

CSCI946 - Assignment 1

Full Name	Student ID	Student Email	Task Log
Robert Mete	5736262	rm700@uowmail.edu.au	Data Processing & Problem Analysis
Kyuta Yasuda	6588773	ky733@uowmail.edu.au	Clustering
Mikael Shahly	8945512	majs774@uowmail.edu.au	Classification
Thamonwan Nitatwichit	8026300	tn304@uowmail.edu.au	Clustering
Huu Thien Pham	7794587	htp898@uowmail.edu.au	Classification
Syed Eisa Alamgir	8188154	sea994@uowmail.edu.au	Problem Analysis
Anas Shahid Raja	8279366	asr968@uowmail.edu.au	Data Processing

All members contributed equally.

Task 1: Problem Analysis and Data Preprocess

Experiment Design

- Defining "Top 10" Products

- Defining "Best" Category

Task 2: Clustering

Data Observation and Approach

Experiment 1

- Data Preparation

- Clustering Model Selection

 - KMean

 - Gaussian Mixture

- Rationales

- Model Evaluation

 - Hyperparameter Tuning

- Experiment 1 Result

- Experiment 2

 - DBSCAN

- Result from experiment 1 and 2

Task 3: Classification

- Approach

- Random Forest Classifier

- Logistic Regression

Results

Task 4: Result Discussion

KMeans vs. GMM and DBSCAN

Clustering Model Limitations and Insights

Cluster Interpretations from Experiment 1

Recommendations on NewChic Dataset for Future Clustering

Random Forest Classifier vs. Logistic Regression

Bibliography

Task 1: Problem Analysis and Data Preprocess

Experiment Design

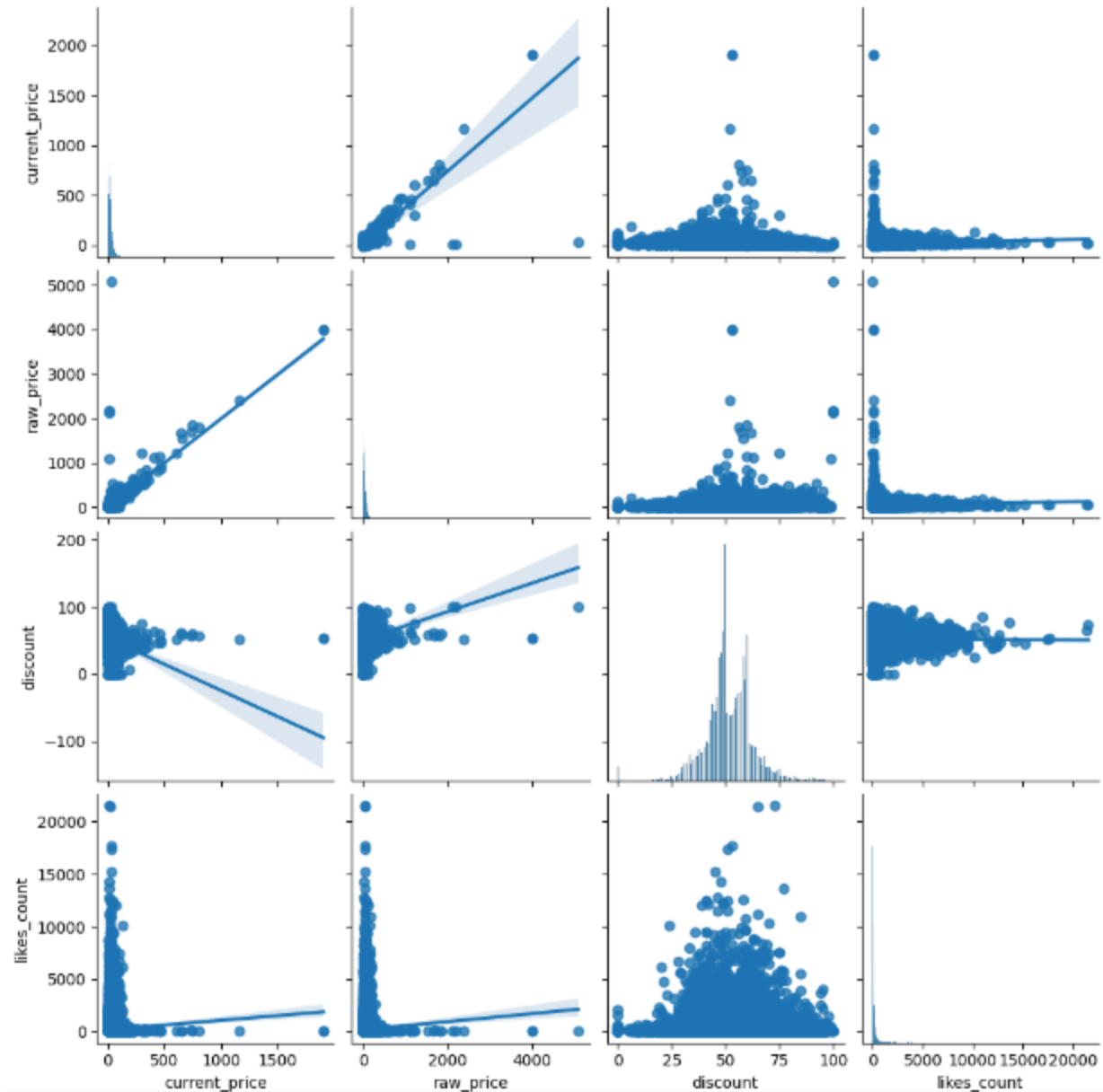
The objective of the problem analysis stage is to thoroughly understand the NewChic dataset, which organises its products into categories. This stage is designed to structure the analysis effectively to address two primary questions: identifying the top 10 products from selected categories and determining the best category among those selected. To effectively answer these questions, it is crucial to undertake an Exploratory Data Analysis (EDA) phase. This phase involves learning about the business domain, framing the problem, and delving deeply into the data to ensure we are well-positioned to meet the two central objectives.

Defining "Top 10" Products

When defining the metric to calculate the "Best" products, it is crucial to consider the dataset's domain, the key stakeholders, and the associated business objectives. Products can be ranked based on various metrics such as sales volume, profit margins, and customer ratings, all of which are valid measures. However, by considering the broader context of the dataset and its relevance to the business, we can establish a metric that is both statistically robust and aligned with the business goals.

The dataset under analysis comes from the e-commerce website NewChic.com, which organises its product records across seven categories, each stored in a separate .csv file. These categories range from men's products to jewellery, with each file containing similar columns but differing in the number of products. Given that NewChic operates in the fashion e-commerce space, this context helps frame the problem of identifying the top 10 products.

In the analysis, **Figure 1 - Pair Plot** demonstrates the relationships between various numeric features, with regression lines highlighting any linear correlations. Notably, the "ID" feature was excluded from these plots. A key observation is the strong linear relationship between "Raw_price" and "Current_price," which aligns with the fact that the "Discount" feature determines the relationship between these two prices.



An interesting pattern emerges in the relationship between "Like_counts" and "Discount," which follows a bell curve. Initially, increasing the discount leads to a sharp rise in "Like_counts," but this relationship peaks, and further discounts actually result in fewer likes. This suggests the existence of an optimal discount level that maximises the number of likes a product receives. Thus, the interaction between "Discount" and "Like_counts" can be used to define a metric that measures the success of NewChic's promotional campaigns.

The advantage of using this metric is that it incorporates "Like_counts," reflecting customer engagement—a key interest for NewChic's stakeholders—while "Discount" represents NewChic's promotional efforts. This dual consideration ensures the metric is both customer-focused and aligned with the company's promotional strategies.

Furthermore, this metric has intuitive interpretations for both high and low values, as outlined below:

- High Discount with High Likes: Typically indicates a successful promotional strategy, where significant price reductions attract customers and lead to favourable product ratings.
- High Discount with Low Likes: Suggests that despite the price incentive, the product fails to meet customer expectations or is in a saturated market.
- Low Discount with High Likes: Strongly indicates a product's inherent appeal, suggesting popularity even without major discounts, likely due to quality, brand reputation, or unique features.
- Low Discount with Low Likes: Might signal a product struggling in the market, possibly due to high competition, poor quality, or ineffective marketing.

From these outcomes, products with both high discounts and high likes are considered the best and will form the basis of our metric. Technically, the metric used to define the best products represents those where NewChic has set an appropriate discount rate, leading to a high number of customer likes. To define this metric, it is important to understand the distribution of both the "Discount" and "Like_counts" features.

Figure 2 - discounts density plot, shown below displays two prominent peaks. The first, higher peak is around the 50% discount mark, suggesting that many products have a discount close to 50%. The second peak is slightly lower, occurring around the 60% discount mark, indicating another common discount level. This suggests that NewChic frequently applies moderate discounts (around 50-60%) to its products.

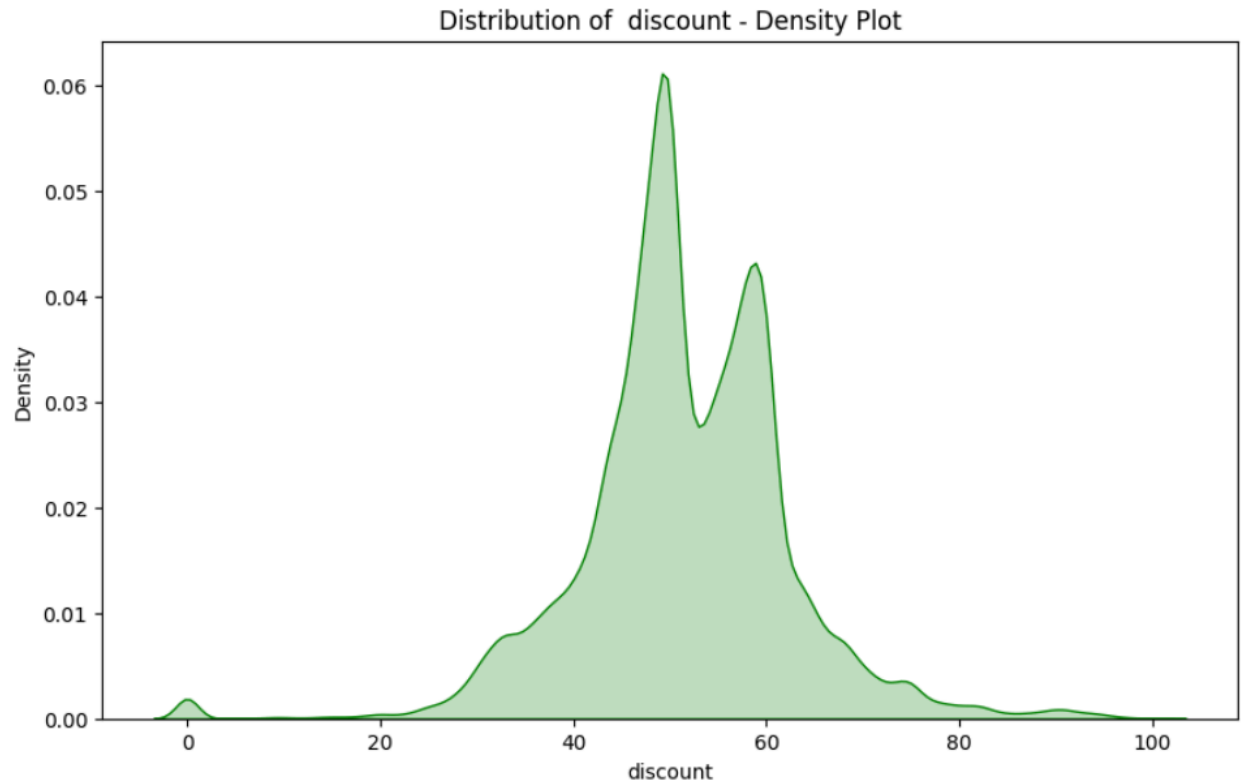
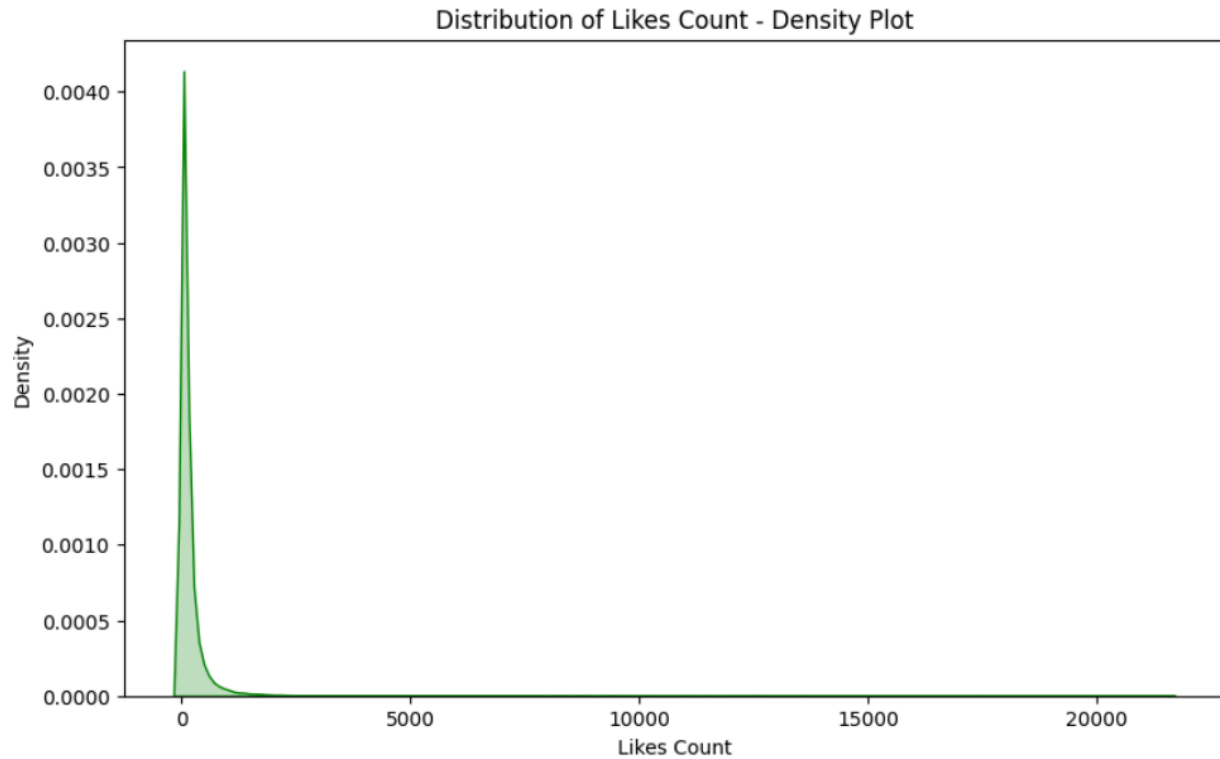
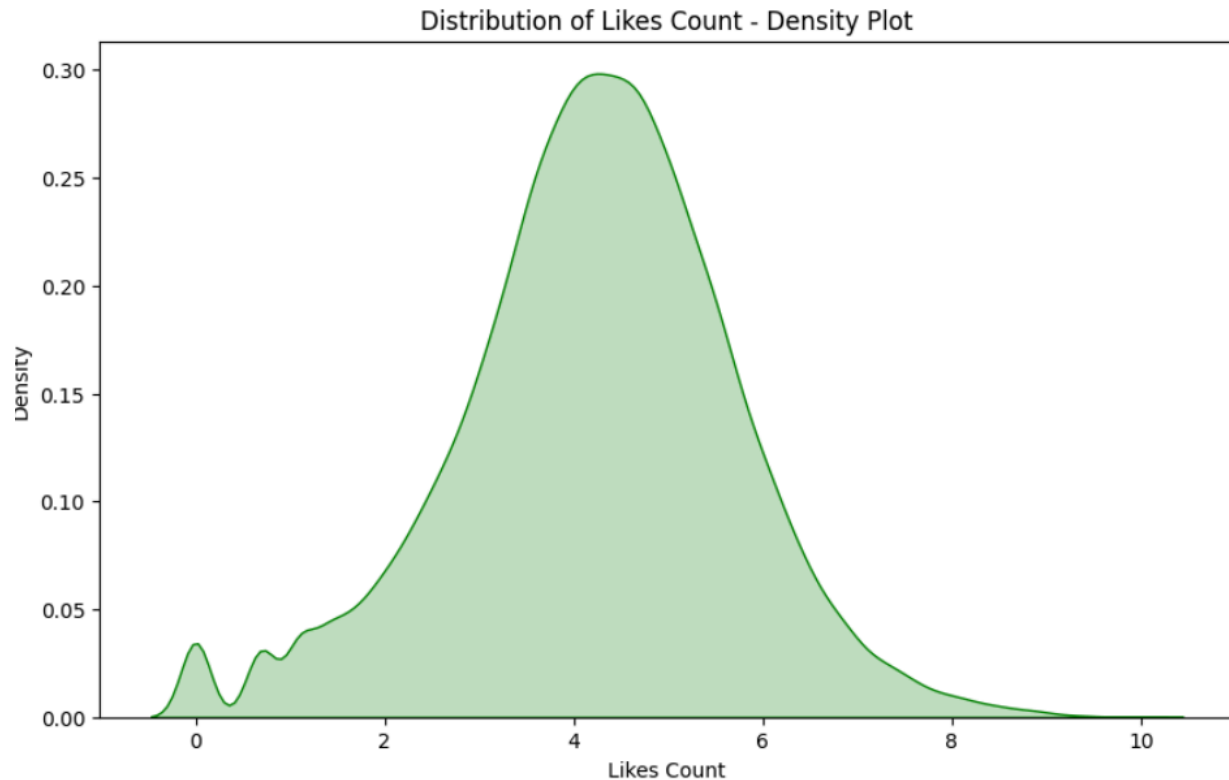


Figure 3 - Like_counts density plot, displayed below reveals that most products have low popularity, as indicated by the high density near the lower "Like_counts" values, with only a few products managing to gather a significant number of likes.



To gain a better understanding, a log transformation of "Like_counts" was applied and plotted below as **Figure 4 - Log Like_counts density plot**. The majority of products tend to receive a moderate number of likes, with 4 being the most common count.



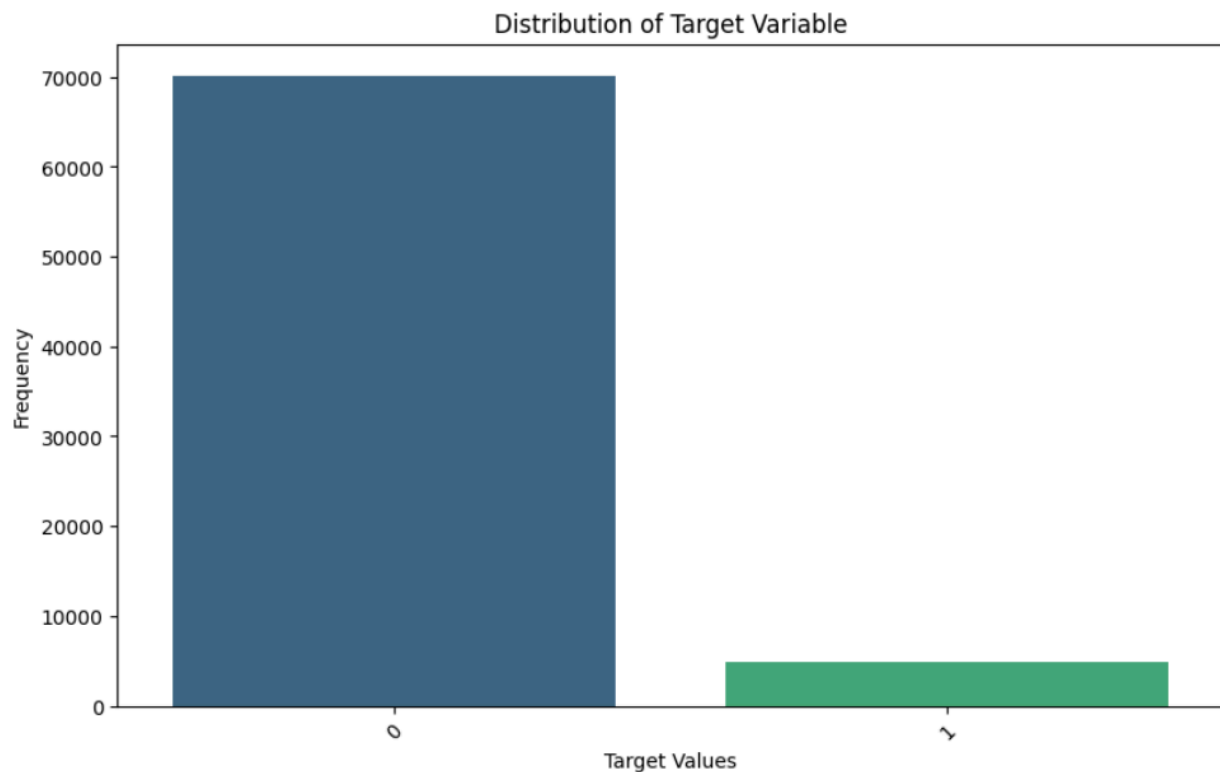
It should be noted that both the "Discount" and the log of "Like_counts" features are on different scales and need to be transformed to a similar scale. The Standard Scaler from sklearn will be applied to both of these features. After this, the metric will simply be the sum of the standardised "Discount" and the standardised log of "Like_counts." This can be expressed as the below equation:

$$\text{Metric} = \text{Discount_Scaled} + \text{Log_Like_Count_Scaled}$$

To create the target variable for classification, a threshold will be used to classify the metric into two groups: 1 indicates a high score, categorising the product as "best," and 0 indicates a low score, categorising the product as "not the best." **Figure 5 - Metric Statistics**, shown below displays the main statistics of the metric. The third percentile of the metric is approximately 0.842, so it was decided to round up and use 1 as the threshold.

discount_likes_feature	
count	7.499900e+04
mean	-1.197517e-16
std	1.389671e+00
min	-7.555051e+00
25%	-8.428188e-01
50%	1.949591e-02
75%	8.421283e-01
max	6.737422e+00

After Applying this threshold, this leaves approx 70,000 records having the class 0 (not good) and roughly 5000 records having the class 1 (good). This can be seen in **Figure 6 - Target Distribution**.



Defining "Best" Category

With the criteria for identifying the 'Best' products clearly established, we can now determine the top-performing category. Our evaluation leverages the output from two classification models: Random Forest and Logistic Regression. These models provide probabilistic predictions for each product, indicating whether they are classified as 'good' or 'not good'. After this step, the products were grouped per category to count the total number of 'Best' products within each category.

To mitigate potential biases arising from unequal record counts across categories, we calculated the ratio of 'Best' products to the total number of products in each category. This normalisation ensures a fair comparison by accounting for differences in category sizes. The categories are then ranked from best to worst based on these ratios. These rankings are presented in a table beneath the classification results, offering a clear visual representation of category performance.

Rank	Category	No. of Products Classified as "Good"	Total Items	Ratio
1	women	1080	14809	0.072929
2	shoes	857	11823	0.072486
3	jewellery	289	4853	0.059551
4	bags	274	6268	0.043714
5	beauty	149	3804	0.039169
6	men	385	10208	0.037716
7	house	457	12791	0.035728
8	accessories	149	6358	0.023435
9	kids	92	4085	0.022521

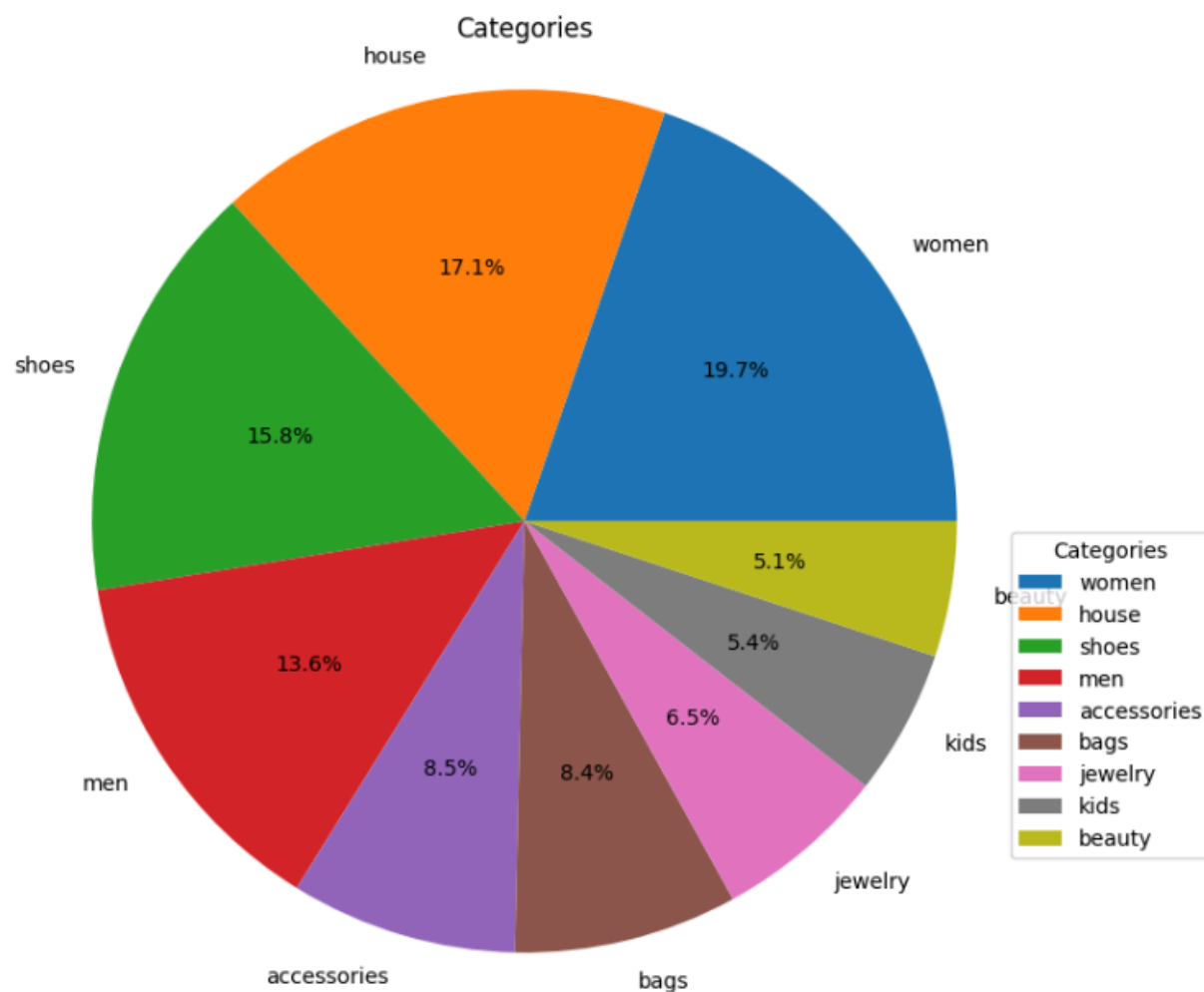
The women's and shoes categories emerged as the top performers, while accessories and kids lagged behind. These findings will be explored in more detail in the following sections.

Data Processing

The data preprocessing phase for the NewChic dataset is geared towards transforming the raw data into a format that is optimised for machine learning algorithms. This phase encompasses two main tasks: enhancing data quality and scaling variables for better model suitability.

Our initial step involved identifying and addressing duplicate records. We discovered 483 records with duplicate ID values. After a thorough review, we decided to retain only the first occurrence of each duplicate to preserve data integrity and minimise redundancy. Additionally, we encountered records displaying a current price of zero, implying a 100% discount. These records were deemed inconsistent with standard e-commerce practices and were subsequently removed from the dataset.

Despite the dominance of Women, Home, and Shoes categories in the dataset, we chose to retain all categories to maximise the potential for discovering significant trends. **Figure 7 - Category Pie Chart**, depicts the distribution of records across various categories, ensuring a holistic view of the data for comprehensive analysis.

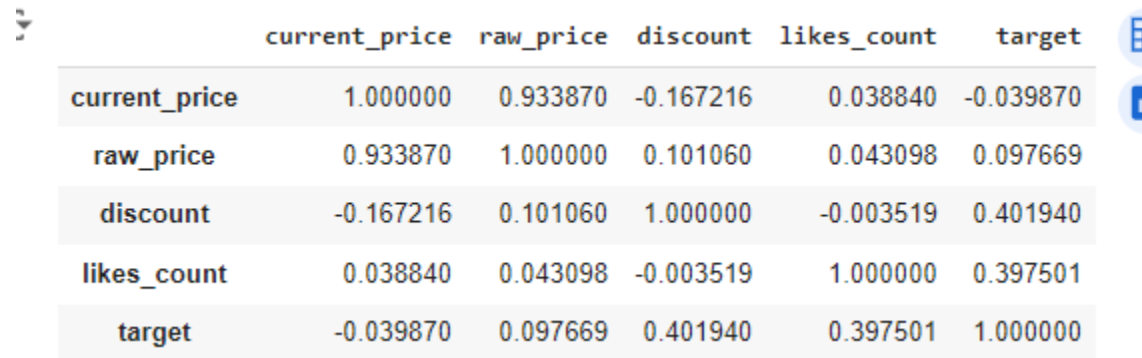


We selected a subset of numeric features as outlined by the project criteria and utilised Scikit-learn's StandardScaler to normalise these features. This standardisation is crucial for many machine learning algorithms, as it ensures that all features contribute equally to the model's performance.

The data was then divided into training and test sets using a 67-33% split, resulting in approximately 50,000 records for training and 25,000 for testing. This division allows us to assess the model's performance on unseen data, providing a reliable measure of its generalizability.

For specific analytical tasks, two distinct datasets are provided: one for classification and another for clustering. Clustering, an unsupervised learning task, does not utilise a target variable, focusing instead on exploring the data's structure. In contrast, classification, a supervised learning task, requires a target variable to train the model effectively.

Figure 8 - Post processing correlation, revealed no significant correlation between the target and any independent variables. However, high multicollinearity was noted between 'current price' and 'raw price,' complicating the assessment of their individual impacts on the dependent variable. This could potentially skew the model's feature importance interpretation. To mitigate this, techniques like Principal Component Analysis (PCA) are recommended to reduce inter-variable correlation, which will be further discussed in subsequent sections.



	current_price	raw_price	discount	likes_count	target
current_price	1.000000	0.933870	-0.167216	0.038840	-0.039870
raw_price	0.933870	1.000000	0.101060	0.043098	0.097669
discount	-0.167216	0.101060	1.000000	-0.003519	0.401940
likes_count	0.038840	0.043098	-0.003519	1.000000	0.397501
target	-0.039870	0.097669	0.401940	0.397501	1.000000

Task 2: Clustering

Data Observation and Approach

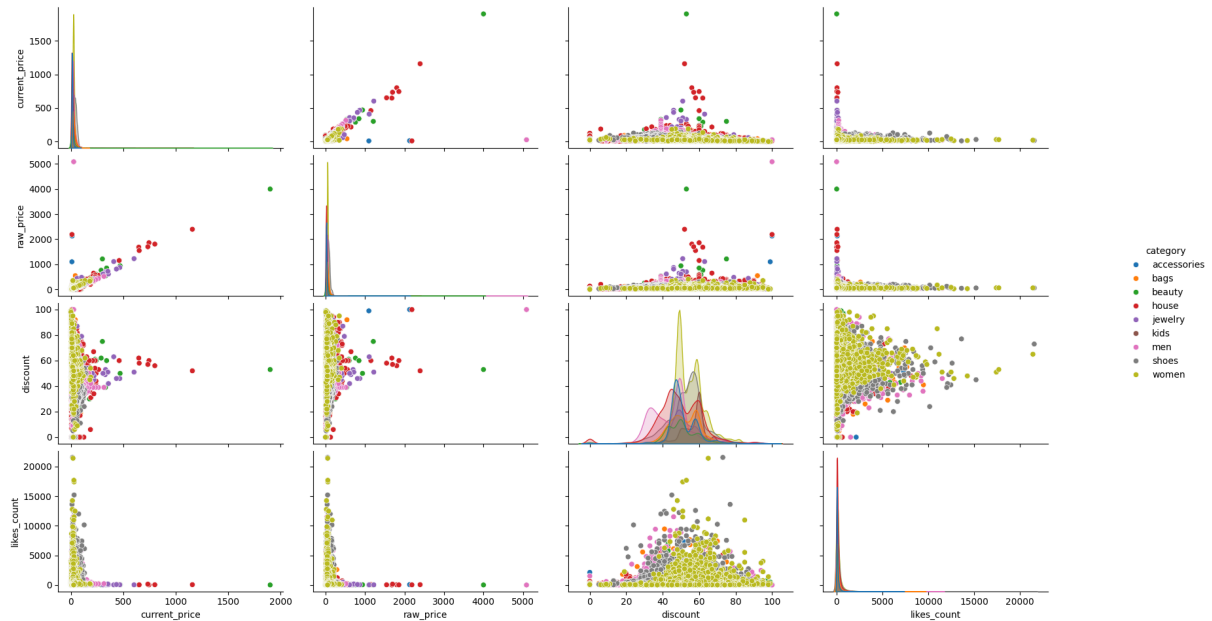


Figure 9A: visualisation display data relationships viewed in category feature

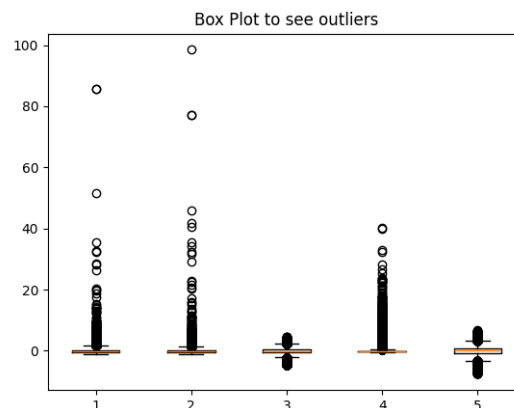


Figure 9B: Box plot for outlier observation

From figure 9A, data visualisation after exploration and pre-processing reveals significant skewness in certain features, particularly in `current_price`, `likes_count`, and `raw_price`. These features exhibit right tails indicative of positive skewness, suggesting the presence of outliers or a long tail, which could potentially affect the clustering results. The `discount` feature, with its considerable number of low or zero values and mild right skewness, may naturally segment the data into products with and without discounts, potentially causing the use of non-parametric methods. Additionally, from figure 9B, outliers displayed in the box plot should be paid attention as they can significantly influence the results of clustering analyses

(Scribbr, n.d.). However, the large number of outlier-like data points can be perceived as special or rare products. Notably, as seen in figure 7 and 9A, some categories, such as "women," have larger proportions compared to others. However, since the aim is to focus on clustering the data points, the proportion of each category may still be considered when identifying the best category later.

Therefore, 2 experiments have been conducted. The first experiment is to use every existing numerical feature except Id and Principal Component Analysis (PCA) before clustering by KMeans and Gaussian Mixture Model regardless of category context, meanwhile the experiment 2 emphasises on feature selection before clustering. These 2 experiments are created in order to differently observe the data structure in the NewChic dataset.

Experiment 1

Data Preparation

As the numerical features in the dataset are current_price, likes_count, raw_price, and discount, clustering method applied onto all features to find the hidden data patterns. Therefore, Principal Component Analysis (PCA) was employed for dimensionality reduction. This step aimed to improve the clustering process by focusing on the most significant features that capture the majority of the variance in the data (HBC Training, n.d.).

Table 1: table demonstrates coefficients of PC1 and PC2 towards each numerical feature

Features	PC1	PC2
current_price	-0.044407	0.710172
raw_price	0.082761	0.699116
discount	0.484356	-0.062437
likes_count	0.262530	0.053347

In the above table, PC1 is significantly influenced by discount and likes_count, with smaller influence from raw_price and a minimal negative influence from current_price, suggesting that PC1 could represent a dimension that relates more to customer engagement regarding pricing strategy (discounts and likes), meanwhile, PC2 is heavily influenced by current_price and raw_price, with current_price being the most significant contributor. Potentially, PC2 represents a dimension primarily related to pricing (both current and raw prices).

Clustering Model Selection

KMean

KMean is a centroid-based algorithm that partitions the data into k clusters by minimising the within-cluster variance. Its strengths are simpleness, computational efficiency, and the ability to work well with spherical clusters. In contrast, it usually struggles with non-spherical clusters.

Also, its sensitivity to outliers is considerable compared to other clustering models and requires the accurate number of clusters to be specified in advance (Scikit-learn developers, n.d.).

Gaussian Mixture

It can provide a probabilistic framework for clustering under the assumption that the data is generated from a mixture of several Gaussian distributions, offering insights into the underlying data structure (Deepchecks, n.d.) GMM can model clusters with varying shapes and sizes, and it provides probabilistic cluster memberships. On the other hand, it can be sensitive to initialization and the assumption of normality may not always be true.

Rationales

The choice of these models allows for a comprehensive comparison of different clustering techniques, with their unique strengths. To illustrate, KMeans provides a baseline with its simplicity, GMM provides a probabilistic approach that can accommodate different cluster shapes..

Model Evaluation

Silhouette Score: Measures how similar an object is to its own cluster compared to other clusters. It ranges from -1 to 1, with higher values indicating better clustering.

Elbow Method: Helps in determining the optimal number of clusters by plotting the sum of squared distances from each point to its assigned cluster centre. This will be applied to only KMeans as acknowledgment of the number of clusters crucial to KMeans performance.

Bayesian Information Criterion (BIC): Evaluates the model fit while penalising for the number of parameters, helping to avoid overfitting (Deepchecks, n.d.). BIC is particularly suitable for evaluating probabilistic models like GMMs because it is derived from the likelihood function, which is central to how GMMs work. The lower BIC score, the better model fitting.

Hyperparameter Tuning

Optuna is selected for hyperparameter tuning due to its efficiency, flexibility, ease of use, and powerful features like adaptive sampling, pruning, and multi-objective optimization. Whether working on simple models or complex deep learning architectures, Optuna provides the tools needed to efficiently explore the hyperparameter space and find the best configuration, ultimately leading to better model performance and reduced computational cost (Roelofs, 2020).

Experiment 1 Result

The clustering results were visualised using plots generated by Optuna, which provided insights into the performance of each model based on the selected metrics. With the same number of trials (25) for KMeans and Gaussian Mixture Model.

KMeans

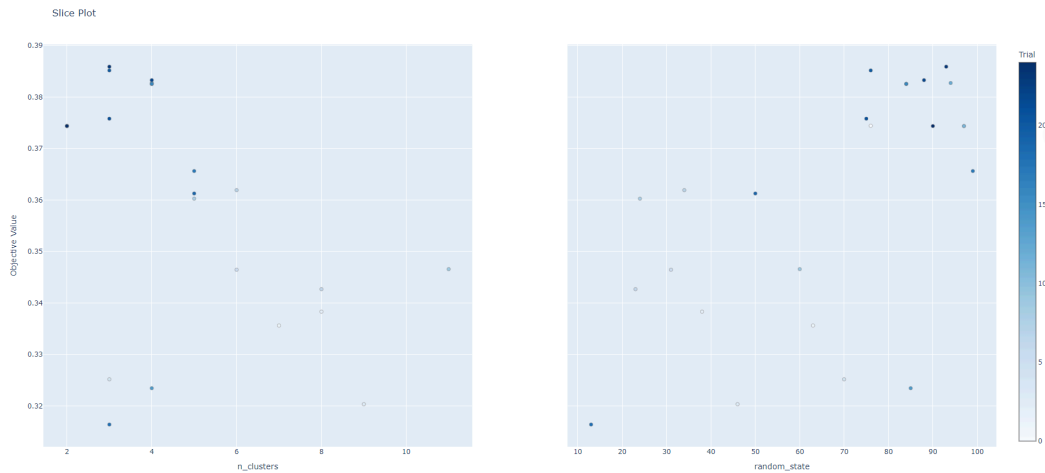


Figure 10: The scatter plot of parameters with the objective value

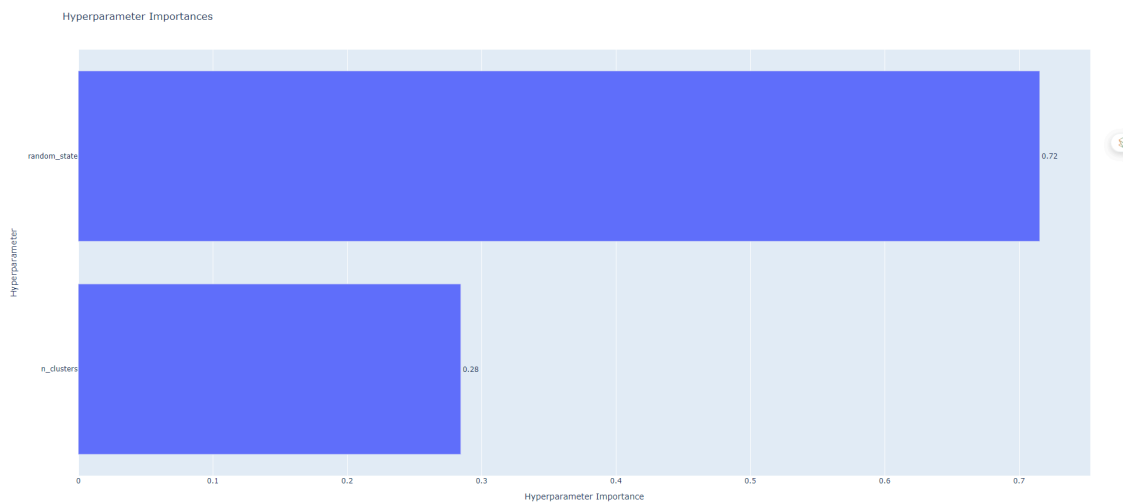


Figure 11: Hyperparameter Importance

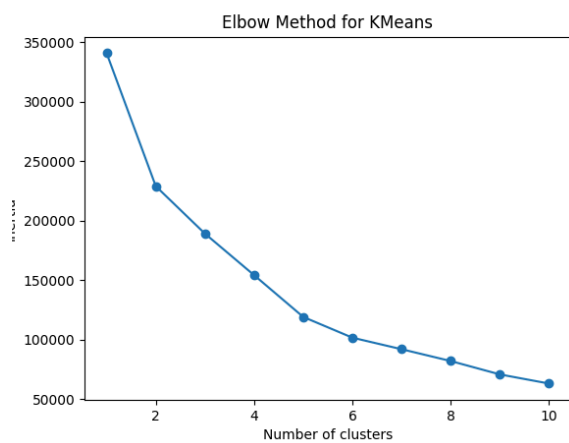


Figure 12: Elbow Method for KMeans

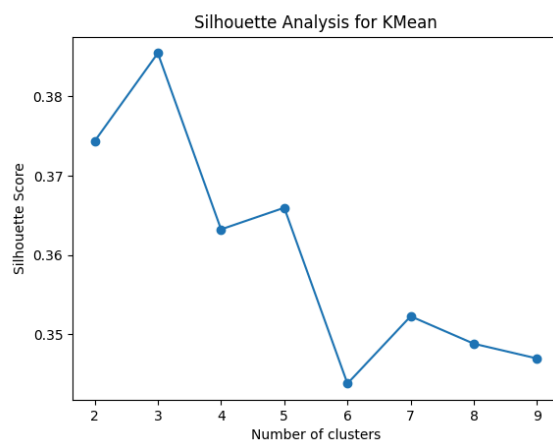


Figure 13: Silhouette analysis for KMeans

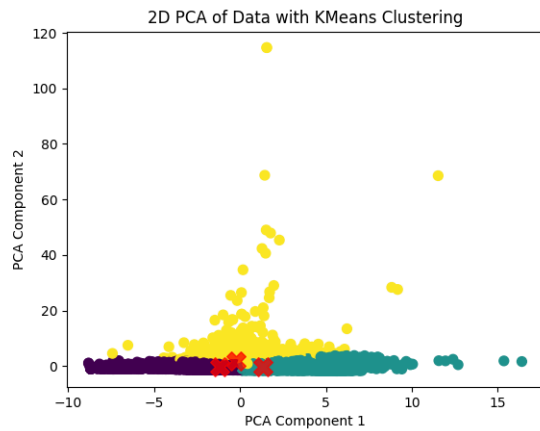


Figure 14: KMeans plot with Optuna tuning

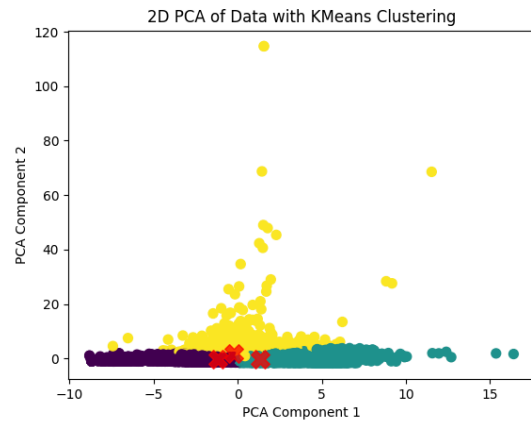


Figure 15: KMeans plot with self-observation

Regarding figure 10 and 11, the importance of parameter 'n_cluster' weighs 72%, leading to n_cluster: 3 and random_state: 93. Notedly, the parameters tuned by Optuna suggest only 3 clusters while the parameters self-observed from Elbow plot and Silhouette analysis also suggest 3 clusters, implicitly telling that the results match. Overall, both KMeans with self-observed parameters and Optuna tuning score low on Silhouette analysis. Overall, the model tried to group products that belong to a particular directional range of PC1 value, either positive or negative, then group the products that have moderate to high values of PC2 together.

Gaussian Mixture Model

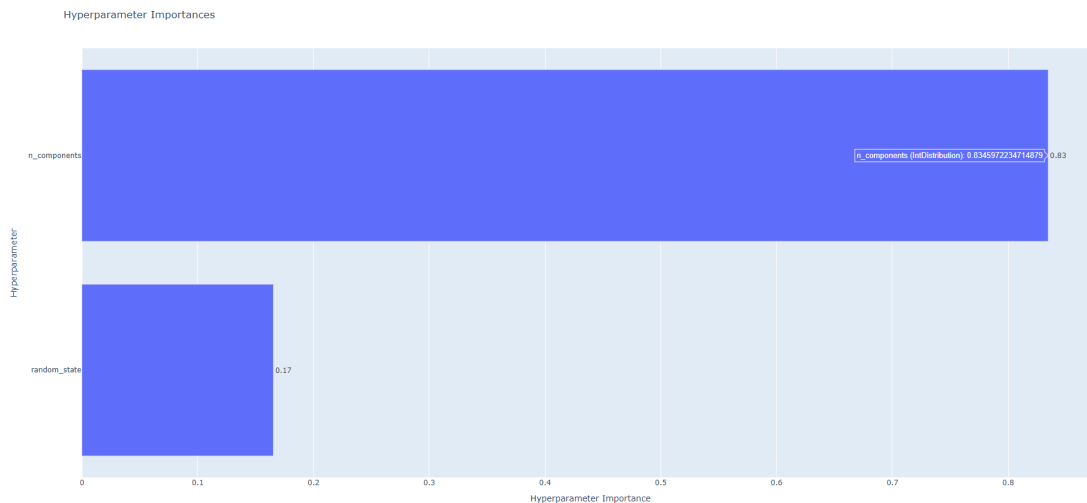


Figure 16: Hyperparameter Importance

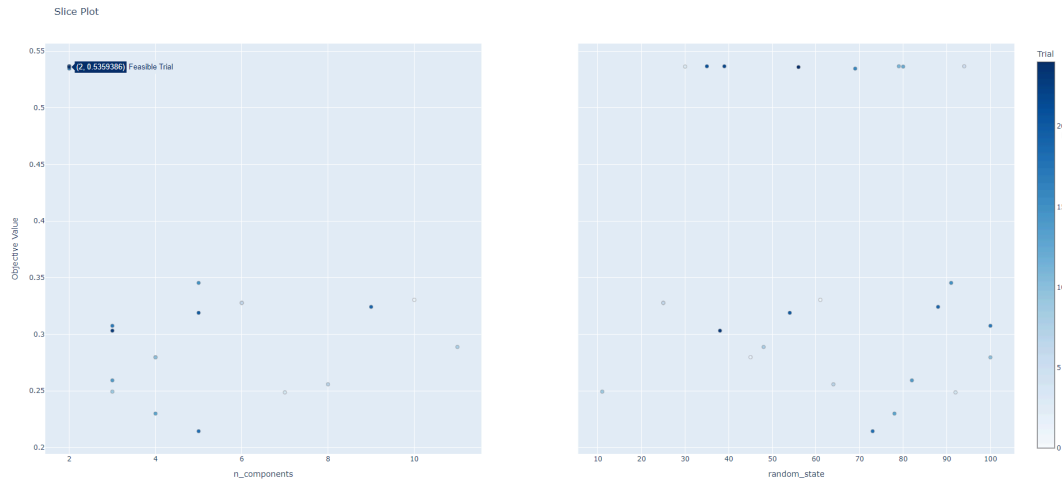


Figure 17: The scatter plot of parameters with the objective value

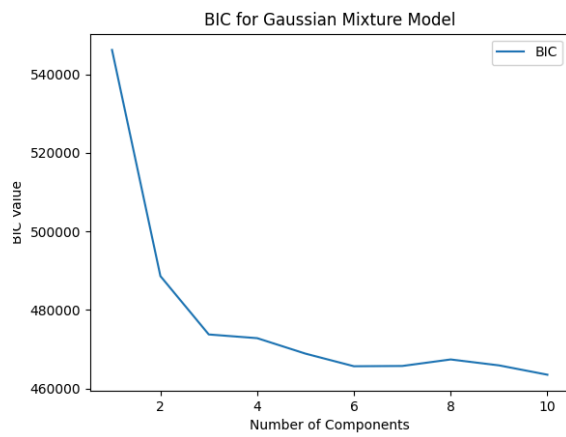


Figure 18: BIC for GMM

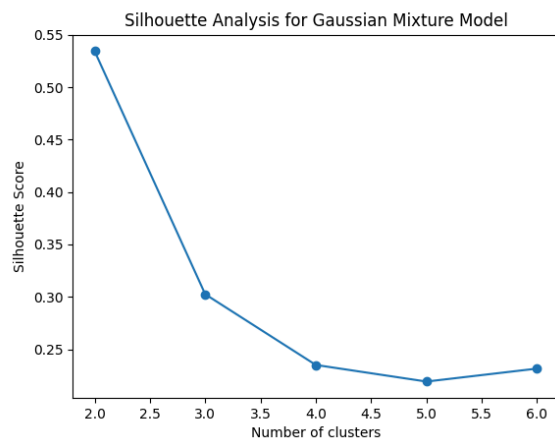


Figure 19: Silhouette analysis for GMM

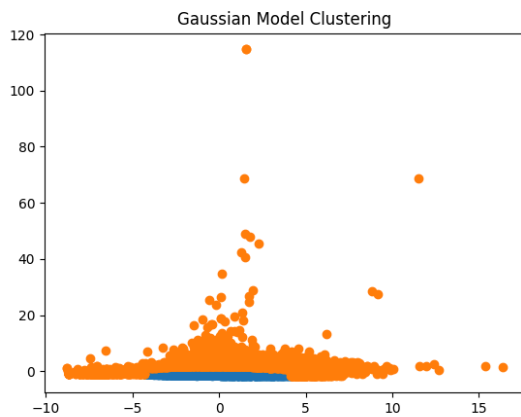


Figure 20: Gaussian with Optuna tuning

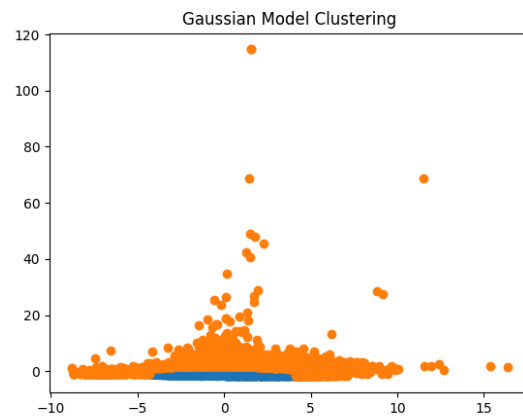


Figure 21: Gaussian with self-observed parameters

According to figure 16 and 17, the importance of parameter 'n_components' weighed 83%, leading to n_cluster: 2 and random_state: 94. Interestingly, the parameters tuned by Optuna suggest 2 clusters and the parameters self-observed from Elbow plot and Silhouette analysis also suggested 2 clusters, implicitly telling the same matter with KMeans. However, the Optuna function in the code was only assigned a Silhouette score for evaluation. From figure 20 and 21, GMM could separate fewer clusters than KMeans. What the model attempted was basically group data points that belong to a small range of PC1, including having remarkably low PC2 values, and the data points that have a wider range of PC1 values as well as PC2 values.

Experiment 2

DBSCAN

DBSCAN is a density-based clustering algorithm that is useful for identifying clusters of varying shapes and sizes. The code iteratively tests different combinations of `eps` (the maximum distance between two points for them to be considered as in the same neighbourhood) and `min_samples` (the number of points in a neighbourhood for a point to be considered as a core point) to find the best clustering configuration based on silhouette score.

DBSCAN

```

eps=0.1, min_samples=5, silhouette_score=-0.45250853759800214
eps=0.1, min_samples=15, silhouette_score=-0.4630611041190063
eps=0.1, min_samples=25, silhouette_score=-0.23046802057682958
eps=0.1, min_samples=35, silhouette_score=-0.21175851990661027
eps=0.1, min_samples=45, silhouette_score=-0.2623717483020558
eps=0.3111111111111111, min_samples=5, silhouette_score=0.2423990629647864
eps=0.3111111111111111, min_samples=15, silhouette_score=0.5192287089429547
eps=0.3111111111111111, min_samples=25, silhouette_score=0.49769432170897826
eps=0.3111111111111111, min_samples=35, silhouette_score=0.4800411342747565
eps=0.3111111111111111, min_samples=45, silhouette_score=0.46433780950357856
eps=0.5222222222222223, min_samples=5, silhouette_score=0.3766288915124283
eps=0.5222222222222223, min_samples=15, silhouette_score=0.39945283503917073
eps=0.5222222222222223, min_samples=25, silhouette_score=0.43273945515464735
eps=0.5222222222222223, min_samples=35, silhouette_score=0.5344753287726508
eps=0.5222222222222223, min_samples=45, silhouette_score=0.5342073457933485
eps=0.7333333333333333, min_samples=5, silhouette_score=0.4291951604015483
eps=0.7333333333333333, min_samples=15, silhouette_score=0.5159838452262553
eps=0.7333333333333333, min_samples=25, silhouette_score=0.5018069559936279
eps=0.7333333333333333, min_samples=35, silhouette_score=0.4612917006715471
eps=0.7333333333333333, min_samples=45, silhouette_score=0.44505308059771537

```

Figure 22: Hyperparameter Tuning for DBSCAN

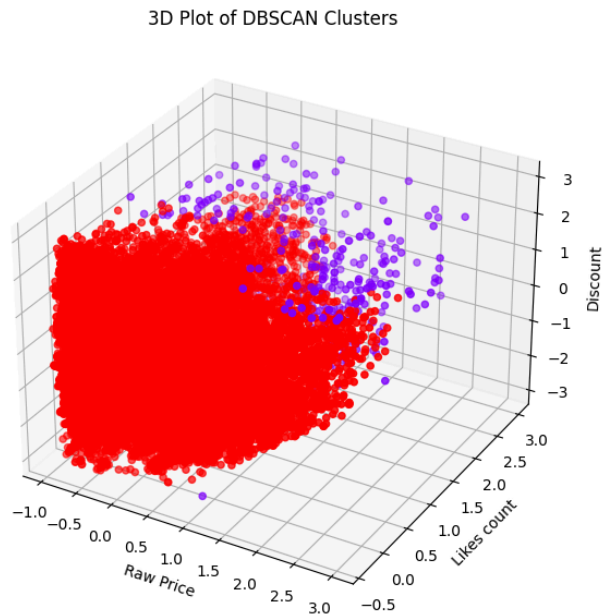


Figure 23: 3D Plot of DBSCAN Clusters

3D Plot of DBSCAN Clusters from a different angle

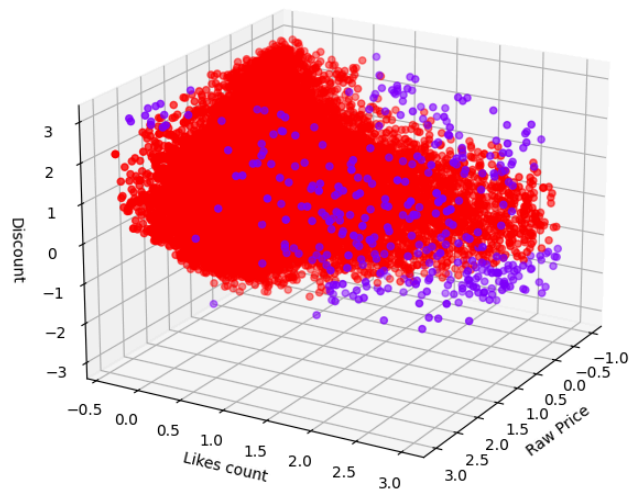


Figure 24: 3D Plot from Different Angle

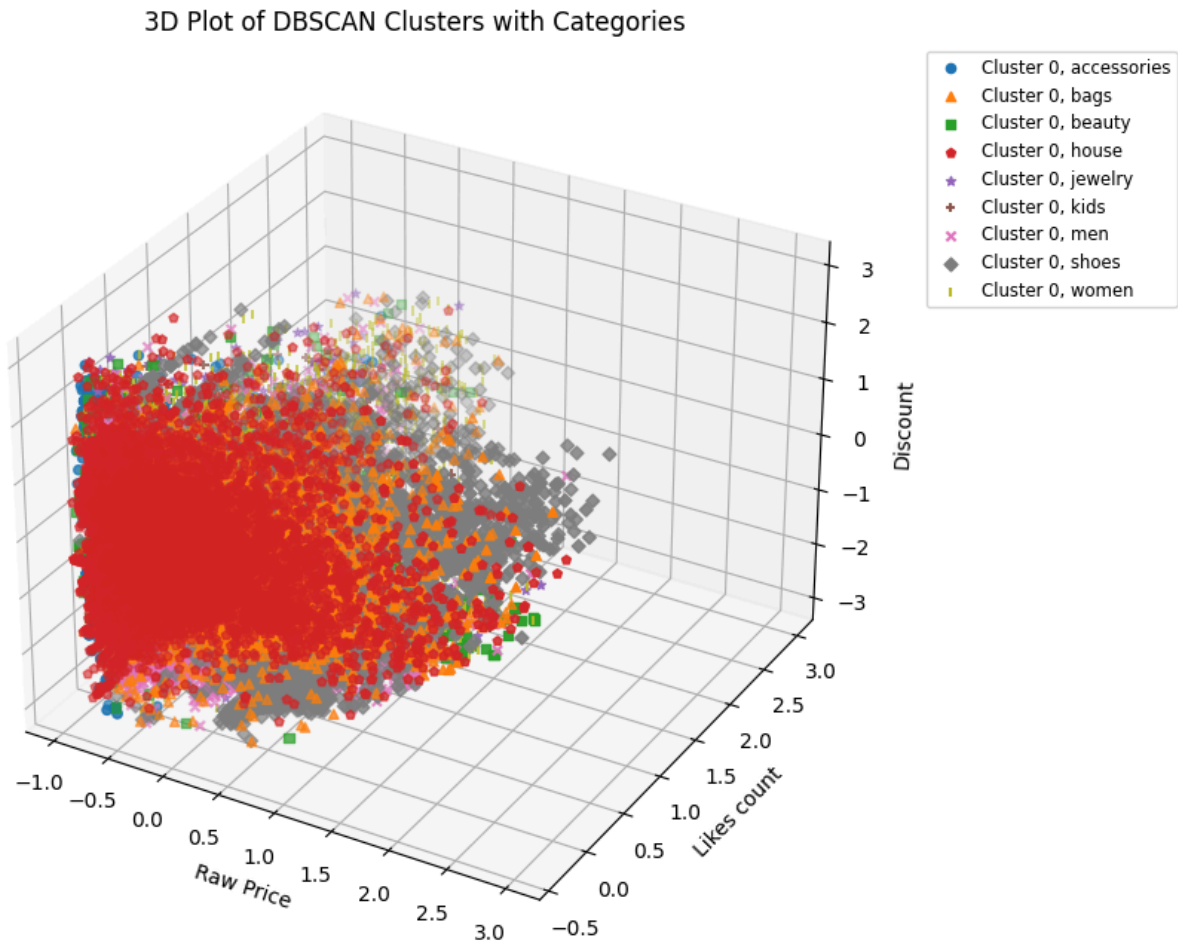


Figure 25: 3D Plot with Categories

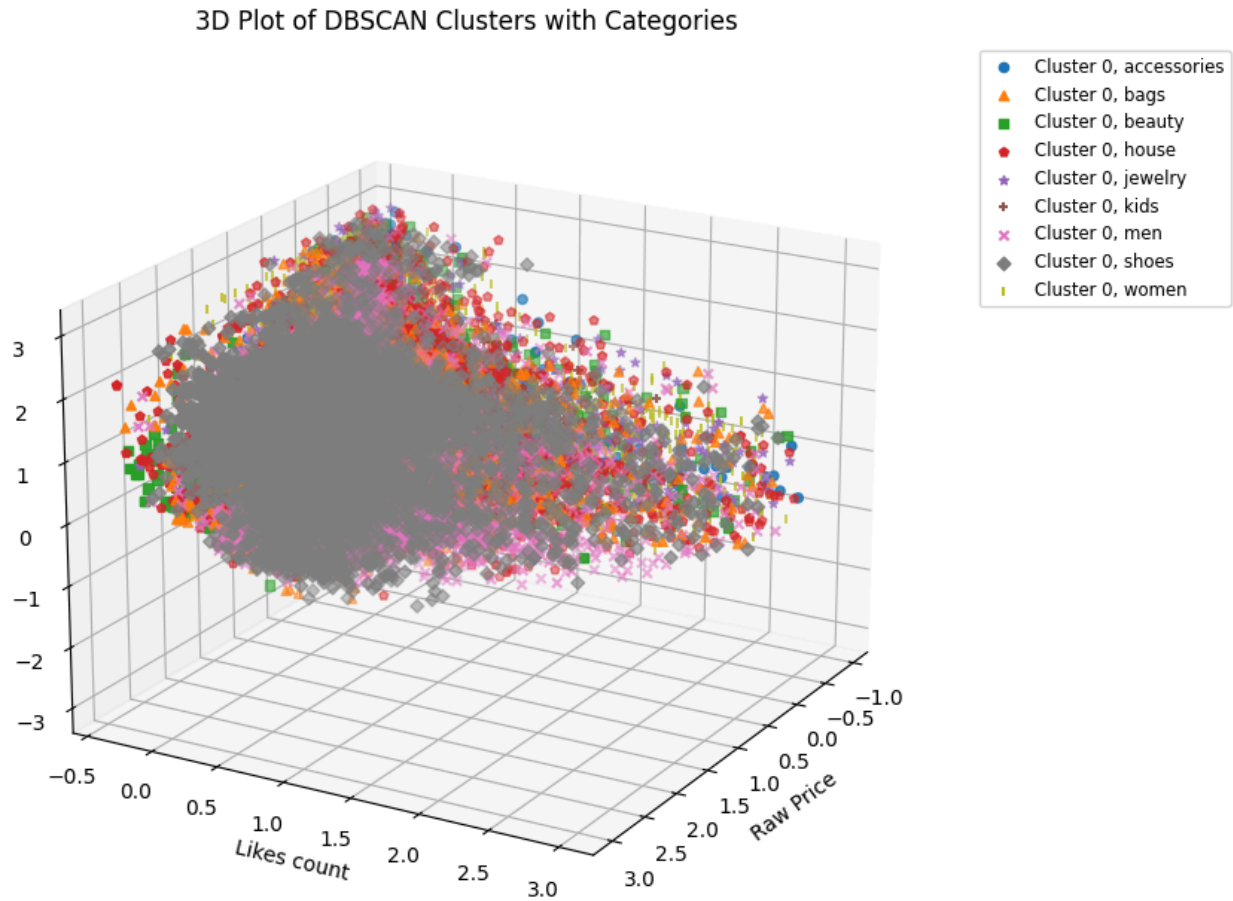


Figure 26: 3D Plot with Categories from Different Angle

Although DBSCAN is known to be robust to outliers, the readability of the plots gets reduced when the distance between the majority of the data points and the outliers is large. Hence, the outliers were removed using z-scores of the data with the threshold of 3. Also, in order to examine relationships between the categories and clusters, the category column was converted into numerical values using the scikit-learn label encoder. These additional data preprocessing were applied specifically for DBSCAN as well as scaling, extracting numerical values and so on.

Two hyperparameters: 'eps' and 'min_samples' were optimised using silhouette scores. With a pair of eps=0.52 and min_samples=35, the model was able to achieve the highest silhouette score of 0.534.

Figure 14 and 15 show that most of the data points fall within a few large clusters, indicated by the dense regions in the plots. These clusters represent groups of data

points that are close together in the feature space defined by 'raw_price', 'likes_count' and 'discount'.

The use of different markers for categories allows us to see how different categories are distributed across clusters. This visualisation can help identify whether certain categories tend to cluster together or if they are evenly distributed across clusters. In figure 16 and 17, it can be seen that some categories such as house, bags, and accessories tend to position on the left side of the cluster, on the other hand, categories such as shoes and men tend to be on the right side of the cluster.

Result from experiment 1 and 2

Metrics\models	KMean	Gaussian Mixture	DBSCAN
Silhouette score	0.38584533173676044	0.5343813629456909	0.5344753287726508
BIC	-	464329.3946224161	-

Task 3: Classification

Approach

Both classification models were trained to correctly classify the target label in our preprocessed data representing the product as "bad" or "good".

Random Forest Classifier

The first classification algorithm that was performed on the preprocessed dataset was a random forest classifier. A random forest classifier is taking multiple decision trees and bagging them by training multiple decision trees on bootstrapped datasets. To introduce even more randomness (and therefore reduce the variance of the model even more), the algorithm only allows a random selection of features to be used for the splits at each node. In order to thoroughly explain the steps of the algorithm some key concepts must be explained.

1. How does a decision tree algorithm function
2. The bias variance tradeoff in classification models
3. Bootstrapping and bagging a model

The Decision Tree algorithm works by recursively splitting a dataset into subsets based on feature values, using a tree-like structure. The splits are based on the feature that gives the optimal separation of classes based on some measure of information gain; typically entropy, Gini impurity or log loss. Deciding which is a matter of hyperparameter tuning.

The tree structure starts with a root node containing the entire dataset and ends with leaf nodes, which assign a class label to the data points that reach them based on the splits. Each leaf node corresponds to a decision about the class based on the feature splits that led to it. At each node in-between these the splits are performed.

Any error made by a classifier on a test set can be decomposed into an irreducible error, a bias term and a variance term (James, Witten, Hastie, & Tibshirani, 2013). A quick example can be made for a mean square error function:

$$E[(f(x) - \hat{f}(x))^2] = \text{bias}(\hat{f}(x))^2 + \text{var}(\hat{f}(x)) + \text{irreducible error}$$

More intuitively; Bias are errors due to overly simplistic models that fail to capture the underlying patterns in the data, leading to systematic inaccuracies. High bias typically results in underfitting. Variance however are errors due to excessive complexity in the model, which captures noise in the training data as if it were a pattern. High variance typically results in overfitting. Therefore it is essential for a model to minimise both. Decision trees are typically low bias/high variance models (complex), to reduce the variance the random forest performs bagging on multiple decision tree models (James, Witten, Hastie, & Tibshirani, 2013).

Finally, bootstrapping refers to resampling the dataset with replacement from the original dataset. Somewhat unintuitively, the bootstrapped dataset remains a reasonable approximation of the actual population the original dataset represents. This means we can train separate decision trees on separate bootstrapped datasets. We can then average the predictions of all these separate decision trees, this process is referred to as bagging (James, Witten, Hastie, & Tibshirani, 2013). The variance reduction of the model by doing this can be shown in the equation:

$$\text{var}(\hat{f}(x)_{\text{avg}}) = \text{var}\left(\frac{1}{\# \text{ Bootstrapped datasets}} \sum_{b=1}^{\# \text{ Bootstrapped datasets}} \hat{f}(b)\right) = \frac{\text{var}(\hat{f}(x))}{\# \text{ Bootstrapped datasets}}$$

Hyperparameter tuning is performed on a validation dataset. This is done through k-fold cross validation with 5 folds. To hyperparameter tune "Optuna" is used again. For our hyperparameter tuning we have some special things to take into consideration for a Random Forest: 1) Likely there is no point in pruning the bagged trees as we likely introduce too much bias then 2) Adding more decisions trees to the forest will likely never hurt accuracy as shown in the equation above, However for large amount of decision trees the reduction in variance will become negligible and only introduce unnecessary training times.

Also, the tree depth is purposely kept somewhat low. The probability of a certain sample being in a certain class is calculated based on the mean predicted class probabilities of the leaf nodes in the forests. The probability of a single tree is the fraction of samples of the class in a leaf node. Therefore, if all samples in a node are the same class we get a probability of 1.0 / 0.0. With a high depth this is more likely to happen resulting in more data samples having a probability of 1.0 of being in class "good" product. This makes it impossible to distinguish which of these samples are the top 10 samples.

Logistic Regression

The second algorithm chosen for this classification experiment is logistic regression, which is relatively popular due to its speed and simplicity. Computation resources are less required, so deploying a logistic regression model is fast and efficient. The model's coefficients are also easy to interpret, and they facilitate better understanding of the relationship between features than other, more robust, models. Logistic regression, however, has setbacks when dealing with non-linear relationships and multicollinearity in the dataset. As a rule of thumb, assumptions about linear relationships between variables are necessary for performing logistic regression. Feature scaling would also optimise its performance.

The NewChic dataset is relatively small in the big data analytics landscape, so logistic regression is a viable option, although most likely not satisfactory for high-stakes classification tasks. On the bright side, if the company likes to interpret the fitting results, they can understand the features' impact and communicate this understanding in case of business decision-making. Moreover, logistic regression is a probabilistic classifier which is helpful for predicting metric-based classes, for example best-selling, most price-sensitive, and most profitable products.

As its name suggests, the model uses the logistic function, also known as the sigmoid function, to predict mapping probabilities between 0 and 1.

$$P(y = 1|x) = \frac{1}{1+e^{-z}} \text{ where } z = wx + b, \text{ and}$$

w as the weight vector, x as the feature vector, and b as the bias term.

z is also known as the linear combination of the features.

Fitting is initialised with a random set of weights, often all zeros. During fitting, the weights are adjusted to minimise the differences between predicted probabilities and actual binary outcomes, i.e. 0 and 1. This process is also known as optimisation, and the function to calculate these differences is called loss function. A common optimization technique is gradient descent.

Results

Confusion matrix and accuracy is shown below for the classification models on the test dataset

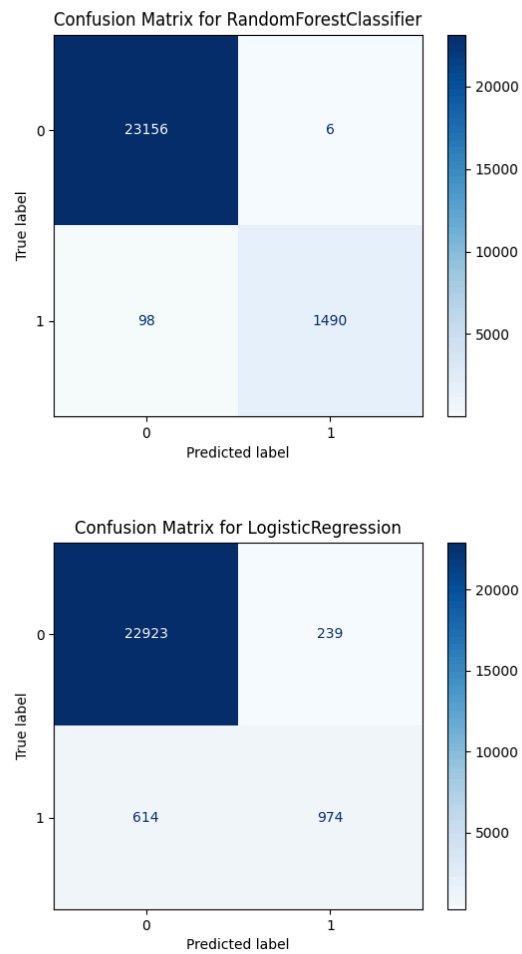


Figure 27: Confusion Matrix for Random Forest Classifier and Logistic Regression Classifier

Metrics\Models	Random Forest Classifier	Logistic Regression
Accuracy	0.995797980	0.965535354
Recall	0.938287154	0.613350126
F1 Score	0.966277562	0.695465905

Index	ID	Name	Predicted Probability
64146	1151213	Soutien-gorge Sexe Adhésif Invisible Sans Bret...	0.998815
50313	1290039	Sandales à velcros bout ouvert à plateforme	0.996930

63991	1432583	T-shirt imprimé patchwork patch	0.996692
64405	1070729	Soutien-gorge Sexy Respirant Sans Armature en ...	0.996666
50316	1310539	Chaussures décontractées en lin	0.996053
51380	1155362	Sandales Perforées Peep Toes Fermeture Éclair ...	0.996022
57904	1357764	SOCOFY Bottines en cuir véritable	0.995998
50381	1268461	Sandales tendance pointure large pour femme	0.995938
51344	1341770	SOCOFY Bottines rétro en cuir véritable à couture	0.995864
64160	1202750	Soutien-gorge sans Armature en Coton avec Ouve...	0.995830

Top 10 Products Predicted by Random Forest

Index	ID	Name	Predicted Probability
38164	1361733	Chemise vintage en couleur pure à col montant	1.0
48386	1124740	Chaussures Souples De Grande Taille Mocassins ...	1.0
48416	1113128	Socofy Chaussures Plates Faites À La Main En C...	1.0
48445	1124740	Chaussures Souples De Grande Taille Mocassins ...	1.0
48475	1113128	Socofy Chaussures Plates Faites À La Main En C...	1.0
50188	1100211	Bottines Plates Doublées de Fourrure	1.0
54981	1159404	SOCOFY Sandales Confortables Plates Avec Bride...	1.0
55676	1107438	Chaussures Plats Décontractées En Suède Mocass...	1.0
55683	1076208	Chaussures Plates Souples En Suède Avec Couleu...	1.0
55715	1214000	Chaussures Plateforme Respirantes en Daim à En...	1.0

Top 10 Products Predicted by Logistic Regression

Model	Category	No. of Products Classified as "Good"	Total Items	Ratio
Random Forest Classifier	shoes	1097	11823	0.092785
	women	1603	14809	0.108245
	house	456	12791	0.035650

	men	348	10208	0.034091
	accessories	128	6358	0.020132
	bags	326	6268	0.052010
	jewellery	269	4853	0.055430
Logistic Regression	shoes	857	11823	0.072486
	women	1080	14809	0.072929
	house	457	12791	0.035728
	men	385	10208	0.037716
	accessories	149	6358	0.023435
	bags	274	6268	0.043714
	jewellery	289	4853	0.059551
	kids	92	4085	0.022521
	beauty	149	3804	0.039169

Top Product Ratio per Category

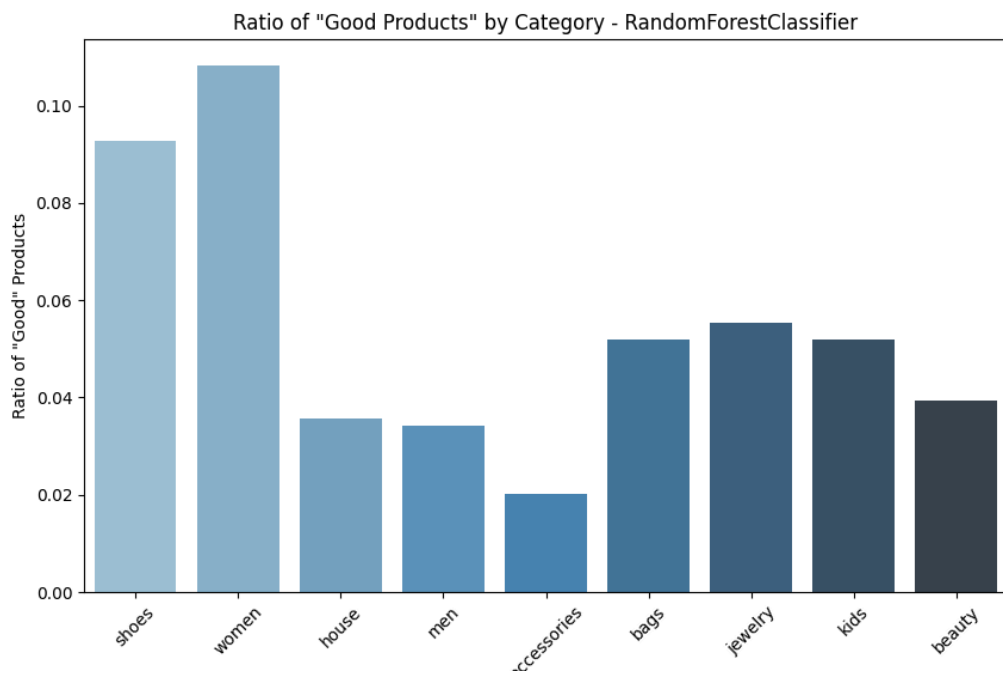


Figure 28: Boxplot of the ratio of datapoints classified as “good” versus total data points per category for Random Forest Classifier

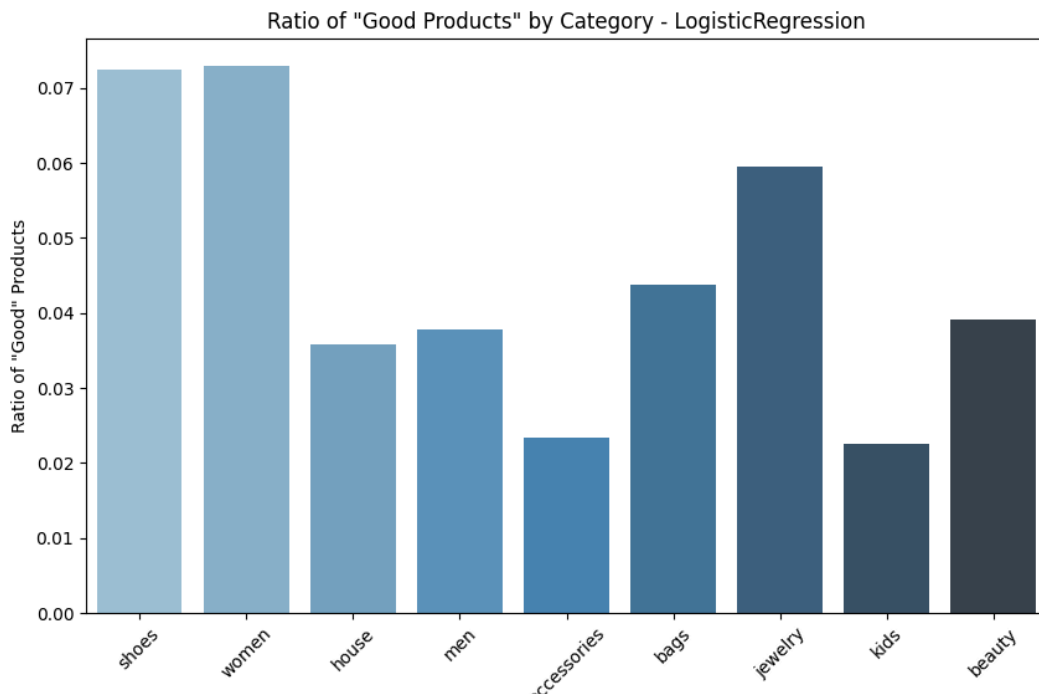


Figure 29: Boxplot of the ratio of datapoints classified as “good” versus total data points per category for Logistic Regression Classifier

Task 4: Result Discussion

KMeans vs. GMM and DBSCAN

In Experiment 1, KMeans exhibited a lower Silhouette Score compared to GMM and DBSCAN, indicating that KMeans produced less well-defined clusters, likely due to the non-spherical structure of the data. None of the models produced well-separated clusters, which may be attributed to the choice of model or the intrinsic characteristics of the dataset. GMM identified a small cluster that likely represents insignificant products or those with low customer attention. In experiment 2, DBSCAN after outlier removal identified larger clusters; however, even these were not well-separated, suggesting that while DBSCAN can capture distinct groupings similar to product categories, it may still struggle with overlapping clusters.

Clustering Model Limitations and Insights

These clusters might be meaningful in certain contexts but require careful interpretation. Advantageously, clustering like DBSCAN somehow could reflect distinct groupings similar to grouping products in categories. On the other hand, for KMeans, the presence

of outliers and potential biases, such as overrepresentation or underrepresentation of certain categories, could affect the model performance which means that data pre-processing could tackle those. Additionally, all clustering models show limitations in this analysis, particularly for KMeans in dealing with non-spherical clusters and exhibiting sensitivity towards those products that behave as outliers. On the evaluation, Silhouette score suggested that GMM and DBSCAN can provide more accuracy in terms of matching similar data points together. However, further analysis with other clustering algorithms to address the issues with small, overlapped clusters could improve cluster separation.

Cluster Interpretations from Experiment 1

Based on the information from Table 1 and the plot in Figure 14, the KMeans and GMM clusters can possibly provide those as follow:

In the yellow cluster, many data points exhibited higher PC2 values, likely corresponding to products with high pricing. This cluster spanned a wide range of PC1 values, from negative to positive, indicating products with varying levels of customer engagement from low to high, coupled with varied discounts. These products are potentially premium items that are popular among customers. In the meantime, the green cluster predominantly consists of data points with higher positive PC1 and lower PC2 values, suggesting products with moderate to high customer engagement and discounts but lower pricing. These are likely to be mid-range or domestic products. In the purple cluster, data points were concentrated around low to moderately negative PC1 values, with low PC2 values as well. This cluster likely represented products with lower pricing and average to low customer engagement and discounts. These could be budget items or less significant products, possibly not worth considering among the top 10 products.

Contrasting with Figure 14, Figure 20 shows that the GMM's orange cluster dominates most of the data points, especially in regions with varying PC1 and PC2 values, indicating a wide range of discounts and customer engagement levels. Some of the top 10 products are likely to be found in this cluster. The dark blue cluster in the GMM plot, with its narrow range of PC1 and low PC2 values, likely represented products with low pricing and limited customer engagement, such as less essential items like bags or accessories.

Recommendations on NewChic Dataset for Future Clustering

To enhance clustering results with the NewChic dataset, consider incorporating additional features, such as time series data that track pricing changes and discount

durations. These features could reveal more complex relationships between the products. The choice of evaluation metrics, such as the Silhouette Score and BIC, remains crucial in identifying the optimal number of clusters and preventing overfitting. Moreover, ensuring that the number of trials in hyperparameter tuning with Optuna is adequate is important; insufficient trials may lead to suboptimal parameter selection, while too many can increase computational costs without significant performance gains.

Random Forest Classifier vs. Logistic Regression

As can be seen from the classification results above, both classification models managed to achieve a high degree of accuracy on the test data. However the logistic regression model only achieved a recall of 0.6133 and an f1 score of 0.6954 on the test data, meaning it frequently does not correctly classify class 1 ("good" product). One of the two classes has a higher priority yet a lower density, so recall and f1 score are more appropriate than accuracy. Given that we are mainly interested in classifying which products are "good", the logistic regression model can therefore be seen as performing worse than the random forest classifier for our use case. The reason it managed to achieve such a high accuracy seems to mainly be caused by the data containing so many more data points of class 0 compared to class 1. Simply predicting all data points as class 0 would have resulted in a good accuracy.

On the other hand, the random forest classifier excelled, overcoming the challenge of imbalance in class distribution. The model is known for its robustness against complexities of the dataset, including non-linear relationships.

In terms of classification, the classifiers managed to do a good job of separating the products into separate classes. Furthermore, both classes have a lot of data points in them. However there are significantly more data points in class 0 as more data points have this as their target value in the training set. As there are only 2 classes, one representing good products and one representing bad products, there are no non-meaningful classes.

The random forest classifier is generally a suitable algorithm for Big Data analytics. One of the main advantages is that the algorithm is parallelizable, as each tree in the forest can be built independently, meaning that the workload can be distributed over multiple processors. Furthermore the model is relatively resistant to overfitting, as the model is designed to have a low variance. This is especially important in Big Data analytics, where overfitting could be a concern due to the sheer volume and amount of features in the data. Furthermore the model is still complex enough to capture non-linear relationships in the data. This is also seen in the classification results above, where the random forest performs better in the test set than logistic regression. However, keep in

mind that for an extremely large dataset a random forest classifier can have high computational costs, such as significant memory usage and long training times, which can be challenging even with parallelization.

Bibliography

Satish Chander P. Vijaya. "Unsupervised Learning Methods for Data Clustering." In Artificial Intelligence in Data Mining, edited by D. Binu and B.R. Rajakumar, 41-64. Academic Press, 2021. ISBN 9780128206010.

<https://doi.org/10.1016/B978-0-12-820601-0.00002-1>.

Khare, A. (2019, September 12). *8 clustering algorithms in machine learning that all data scientists should know*. freeCodeCamp.

<https://www.freecodecamp.org/news/8-clustering-algorithms-in-machine-learning-that-all-data-scientists-should-know/>

Scikit-learn developers. (n.d.). *Clustering performance evaluation*. Scikit-learn.

Retrieved August 21, 2024, from

<https://scikit-learn.org/stable/modules/clustering.html#clustering-performance-evaluation>

Roelofs, R. (2020, May 12). *State-of-the-art machine learning hyperparameter optimization with Optuna*. Towards Data Science.

<https://towardsdatascience.com/state-of-the-art-machine-learning-hyperparameter-optimization-with-optuna-a315d8564de1>

HBC Training. (n.d.). Principal component analysis. HBC Training. Retrieved August 21, 2024, from

https://hbctraining.github.io/DGE_workshop/lessons/principal_component_analysis.html

Deepchecks. (n.d.). Gaussian mixture model. Deepchecks. Retrieved August 21, 2024, from <https://deepchecks.com/glossary/gaussian-mixture-model/>

Ester, M., Kriegel, H.-P., Sander, J., & Xu, X. (1996). A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning with Applications in R*. Springer.

