Predicting Customer Preferences and Sales for Cosmetic Products using Deep Learning

Sneh Pahuja

Descriptive Statistics and Feature Overview

The dataset consists of 18 features and a binary target variable, *Sales_Prediction*, which indicates whether the customer will purchase the product (1) or not (0). There are 7 continuous features: *Price*, *Rating*, *Stock_Level*, *Age*, *Likes*, *Shares*, and *Comments*. The binary categorical variables include *Ingredients*, *Gender*, *Campaign*, and *Seasonal_Factor*. Additionally, there are 5 categorical variables such as *Brand*, *Location*, *Income_Level*, *Product_Type*, and *Skin_Type*. The dataset also includes 2 ID variables: *Customer_ID* and *Product_ID*.

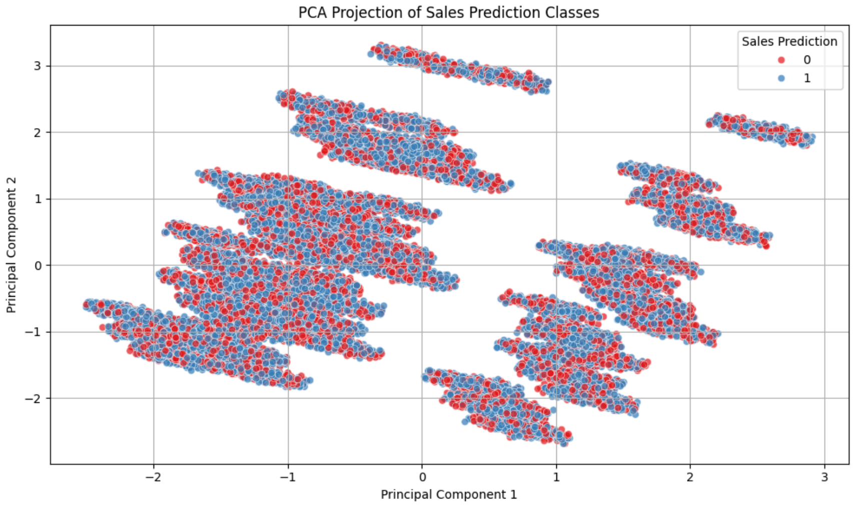
Price has a mean of 76.75 and a standard deviation of 40.31, indicating moderate variability in product pricing. *Stock_Level* has a mean of 107.65 with a standard deviation of 55.14, reflecting wide variation in stock levels, however this feature was removed as the model is focused on customer behaviours and purchases and has no logical association. Engagement features like *Likes*, *Shares*, and *Comments* have standard deviations of 28.97, 14.37, and 8.67, respectively, suggesting consistency in shares and comments, though *Likes* show more variability. *Age* has a relatively large standard deviation (15.03), implying a broad age distribution among customers. The *Rating* feature has a mean of 3.03 and a standard deviation of 1.17, indicating ratings tend to center around the middle value. The categorical variables show balanced frequencies, except for *Campaign* (60-40 split) and *Seasonal_Factor* (67-33 split). Brand 3 is the most commonly occurring brand in the dataset suggesting that it may be the most widely stocked, however it does not necessarily mean its the most popular brand.

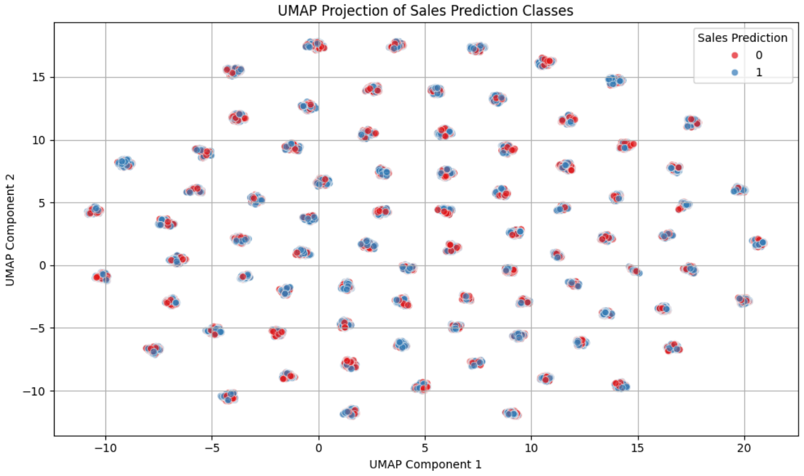
Data Visualization and Exploration

In the exploration phase, I performed a correlation analysis of continuous variables, finding no significant relationships (values ranging from -0.077 to 0.069). I performed a Point-Biserial correlation to assess the relationship between continuous variables and the binary target variable. Most continuous features (*Price*, *Rating*, *Stock_Level*, *Age*, *Likes*, *Shares*) had weak or negligible correlations with *Sales_Prediction*, with high p-values indicating no statistical significance. The only exception was *Comments*, which showed a very weak but statistically significant negative correlation.

I then used Cramér's V assess categorical variables' association with *Sales_Prediction*, and the values were very low, suggesting weak associations. No categorical feature demonstrated a strong relationship with the target variable, indicating these features may not significantly contribute to distinguishing between purchasers and non-purchasers. Further analysis of class separation showed minimal differences in the average feature values between the two classes, which means there is no clear separability.

I also applied PCA and UMAP for dimensionality reduction and visualization but found no clear class separation in the 2D space. The Silhouette Score (~0.0001) confirmed almost no cluster separation between purchasers and non-purchasers. The highest Mutual Information value (~0.0068 for *Campaign*) was very low, which suggests the features offer minimal predictive information for distinguishing between the classes.





Data Preprocessing and Feature Engineering

For data preprocessing, I removed duplicates and handled missing values by filling categorical columns with the mode and numerical columns with zeros. To enhance model performance, I engineered several new features, including *Mean_Sale_Customer*, *Mean_Sale_Product*, *Mean_Sale_Type*, and *Mean_Sale_Brand*, which capture average purchase behavior at the customer, product, type, and brand levels. Additional features like *Total_Spent*, *Avg_Spent*, and *Num_Transactions* capture aggregate customer spending behavior, while *Campaign_Response_Rate* reflects the effectiveness of campaigns in eliciting responses from different customers.

Further, I created *Product_Freq*, *Product_Avg_Spent*, and *Product_Success to* measure product popularity, spending patterns, and success in actual purchases. I also binned the *Age* feature into categories (e.g., '18-24', '25-34') to reduce sensitivity to small age variations. The dataset was split first into training, validation, and test sets to avoid data leakage. Categorical features, such as *Location*, *Income_Level*, and *Age_Group*, were one-hot encoded, while numerical features were standardized using *StandardScaler* to ensure they were on the same scale for distance-based models like KMeans.

Model Development and Hyperparameter Tuning

I explored multiple modeling techniques, including creating a manually tuned Artificial Neural Network (ANN) and machine learning models such as Logistic Regression and Random Forest. The ANN model had an architecture with 4 hidden layers (100, 64, 48, and 16 units), ReLU activation, dropout (40%), and L1 regularization (0.01) to prevent overfitting. The Adam optimizer was used with a learning rate of 0.00005, and binary cross-entropy loss was applied for binary classification. However, the model still showed a slight overfitting tendency with an accuracy of 51%, indicating limited generalization. Precision for Class 1 (purchasers) was 61%, but recall for Class 0 (non-purchasers) was low at 44%, suggesting the model struggles to identify non-purchasers. However in the context of the business, higher precision for class 1 may be ideal to prevent lost sales by not targeting those customers who may actually make purchases.

To check whether any other hyperparameter tuning could be done, I used Optuna. However, the model performance did not significantly improve, so I stuck with the manual ANN. Other models, like Logistic Regression, achieved 50.96% accuracy, with moderate precision (60.66%) and recall (54.01%). The AUC-ROC score was 0.5022, which is close to random, indicating limited discriminatory power. The Random Forest model achieved a higher accuracy of 52.89%, with stronger recall for Class 1 (63.44%) but weaker recall for Class 0 (37%).

I tried stacking the Random Forest with the ANN but that too could not distinguish well between the classes.

I also applied KMeans clustering to segment customers into 15 clusters based on their purchasing behavior only by creating a separate customer attributes table and then mapping the cluster id back to the transactions table.

The model achieved an accuracy of 50.5%, which is moderate and close to random guessing. Precision for Class 1 was 60.48%, while recall was 51.39%, with an F1-Score of 55.57%. The error metrics, such as Mean Absolute Error (0.4927) and Root Mean Squared Error (0.5765), indicate moderate prediction errors. The AUC-ROC score of 0.5057 further confirms the model's limited ability to distinguish between the two classes. The confusion matrix showed significant misclassifications with many false positives and false negatives.

Though the model did not perform better than the others logically, I feel that this is the best constructed as it tries to find specific patterns amongst different groups of users and rather than treating them all the same.

Conclusion

The current models show poor performance, with accuracy and precision close to random guessing, particularly for Class 0 (non-purchasers). Despite various preprocessing steps, feature engineering, and exploration of different models (ANN, Logistic Regression, Random Forest), the ability to predict customer purchases remains limited. Future improvements could involve more sophisticated feature engineering, adding more variables that better help in class separation, sequential or time-series data, better handling of class imbalance, and exploring more advanced models or techniques.

In addition to classification, a basic collaborative filtering system was implemented using Singular Value Decomposition (SVD). The goal was to build a product recommendation system:

- A user-item interaction matrix was created from Sales_Prediction values.
- SVD (TruncatedSVD with 20 latent components) was applied to reduce dimensionality.
- The resulting latent vectors were used to estimate affinity scores between customers and products.

I built a simple recommendation function to generate top-N product suggestions for any given customer. For instance, for Customer_ID = 5, the system identified the top 5 products with the highest predicted interest. While exploratory with no labels to test with, this model illustrates potential for adding recommendation functionality for more targeted marketing.