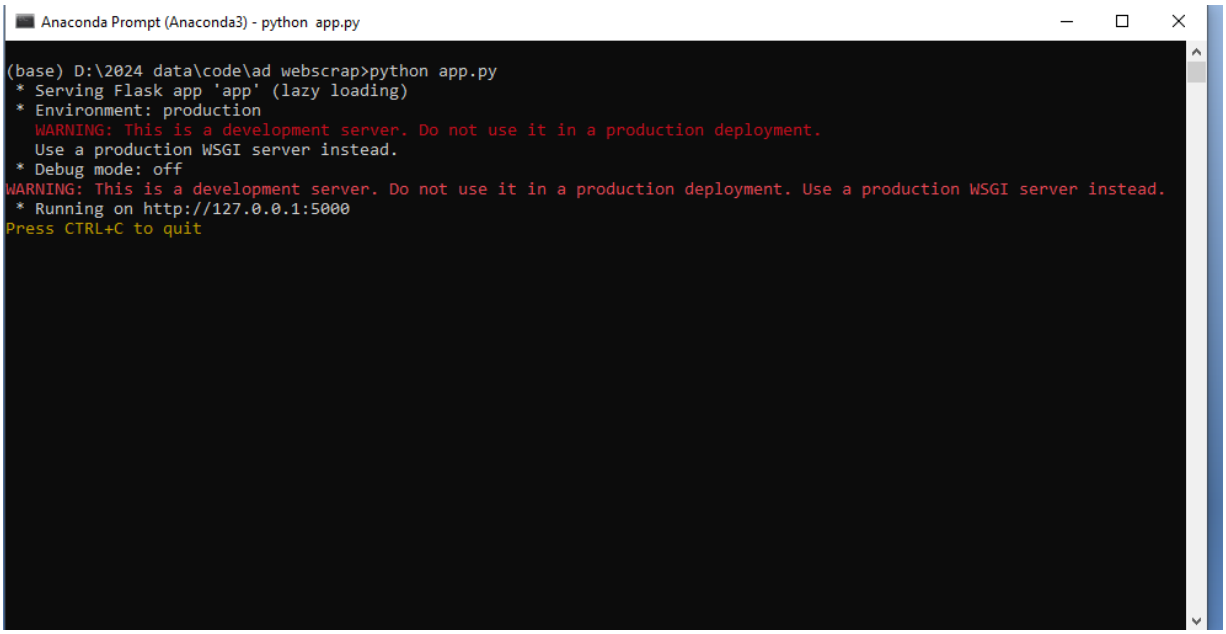
Scalability Testing is performed to verify the performance of the software with respect to its capacity to scale up or down the count of the user request loads.

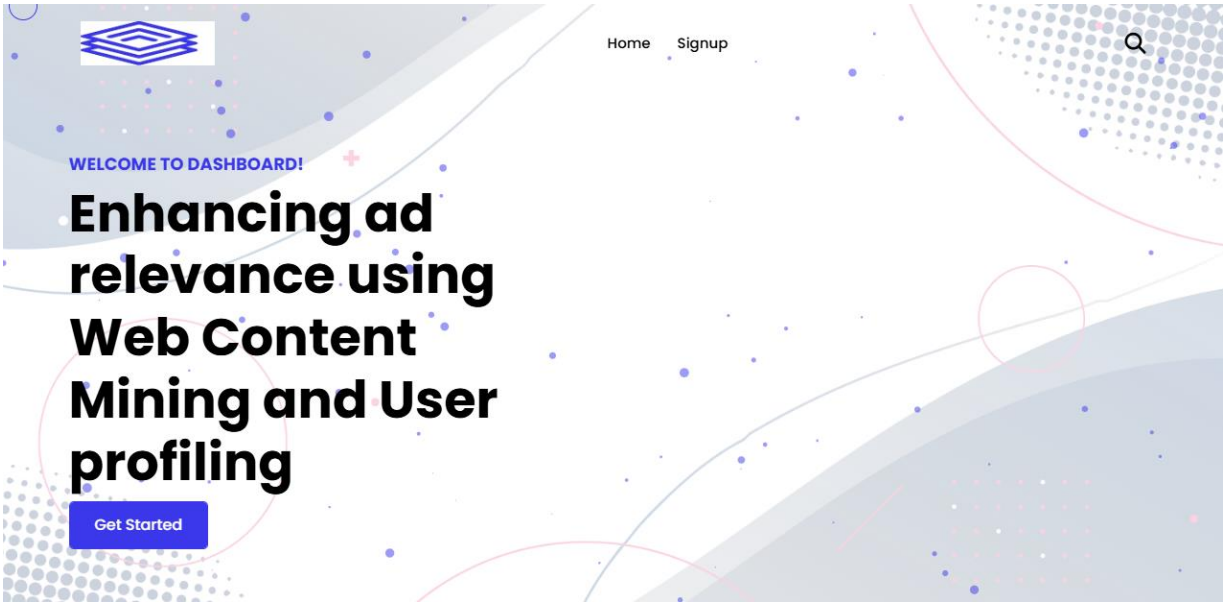
4.2 TEST CASES:

S.NO	INPUT	If available	If not available
1	User signup	User get registered into the application	There is no process
2	User signin	User get login into the application	There is no process
3	Enter input for prediction	Prediction result displayed	There is no process

Chapter 5: Results and Evaluation

SCREENS:





Sign up

Your Username

Your Name

Your Email

Your Phone Number

Password

Register

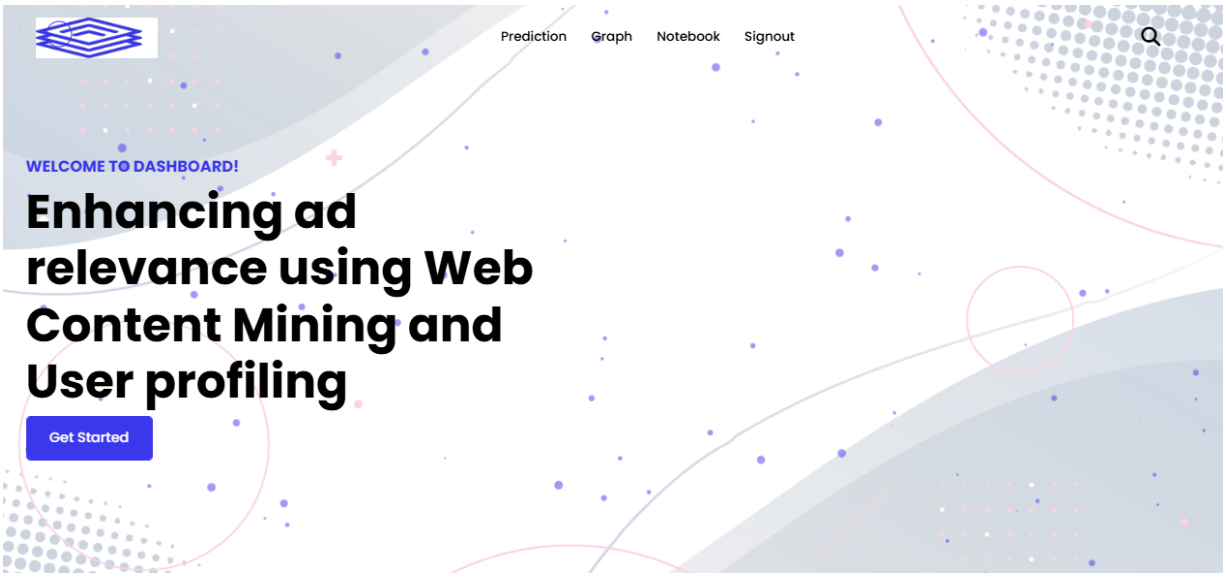
Already have an account? Sign in

Welcome Back

admin

Login

Register here! Sign Up



FORM

Daily Time Spent on Site:

21

Age:

15

Area Income:

25.25

Daily Internet Usage:

521

Gender:

1

Month:

12

Week :

51

Day:

2

Category Bank:

0

Category Electronic:

0

Category Fashion:

0

Category Finance:

0

Category Food:

0

Category Furniture:

0

Category Health:

0

Category House:

0

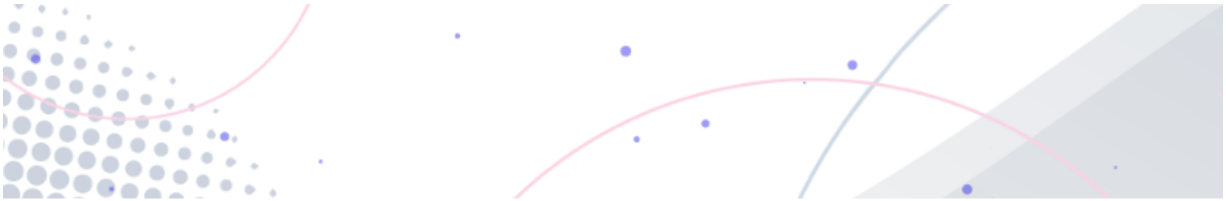
Category Otomotif:

0

Category Travel:

1

Predict



RESULT

They Clicked on AD

RESULT

They didn't Clicked on AD

Chapter 6: Conclusion and Future Scopes

This study successfully demonstrated the effectiveness of leveraging Web Content Mining and User Profiling to enhance advertisement relevance. By analyzing user behavior and ad interaction patterns from the Clicked Ads Dataset, a robust framework was developed, incorporating multiple machine learning algorithms to predict user preferences accurately. Among the implemented approaches, the Voting Classifier, integrating Adaboost and ExtraTree classifiers, achieved the highest performance, showcasing its ability to capture complex relationships within the data. This ensemble method effectively combined the strengths of individual models, delivering superior prediction accuracy and optimizing ad personalization. The results underscore the importance of advanced predictive techniques in improving ad targeting, ensuring higher user engagement, and maximizing advertising efficiency. By addressing the challenges of ad irrelevance and inefficiency, the proposed system provides a scalable solution for creating more meaningful and impactful digital marketing strategies. The findings highlight the potential of ensemble learning to transform user-centric advertisement delivery.

Future Scope:

Future work can explore incorporating additional advanced ensemble techniques and deep learning models to further enhance prediction accuracy and system robustness. Techniques such as stacking or gradient boosting could be investigated to capture even more complex patterns in user behavior and ad interactions. Enhancing feature engineering processes, including deriving higher-order features from existing data, can also improve model performance. Expanding the evaluation metrics and datasets will help refine the system, ensuring a more comprehensive approach to optimizing advertisement relevance.

References

[1]Barbosa, B., Saura, J. R., Zekan, S. B., & Ribeiro-Soriano, D. (2024). RETRACTED ARTICLE: Defining content marketing and its influence on online user behavior: a data-driven prescriptive analytics method. *Annals of Operations Research*, *337*(Suppl 1), 17-17.

[2] Wei, W., Ren, X., Tang, J., Wang, Q., Su, L., Cheng, S., ... & Huang, C. (2024, March). Llmrec: Large language models with graph augmentation for recommendation. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining* (pp. 806-815).

[3] Liao, S. H., Widowati, R., & Hsieh, Y. C. (2021). Investigating online social media users' behaviors for social commerce recommendations. *Technology in Society*, *66*, 101655.

[4] Samanta, D., Dutta, S., Galety, M. G., & Pramanik, S. (2022). A novel approach for web mining taxonomy for high-performance computing. In *Cyber Intelligence and Information Retrieval: Proceedings of CIIR 2021* (pp. 425-432). Springer Singapore.

[5] Hasan, M. K., Ghazal, T. M., Alkhalifah, A., Abu Bakar, K. A., Omidvar, A., Nafi, N. S., &Agbinya, J. I. (2021). Fischer linear discrimination and quadratic discrimination analysis–based data mining technique for internet of things framework for Healthcare. *Frontiers in Public Health*, *9*, 737149.

[6] Keqin Bao, Jizhi Zhang, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. 2023. TALLRec: An Effective and Efficient Tuning Framework to Align Large Language Model with Recommendation. arXiv preprint arXiv:2305.00447 (2023).

[7] Chong Chen, Weizhi Ma, Min Zhang, et al. 2023. Revisiting negative sampling vs. non-sampling in implicit recommendation. TOIS 41, 1 (2023), 1--25.

[8] Chong Chen, Min Zhang, Yongfeng Zhang, et al. 2020. Efficient neural matrix factorization without sampling for recommendation. TOIS 38, 2 (2020), 1--28.

[9]Mengru Chen, Chao Huang, Lianghao Xia, Wei Wei, et al. 2023. Heterogeneous graph contrastive learning for recommendation. In Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining. 544--552.

[10] Zheng Chen. 2023. PALR: Personalization Aware LLMs for Recommendation. arXiv preprint arXiv:2305.07622 (2023).

[11]Sunhao Dai, Ninglu Shao, Haiyuan Zhao, Weijie Yu, Zihua Si, Chen Xu, Zhongxiang Sun, Xiao Zhang, and Jun Xu. 2023. Uncovering ChatGPT's Capabilities in Recommender Systems. arXiv preprint arXiv:2305.02182 (2023).
[12]Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In ACM International World Wide Web Conference. 417--426.

[13] Xinyu Fu, Jiani Zhang, et al. 2020. Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding. In ACM International World Wide Web Conference. 2331--2341.

[14] He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. 2022. Masked autoencoders are scalable vision learners. In CVPR. 16000--16009.

[15] Ruining He and Julian McAuley. 2016. VBPR: visual bayesian personalized ranking from implicit feedback. In AAAI, Vol. 30.

[16] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In ACM SIGIR Conference on Research and Development in Information Retrieval. 639--648.

[17] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. arXiv preprint arXiv:1904.09751 (2019).

[18] Tinglin Huang, Yuxiao Dong, Ming Ding, Zhen Yang, Wenzheng Feng, Xinyu Wang, and Jie Tang. 2021. MixGCF: An Improved Training Method for Graph Neural Network-based Recommender Systems. In ACM SIGKDD Conference on Knowledge Discovery and Data Mining.

[19] Wang-Cheng Kang, Jianmo Ni, Nikhil Mehta, Maheswaran Sathiamoorthy, Lichan Hong, et al. 2023. Do LLMs Understand User Preferences? Evaluating LLMs On User Rating Prediction. arXiv preprint arXiv:2305.06474 (2023).

[20] Hyeyoung Ko, Suyeon Lee, Yoonseo Park, and Anna Choi. 2022. A survey of recommendation systems: recommendation models, techniques, and application fields. Electronics 11, 1 (2022), 141.