Online Shoppers Purchasing Intention

Abstract

Abstract - This research analyzed online shopper behavior using the Online Shoppers Purchasing Intention Dataset, applying both supervised (Decision Tree, Random Forest) and unsupervised (Kmeans clustering) machine learning methods. This research analyzes machine learning models to predict customer purchases using clickstream and supplementary customer dataIt emphasizes the role of features such as ExitRates and PageValues in improving prediction accuracy.It emphasizes the role of features such as ExitRates and PageValues in improving prediction accuracy. [1] Random forest was found to portray the best results, while K-means uncovered distinct browsing-based customer segments. Model performance was assessed with relevant metrics, offering valuable insights for e-commerce platforms to enhance user experience and target marketing strategies for better conversion rates.This research bridges data-driven behavioral analysis with practical applications for online retail optimization.

Irin Thomas
irinthomas0@gmail.com

INTRODUCTION:

In today's data-driven marketplace, understanding customer behavior is essential for e-commerce businesses aiming to enhance customer retention, boost sales, and maximize revenue. [2] This research focuses on analyzing customer purchase intentions using the UCI Online Shoppers Purchasing Intention Dataset, which captures user interaction data across over 12,000 browsing sessions, only a small fraction of which result in completed transactions. the dataset presents a realistic imbalanced classification challenge commonly faced in real-world e-commerce scenarios. To address this, the study applies a combination of supervised learning methods—Decision Tree and Random
Forest—and unsupervised clustering via Kmeans, all structured within the CRISP-DM framework. This dual analytical approach enables both predictive modeling of conversion likelihood and discovery of distinct shopper profiles based on interaction patterns.

The primary aim is to uncover behavioral patterns and identify influential factors that drive purchasing decisions. The insights derived from this analysis hold practical value for online retailers seeking to tailor user experiences, refine marketing strategies, and ultimately improve conversion rates.

II. LITERATURE REVIEW:

1. Decision Trees in Predicting Shopping Intention

Suchacka and Stemplewski [3] investigated how decision trees can be used to predict consumers' intention to shop online. Using a dataset of 305 participants engaged in grocery shopping, the study deployed the C5.0 decision tree algorithm. Results indicated a high prediction accuracy of 90%, emphasizing the strength of decision trees in modeling consumer behavior.This article was included for its direct alignment with this project's methodology, particularly its use of decision tree classification. It sets a benchmark for the effectiveness of decision trees in e-commerce analytics.

While the study demonstrates impressive accuracy, it is limited by its small, localized dataset and lack of model comparison. The project presented here expands on this by combining decision trees with unsupervised methods like K-Means Clustering to broaden the model perspective and generalizability.

2. Understanding consumer's internet purchase intention in Malaysia

Delafrooz, Paim, and Khatibi [4] explored how consumer attitudes in Malaysia affect their intention to shop online. The study applied the Technology Acceptance Model (TAM) and Theory of Reasoned Action (TRA), revealing that perceived usefulness, ease of use, and subjective norms significantly influence online shopping behavior. This source was selected to incorporate a behavioral and psychological viewpoint into the study, complementing the technical analysis. It bridges human-centered theories with technological acceptance, which are essential for understanding motivation beyond algorithmic output.

The primary limitation of this study is its exclusive use of surveys, lacking the integration of predictive modeling. This project addresses this gap by not only analyzing behavioral features but also applying machine learning techniques to predict outcomes based on such features.

3. Optimizing Classification Methods for Online Buyers' Purchase Intentions in Bangladesh

Hoque et al. [5]evaluated several machine learning classification techniques to predict online buyer behavior in Bangladesh. Among models tested—Logistic Regression, Decision Trees, Random Forest, and KNearest

Neighbors—the Random Forest algorithm delivered the highest accuracy: 99.9% (training) and 89.7% (testing). This study was selected for its comparative evaluation of algorithms and relevance to real-world classification challenges. It offers insight into advanced model performance in a developing e-commerce market.

Though powerful in predictive accuracy, the study neglects clustering or segmentation analysis—key components of this project. By combining supervised and unsupervised learning techniques, the current study contributes a more holistic approach to identifying and predicting user segments and intentions.

4. An Exploration of Clustering Algorithms for Customer Segmentation in the UK Retail Market

Customer segmentation is an essential approach in retail analytics, enabling businesses to tailor services and improve customer satisfaction. [6] This study evaluated several machine learning clustering techniques — including K-Means, GMM, DBSCAN, BIRCH, and Agglomerative — using the RFM (Recency, Frequency, Monetary) model and PCA to process a large UK retail dataset. Results indicated that the Gaussian Mixture Model performed best with a Silhouette Score of 0.80, outperforming other methods.

Prior research has utilized individual clustering techniques for segmentation, such as the ensemble clustering by Hicham and Karim [7], or the variable-based selection approach by Turkmen [8]. However, these studies lacked direct algorithm comparisons or dimensionality reduction integration like PCA, as done here.
While this study offers a broader comparison, it still leaves room for exploring more adaptive or hybrid models such as spectral clustering or deep learning-based segmentation.

Moreover, though historical works like those by Ramanathan et al. [9] and Oussous et al. [10] emphasize the role of Big Data and BI in understanding customer profiles, few link these technologies with advanced unsupervised learning frameworks. This gap indicates the need for research integrating newer AI models for real-time segmentation or contextual behavioral analysis.

5. Using DBSCAN to Identify Customer Segments with High Churn Risk on Amazon Consumer Behaviour Data.

This study applies the DBSCAN algorithm to segment Amazon customers based on variables like demographics, shopping behavior, reviews, and recommendation interactions [11]. DBSCAN was chosen due to its ability to detect clusters of varying densities and shapes without predefining the number of segments. The model identified groups with differing churn risks, providing actionable insights for improving customer retention strategies.

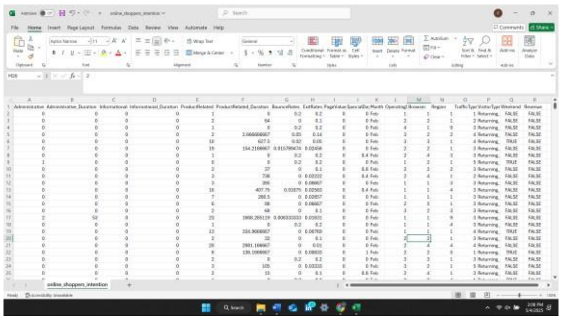
Previous literature has focused on outlier detection and clickstream data analysis However, few studies directly utilized DBSCAN for churn analysis, and even fewer addressed how demographic and behavioral attributes combine to indicate customer risk levels.

Although the paper effectively demonstrates DBSCAN's utility in detecting at-risk customers, it does not explore sentiment analysis or textual review mining. Future work could integrate NLP or deep clustering to enhance personalization and prediction.

III. METHODOLOGY:

This study utilizes the "Online Shoppers Purchasing Intention Dataset" from the UCI Machine Learning Repository, containing 12,330 browsing sessions and 18 attributes capturing user behavior as shown in **Figure**

1.



	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated	ProductRelated_Duration	BounceRates	ExitRates	PageValues	SpecialDa	Month	Operating	Browser	Region	TrafficType	VisitorType	Weekend	Revenue
2	0	0	0	0	1	0	0.2	0.2	0	0	Feb	1	1	1	1	Returning_	FALSE	FALSE
3	0	0	0	0	2	64	0	0.1	0	0	Feb	2	2	1	2	Returning_	FALSE	FALSE
4	0	0	0	0	1	0	0.2	0.2	0	0	Feb	4	1	9	3	Returning_	FALSE	FALSE
5	0	0	0	0	2	2.666666667	0.05	0.14	0	0	Feb	3	2	2	4	Returning_	FALSE	FALSE
6	0	0	0	0	10	627.5	0.02	0.05	0	0	Feb	3	3	1	4	Returning_	TRUE	FALSE
7	0	0	0	0	19	154.2166667	0.015789474	0.02456	0	0	Feb	2	2	1	3	Returning_	FALSE	FALSE
8	0	0	0	0	1	0	0.2	0.2	0	0.4	Feb	2	4	3	3	Returning_	FALSE	FALSE
9	1	0	0	0	0	0	0.2	0.2	0	0	Feb	1	2	1	5	Returning_	TRUE	FALSE
10	0	0	0	0	2	37	0	0.1	0	0.8	Feb	2	2	2	3	Returning_	FALSE	FALSE
11	0	0	0	0	3	738	0	0.02222	0	0.4	Feb	2	4	1	2	Returning_	FALSE	FALSE
12	0	0	0	0	3	395	0	0.06667	0	0	Feb	1	1	3	3	Returning_	FALSE	FALSE
13	0	0	0	0	16	407.75	0.01875	0.02583	0	0.4	Feb	1	1	4	3	Returning_	FALSE	FALSE
14	0	0	0	0	7	280.5	0	0.02857	0	0	Feb	1	1	1	3	Returning_	FALSE	FALSE
15	0	0	0	0	6	98	0	0.06667	0	0	Feb	2	5	1	3	Returning_	FALSE	FALSE
16	0	0	0	0	2	68	0	0.1	0	0	Feb	3	2	3	3	Returning_	FALSE	FALSE
17	2	53	0	0	23	1668.285119	0.008333333	0.01631	0	0	Feb	1	1	9	3	Returning_	FALSE	FALSE
18	0	0	0	0	1	0	0.2	0.2	0	0	Feb	1	1	4	3	Returning_	FALSE	FALSE
19	0	0	0	0	13	334.9666667	0	0.00769	0	0	Feb	1	1	1	4	Returning_	TRUE	FALSE
20	0	0	0	0	2	32	0	0.1	0	0	Feb	2	2	1	3	Returning_	FALSE	FALSE
21	0	0	0	0	20	2981.166667	0	0.01	0	0	Feb	2	4	4	4	Returning_	FALSE	FALSE
22	0	0	0	0	8	136.1666667	0	0.01667	0	1	Feb	2	2	5	1	Returning_	TRUE	FALSE
23	0	0	0	0	2	0	0.2	0.2	0	0	Feb	3	3	1	3	Returning_	FALSE	FALSE
24	0	0	0	0	3	105	0	0.03333	0	0	Feb	3	2	1	5	Returning_	FALSE	FALSE
25	0	0	0	0	2	15	0	0.1	0	0.8	Feb	2	4	1	3	Returning_	FALSE	FALSE

Figure 1-Online Shoppers Purchasing Intention Dataset

These attributes encompass a range of user interactions on an e-commerce platform, including metrics such as page views, time spent on pages, referral sources, and visitor categorization. The binary target variable, 'Revenue', indicates whether a session culminated in a transaction. Key features include categorical variables like Month, VisitorType, and Weekend, alongside numerical variables such as Administrative pages, Administrative duration, ProductRelated pages, ProductRelated duration, PageValues, ExitRates, and BounceRates.

Features	**Variable names**
Categorical features	Month, visitor type, Weekend
Numerical features	Administrative, Administrative Information, product-related, product-related duration, page values, exit rates, bounce rates
Target	Revenue

The project followed the CRISP-DM methodology. The **Business Understanding** stage focused on identifying patterns that predict purchase behavior to help ecommerce businesses improve conversion rates through personalized strategies. In the Data Understanding phase, we conducted initial exploration through descriptive statistics and correlation analyses. This involved checking the data structure using df.head() and df.info(), and examining descriptive statistics with df.describe(). Missing values were identified using df.isnull().sum().Exploratory data analysis revealed notable correlations between features such as ProductRelated and ProductRelated_Duration, as well as BounceRates and ExitRates. .During **Data Preparation**, we addressed missing values, encoded categorical variables (Month, VisitorType, Weekend), and applied feature scaling to standardize numerical measurementsFor **Modeling**, Decision Tree, Random Forest, and K-means clustering algorithms were selected to provide both predictive and segmentation insights. These models were chosen for their interpretability and suitability for classification and unsupervised grouping.The **Evaluation** phase involved assessing the performance of the classification models (Decision Tree and Random Forest) using relevant metrics such as accuracy, precision, recall, F1score, and AUC-ROC. The K-Means clustering was evaluated using the
Silhouette Score to determine the quality of the clusters. Although **Deployment** was not implemented, the findings can inform realworld applications such as targeted advertising and user experience
personalization. [12]

The Decision Tree and Random Forest models were chosen for their classification capabilities, while K-Means Clustering was used for customer segmentation based on behavioral patterns. Together, these models facilitate a comprehensive understanding of the data and provide actionable insights.

IV. RESULTS AND DISCUSSIONS:

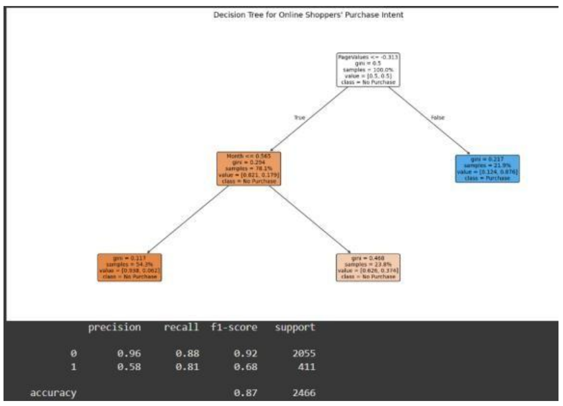
This study employs both supervised and unsupervised machine learning approaches to understand online shopping behavior. For supervised learning, we implemented Decision Tree and Random Forest classifiers to predict purchase intentions, while Kmeans clustering provided unsupervised segmentation of customer behavioral patterns.

The dataset was initially assessed for structure and cleanliness. No missing values were found, as shown in **Appendix Figure 1.** To prepare the data for analysis, categorical features were transformed using Label Encoding, while numerical features intended for clustering were standardized to ensure equal contribution. A correlation matrix was used to examine multicollinearity **as shown in Appendix Figure 2.** Features like ProductRelated and BounceRates, which showed high correlation (p > 0.7) with ProductRelated_Duration and ExitRates, respectively, were dropped to improve model performance.

Exploratory Data Analysis (EDA) uncovered several important trends. A correlation matrix was generated to identify highly correlated features. A notable finding was the seasonal pattern in purchases, with certain months showing higher conversion rates than others. Visuals such as histograms, boxplots, and line graphs illustrated that PageValues and ProductRelated_Duration were positively associated with purchases, while higher ExitRates were more frequent in nonpurchasing sessions, **as shown in Appendix Figure 3.** These patterns shaped feature selection for both supervised and unsupervised models.

SUPERVISED LEARNING RESULTS:

The Decision Tree classifier was implemented with strategic hyperparameter tuning—setting max_depth=5, applying cost-complexity pruning (ccp_alpha=0.01), and using class_weight='balanced' to address the dataset's inherent imbalance. The initial model achieved an accuracy of 0.82, which improved to 0.87 ,**as shown in Appendix Figure 6,**after tuning parameters as shown in **Figure 2.**

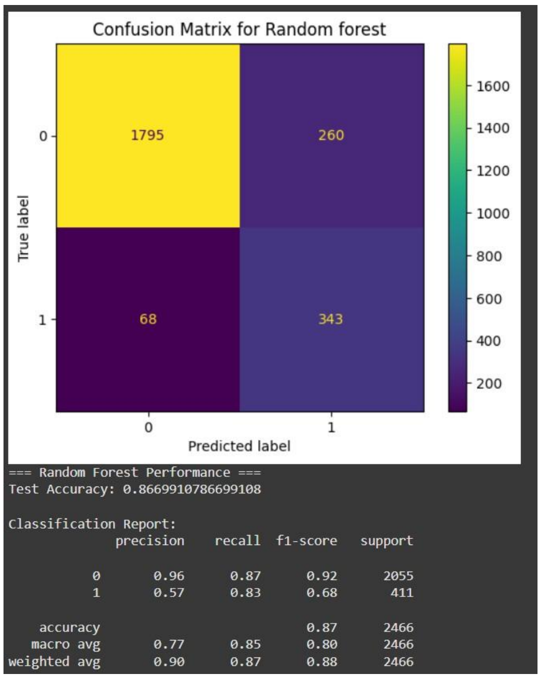


```
              precision    recall  f1-score   support

           0       0.96      0.88      0.92      2055
           1       0.58      0.81      0.68       411

    accuracy                           0.87      2466
```

Figure 2-Decision Tree and Classification report.

To better align with business needs, the decision threshold was raised from 0.5 to 0.7, enhancing **precision**, meaning predictions of "purchase" were more reliable even if recall decreased. Feature importance analysis revealed that PageValues was the most significant predictor, followed by ExitRates and ProductRelated_Duration, as shown in **Appendix Figure 4.**

An ensemble of 100 decision trees was trained (n_estimators=100, max_depth=5, min_samples_split=50) using all available CPU cores (n_jobs=-1). [13]It achieved an accuracy of 0.87, demonstrating enhanced reliability and robustness by averaging predictions across multiple trees. The model's performance was evaluated using metrics such as accuracy, precision, recall, F1-score as shown **in Appendix Figure 5,** and AUC-ROC curve. The **confusion matrix** and classification report confirmed that while false negatives were still present due to the class imbalance, overall model robustness and precision increased, making the Random Forest the superior predictive tool in this study as shown in **Figure 3.**



```
=== Random Forest Performance ===
Test Accuracy: 0.8669910786699108

Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.87      0.92      2055
           1       0.57      0.83      0.68       411

    accuracy                           0.87      2466
   macro avg       0.77      0.85      0.80      2466
weighted avg       0.90      0.87      0.88      2466
```

Figure 3-Confusion Matrix and Classification Report

Unsupervised Learning Results:

K-means clustering was applied to identify natural groupings in shopping behavior. To determine the optimal number of clusters, both the elbow method, **as shown in Appendix Figure 7,** and the silhouette score were used. Evaluating cluster quality using the Silhouette Score is recommended by Lleti et al. (2004). [14] The optimal "k" is the one that results in the highest Silhouette score; thus, k=2 was selected as the optimal number of clusters, as shown in **Figure 4.**

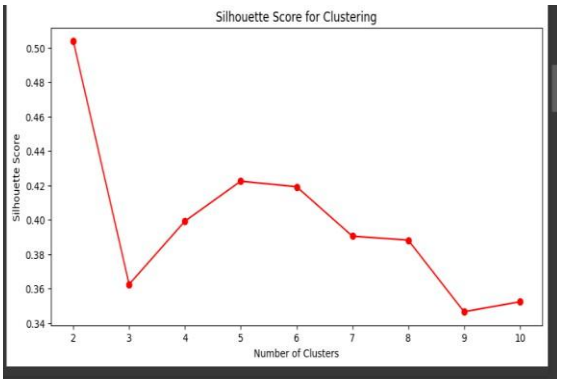


Figure 4-Optimal K with Silhouette score

The clustering analysis utilized features like PageValues, ProductRelated_Duration, and ExitRates, excluding highly correlated features. A correlation matrix was used, and features with a correlation coefficient above 0.7 were considered highly correlated,**as shown in Appendix Figure 8.**

The K-Means algorithm converged after 28 iterations, resulting in two distinct shopper segments. The characteristics of these segments are visualized in 2D , **as shown in Appendix Figure 9**,and 3D plots as shown in **Figure 5.**

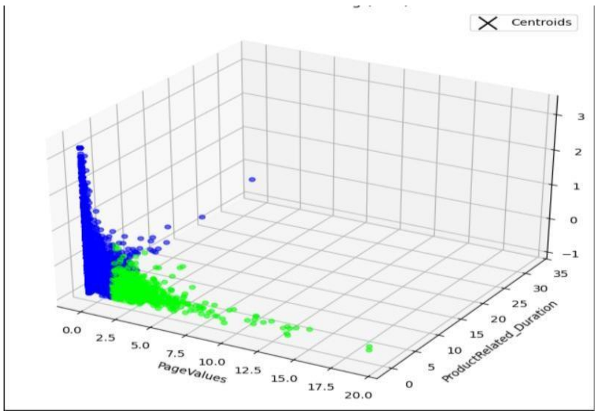


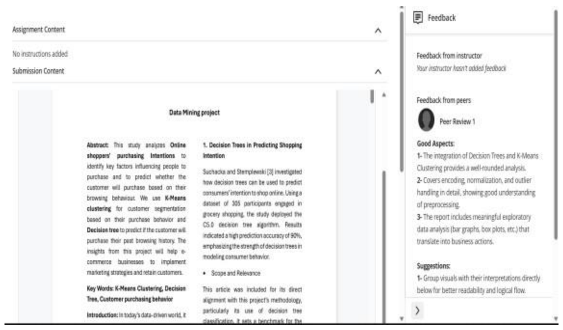
Figure 5-3D plot of clusters

The datapoints in the dataset where mainly 0s and 1s and after standardization all the data points where found to be in between the range of 0 and 1 which made the cluster look like it is grouped in a singe area and not separated.The analysis of the **Cluster 1 (Majority 85% of shoppers, i.e., blue cluster)** revealed that the majority of shoppers exhibit lower 'PageValues', 'ProductRelated_Duration', and higher 'ExitRates' compared to Cluster 2. This suggests a large group of users who exhibit less engagement with the website, potentially representing casual browsers or window shoppers with a higher propensity to abandon their sessions. While **Cluster 2 (Minority, i.e., green cluster)** is smaller (approximately 15%) displayed significantly higher PageValues and ProductRelated_Duration, indicative of a more engaged segment actively exploring product details. However, they also exhibit higher 'ExitRates'. This could suggest that while they are interested in products, they may encounter issues during the checkout process or find the website navigation challenging.

Results: The findings from both the supervised and unsupervised analyses provide a holistic view of online shopper behavior. The most significant insights from this integrated analysis can be summed up to say that Two distinct shopping behavior patterns naturally emerge from the data.ageValues serves as the strongest purchase predictor.User engagement metrics like ProductRelated_Duration significantly impact purchase decisions.Exit rates are inversely related to purchase probability. The identified shopper segments allow for tailored strategies, such as targeted marketing campaigns for the highly engaged segment and website improvements to reduce exit rates for both groups. The validation of the supervised models through standard metrics provides confidence in the reliability of the identified predictors.

V. PEER REVIEW:

The peer review, as shown in **Figure 6**, provided valuable insights that improved my analysis and communication. I implemented several valuable suggestions. Grouping visuals with their interpretations greatly improved readability, while including specific accuracy and silhouette scores strengthened the evaluation section. This process also encouraged me to adopt a more concise writing style, enhancing clarity and accessibility.


Assignment Content

No instructions added

Submission Content

Data Mining project

Abstract: This study analyzes **Online shoppers' purchasing Intentions** to identify key factors influencing people to purchase and to predict whether the customer will purchase based on their browsing behaviour. We use **K-Means clustering** for customer segmentation based on their purchase behavior and **Decision tree** to predict if the customer will purchase their past browsing history. The insights from this project will help e-commerce businesses to implement marketing strategies and retain customers.

Key Words: K-Means Clustering, Decision Tree, Customer purchasing behavior

Introduction: In today's data-driven world, it

1. Decision Trees in Predicting Shopping Intention

Suchacka and Stemplewski [3] investigated how decision trees can be used to predict consumers' intention to shop online. Using a dataset of 305 participants engaged in grocery shopping, the study deployed the C5.0 decision tree algorithm. Results indicated a high prediction accuracy of 90%, emphasizing the strength of decision trees in modeling consumer behavior.

- Scope and Relevance

This article was included for its direct alignment with this project's methodology, particularly its use of decision tree classification. It sets a benchmark for the

Feedback

Feedback from instructor

Your instructor hasn't added feedback

Feedback from peers

Peer Review 1

Good Aspects:

1- The integration of Decision Trees and K-Means Clustering provides a well-rounded analysis.

2- Covers encoding, normalization, and outlier handling in detail, showing good understanding of preprocessing.

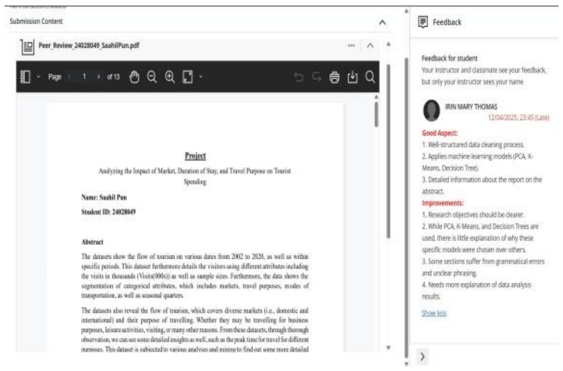
3- The report includes meaningful exploratory data analysis (bar graphs, box plots, etc.) that translate into business actions.

Suggestions:

1- Group visuals with their interpretations directly below for better readability and logical flow.

Figure 6-Feedback that I got

In reflecting on my feedback to a peer, as shown in **Figure 7,** I emphasized the importance of clearer research objectives and stronger justification for model choices. This helped me appreciate the need for clear communication and grammatical accuracy in presenting data analysis.


Submission Content

Peer_Review_24028049_SaahilPun.pdf

Page 1 of 13

Project

Analyzing the Impact of Market, Duration of Stay, and Travel Purpose on Tourist Spending

Name: Saahil Pun

Student ID: 24028049

Abstract

The datasets show the flow of tourism on various dates from 2002 to 2020, as well as within specific periods. This dataset furthermore details the visitors using different attributes including the visits in thousands (Visits(000s)) as well as sample sizes. Furthermore, the data shows the segmentation of categorical attributes, which includes markets, travel purposes, modes of transportation, as well as seasonal quarters.

The datasets also reveal the flow of tourism, which covers diverse markets (i.e., domestic and international) and their purpose of travelling. Whether they may be travelling for business purposes, leisure activities, visiting, or many other reasons. From these datasets, through thorough observation, we can see some detailed insights as well, such as the peak time for travel for different purposes. This dataset is subjected to various analyses and mining to find out some more detailed

Feedback

Feedback for student

Your instructor and classmate see your feedback, but only your instructor sees your name

IRIN MARY THOMAS

12/04/2025, 23:45 (Late)

Good Aspect:

1. Well-structured data cleaning process.

2. Applies machine learning models (PCA, K-Means, Decision Tree).

3. Detailed information about the report on the abstract.

Improvements:

1. Research objectives should be clearer.

2. While PCA, K-Means, and Decision Trees are used, there is little explanation of why these specific models were chosen over others.

3. Some sections suffer from grammatical errors and unclear phrasing.

4. Needs more explanation of data analysis results.

Show less

Figure 7-The feedback that I provided

Overall, the peer review process enhanced my ability to structure my findings effectively and communicate them clearly, contributing to my professional development in data science.

VI. CONCLUSION AND RECOMMENDATIONS:

This study successfully implemented and analyzed both supervised and unsupervised machine learning models to predict online shopping purchase intentions and identify distinct customer segments. The primary objective was to uncover key predictors of purchase intention and identify customer segments and was effectively achieved. The analysis of the UCI Online Shoppers Purchasing

Intention dataset yielded several valuable insights.

Firstly the supervised models particularly the Decision Tree and Random Forest, identified most influential predictors of purchase intention being 'PageValues' which emerged as the dominant predictor and highlighting the importance of presenting high-value content, 'ProductRelated_Duration' demonstrated that meaningful engagement with product information significantly increases conversion likelihood, and 'ExitRates' revealed potential friction points that deter purchases.Secondly, the K-means clustering effectively segmented the users into two primary groups which is: a large segment of less engaged, casual browsers (Cluster 1), a smaller but significant segment of highly engaged potential buyers (Cluster 2) who, despite showing strong initial interest, exhibit higher exit rates, suggesting potential friction points in their purchasing journey. Finally, the Random Forest model demonstrated a marginal but noteworthy improvement in predictive performance (accuracy = 0.89, F1-score = 0.66) over the Decision Tree (accuracy = 0.87, F1-score = 0.61), confirming our hypothesis that ensemble methods would yield better results for this complex prediction task.

For future work, several recommendations can be made, like implementing more advanced ensemble methods, like Gradient Boosting or XGBoost to potentially improve prediction accuracy. Exploring feature engineering to create more predictive variables, especially regarding temporal aspects of user behaviour. Also prioritize improving high-exit pages, especially for Cluster 2 users who show high intent but abandon sessions mid-funnel. Additionally, applying more sophisticated clustering techniques like DBSCAN or hierarchical clustering can help identify more nuanced customer segments.

Peer feedback was valuable in improving both the structure and clarity of the report. By pairing each visual directly with its interpretation, the flow became more intuitive and easier to follow. Incorporating specific metrics such as accuracy and silhouette scores added precision to model evaluations. Finally, tightening the language to reduce repetition helped create a more engaging and professional narrative.This feedback cycle has enhanced my ability to communicate technical findings clearly which is an essential skill in collaborative data science environments and has reinforced the importance of presenting insights in a concise, actionable format.

VII. REFERENCES

[1] M. .Anitha, K. B. Ramya and M. Zeenath, “Predicting Online Shopping Behavior Through Clickstream,” *International Journal of Advance Research, Science and Technology,* vol. 14, no. 5, pp. 20-30, 2024.

[2] Y. Zhao, K. Cormican and S. Sampaio, “Clicks vs. bricks: Exploring the critical success factors for consumer purchase intention in e-commerce,” *Procedia Computer Science,* vol. 239, pp. 590-597, 2024.

[3] D. Šebalj, J. Franjković and K. Hodak, “Shopping intention prediction using decision trees,” *Millenium - Journal of Education, Technologies, and Health,* Vols. 2,no.4, no. 10.29352/mill0204.01.0015., pp. 1322, 2017.

[N. Delafrooz, L. Paim and A. Khatibi,

[4] "Understanding consumer's internet purchase intention in Malaysia," *ReasearchGate,* vol. 5, p. 11, 2011.

[5] I. Ahmed, M. M. Hoque, N. banik, A. Rahman, M. Nur-E-Alam and M. A. Islam, "Optimizing classification methods for online buyers' purchase intentions in Bangladesh," *International Journal of Recent Technology and Engineering (IJRTE),* vol. 12, no. 10.35940/ijrte.E7987.12060324, pp. 2277-3878, 2024.

[6] J. J. John, O. Shobayo and B. Ogunleye, "An Exploration of Clustering Algorithms for Customer Segmentation in the UK Retail Market," *Analytics,* vol. 2, no. 10.3390/analytics2040042, pp. 809823, 2023.

[7] H. N. and S. Karim, "Analysis of Unsupervised Machine Learning Techniques for an Efficient Customer Segmentation Using Clustering Ensemble," *International Journal of Advanced Computer Science and Applications (IJACSA),* vol. 13, pp. 122-130, 2022.

[8] B. Turkmen, "Customer Segmentation with Machine Learning for Online Retail Industry," *European Journal of Social and Behavioural Sciences,* vol. 31, no. 2022, pp. 111136, 2022.

[9] U. Ramanathan, N. Subramanian, W. Yu and R. Vijaygopal, " Impact of Customer Loyalty and Service Operations on Customer Behaviour and Firm Performance," *Production Planning & Control,* vol. 28, no. 2017, pp. 478-488, 2017.

[10] A. Oussous, F. Z. Benjelloun, A. A. Lahcen and S. Belfkih, "Big Data Technologies: A Survey," *Journal of King Saud University - Computer and Information Sciences,* vol. 30, no. 4, pp. 431-448, 2018.

[11] G. A and R. Syam, " Using DBSCAN to Identify Customer Segments with High Churn Risk on Amazon Consumer Behavior Data," *TechRxiv (Preprint),* vol. 1, no. 0.36227/techrxiv.170619281.19861929.v1, January 2024.

[12] C. Shearer, "The CRISP-DM Model: The New Blueprint for Data Mining," *Journal of Data Warehousing,* vol. 5, no. 4, pp. 13-23, 2000.

[13] S. Saxena, "A Beginner's Guide to Random Forest Hyperparameter Tuning," Analytics Vidya, 17 December 2024. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/03/beginners-guiderandom-forest-hyperparametertuning/. [Accessed 15 April 2024].

[14] M. Sánchez, L. Sarabia, M. C. Ortiz and R. Lleť ı, "Selecting variables for k-means cluster analysis by using a genetic algorithm that optimises the silhouettes," *ScienceDirect,* vol. 515, no. 1, pp. 87-100, 2004.

[15] C. Sakar and Y. Kastro, "Online Shoppers Purchasing Intention Dataset," UCI Machine Learning Repository , 30 8 2018. [Online].

Available:
https://archive.ics.uci.edu/dataset/468/online+shoppers+purchasing+intention+dataset. [Accessed 23 03 2024].

[1] V. Winland and E. Kavlakoglu, "What is k-means clustering?," IBM, 26 6 2024. [Online]. Available: https://www.ibm.com/think/topics/k-means-clustering. [Accessed 29 3 2025].

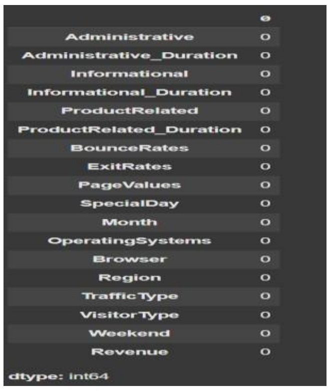
VIII. APPENDIX:

K-Means Clustering Code link: https://colab.research.google.com/drive

/12Chm3RPyZqmrIMRHdVSG03gz1VtXMre?usp=sharing

Decision Tree and Random Fores Code Link: https://colab.research.google.com/drive/1zaKeMWl7xKLBccWVVkvB2KbXDZSak6X0?usp=sharing

Figure 1. NO Null values



```
                           0
Administrative             0
Administrative_Duration    0
Informational              0
Informational_Duration     0
ProductRelated             0
ProductRelated_Duration    0
BounceRates                0
ExitRates                  0
PageValues                 0
SpecialDay                 0
Month                      0
OperatingSystems           0
Browser                    0
Region                     0
TrafficType                0
VisitorType                0
Weekend                    0
Revenue                    0

dtype: int64
```

Figure 2: Feature Correlation Heatmap

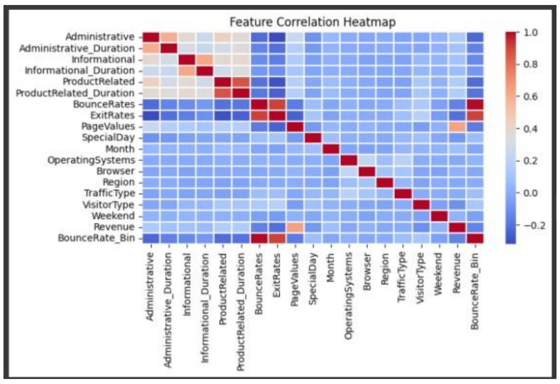


Figure 3. BOX Plot for Average Exit Rates on High Value pages

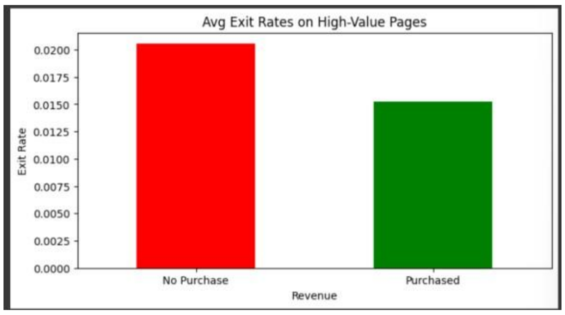
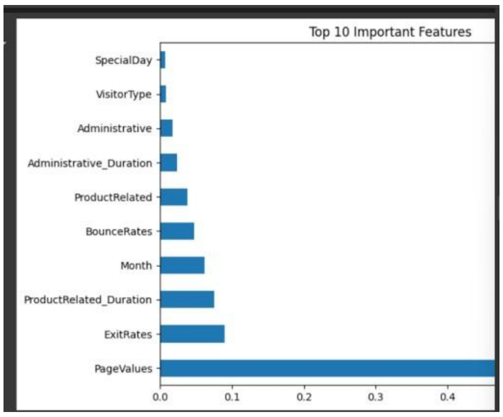
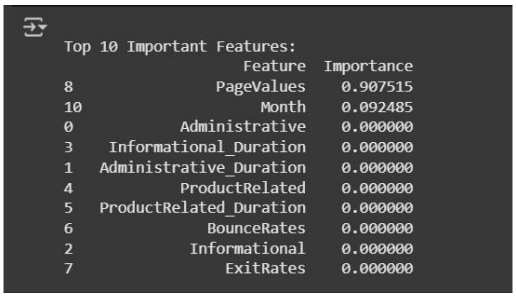


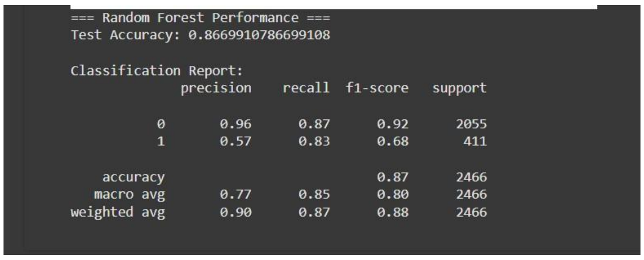
Figure 4. Top 10 Important Feature





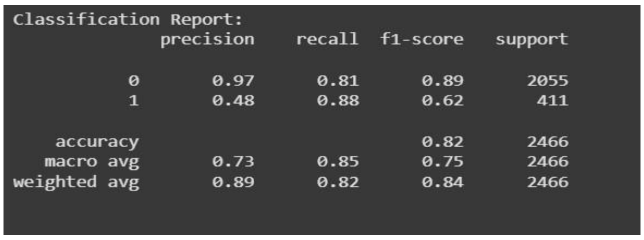
```
Top 10 Important Features:
                    Feature  Importance
8                PageValues    0.907515
10                    Month    0.092485
0            Administrative    0.000000
3    Informational_Duration    0.000000
1   Administrative_Duration    0.000000
4            ProductRelated    0.000000
5   ProductRelated_Duration    0.000000
6               BounceRates    0.000000
2             Informational    0.000000
7                 ExitRates    0.000000
```

Figure 5. Accuracy, Precision, Recall, F1score Report for Randon forest

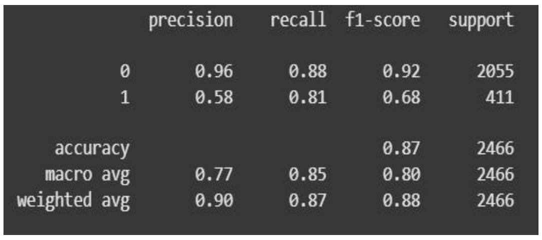


```
=== Random Forest Performance ===
Test Accuracy: 0.8669910786699108

Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.87      0.92      2055
           1       0.57      0.83      0.68       411

    accuracy                           0.87      2466
   macro avg       0.77      0.85      0.80      2466
weighted avg       0.90      0.87      0.88      2466
```

Figure 6. Classification report for Decision Tree before tuning and after tuning



```
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.81      0.89      2055
           1       0.48      0.88      0.62       411

    accuracy                           0.82      2466
   macro avg       0.73      0.85      0.75      2466
weighted avg       0.89      0.82      0.84      2466
```



```
              precision    recall  f1-score   support

           0       0.96      0.88      0.92      2055
           1       0.58      0.81      0.68       411

    accuracy                           0.87      2466
   macro avg       0.77      0.85      0.80      2466
weighted avg       0.90      0.87      0.88      2466
```

Figure 7. Elbow Method for K- Means Clustering

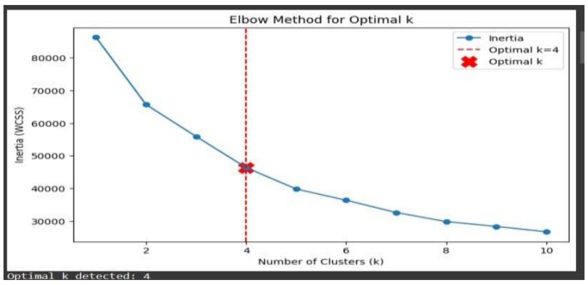


Figure 8. Correlation Matrix for K-means Clustering

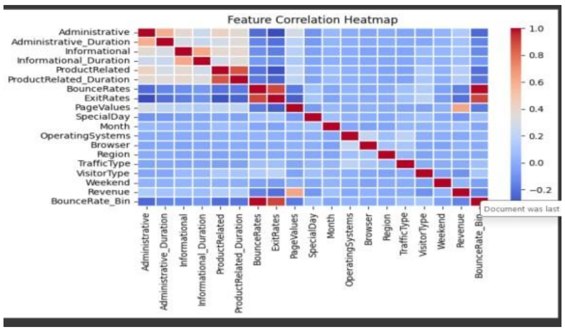
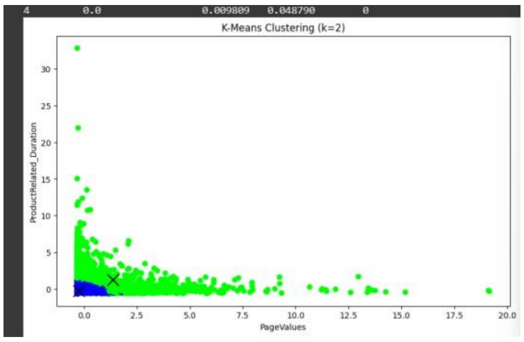


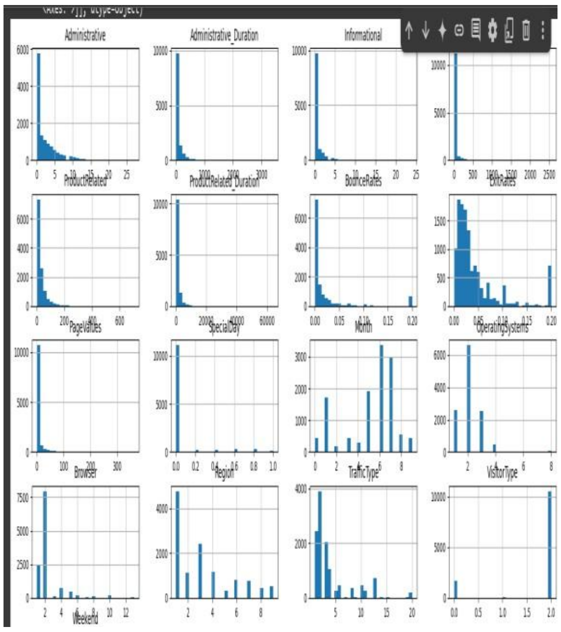
Figure 9. 2D visual of Cluster with k=2



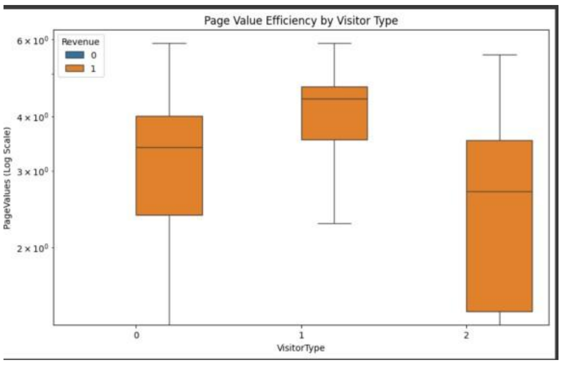
Other Relevant Figures not included in the report:

K-Means Clustering:

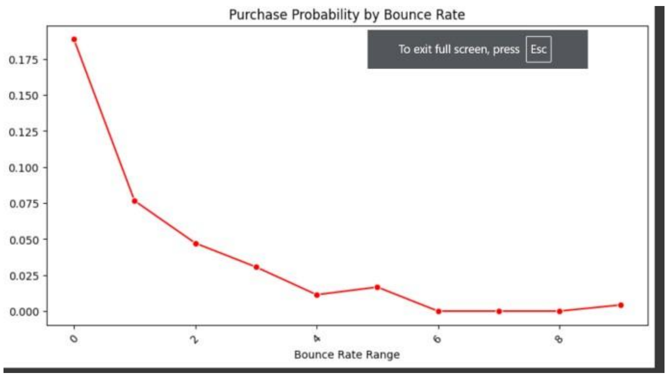
1. Histogram of all variables to find outliers



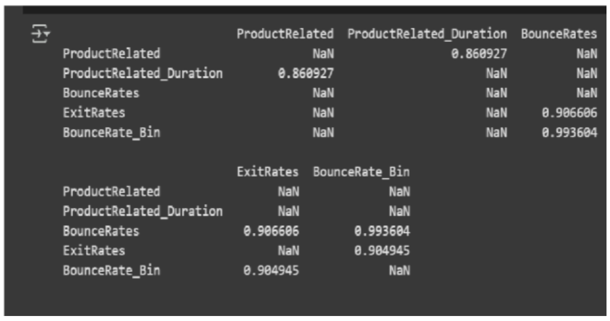
2. Comparing Page value with Histogram:



3. Purchase probability by bounce rates:

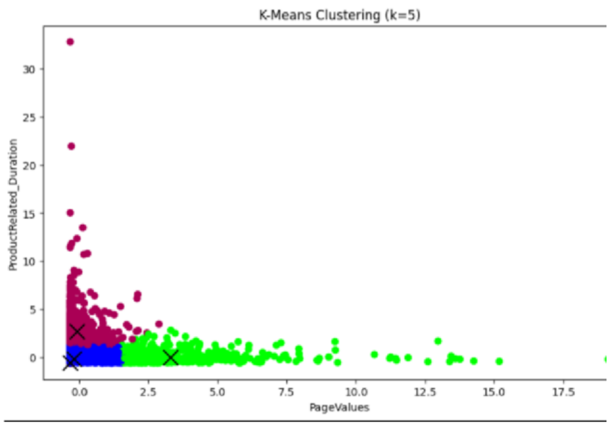


4. Finding the highly correlated variables:

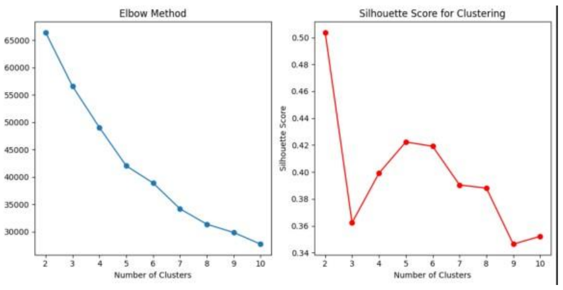


```
                         ProductRelated  ProductRelated_Duration  BounceRates
ProductRelated                      NaN                 0.860927          NaN
ProductRelated_Duration        0.860927                      NaN          NaN
BounceRates                         NaN                      NaN          NaN
ExitRates                           NaN                      NaN     0.906606
BounceRate_Bin                      NaN                      NaN     0.993604

                         ExitRates  BounceRate_Bin
ProductRelated                 NaN             NaN
ProductRelated_Duration        NaN             NaN
BounceRates               0.906606        0.993604
ExitRates                      NaN        0.904945
BounceRate_Bin            0.904945             NaN
```

5. 2D plot of cluster when k =4

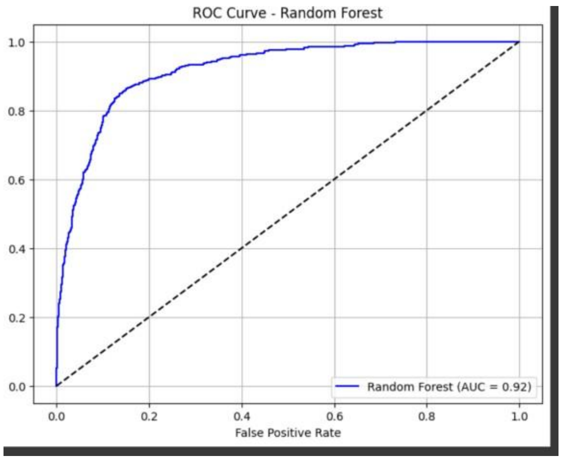


6. Comparison of K value with the elbow method and Silhoutte Score

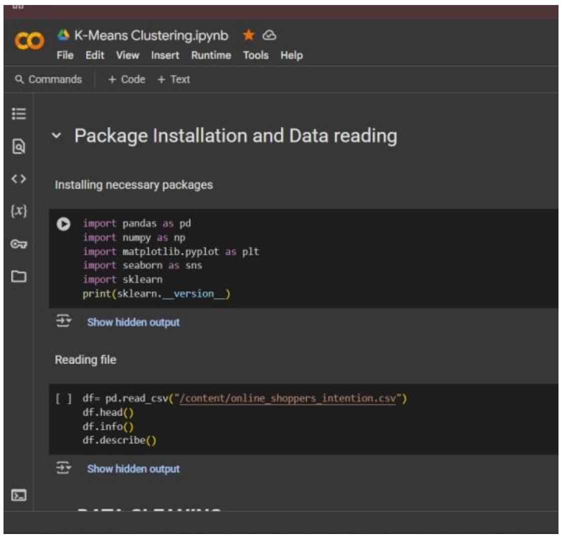


Random Forest:

7.ROC Curve for Random Forest:



K-Means Clustering Codes:



Package Installation and Data reading

Installing necessary packages

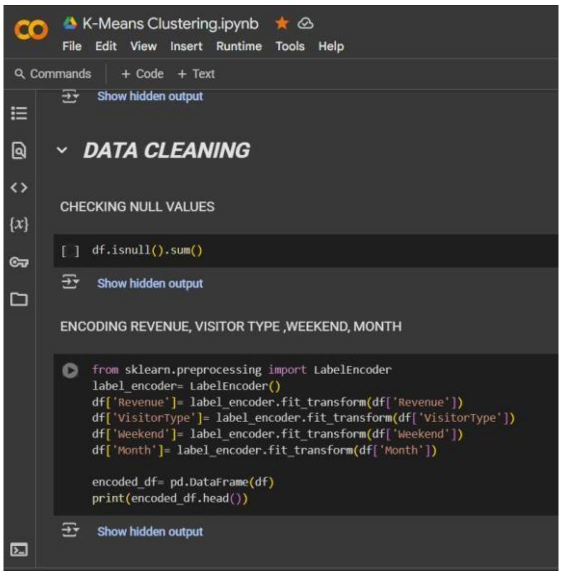
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
print(sklearn.__version__)
```

Show hidden output

Reading file

```python
df= pd.read_csv("/content/online_shoppers_intention.csv")
df.head()
df.info()
df.describe()
```

Show hidden output



Show hidden output

DATA CLEANING

CHECKING NULL VALUES

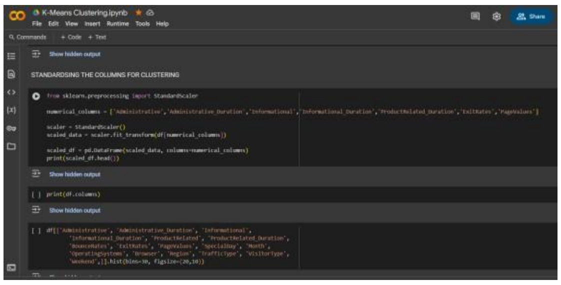
```python
df.isnull().sum()
```

Show hidden output

ENCODING REVENUE, VISITOR TYPE ,WEEKEND, MONTH

```python
from sklearn.preprocessing import LabelEncoder
label_encoder= LabelEncoder()
df['Revenue']= label_encoder.fit_transform(df['Revenue'])
df['VisitorType']= label_encoder.fit_transform(df['VisitorType'])
df['Weekend']= label_encoder.fit_transform(df['Weekend'])
df['Month']= label_encoder.fit_transform(df['Month'])

encoded_df= pd.DataFrame(df)
print(encoded_df.head())
```

Show hidden output

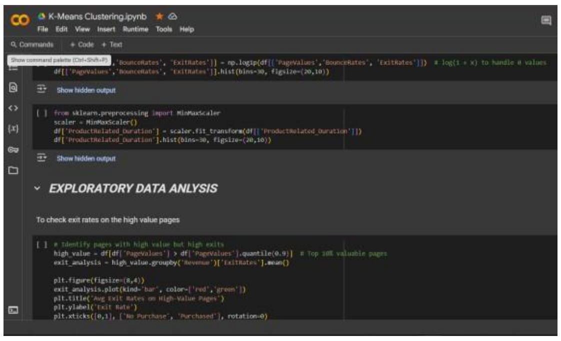


STANDARDSING THE COLUMNS FOR CLUSTERING

```python
from sklearn.preprocessing import StandardScaler

numerical_columns = ['Administrative','Administrative_Duration','Informational','Informational_Duration','ProductRelated_Duration','ExitRates','PageValues']

scaler = StandardScaler()
scaled_data = scaler.fit_transform(df[numerical_columns])

scaled_df = pd.DataFrame(scaled_data, columns=numerical_columns)
print(scaled_df.head())
```

```python
print(df.columns)
```

```python
df[['Administrative', 'Administrative_Duration', 'Informational',
        'Informational_Duration', 'ProductRelated', 'ProductRelated_Duration',
        'BounceRates', 'ExitRates', 'PageValues', 'SpecialDay', 'Month',
        'OperatingSystems', 'Browser', 'Region', 'TrafficType', 'VisitorType',
        'Weekend',]].hist(bins=30, figsize=(20,10))
```

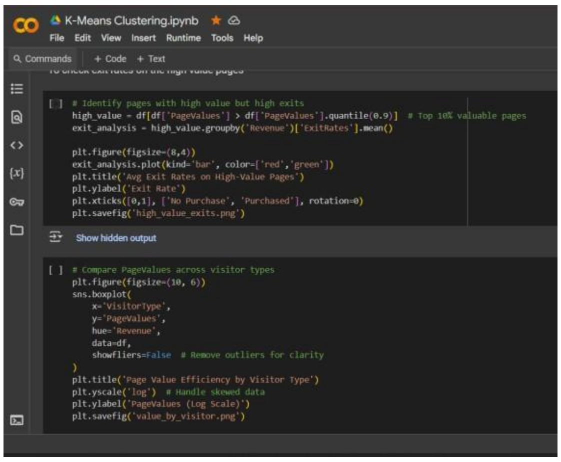


```python
,'BounceRates', 'ExitRates']] = np.log1p(df[['PageValues','BounceRates', 'ExitRates']])  # log(1 + x) to handle 0 values
    df[['PageValues','BounceRates', 'ExitRates']].hist(bins=30, figsize=(20,10))
```

```python
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
df['ProductRelated_Duration'] = scaler.fit_transform(df[['ProductRelated_Duration']])
df['ProductRelated_Duration'].hist(bins=30, figsize=(20,10))
```

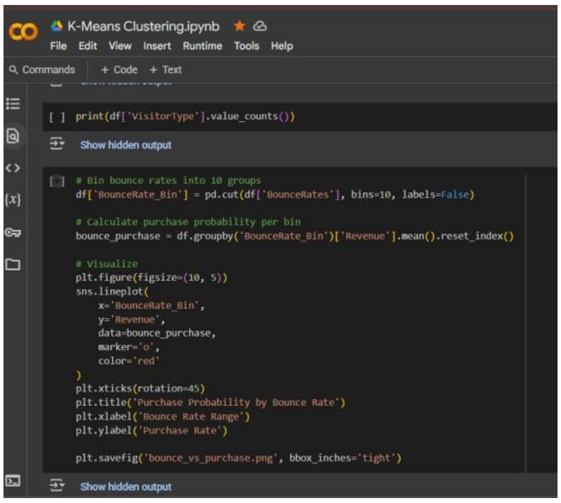
EXPLORATORY DATA ANLYSIS

To check exit rates on the high value pages



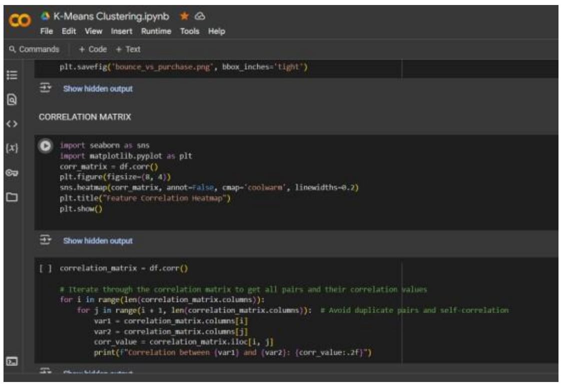
```python
# Identify pages with high value but high exits
high_value = df[df['PageValues'] > df['PageValues'].quantile(0.9)]  # Top 10% valuable pages
exit_analysis = high_value.groupby('Revenue')['ExitRates'].mean()

plt.figure(figsize=(8,4))
exit_analysis.plot(kind='bar', color=['red','green'])
plt.title('Avg Exit Rates on High-Value Pages')
plt.ylabel('Exit Rate')
plt.xticks([0,1], ['No Purchase', 'Purchased'], rotation=0)
plt.savefig('high_value_exits.png')
```

```python
# Compare PageValues across visitor types
plt.figure(figsize=(10, 6))
sns.boxplot(
    x='VisitorType',
    y='PageValues',
    hue='Revenue',
    data=df,
    showfliers=False  # Remove outliers for clarity
)
plt.title('Page Value Efficiency by Visitor Type')
plt.yscale('log')  # Handle skewed data
plt.ylabel('PageValues (Log Scale)')
plt.savefig('value_by_visitor.png')
```



```python
print(df['VisitorType'].value_counts())
```

```python
# Bin bounce rates into 10 groups
df['BounceRate_Bin'] = pd.cut(df['BounceRates'], bins=10, labels=False)

# Calculate purchase probability per bin
bounce_purchase = df.groupby('BounceRate_Bin')['Revenue'].mean().reset_index()

# Visualize
plt.figure(figsize=(10, 5))
sns.lineplot(
    x='BounceRate_Bin',
    y='Revenue',
    data=bounce_purchase,
    marker='o',
    color='red'
)
plt.xticks(rotation=45)
plt.title('Purchase Probability by Bounce Rate')
plt.xlabel('Bounce Rate Range')
plt.ylabel('Purchase Rate')

plt.savefig('bounce_vs_purchase.png', bbox_inches='tight')
```

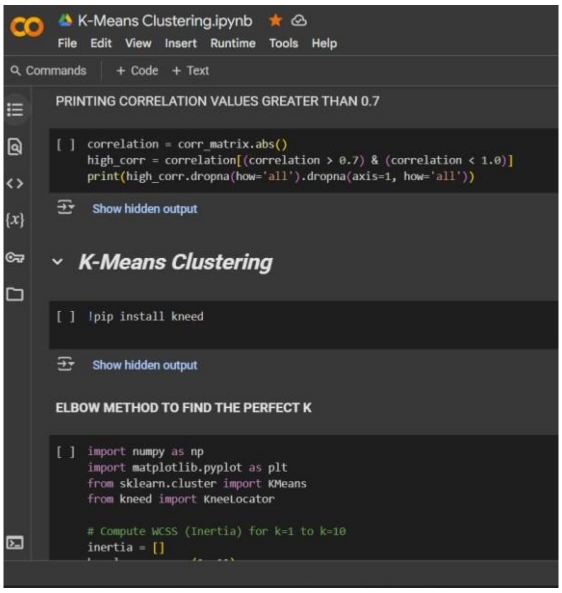


```python
plt.savefig('bounce_vs_purchase.png', bbox_inches='tight')
```

CORRELATION MATRIX

```python
import seaborn as sns
import matplotlib.pyplot as plt
corr_matrix = df.corr()
plt.figure(figsize=(8, 4))
sns.heatmap(corr_matrix, annot=False, cmap='coolwarm', linewidths=0.2)
plt.title("Feature Correlation Heatmap")
plt.show()
```

```python
correlation_matrix = df.corr()

# Iterate through the correlation matrix to get all pairs and their correlation values
for i in range(len(correlation_matrix.columns)):
    for j in range(i + 1, len(correlation_matrix.columns)):  # Avoid duplicate pairs and self-correlation
        var1 = correlation_matrix.columns[i]
        var2 = correlation_matrix.columns[j]
        corr_value = correlation_matrix.iloc[i, j]
        print(f"Correlation between {var1} and {var2}: {corr_value:.2f}")
```



PRINTING CORRELATION VALUES GREATER THAN 0.7

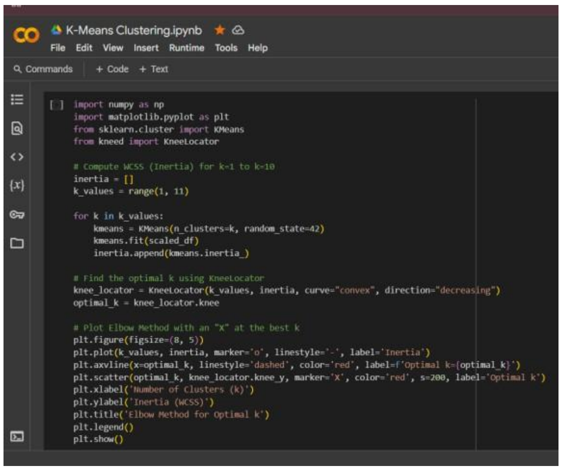
```python
correlation = corr_matrix.abs()
high_corr = correlation[(correlation > 0.7) & (correlation < 1.0)]
print(high_corr.dropna(how='all').dropna(axis=1, how='all'))
```

K-Means Clustering

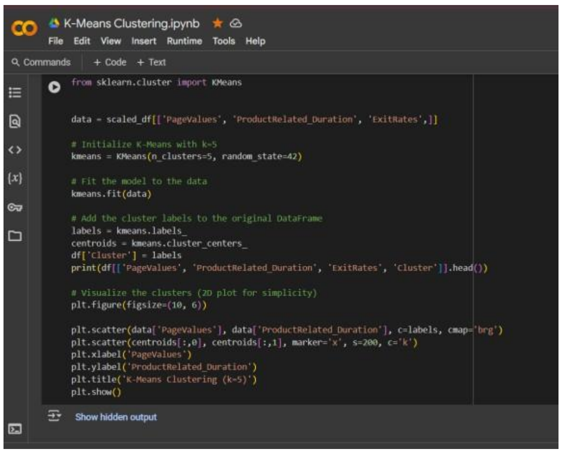
```python
!pip install kneed
```

ELBOW METHOD TO FIND THE PERFECT K

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from kneed import KneeLocator

# Compute WCSS (Inertia) for k=1 to k=10
inertia = []
```

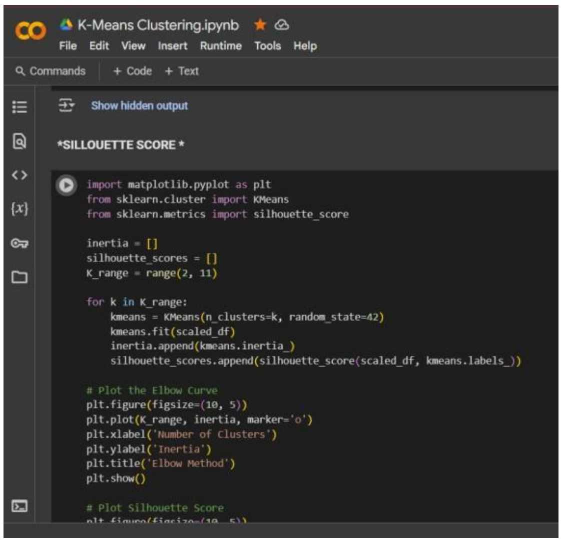


```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from kneed import KneeLocator

# Compute WCSS (Inertia) for k=1 to k=10
inertia = []
k_values = range(1, 11)

for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(scaled_df)
    inertia.append(kmeans.inertia_)

# Find the optimal k using KneeLocator
knee_locator = KneeLocator(k_values, inertia, curve="convex", direction="decreasing")
optimal_k = knee_locator.knee

# Plot Elbow Method with an "X" at the best k
plt.figure(figsize=(8, 5))
plt.plot(k_values, inertia, marker='o', linestyle='-', label='Inertia')
plt.axvline(x=optimal_k, linestyle='dashed', color='red', label=f'Optimal k={optimal_k}')
plt.scatter(optimal_k, knee_locator.knee_y, marker='X', color='red', s=200, label='Optimal k')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Inertia (WCSS)')
plt.title('Elbow Method for Optimal k')
plt.legend()
plt.show()
```



```python
from sklearn.cluster import KMeans


data = scaled_df[['PageValues', 'ProductRelated_Duration', 'ExitRates',]]

# Initialize K-Means with k=5
kmeans = KMeans(n_clusters=5, random_state=42)

# Fit the model to the data
kmeans.fit(data)

# Add the cluster labels to the original DataFrame
labels = kmeans.labels_
centroids = kmeans.cluster_centers_
df['Cluster'] = labels
print(df[['PageValues', 'ProductRelated_Duration', 'ExitRates', 'Cluster']].head())

# Visualize the clusters (2D plot for simplicity)
plt.figure(figsize=(10, 6))

plt.scatter(data['PageValues'], data['ProductRelated_Duration'], c=labels, cmap='brg')
plt.scatter(centroids[:,0], centroids[:,1], marker='x', s=200, c='k')
plt.xlabel('PageValues')
plt.ylabel('ProductRelated_Duration')
plt.title('K-Means Clustering (k=5)')
plt.show()
```

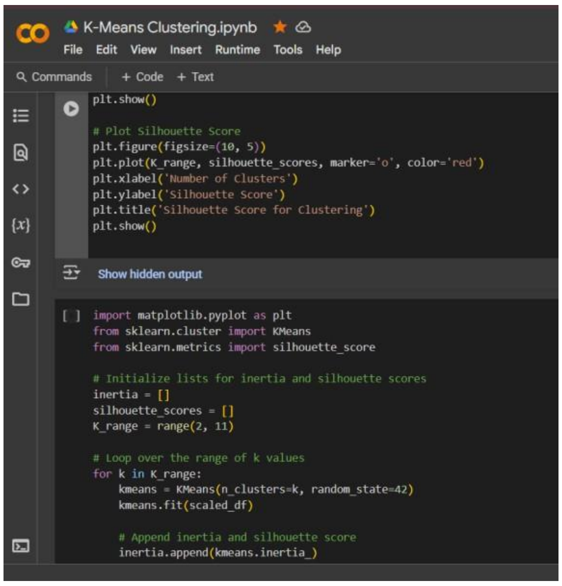
Show hidden output



Show hidden output

*SILLOUETTE SCORE *

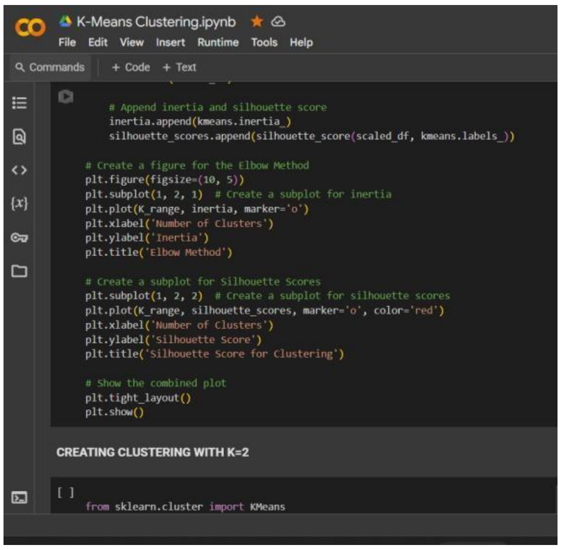
```python
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

inertia = []
silhouette_scores = []
K_range = range(2, 11)

for k in K_range:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(scaled_df)
    inertia.append(kmeans.inertia_)
    silhouette_scores.append(silhouette_score(scaled_df, kmeans.labels_))

# Plot the Elbow Curve
plt.figure(figsize=(10, 5))
plt.plot(K_range, inertia, marker='o')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
plt.title('Elbow Method')
plt.show()

# Plot Silhouette Score
```



```python
plt.show()

# Plot Silhouette Score
plt.figure(figsize=(10, 5))
plt.plot(K_range, silhouette_scores, marker='o', color='red')
plt.xlabel('Number of Clusters')
plt.ylabel('Silhouette Score')
plt.title('Silhouette Score for Clustering')
plt.show()
```

Show hidden output

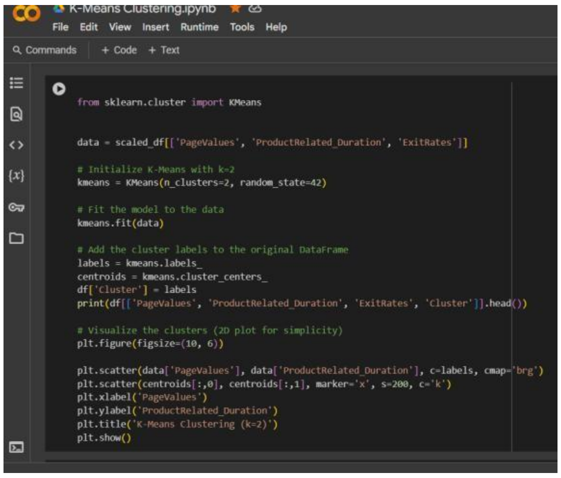
```python
[ ] import matplotlib.pyplot as plt
    from sklearn.cluster import KMeans
    from sklearn.metrics import silhouette_score

    # Initialize lists for inertia and silhouette scores
    inertia = []
    silhouette_scores = []
    K_range = range(2, 11)

    # Loop over the range of k values
    for k in K_range:
        kmeans = KMeans(n_clusters=k, random_state=42)
        kmeans.fit(scaled_df)

        # Append inertia and silhouette score
        inertia.append(kmeans.inertia_)
```



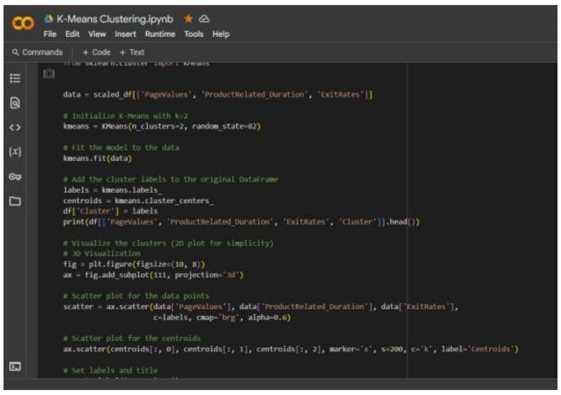
```python
        # Append inertia and silhouette score
        inertia.append(kmeans.inertia_)
        silhouette_scores.append(silhouette_score(scaled_df, kmeans.labels_))

    # Create a figure for the Elbow Method
    plt.figure(figsize=(10, 5))
    plt.subplot(1, 2, 1)  # Create a subplot for inertia
    plt.plot(K_range, inertia, marker='o')
    plt.xlabel('Number of Clusters')
    plt.ylabel('Inertia')
    plt.title('Elbow Method')

    # Create a subplot for Silhouette Scores
    plt.subplot(1, 2, 2)  # Create a subplot for silhouette scores
    plt.plot(K_range, silhouette_scores, marker='o', color='red')
    plt.xlabel('Number of Clusters')
    plt.ylabel('Silhouette Score')
    plt.title('Silhouette Score for Clustering')

    # Show the combined plot
    plt.tight_layout()
    plt.show()
```

CREATING CLUSTERING WITH K=2

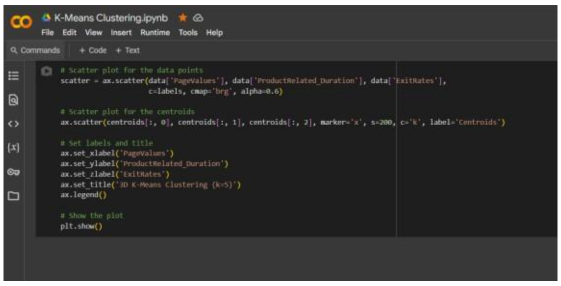
```python
[ ] from sklearn.cluster import KMeans
```



```python
from sklearn.cluster import KMeans


data = scaled_df[['PageValues', 'ProductRelated_Duration', 'ExitRates']]

# Initialize K-Means with k=2
kmeans = KMeans(n_clusters=2, random_state=42)

# Fit the model to the data
kmeans.fit(data)

# Add the cluster labels to the original DataFrame
labels = kmeans.labels_
centroids = kmeans.cluster_centers_
df['Cluster'] = labels
print(df[['PageValues', 'ProductRelated_Duration', 'ExitRates', 'Cluster']].head())

# Visualize the clusters (2D plot for simplicity)
plt.figure(figsize=(10, 6))

plt.scatter(data['PageValues'], data['ProductRelated_Duration'], c=labels, cmap='brg')
plt.scatter(centroids[:,0], centroids[:,1], marker='x', s=200, c='k')
plt.xlabel('PageValues')
plt.ylabel('ProductRelated_Duration')
plt.title('K-Means Clustering (k=2)')
plt.show()
```

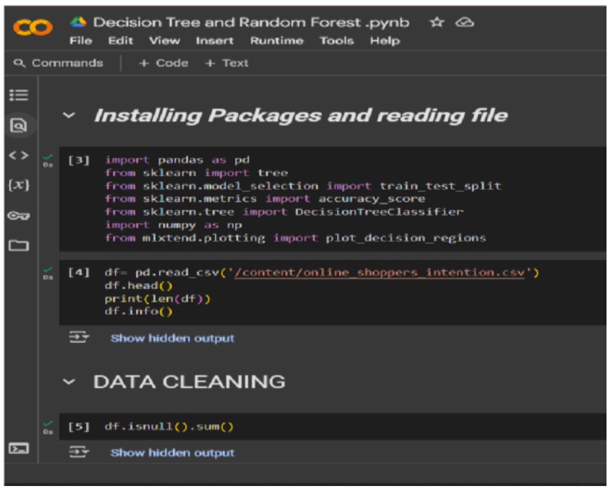


```python
data = scaled_df[['PageValues', 'ProductRelated_Duration', 'ExitRates']]

# Initialize K-Means with k=2
kmeans = KMeans(n_clusters=2, random_state=82)

# Fit the model to the data
kmeans.fit(data)

# Add the cluster labels to the original DataFrame
labels = kmeans.labels_
centroids = kmeans.cluster_centers_
df['Cluster'] = labels
print(df[['PageValues', 'ProductRelated_Duration', 'ExitRates', 'Cluster']].head())

# Visualize the clusters (2D plot for simplicity)
# 3D Visualization
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')

# Scatter plot for the data points
scatter = ax.scatter(data['PageValues'], data['ProductRelated_Duration'], data['ExitRates'],
                     c=labels, cmap='brg', alpha=0.6)

# Scatter plot for the centroids
ax.scatter(centroids[:, 0], centroids[:, 1], centroids[:, 2], marker='x', s=200, c='k', label='Centroids')

# Set labels and title
```



```python
# Scatter plot for the data points
scatter = ax.scatter(data['PageValues'], data['ProductRelated_Duration'], data['ExitRates'],
                     c=labels, cmap='brg', alpha=0.6)

# Scatter plot for the centroids
ax.scatter(centroids[:, 0], centroids[:, 1], centroids[:, 2], marker='x', s=200, c='k', label='Centroids')

# Set labels and title
ax.set_xlabel('PageValues')
ax.set_ylabel('ProductRelated_Duration')
ax.set_zlabel('ExitRates')
ax.set_title('3D K-Means Clustering (k=5)')
ax.legend()

# Show the plot
plt.show()
```

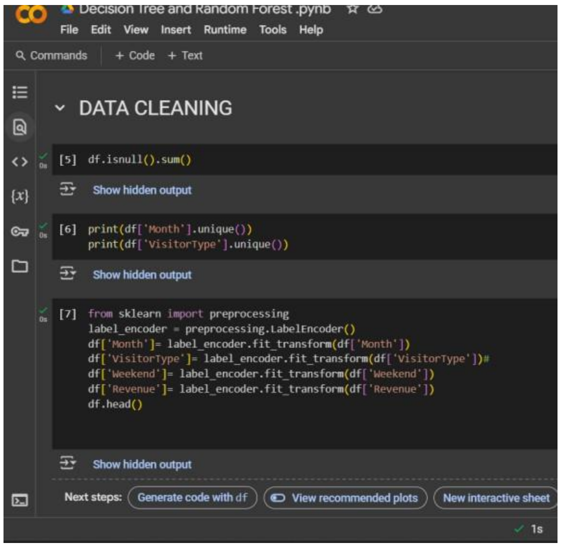
Decision Tree:


Decision Tree and Random Forest .pynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text
Installing Packages and reading file

```
[3] import pandas as pd
    from sklearn import tree
    from sklearn.model_selection import train_test_split
    from sklearn.metrics import accuracy_score
    from sklearn.tree import DecisionTreeClassifier
    import numpy as np
    from mlxtend.plotting import plot_decision_regions

[4] df= pd.read_csv('/content/online_shoppers_intention.csv')
    df.head()
    print(len(df))
    df.info()
```

Show hidden output
DATA CLEANING

```
[5] df.isnull().sum()
```


Decision Tree and Random Forest .pynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text
DATA CLEANING

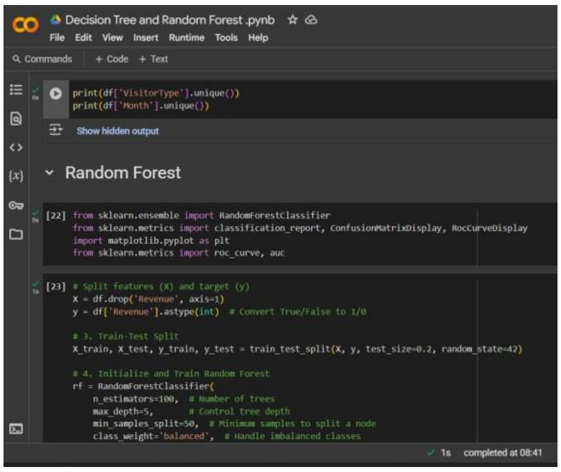
```
[5] df.isnull().sum()
```

```
[6] print(df['Month'].unique())
    print(df['VisitorType'].unique())
```

```
[7] from sklearn import preprocessing
    label_encoder = preprocessing.LabelEncoder()
    df['Month']= label_encoder.fit_transform(df['Month'])
    df['VisitorType']= label_encoder.fit_transform(df['VisitorType'])#
    df['Weekend']= label_encoder.fit_transform(df['Weekend'])
    df['Revenue']= label_encoder.fit_transform(df['Revenue'])
    df.head()
```

Show hidden output
Next steps: Generate code with df View recommended plots New interactive sheet
1s

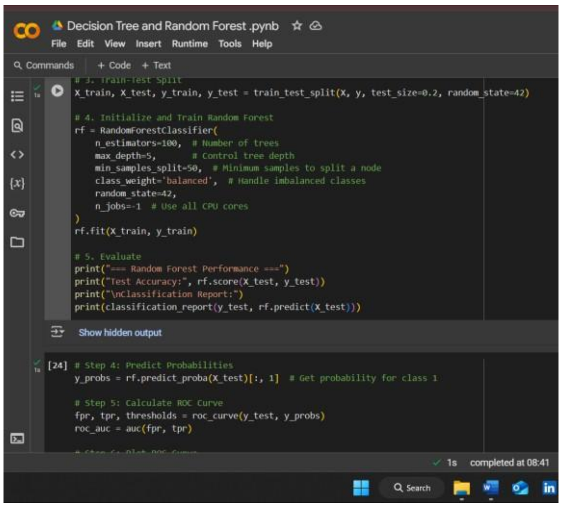
Random Forest:


Decision Tree and Random Forest .pynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text

```
print(df['VisitorType'].unique())
print(df['Month'].unique())
```

Show hidden output
Random Forest

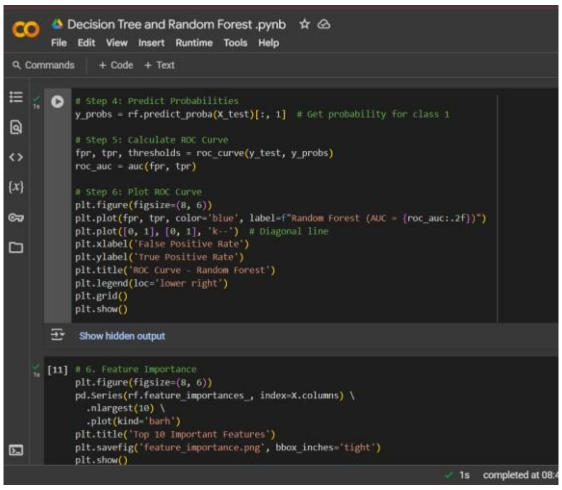
```
[22] from sklearn.ensemble import RandomForestClassifier
     from sklearn.metrics import classification_report, ConfusionMatrixDisplay, RocCurveDisplay
     import matplotlib.pyplot as plt
     from sklearn.metrics import roc_curve, auc

[23] # Split features (X) and target (y)
     X = df.drop('Revenue', axis=1)
     y = df['Revenue'].astype(int)  # Convert True/False to 1/0

     # 3. Train-Test Split
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

     # 4. Initialize and Train Random Forest
     rf = RandomForestClassifier(
         n_estimators=100,  # Number of trees
         max_depth=5,       # Control tree depth
         min_samples_split=50,  # Minimum samples to split a node
         class_weight='balanced',  # Handle imbalanced classes
```


Decision Tree and Random Forest .pynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text

```
# 3. Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 4. Initialize and Train Random Forest
rf = RandomForestClassifier(
    n_estimators=100,  # Number of trees
    max_depth=5,       # Control tree depth
    min_samples_split=50,  # Minimum samples to split a node
    class_weight='balanced',  # Handle imbalanced classes
    random_state=42,
    n_jobs=-1  # Use all CPU cores
)
rf.fit(X_train, y_train)

# 5. Evaluate
print("=== Random Forest Performance ===")
print("Test Accuracy:", rf.score(X_test, y_test))
print("\nClassification Report:")
print(classification_report(y_test, rf.predict(X_test)))
```

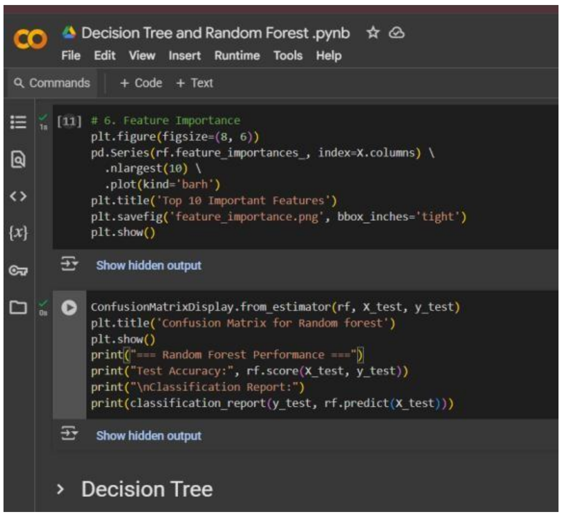
```
[24] # Step 4: Predict Probabilities
     y_probs = rf.predict_proba(X_test)[:, 1]  # Get probability for class 1

     # Step 5: Calculate ROC Curve
     fpr, tpr, thresholds = roc_curve(y_test, y_probs)
     roc_auc = auc(fpr, tpr)
```

1s completed at 08:41
Search


Decision Tree and Random Forest .pynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text

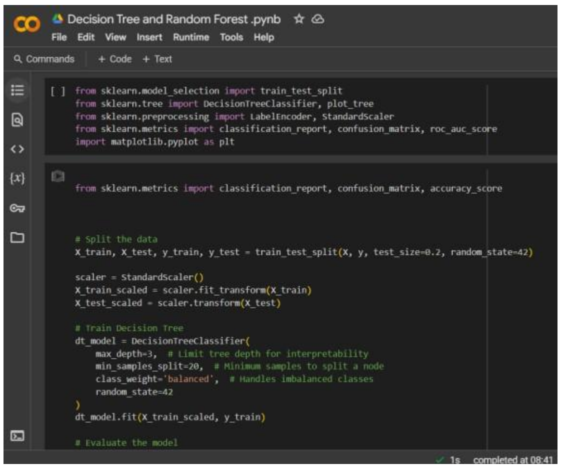
```
# Step 4: Predict Probabilities
y_probs = rf.predict_proba(X_test)[:, 1]  # Get probability for class 1

# Step 5: Calculate ROC Curve
fpr, tpr, thresholds = roc_curve(y_test, y_probs)
roc_auc = auc(fpr, tpr)

# Step 6: Plot ROC Curve
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', label=f"Random Forest (AUC = {roc_auc:.2f})")
plt.plot([0, 1], [0, 1], 'k--')  # Diagonal line
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - Random Forest')
plt.legend(loc='lower right')
plt.grid()
plt.show()
```

```
[11] # 6. Feature Importance
     plt.figure(figsize=(8, 6))
     pd.Series(rf.feature_importances_, index=X.columns) \
       .nlargest(10) \
       .plot(kind='barh')
     plt.title('Top 10 Important Features')
     plt.savefig('feature_importance.png', bbox_inches='tight')
     plt.show()
```


Decision Tree and Random Forest .pynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text

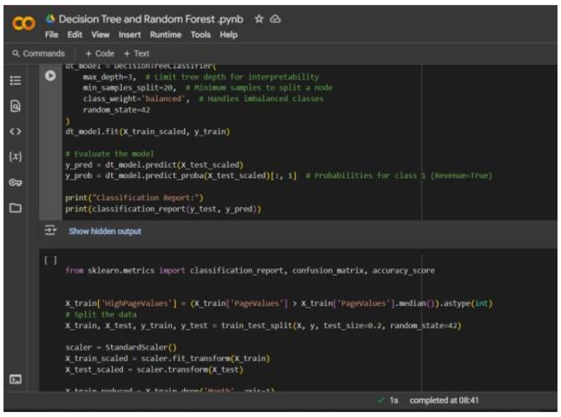
```
[11] # 6. Feature Importance
     plt.figure(figsize=(8, 6))
     pd.Series(rf.feature_importances_, index=X.columns) \
       .nlargest(10) \
       .plot(kind='barh')
     plt.title('Top 10 Important Features')
     plt.savefig('feature_importance.png', bbox_inches='tight')
     plt.show()
```

```
ConfusionMatrixDisplay.from_estimator(rf, X_test, y_test)
plt.title('Confusion Matrix for Random forest')
plt.show()
print("=== Random Forest Performance ===")
print("Test Accuracy:", rf.score(X_test, y_test))
print("\nClassification Report:")
print(classification_report(y_test, rf.predict(X_test)))
```


Decision Tree and Random Forest .pynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text

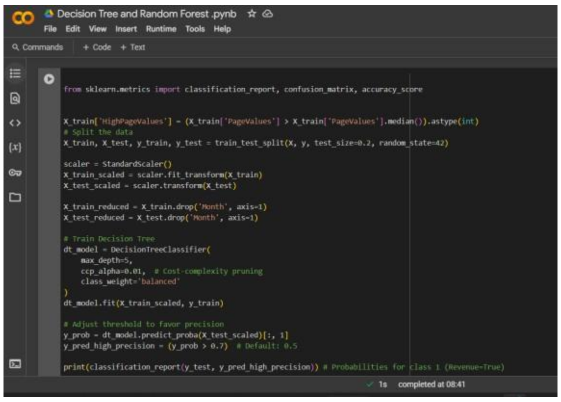
```
[ ] from sklearn.model_selection import train_test_split
    from sklearn.tree import DecisionTreeClassifier, plot_tree
    from sklearn.preprocessing import LabelEncoder, StandardScaler
    from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score
    import matplotlib.pyplot as plt
```

```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score



# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

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

# Train Decision Tree
dt_model = DecisionTreeClassifier(
    max_depth=3,  # Limit tree depth for interpretability
    min_samples_split=20,  # Minimum samples to split a node
    class_weight='balanced',  # Handles imbalanced classes
    random_state=42
)
dt_model.fit(X_train_scaled, y_train)

# Evaluate the model
```

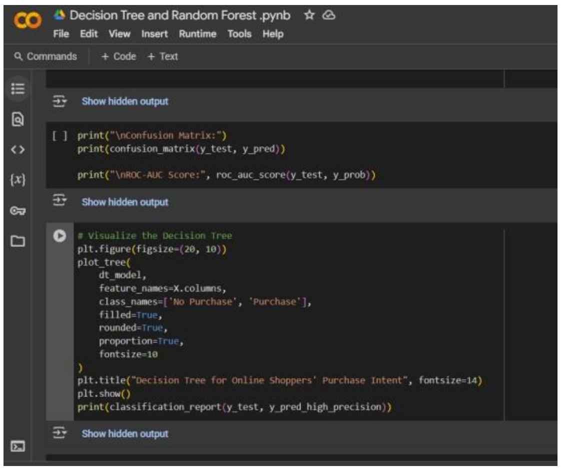

Decision Tree and Random Forest .pynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text

```
dt_model = DecisionTreeClassifier(
    max_depth=3,  # Limit tree depth for interpretability
    min_samples_split=20,  # Minimum samples to split a node
    class_weight='balanced',  # Handles imbalanced classes
    random_state=42
)
dt_model.fit(X_train_scaled, y_train)

# Evaluate the model
y_pred = dt_model.predict(X_test_scaled)
y_prob = dt_model.predict_proba(X_test_scaled)[:, 1]  # Probabilities for class 1 (Revenue=True)

print("Classification Report:")
print(classification_report(y_test, y_pred))
```

```
[ ]
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score


X_train['HighPageValues'] = (X_train['PageValues'] > X_train['PageValues'].median()).astype(int)
# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

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

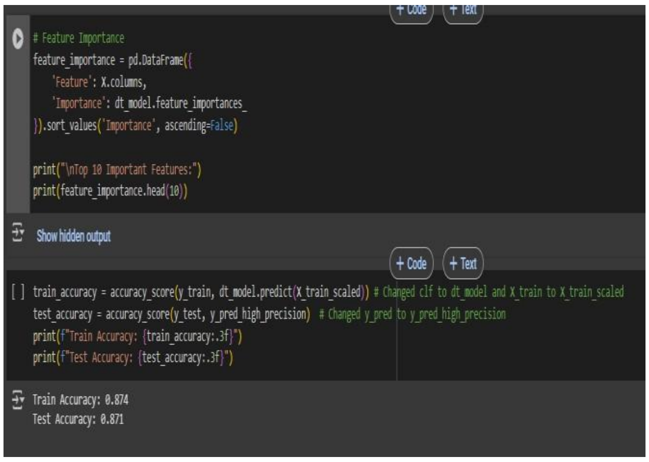

Decision Tree and Random Forest .pynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text

```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score


X_train['HighPageValues'] = (X_train['PageValues'] > X_train['PageValues'].median()).astype(int)
# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

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

X_train_reduced = X_train.drop('Month', axis=1)
X_test_reduced = X_test.drop('Month', axis=1)

# Train Decision Tree
dt_model = DecisionTreeClassifier(
    max_depth=5,
    ccp_alpha=0.01,  # Cost-complexity pruning
    class_weight='balanced'
)
dt_model.fit(X_train_scaled, y_train)

# Adjust threshold to favor precision
y_prob = dt_model.predict_proba(X_test_scaled)[:, 1]
y_pred_high_precision = (y_prob > 0.7)  # Default: 0.5

print(classification_report(y_test, y_pred_high_precision)) # Probabilities for class 1 (Revenue=True)
```


Decision Tree and Random Forest .pynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text
Show hidden output

```
[ ] print("\nConfusion Matrix:")
    print(confusion_matrix(y_test, y_pred))

    print("\nROC-AUC Score:", roc_auc_score(y_test, y_prob))
```

```
# Visualize the Decision Tree
plt.figure(figsize=(20, 10))
plot_tree(
    dt_model,
    feature_names=X.columns,
    class_names=['No Purchase', 'Purchase'],
    filled=True,
    rounded=True,
    proportion=True,
    fontsize=10
)
plt.title("Decision Tree for Online Shoppers' Purchase Intent", fontsize=14)
plt.show()
print(classification_report(y_test, y_pred_high_precision))
```



```
# Feature Importance
feature_importance = pd.DataFrame({
    'Feature': X.columns,
    'Importance': dt_model.feature_importances_
}).sort_values('Importance', ascending=False)

print("\nTop 10 Important Features:")
print(feature_importance.head(10))
```

```
[ ] train_accuracy = accuracy_score(y_train, dt_model.predict(X_train_scaled)) # Changed clf to dt_model and X_train to X_train_scaled
    test_accuracy = accuracy_score(y_test, y_pred_high_precision)  # Changed y_pred to y_pred_high_precision
    print(f"Train Accuracy: {train_accuracy:.3f}")
    print(f"Test Accuracy: {test_accuracy:.3f}")
```

```
Train Accuracy: 0.874
Test Accuracy: 0.871
```