NewChic Dataset Analysis

Group 17

Karan Goel: SN 7836685
Alvin Jose: SN 8066358
Ashutosh Bhosale: SN 7795786
Banin Sensha Shreshta: SN 8447196
Gaurav Adarsh Santosh: SN 7032663
Lino Thankachan: SN 7926017
Rishab Manokaran: SN 7863974

CSCI946 Big Data Analytics Assignment 1
August 23, 2024

1 Introduction

In the rapidly evolving e-commerce landscape, leveraging data is essential for making informed decisions regarding consumer behavior and business strategy. This report delves into an extensive analysis of the NewChic.com dataset, focusing on identifying the top 10 products from selected categories and determining the best-performing category.

Our approach began with data preparation and hypothesis testing to ensure accuracy and relevance. We employed two clustering algorithms—K-Means and Agglomerative Clustering (Hierarchical)—to uncover insights into customer behavior and preferences. By analyzing these clusters, we aimed to understand which products resonate most with consumers.

Furthermore, we utilized classification algorithms, including K-Nearest Neighbors and Decision Trees, to predict product popularity and assess the effectiveness of our model in forecasting trends.

This report provides a comprehensive overview of the data analysis process, highlighting the identification of best categories and top products. By comparing the performance of different clustering techniques, we offer actionable insights to enhance NewChic's strategic decisions and optimize product offerings in the competitive e-commerce market.

2 Task 1: Problem Analysis

The dataset used in this analysis was compiled by merging nine different CSV files, each containing product data from distinct categories: accessories, bags, beauty, house, jewelry, kids, men, shoes, and women. This consolidation resulted in a comprehensive DataFrame consisting of 74,999 entries with 22 columns. For the purpose of this analysis, the focus was narrowed down to six key features:

- **Category**: The type or category of the product (e.g., accessories, bags, etc.).
- **Name**: The specific name or identifier of the product.
- **Current Price**: The current selling price of the product.
- **Raw Price**: The original price of the product before any discounts.
- **Likes Count**: The number of likes or upvotes the product has received.
- **Discount**: The discount applied to the product, calculated as a percentage or fixed amount off the raw price.

Objective: The main objectives were to clean and prepare the data for analysis, explore the relationships between the selected features, and test specific hypotheses related to pricing, discounts, and customer preferences.

2.1 Data Characteristics and Challenges

The dataset presents several characteristics and challenges that must be addressed:

- **Diversity of Categories**: The data spans nine distinct product categories, each likely to have different pricing structures, customer preferences, and discount strategies. This diversity requires careful handling during analysis to ensure meaningful comparisons and insights.

- **Size and Complexity**: With nearly 75,000 entries, the dataset is both large and complex, presenting potential computational challenges. Effective preprocessing and outlier detection are crucial to manage these complexities and ensure accurate analysis.
- **Missing Data**: Certain columns, particularly those related to `likes_count` and `discount`, may contain missing values that need to be handled carefully to avoid biasing the analysis.
- **Outliers**: Given the range of products and categories, there may be significant outliers, particularly in numerical features such as `likes_count` and `current_price`. These outliers could skew the results if not properly managed.
- **Interdependencies Among Features**: Features such as `current_price`, `raw_price`, and `discount` are inherently related. Understanding these interdependencies is crucial for accurate analysis and hypothesis testing.

2.2 Hypotheses and Analytical Focus

Given the dataset's characteristics, the analysis focuses on the following key areas, each accompanied by formal hypotheses:

- **Price Sensitivity and Customer Engagement**: Examining the relationship between `current_price` and `likes_count` to understand how price influences customer engagement.
  - **Null Hypothesis (H_0)**: There is no significant correlation between the current price of a product and the number of likes it receives.
  - **Alternative Hypothesis (H_A)**: There is a significant correlation between the current price of a product and the number of likes it receives.
- **Impact of Discounts**: Exploring how discounts affect `likes_count` to uncover whether discounted products receive more attention.
  - **Null Hypothesis (H_0)**: There is no significant relationship between the discount applied to a product and the number of likes it receives.
  - **Alternative Hypothesis (H_A)**: There is a significant relationship between the discount applied to a product and the number of likes it receives.
- **Category-Specific Preferences**: Testing whether customer engagement, as measured by `likes_count`, differs significantly across product categories.
  - **Null Hypothesis (H_0)**: The average number of likes is the same across all product categories.
  - **Alternative Hypothesis (H_A)**: The average number of likes differs across at least one product category.

These hypotheses guide the data preprocessing steps, ensuring that the dataset is prepared in a way that allows for robust and insightful analysis.

3 Task 1: Data Preprocess

The data preprocessing involved several critical steps to ensure the dataset was suitable for detailed analysis. Below is a breakdown of each step:

3.1 Data Loading and Concatenation

The first step was to load the nine CSV files into individual DataFrames and then concatenate them into a single unified DataFrame. This consolidation was essential for handling and analyzing the data as a cohesive unit.

```
data_frames = {
    file.split('.')[0]: pd.read_csv(os.path.join(self.base_path, file))
    for file in self.files
}
self.df = pd.concat(data_frames.values(), ignore_index=True)
```

3.2 Feature Selection and Handling Missing Values

The next step involved selecting the six key features (category, name, current_price, raw_price, likes_count, discount) and handling any missing values. Missing values in `likes_count` and `discount` were filled with zeros, while missing values in `current_price` were filled with the median value of that feature.

```
chosen_columns = {
    'category', 'name', 'current_price', 'raw_price', 'likes_count', 'discount'
}
df = self.df.loc[:, self.df.columns.intersection(chosen_columns)]

fill_values = {
    'likes_count': 0,
    'discount': 0,
    'current_price': df['current_price'].median()
}
for column, value in fill_values.items():
    df[column] = df[column].fillna(value)
```

3.3 Scaling Numerical Features

To ensure that numerical features like `current_price`, `raw_price`, `discount`, and `likes_count` were on a comparable scale, the `StandardScaler` was applied. This step standardizes the data, ensuring that each feature contributes equally to the analysis.

```
numerical = df[['current_price', 'raw_price', 'discount', 'likes_count']]
scaled_features = self.scaler.fit_transform(numerical)
df[['current_price', 'raw_price', 'discount', 'likes_count']] = scaled_features
```

3.4 Encoding Categorical Features

Categorical features such as `category` and `name` were encoded using `LabelEncoder`. This step converted these features into numerical values, which are more suitable for machine learning models and statistical analysis.

```
df['category'] = self.le_category.fit_transform(df['category'])
df['name'] = self.le_name.fit_transform(df['name'])
```

3.5 Outlier Removal

Outliers can skew analysis results, especially in a dataset with numerical features. Outliers were identified using the Z-score method, and any values exceeding the threshold of 3 were removed to ensure the dataset's integrity.

```
z_scores = np.abs(stats.zscore(df.select_dtypes(include=[np.number])))
inliers = (z_scores < 3).all(axis=1)
df = df[inliers]
```

3.6 Task 1: Results

To test Hypotheses 1 and 2, we first constructed a correlation matrix and computed the corresponding p-values, using a significance level (α) of 0.05 Figure 1. Additionally, to address Hypothesis 3, we performed an ANOVA test to determine if the average `likes_count` varies across product categories. Figure 2

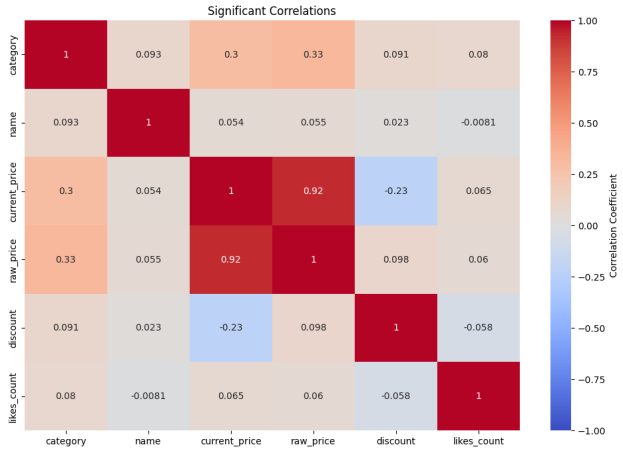

Figure 1: Correlation Matrix

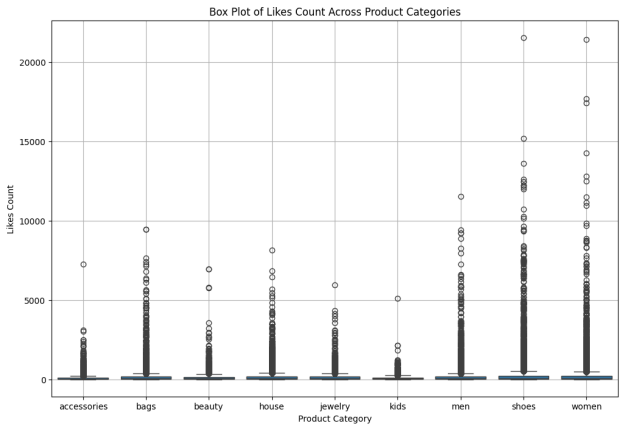

Figure 2: Box Chart for Categories

Results: In both cases, we rejected the null hypotheses.

Based on these findings, we identified the top 7 categories and top 10 products by evaluating `likes_count`, `current_price`, and `discount`.

Detailed results are provided in Section 6.

4 Task 2: Clustering

Our objective is to cluster customer preferences to determine the types of products that are most favored by different customer segments. We proceed as follows:

- **Feature Selection**: The analysis focused on the current_price and discount columns. The raw_price and likes_count columns were excluded as they were not necessary for the clustering process.
- **Target Data**: The name and category columns were retained as target data. Although these columns were not used directly for clustering, they were later employed for interpreting the resulting clusters.

4.1 Elbow Method

The Elbow Method was employed to determine the optimal number of clusters for the KMeans algorithm. This method involves plotting the Within-Cluster Sum of Squares (WSS) against the number of clusters to

identify the "elbow point," which indicates the number of clusters beyond which adding more clusters does not significantly reduce the WSS. Figure 3

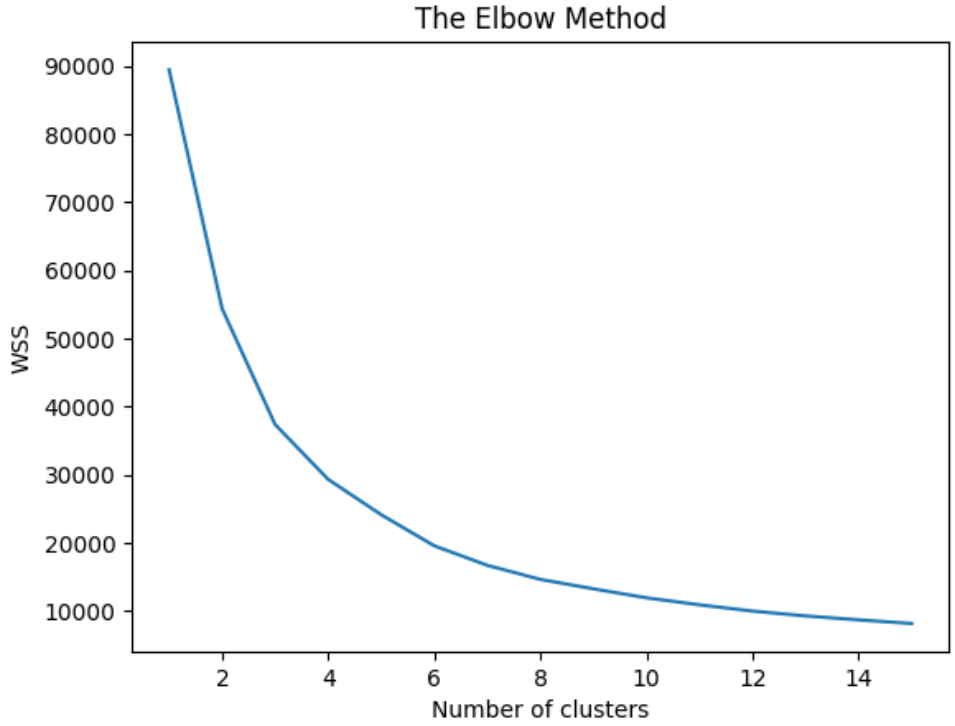

Figure 3: WSS

Based on this method, we determined the optimal value for k to be **7**.

4.2 Clustering Algorithms

Two clustering algorithms were employed:

- **KMeans**: A centroid-based clustering algorithm that partitions the dataset into k clusters, with each cluster represented by the mean of the points within it.
- **Agglomerative Clustering**: A hierarchical clustering technique that builds nested clusters by merging or splitting them successively.

Both algorithms were executed with $k = 7$ clusters, and the results were evaluated using the following metrics:

- **Silhouette Score**: Measures how similar an object is to its own cluster compared to other clusters.
- **Calinski-Harabasz Index**: The ratio of the sum of between-clusters dispersion and within-cluster dispersion.
- **Davies-Bouldin Index**: The average similarity ratio of each cluster with its most similar cluster.

The clustering process is implemented as follows:

```
def perform_clustering(self):
    algorithms = {
        "KMeans": KMeans(n_clusters=7, max_iter=1000),
```

```
    "Agglomerative": AgglomerativeClustering(n_clusters=7, linkage='ward')
}

for name, algorithm in algorithms.items():
    labels = algorithm.fit_predict(self.X)

    # Store the calculated metrics
    self.metrics["Silhouette Score"][name] = silhouette_score(
        self.X, labels)
    self.metrics["Calinski-Harabasz Index"][name] = calinski_harabasz_score(
        self.X, labels)
    self.metrics["Davies-Bouldin Index"][name] = davies_bouldin_score(
        self.X, labels)
```

4.3 Visualization

Clusters were visualized by plotting the 'current_price' against the 'discount' and coloring the points based on their cluster assignments. Additionally, top products were highlighted in red to emphasize their positioning within the clusters. Figure 4 and 5

This helped us understand customer preferences regarding pricing and discounts.

4.4 Justification and Selection of Algorithms

- **KMeans Clustering:**
  - **Reason for Selection:** KMeans is efficient for large datasets and partitions data into 'K' clusters based on proximity to the cluster mean.
  - **Strengths:** Computationally faster, works well with spherical clusters, easy to implement and interpret.
  - **Weaknesses:** Requires pre-specification of 'K', struggles with non-spherical clusters.
- **Agglomerative Clustering:**
  - **Reason for Selection:** Hierarchical technique capturing complex relationships, useful for understanding data structure.
  - **Strengths:** Does not require pre-specification of clusters, can visualize clustering with a dendrogram.
  - **Weaknesses:** Computationally expensive, less effective for large datasets.

4.5 Task 2: Results

This clustering analysis successfully grouped similar products based on their current price and discount. The KMeans algorithm, with $k = 7$ clusters, demonstrated the best performance according to the Silhouette Score, although both algorithms provided valuable insights. The visualizations highlighted the distribution of clusters and the positioning of top products, offering a clear view of the clustering results.

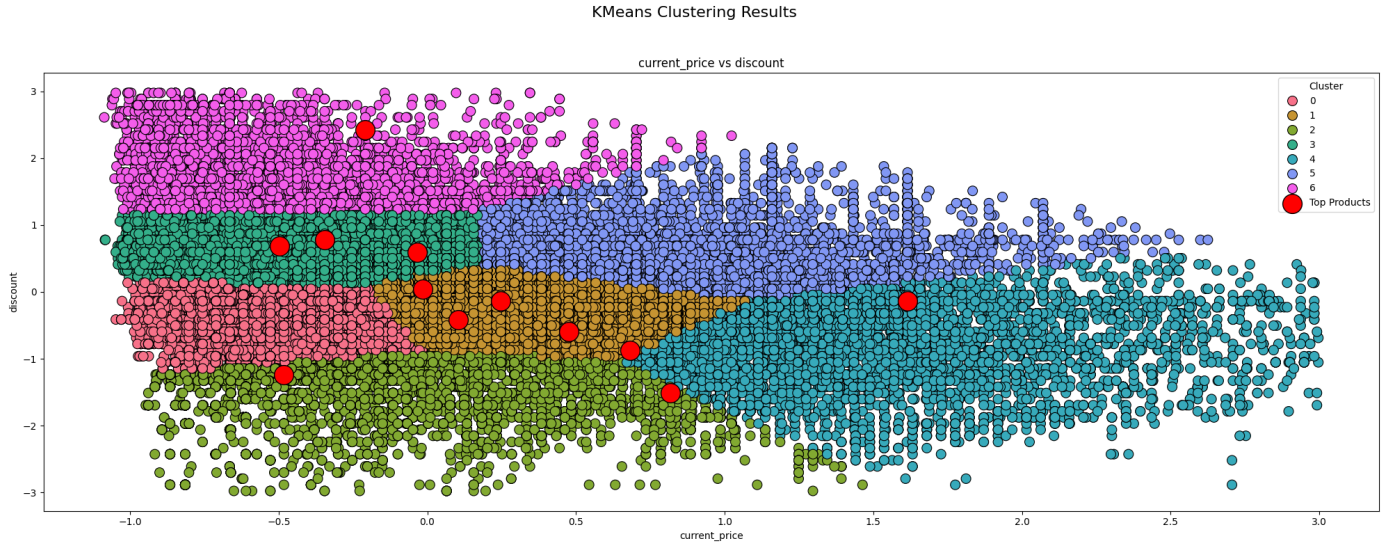

Figure 4: Kmeans Cluster

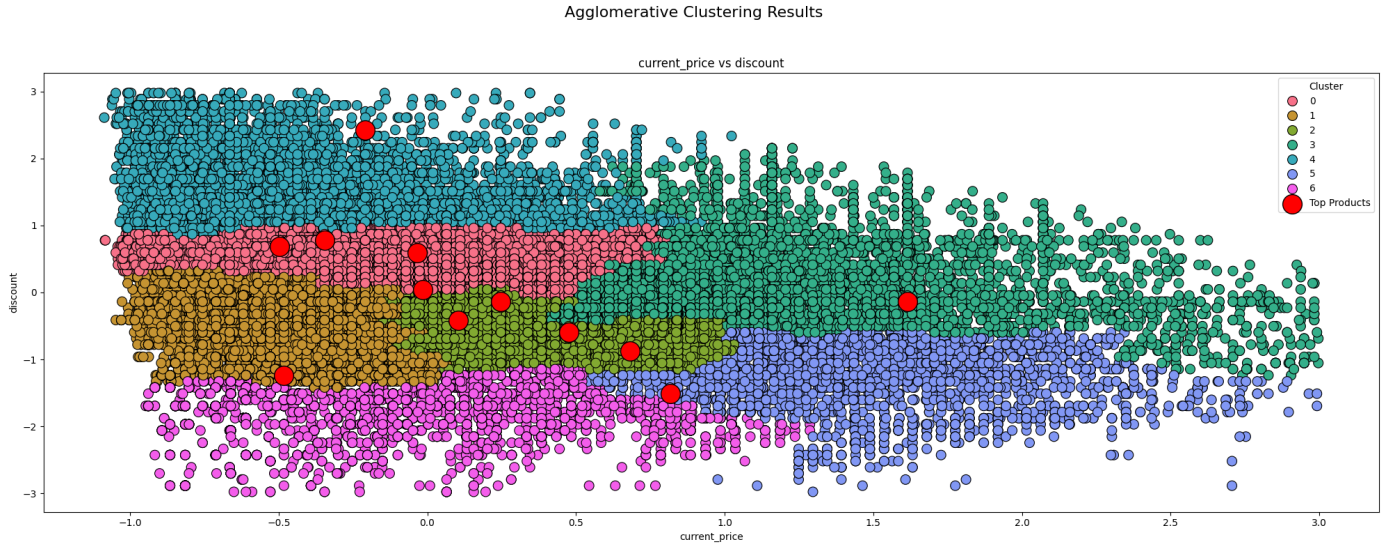

Figure 5: Hierarchy Clusters

Algorithm	Silhouette Score	Calinski-Harabasz Index	Davies-Bouldin Index
KMeans	0.4281	48062.9940	0.7600
Agglomerative	0.3829	20383.9248	0.8066

Table 1: Comparison of Clustering Algorithms

5 Task 3: Classification

Our objective for the classification task is to predict the popularity of a product based on its current price, discount, and category. We employed two different machine learning algorithms: K-Nearest Neighbors (KNN) and Decision Trees.

5.1 Data Preprocessing

To prepare the data for classification, the following steps were undertaken:

1. **Popularity Definition**: A new binary target variable, *popularity*, was created based on the *likes_count* feature. Products with a *likes_count* higher than the dataset's mean were classified as popular (`popularity`

= 1), and the rest as not popular (popularity = 0).

2. **Feature Selection**: The features selected for the classification task included *category*, *current price*, and *discount*. The target variable was *popularity*.

3. **Train-Test Split**: The dataset was split into training and testing sets, with 80% of the data used for training and 20% reserved for testing.

5.2 K-Nearest Neighbors (KNN)

5.2.1 Algorithm Overview

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. The choice of k is critical; a smaller k can lead to high variance, while a larger k may increase bias.

The KNN model was trained using a range of odd values for k between 13 and 19, with the Manhattan distance metric. The best parameters were selected through GridSearchCV, optimizing for accuracy.

```
def train_knn(self):
    knn = KNeighborsClassifier()
    param_grid = {
        'n_neighbors': [i for i in range(13, 20, 2)],
        'weights': ['uniform'],
        'metric': ['manhattan']
    }
    grid_search = GridSearchCV(
        knn, param_grid, cv=self.skf, scoring='accuracy')
    grid_search.fit(self.X_train, self.y_train)

    # Best KNN model
    self.knn_model = grid_search.best_estimator_
    print(f"Best KNN Parameters: {grid_search.best_params_}")

    # Cross-validation scores
    cv_scores = cross_val_score(
        self.knn_model, self.X_train, self.y_train, cv=self.skf, scoring='accurac
    print(f'KNN Average Accuracy: {cv_scores.mean()
```

5.3 Decision Tree

5.3.1 Algorithm Overview

A Decision Tree is a supervised learning algorithm that splits the dataset into subsets based on feature values, creating a tree-like structure of decisions. Each internal node of the tree represents a decision based on a feature, each branch represents the outcome of that decision, and each leaf node represents a class label or regression value. Decision Trees can be used for both classification and regression tasks.

Key parameters for Decision Trees include:

- **Criterion**: The function used to measure the quality of a split (e.g., Gini impurity, entropy).

- **Max Depth**: The maximum depth of the tree, controlling the complexity and preventing overfitting.
- **Min Samples Split**: The minimum number of samples required to split an internal node.
- **Min Samples Leaf**: The minimum number of samples required to be at a leaf node.

The following Python code demonstrates the implementation of the KNN and Decision Tree algorithms for classifying product popularity.

```
def train_decision_tree(self):
    print("Decision Tree")
    dt = DecisionTreeClassifier(random_state=42)
    param_grid = {
        'criterion': ['gini', 'entropy'],
        'max_depth': [None, 10, 20, 30, 40, 50],
        'min_samples_split': [2, 5, 10, 15],
        'min_samples_leaf': [1, 2, 4, 6]
    }
    grid_search = GridSearchCV(
        dt, param_grid, cv=self.skf, n_jobs=-1, verbose=2)
    grid_search.fit(self.X_train, self.y_train)

    # Best Decision Tree model
    self.dt_model = grid_search.best_estimator_
    print(f"Best Decision Tree Parameters: {grid_search.best_params_}")

    # Cross-validation scores
    cv_scores = cross_val_score(
        self.dt_model, self.X_train, self.y_train, cv=self.skf, scoring='accuracy
    print(f'Decision Tree Average Accuracy: {cv_scores.mean() * 100:.2f}%')
    print(f'Decision Tree Standard Deviation: {cv_scores.std() * 100:.2f}%')
```

5.4 Justification and Selection of Algorithms

- **K-Nearest Neighbors (KNN):**
  - **Reason for Selection:** KNN is a straightforward and intuitive algorithm that works well for classification tasks where decision boundaries are not linear.
  - **Strengths:**
    - **Simplicity:** Easy to implement and understand.
    - **Flexibility:** Can be used for both classification and regression.
    - **No Assumptions:** Makes no assumptions about the underlying data distribution.
    - **Adaptability:** Effective with small to medium-sized datasets.
- **Decision Trees:**
  - **Reason for Selection:** Decision Trees are chosen for their interpretability and ability to handle non-linear relationships in the data.

- **Strengths:**
  - **Interpretability:** Easy to interpret and visualize, allowing us to understand the decision-making process and the importance of different features.
  - **Non-Linear Relationships:** Can capture complex, non-linear relationships between features and the target variable.
  - **Feature Importance:** Provides insights into which features contribute most to the predictions.
  - **No Need for Feature Scaling:** Does not require feature scaling, simplifying preprocessing.
  - **Handling of Missing Values:** Can handle missing values in the dataset, making them robust to incomplete data.
  - **Versatility:** Suitable for both classification and regression tasks.

5.5 Task 3: Results

Both algorithms achieved an accuracy of 77% on the dataset, with good precision, recall, and F1 scores. The Decision Tree performed slightly better than K-Nearest Neighbors.

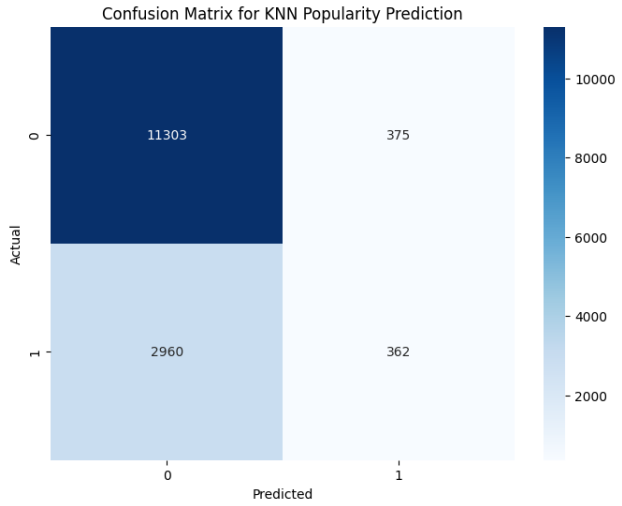

Figure 6: Confusion Matrix for K-Nearest Neighbors

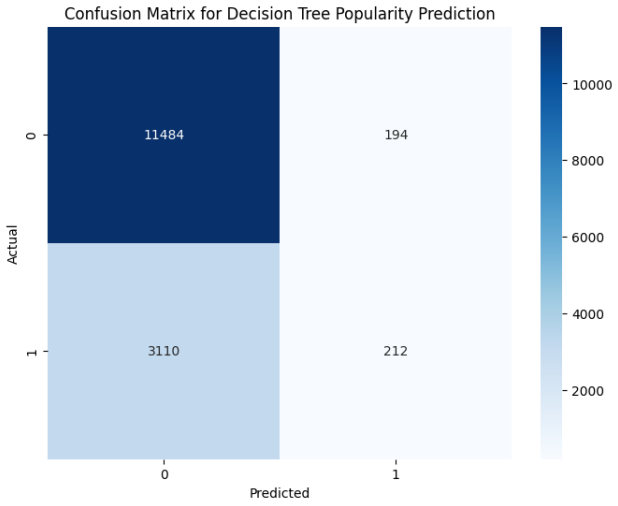

Figure 7: Confusion Matrix for Decision Tree

6 Task 4: Result Discussion

6.1 Are the clusters well-separated from each other?

The clusters are indeed well-separated. We determined the optimal number of clusters k to be 7 by using the Within-Cluster Sum of Squares (WSS) criterion. Plotting the clusters against features such as discount and current price clearly shows distinct separations between clusters. This indicates that the clustering process has effectively identified separate groups within the data.

6.2 Did the classifiers separate products into different classes effectively?

The classifiers demonstrated competent performance in distinguishing products into different classes. The K-Nearest Neighbors (KNN) classifier achieved an accuracy of 77%, while the Decision Trees classifier achieved

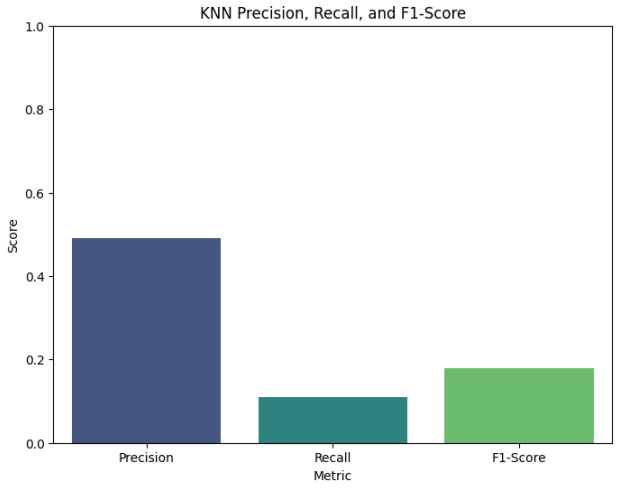

Figure 8: Performance Metrics for K-Nearest Neighbors

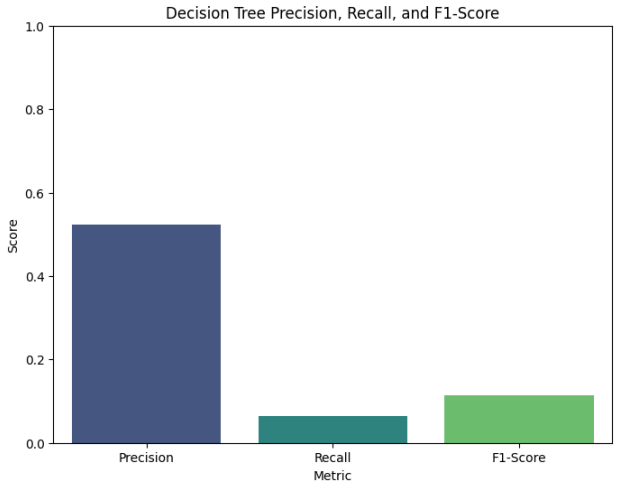

Figure 9: Performance Metrics for Decision Tree

a slightly higher accuracy of 78%. Further analysis revealed that Decision Trees outperformed KNN in terms of Precision, Recall, and F1 Score, suggesting that Decision Trees provide a more balanced classification performance in this context.

6.3 Do any of the clusters/classes have only a few points?

No, all clusters contain a sufficient number of data points. The distribution across clusters is balanced, with none of the clusters having a disproportionately small number of points. This implies that the data is well-represented across all clusters, contributing to the robustness of the analysis.

6.4 Are there meaningful and non-meaningful clusters/classes with respect to the analytics problems outlined in Task 1?

Meaningful Clusters: The analysis reveals that clusters containing products with low current prices, even in the absence of significant discounts, tend to include well-liked products. This suggests that price is a key factor in customer preference, providing actionable insights into pricing strategies.
Non-Meaningful Clusters: Some clusters encompass information from various categories without a clear pattern or insight. These clusters do not provide specific actionable information and may reflect a more random distribution of product attributes.

6.5 What are the advantages and shortcomings of clustering and classification algorithms in this analytics case? Which algorithm provides more valuable results?

Clustering (K-Means):

- **Advantages:** Effective at identifying natural groupings within the data and requires relatively few computational resources. Useful for segmenting products based on attributes like price and discount.
- **Shortcomings:** Sensitive to the initial placement of cluster centroids and the scaling of data. It may not perform well with outliers and requires the number of clusters to be specified in advance.

Classification (KNN and Decision Trees):

- **Advantages:** KNN is effective at capturing local patterns, while Decision Trees offer interpretability and can handle complex relationships in the data. Both models provide insights into how features like price and discount influence product classification.
- **Shortcomings:** KNN can be computationally expensive with large datasets, and Decision Trees may overfit if not properly tuned. The performance of KNN may degrade with increased dataset size.

Which Algorithm Provides Greater Value: In this case, K-Means clustering was particularly valuable as it provided insights into customer behavior and product segmentation. Although Decision Trees performed better in classification tasks, the computational expense of KNN and the interpretability of Decision Trees suggest that clustering offered more actionable insights for understanding product groupings and pricing strategies.

6.6 Are the examined algorithms suitable for Big Data analytics? If so, why?

K-Means Clustering: K-Means is suitable for large datasets due to its relatively low computational complexity. However, with very large datasets, it may become computationally intensive and sensitive to high-dimensional data.
KNN: KNN is generally not well-suited for Big Data analytics because it requires storing the entire training dataset and has high computational costs, especially for large datasets.
Decision Trees: Decision Trees are more scalable and can handle larger datasets better than KNN. They are effective in managing complexity and high-dimensional data, though they may require careful tuning to avoid overfitting with very large datasets.

6.7 Will data preprocessing affect clustering and classification results? If so, how?

Yes, data preprocessing significantly impacts both clustering and classification results.

- **Scaling:** Standardizing or normalizing data ensures that all features contribute equally to distance calculations, which is crucial for algorithms like K-Means and KNN.
- **Outliers:** Removing outliers enhances the robustness of clustering and classification algorithms, preventing them from being unduly influenced by extreme values.

Proper preprocessing improves the quality of the clusters and the accuracy of the classifiers by ensuring that the data is clean, well-scaled, and appropriately prepared for analysis.

Table 2: Top 7 Categories (based on average likes count)

Category
Shoes
Women
Jewelry
House
Bags
Men
Beauty

Table 3: Top 10 Products in the Top 7 Categories

Product
Sandales élastiques à strass perle
Chaussures Vintage Rétro Respirantes Maille Ballerines À Fleur Avec Noeud Chinois
Sac de rangement étanche
Combinaison ample sans manches
Tongs plats en suède à entredoigt à fleur décorative
Bottes Vintage à Imprimé Fleurs
Manteau à Capuche Imprimé avec Poches
12 couleurs maquillage ombre à paupières stylo
Pompes papillon peintes à la main
Robe Casual Couleur Unie

References

Bishop, C. M. (2006). *Pattern recognition and machine learning.* Springer. Retrieved from `https://www.springer.com/gp/book/9780387310732`

Developers, N. (2024). *Numpy: The fundamental package for scientific computing with python.* Retrieved from `https://numpy.org`

Géron, A. (2019). *Hands-on machine learning with scikit-learn, keras, and tensorflow.* O'Reilly Media. Retrieved from `https://www.oreilly.com/library/view/hands-on-machine/9781492032632/`

learn Developers, S. (2024). Scikit-learn user guide. *Scikit-learn Documentation.* Retrieved from `https://scikit-learn.org/stable/user_guide.html`

Team, P. D. (2024). *Pandas: Python data analysis library.* Retrieved from `https://pandas.pydata.org`

VanderPlas, J. (2016). *Python data science handbook: Essential tools for working with data.* O'Reilly Media. Retrieved from `https://jakevdp.github.io/PythonDataScienceHandbook/`