IEEE
UNIVERSITY OF COLOMBO SCHOOL OF COMPUTING
STUDENT BRANCH
UCSC

IntelliHack 5.0

TEAM HYPER TUNERS

TASK 02

Customer Segmentation

intelliHack 5.0
Artificial . But Intelligent

Task 02

Customer Segmentation

Contents Page no

Introduction

Understanding customer behavior is crucial for enhancing business strategies and improving customer satisfaction. This project aims to perform customer segmentation on an e-commerce platform using clustering techniques to identify distinct customer groups based on their purchasing patterns and behavior.

We explored various clustering approaches, including KMeans and DBSCAN to categorize customers into three hidden clusters: Bargain Hunters, High Spenders, and Window Shoppers. The model was evaluated based on clustering performance and interpretability, using key customer features such as total purchases, average cart value, total time spent, product clicks, customer id and discount counts.

This report presents a comprehensive exploratory data analysis (EDA), details the data preprocessing steps, outlines the model selection and evaluation process, and provides insights into the identified customer clusters.

Objective : To identify distinct customer segments (clusters) based on their behavior

Approach

We were given a data set containing 999 data points: *total_purchases, avg_cart_value, total_time_spent, product_click, discount_counts, and customer_id.* The goal was to cluster the dataset into three segments

1. Bargain Hunters: These customers are deal-seekers who make frequent purchases of low-value items and heavily rely on discounts.
2. High Spenders: These customers are premium buyers who focus on high-value purchases and are less influenced by discounts.
3. Window Shoppers: These customers browse without making purchases, influenced by discounts but not necessarily purchasing.

First the CSV file was renamed to *customer_behavior_analytics.csv*, the dataset was loaded, and basic information and the following statistics were displayed.

- *total_purchases*: 979 non-null values, with a data type of float64.
- *avg_cart_value*: 979 non-null values, with a data type of float64.
- *total_time_spent*: 999 non-null values, with a data type of float64.
- *product_click*: 979 non-null values, with a data type of float64.
- *discount_counts*: 999 non-null values, with a data type of float64.

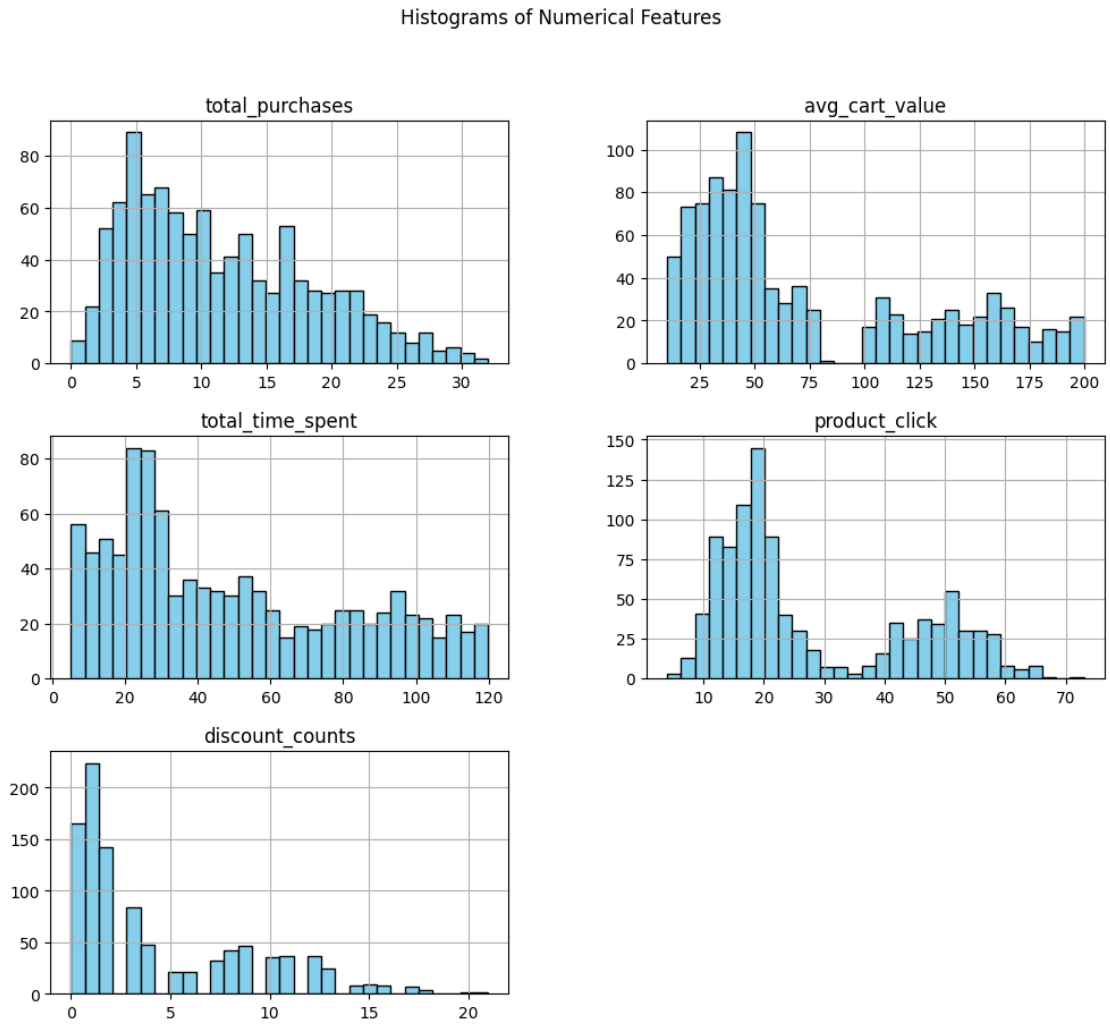
For filling the null values, the good practice is to use the median (if the data is skewed or contains outliers) or the mean (if the data is normally distributed). Since the data in *total_purchases*, *avg_cart_value*, and *product_click* were skewed, I used the median to fill the 20 missing (null) values in these columns.

The *customer_id* column was identified as a non-informative for the clustering process, so it was removed f

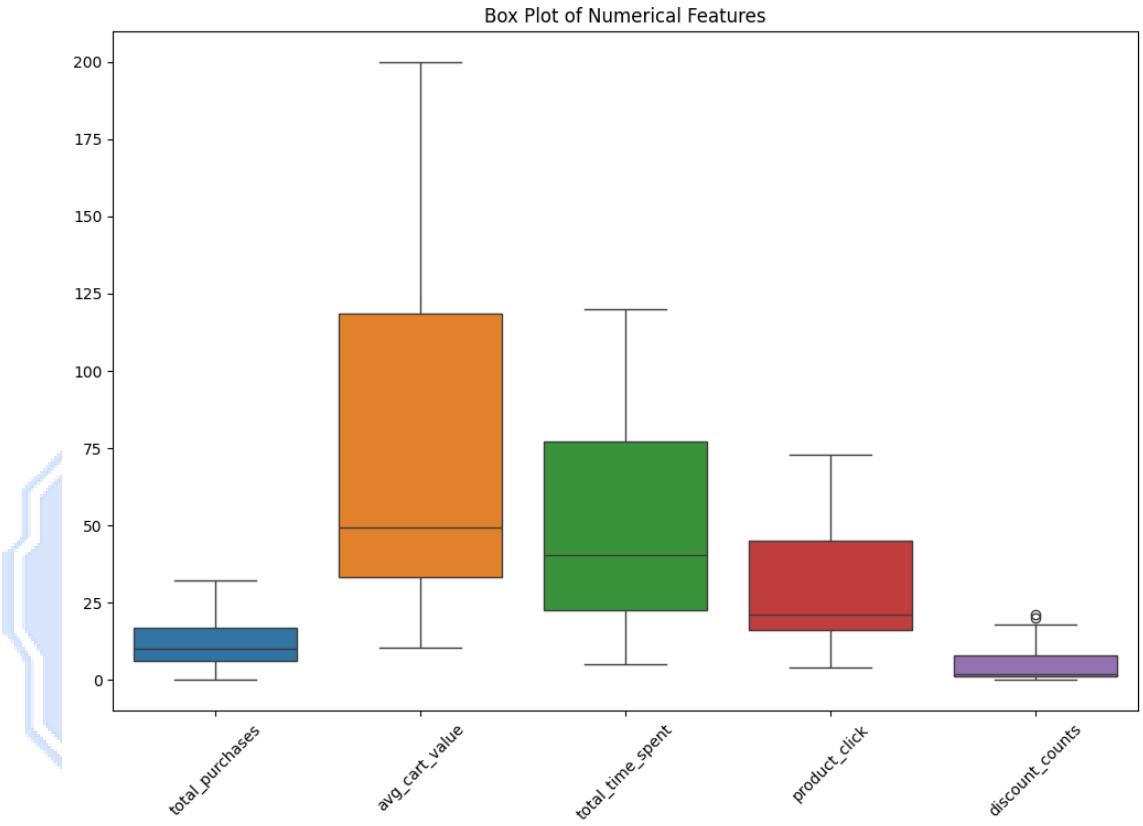
Exploratory Data Analysis (EDA)

Next, the Exploratory Data Analysis process was carried out to explore the dataset. During the visualization phase, to visualize the distribution and spread of the numerical features, histograms and box plots were created for each column.

Histograms were used to examine the distribution of the data and assess the skewness or normality of each feature.



Box plots were used to identify any outliers and understand the spread of the data.



After performing the EDA, the skewness of the numerical features was calculated. Skewness measures the asymmetry of the data distribution. A skewness value of 0 indicates a perfectly symmetrical distribution.

Positive skewness (right skew) means the tail on the right side of the distribution is longer or fatter, and the data is skewed towards the lower end.

The skewness values are as follows,

- *total_purchases*: 0.656645
- *avg_cart_value*: 0.816758
- *total_time_spent*: 0.564760
- *product_click*: 0.718501
- *discount_counts*: 1.070385

All numerical features show positive skewness, indicating that the data for each feature is skewed to the right. This means that most of the values are concentrated at the lower end of the scale, with a few higher values pulling the mean to the right.

The highest skewness is observed in the *discount_counts* feature, which indicates a relatively strong right skew. This suggests that most customers have fewer discount counts, while a small number of customers might have significantly higher discount counts.

Since the distributions of the numerical features are positively skewed, we wanted to reduce this skewness and make the data closer to a normal distribution. Typically, for data to be normally distributed, the skewness value should be close to zero. However, since the distributions were not normally distributed, transformation techniques were applied to reduce the skewness and make the data more normally distributed.

Transformation Techniques

1. Log Transformation

This transformation compresses higher values more significantly than the lower values, reducing the overall skewness. By applying this, we make the data more normally distributed, which is often important for clustering algorithms like KMeans, as they perform better when features are normally distributed or closer to normal.

Log transformation is suitable here because the *avg_cart_value*, *product_click*, and *discount_counts* values are concentrated at the lower end, with some higher values pulling the mean to the right. The log transformation helps compress this skew.

2. Square Root Transformation

The square root is typically used when the data contains counts or other types of positive data that aren't as highly skewed as those suited for the log transformation. It is especially helpful for data where the values are still right-skewed but do not have as extreme outliers. The square root transformation helps in making the distribution more symmetrical without distorting the data too much.

For features like *total_purchases* and *total_time_spent*, which have moderate skewness, we applied the square root transformation.

These transformations help to reduce the skewness of the data, making it more normally distributed.

Next, a **pairplot** was obtained to explore the relationships between each pair of features in the dataset. This is a useful visualization tool EDA that creates scatterplots for each pair of features, along with histograms for the individual features along the diagonal.

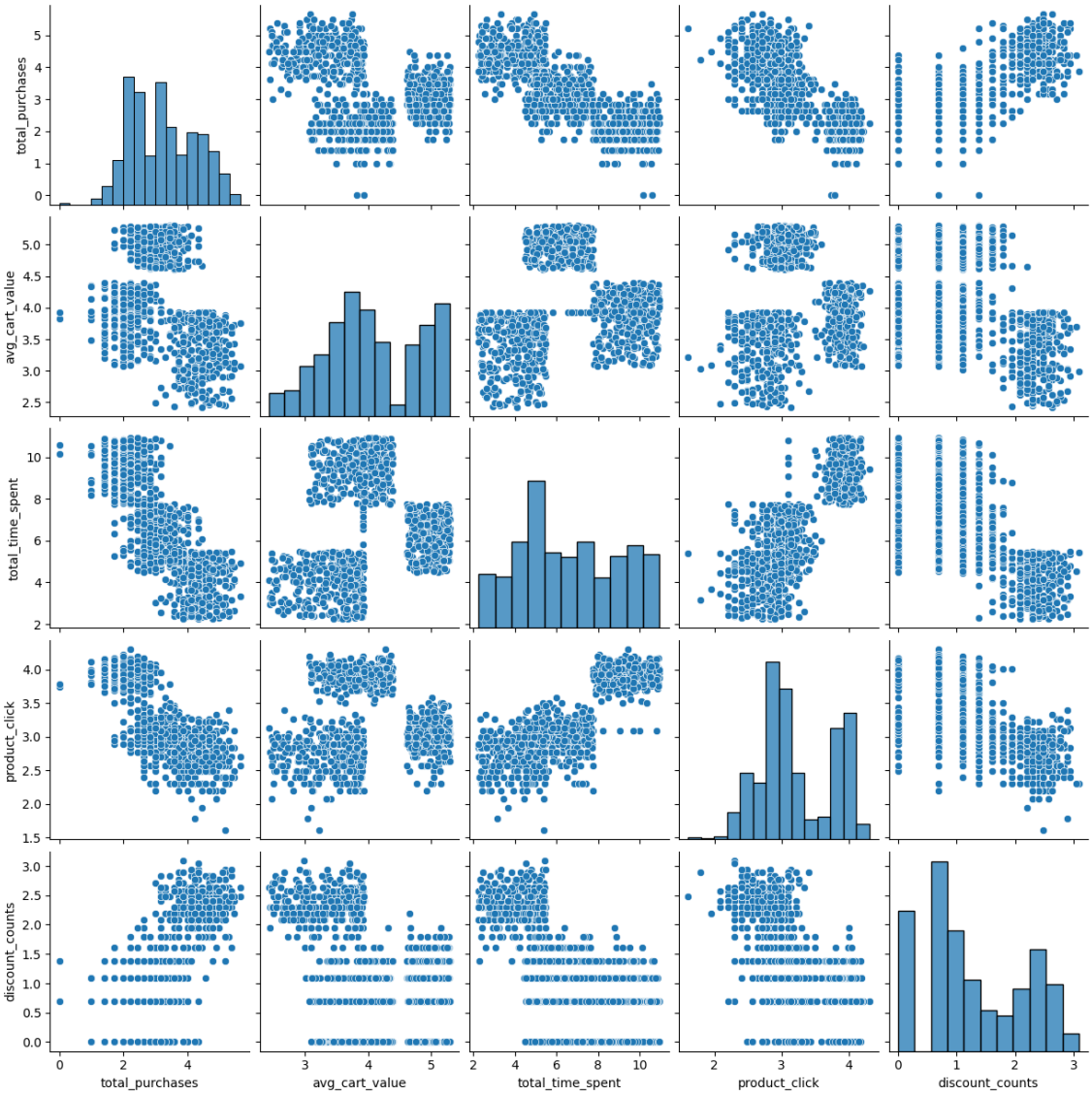
Purpose of Using Pairplot

- Identifying Relationships Between Feature Pairs
  A pairplot helps to visually identify any correlations or relationships between pairs of features. By looking at the scatterplots, we can quickly assess if there are any linear, non-linear, or no relationships between the features.

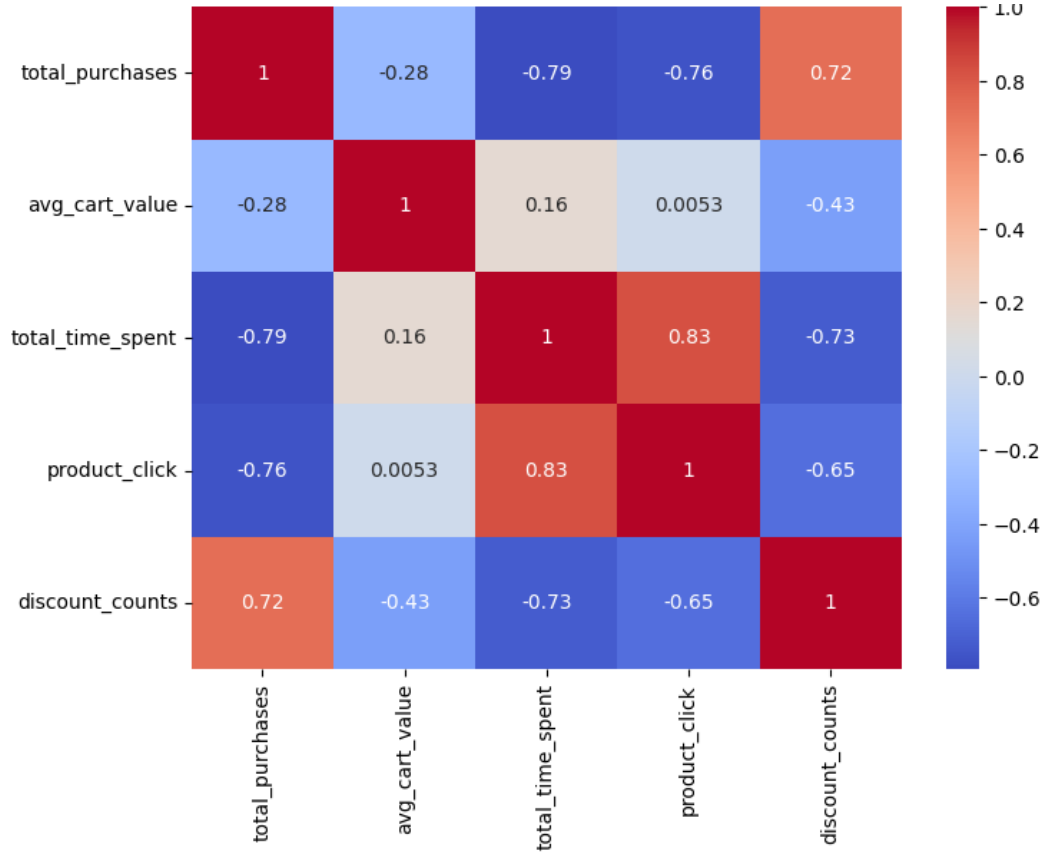
- Identifying Clusters or Groupings
  The pairplot can also help identify if any natural groupings or clusters exist in the data, which is useful for tasks like clustering. If certain features are highly correlated, we may see distinct groupings or clusters in the scatterplots.

- Detecting Outliers
  Pairplots also make it easier to detect potential outliers. If any data points are far away from the general trend or scatter of the other data points, they will stand out in the scatterplots, allowing us to decide whether to handle them.

- Visualizing the Distribution of Individual Features
  The diagonal histograms or density plots give a clear view of the distribution of each individual feature. These plots can help confirm if the transformations improved the distribution, reducing skewness and making the data closer to normal.



Next, a **correlation heatmap** was obtained to analyze the relationships between the features. This is a useful tool for visualizing the pairwise correlations between numerical features in a dataset. It uses colors to represent correlation coefficients, with values ranging from -1 (perfect negative correlation) to +1 (perfect positive correlation), and 0 indicating no correlation.



Why Draw a Correlation Heatmap?

- Identifying Relationships Between Features

The heatmap allows us to quickly identify which features are highly correlated, moderately correlated, or not correlated at all. Features with high correlation can often be combined or used together effectively in models, while uncorrelated features may be treated as independent inputs.

When two or more features are highly correlated, it can lead to redundant information and negatively impact machine learning algorithms like KMeans. Identifying such correlations helps decide if any features can be removed or combined to improve model performance.

The heatmap can also suggest potential feature engineering opportunities, such as creating new features by combining highly correlated ones.

From the heatmap, it was identified that:

- *Total_time_spent* and *Product_click* had a high correlation of 0.83. These two features have a strong relationship, suggesting that they could be combined to create a new feature representing user engagement. This new feature could potentially capture the overall level of activity of a user.

- We decided to combine the two correlated features (*total_time_spent* and *product_click*) into a single feature. PCA was applied to reduce the dimensionality and combine these two features into one, while retaining key information. The dataset then had a new feature *time_click_combined* which captures the information from both *total_time_spent* and *product_click.*

After performing the combination, the following was observed because of what we chose not to combine the features into new ones.

- Increased Inertia
  Combining features like *total_time_spent* and *product_click* caused the clusters to become more spread out. This increased the inertia (sum of squared distances from each point to its assigned cluster center), which is undesirable in clustering. A higher inertia indicates that the data points are less tightly grouped around the centroids, which reduces the effectiveness of the clustering algorithm.

- Decreased Silhouette Score
  The silhouette score measures how similar an object is to its own cluster compared to other clusters. After combining the features, the silhouette score decreased, indicating that the quality of the clustering reduced. A lower silhouette score suggests that the newly created features did not improve the separation between clusters.

Scaling is the process of standardizing or normalizing the range of features in a dataset. This is crucial for machine learning algorithms, especially those that use distance-based metrics like KMeans clustering. Features with larger ranges or different units of measurement can disproportionately affect the model, leading to biased or inaccurate results.

Scaling ensures that each feature contributes equally to the analysis, improving the performance and convergence of the algorithm. There are different scaling techniques, each with specific advantages depending on the distribution and characteristics of the data.

1. Standardization (**StandardScaler**)
   This method centers the data around zero and scales it to have a unit variance. It is best used when the data follows a normal distribution, as it doesn't change the shape of the data, just rescales it.

2. **RobustScaler**
   This scaler uses the median and interquartile range (IQR) to scale the data. It is more robust to outliers compared to StandardScaler. It is ideal for datasets with many extreme values or skewed distributions.

3. **MinMaxScaler**
   This method scales the data to a fixed range, typically between 0 and 1. It is effective when the data is bounded or when you want to preserve the relationships between the features. MinMax scaling is sensitive to outliers, so it may not work well if your data has extreme values.

Scaling Results

1. StandardScaler:
   - Silhouette Score: 0.62
   - Inertia: Around 850

   After scaling with StandardScaler, the silhouette score was 0.62, which is a decent value indicating that the clustering was somewhat effective. The inertia value was around 850, suggesting that the clusters were relatively well-formed but could be improved further.

2. RobustScaler:
   - Silhouette Score: 6.11
   - Inertia: 183

   When using the Robust Scaler, the inertia significantly decreased to 183, which indicates that the clusters became much tighter and more well-defined. However, the silhouette score of 6.11 seems quite high, which may be due to an issue with how the scaling impacted the overall clustering structure possibly resulting in poor cluster separation.

3. MinMaxScaler:
   - Silhouette Score: 0.70

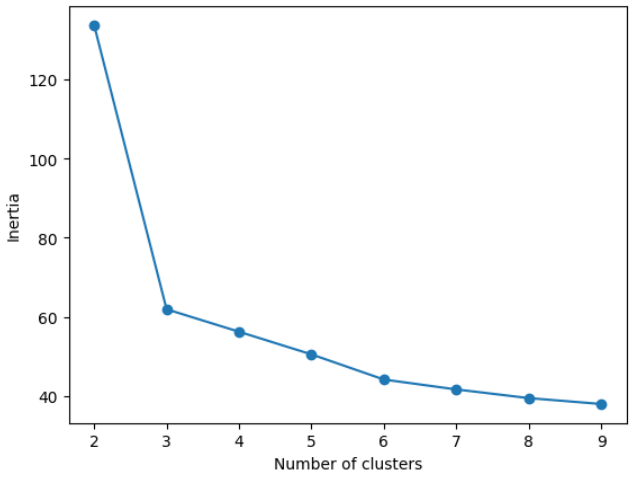
The MinMax Scaler provided the best performance, with a silhouette score of 0.70, indicating well-separated and compact clusters. This suggests that scaling the data within a specific range improves the clustering quality and separation between different customer segments.

Elbow Method for Optimal Number of Clusters

After scaling the data, we used the Elbow Method to determine the optimal number of clusters for the KMeans clustering algorithm. The Elbow Method involves plotting the inertia against the number of clusters.

- Inertia is a measure of how tightly the data points are grouped around the cluster centroids. A lower inertia value indicates that the clusters are more compact, and the algorithm has done a better job in grouping the data points.
- By plotting inertia against the number of clusters, we observe a "bend" or "elbow" in the plot. The optimal number of clusters corresponds to the point where adding more clusters doesn't lead to a significant reduction in inertia.

After plotting, the "elbow" point indicated that the optimal number of clusters is 3, which aligns with our initial assumption based on the customer segmentation task.



Silhouette Score

To evaluate the quality of the clusters, we used the Silhouette Score. The silhouette score measures how similar an object is to its own cluster compared to other clusters, with a value closer to +1 indicating well-clustered points, and values near 0 or negative indicating poor clustering.

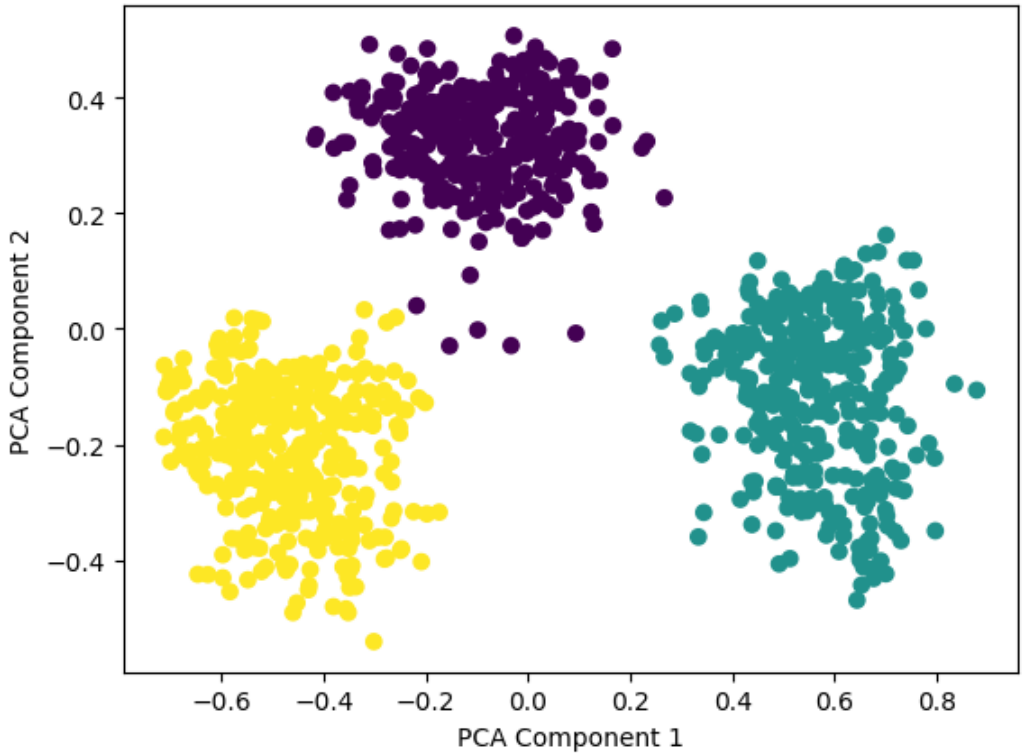
- Silhouette Score: 0.5557
  This score suggests that the clusters are well-separated and well-defined. While not perfect (as

KMeans Algorithm

The KMeans clustering algorithm was used with 3 clusters. We chose KMeans because:

1. Clear Clusters: It works well when data has distinct, spherical clusters, which fits the customer segmentation task.
2. Efficiency: KMeans is computationally efficient and scales well with larger datasets (like 999 data points here).
3. Simplicity: It's easy to implement and interpret, which is important for profiling customer behaviors.
4. Numerical Data: It works well with numerical data, like total purchases and product clicks.
5. Better than Alternatives: Other methods like Hierarchical Clustering and DBSCAN are slower or less effective for this dataset, while GMM is more complex and harder to interpret.

To visualize the clusters using PCA (Principal Component Analysis), we'll reduce the data from its higher-dimensional space to 2 dimensions, allowing us to plot the clusters in a 2D scatter plot.



The Silhouette Score ranges from -1 to +1, where a value closer to +1 indicates

Well-separated clusters, and values near 0 suggest overlapping clusters. A value closer to -1 indicates that the clusters may be incorrect.

Since we got a Silhouette Score of 0.5557, that's actually a pretty good result. It indicates that your clusters are well-defined and there is a decent separation between them. A higher Silhouette Score would obviously be better, but anything above 0.5 is usually considered a solid outcome for clustering tasks.

And the goal with KMeans is to minimize inertia. Lower inertia means that your clusters are more tightly packed and have a better fit to the data, but it's important to balance that with the Silhouette Score to ensure good cluster separation.

Challenges

- Dealing with Skewed Data: The features in the dataset were positively skewed, which could negatively affect clustering performance. Applying transformations was necessary to reduce skewness but finding the right transformation and ensuring that it improved clustering was a bit challenging.
- Feature Correlation: Some features were highly correlated which led to the temptation to combine them. However, combining features resulted in worse clustering performance, as measured by increased inertia and lower silhouette scores.
- Scaling Methods: Choosing the right scaling method was another challenge. After testing multiple scaling techniques, the MinMax Scaler produced the best silhouette score, but there were also cases where other methods did not work as expected.

Insights

- Feature Relationships: The EDA revealed important relationships between features, such as the strong correlation between *total_time_spent* and *product_click*. This suggests that user engagement is a key factor in clustering customers.
- Cluster Profiles: KMeans clustering identified three meaningful customer segments
    - Bargain Hunters: Customers who heavily rely on discounts and make frequent, low-value purchases.
    - High Spenders: Customers who focus on high-value purchases and are less influenced by discounts.
    - Window Shoppers: Customers who browse products but don't make many purchases.

- Clustering Quality: While the silhouette score (0.5557) indicated reasonable clustering, there is still room for improvement in terms of tighter cluster formation and better separation.

Suggestions for Improvement

- Experiment with More Advanced Clustering Algorithms: KMeans is effective but might not always be the best choice for complex data. Algorithms like DBSCAN or Gaussian Mixture Models (GMM) could provide better clustering, especially when dealing with non-spherical clusters.
- Feature Engineering: Combining features based on correlation could lead to better performance, but further exploration is needed. New features that capture customer behavior patterns more effectively could improve clustering results.
- Hyperparameter Tuning: While the number of clusters was selected using the Elbow Method, further tuning (testing different numbers of clusters or experimenting with initialization methods) could help improve the clustering results.
- Outlier Handling: While outliers were detected using box plots, further handling of extreme outliers (removing or transforming them) might lead to better-defined clusters.

Conclusion

In this analysis, customer segmentation was performed using the KMeans clustering algorithm to identify distinct customer segments based on purchasing behavior and engagement patterns. Through exploratory data analysis (EDA), transformation techniques, and scaling methods, the data was effectively prepared for clustering. The Elbow Method confirmed that the optimal number of clusters was three, aligning with the objective of identifying Bargain Hunters, High Spenders, and Window Shoppers.

The MinMax scaling method provided the best clustering performance, with a silhouette score of **0.56**, indicating well-separated and cohesive clusters. Despite challenges such as right-skewed data and high correlation between certain features, careful feature engineering and scaling adjustments improved the clustering quality. The insights gained from this analysis offer valuable profiling information, which can help businesses develop targeted marketing strategies and improve customer engagement.

Future improvements could involve experimenting with alternative clustering algorithms such as Gaussian Mixture Models (GMM) and DBSCAN, incorporating additional behavioral features, and exploring temporal patterns to enhance the accuracy and depth of customer segmentation.

Acknowledgement

We, Team HyperTuners, express our sincere gratitude to the UCSC team for organizing the IntelliHack 5.0 competition. Your efforts in providing a challenging platform and valuable resources have inspired us to explore innovative solutions in customer segmentation through clustering. Thank you for this opportunity to grow, learn, and compete.

Team Members:

- Janitha Rajapaksha
- Sanjula Weerasekara
- Hasindu Nimesh
- Veenavee Samarasinghe