BDM 3035
Big Data Capstone Project

Elevating SEPHORA's Product Experience 😊 with an Innovative Recommender System.✨

Submitted By:

Dipti Baral
Koshish Aryal
Punam Bhattarai
Shweta Laljibhai Thummar
Uttam Tamang

Submitted To:

Meysam Effati

Contents

Introduction

This project plans to perform a deep analysis on Sephora's skincare products - their reviews as well as to unravel some of the key insights carrying how the products are performing and how do the customers prefer it. However, by using a rich dataset that contains extensive details for 8000+ beauty products and around a million user reviews provided by beautia, we aim to provide insights into how beauty products are priced, brand popularity, and execution of ingredient usage. As always, we will start with deep-dive data preparation, some exploratory data analysis (EDA) to pinpoint trends and patterns in product categories, prices, and how brands are doing. In addition, we will also create a custom recommender system to recommend products according to customer purchase history and review trends. Finally, we will generate exciting and interactive data visualizations to help us present our findings in an accessible and compelling way. This is an extensive report intended to offer significant information on.

In order to do so, we use Singular Value Decomposition (SVD) which is a matrix factorization technique used in collaborative filtering for recommendation systems. It decomposes a matrix into three other matrices, which can be used to approximate the original matrix. This is particularly useful for dimensionality reduction and capturing latent factors in the data.

Data Collection and Preparation

After finalizing the topic of our project, the first and most critical task was data preparation. This step is crucial because it involves converting raw data into a clean, structured, and usable format, setting the foundation for accurate and insightful analysis. The process encompasses several key activities, each essential to ensure that the data is correct, comprehensive, and ready for exploratory data analysis (EDA).

Key Steps in Our Data Preparation Process:

1. Data Collection

We sourced our dataset from Kaggle, ensuring access to a diverse and extensive data pool relevant to our project. This stage involved selecting the appropriate dataset that aligns with our research objectives.

2. Data Cleaning

In this step, we focused on improving data quality by addressing missing values, correcting errors, handling outliers, and standardizing formats. This process was vital to ensure that the data is accurate and reliable, thus laying the groundwork for subsequent analysis.

3. Data Transformation

To enhance the usability of the data, we performed several transformation tasks. This included creating new features, scaling and normalizing data, encoding categorical variables, and reducing dimensionality. These transformations made the data more suitable for analysis, particularly for machine learning models.

4. Data Integration and Aggregation

We combined data from multiple sources, integrating them into a unified dataset. Additionally, we summarized the data at various levels of granularity, providing a more comprehensive view and enabling deeper insights during analysis.

5. Data Sampling

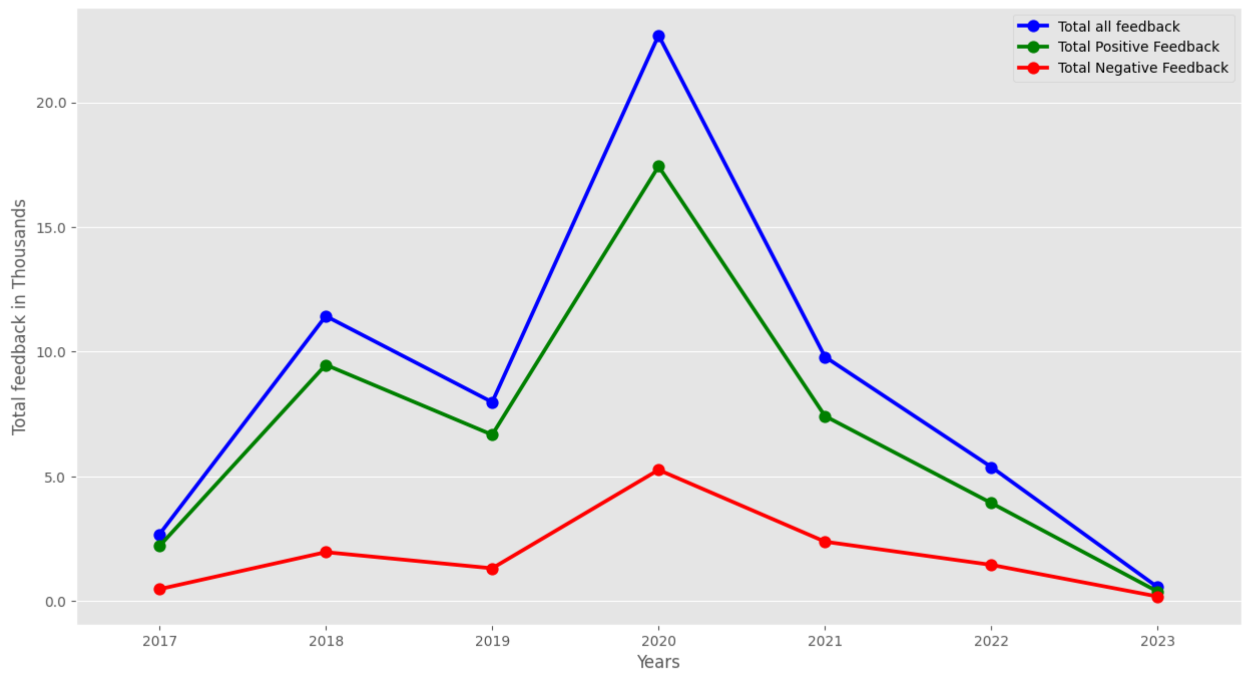
To streamline our analysis, we selected representative subsets of the data. This sampling ensured that the data used for analysis was manageable in size while still being reflective of the entire dataset.

6. Data Quality Assessment

The final step involved verifying the reliability of our data. We conducted thorough checks for consistency, completeness, and validity, ensuring that the dataset was of high quality and ready for accurate analysis.

This meticulous preparation was essential for conducting a robust and insightful exploratory data analysis, ultimately contributing to the success of our project.

Some Visualization:



The graph shows the trend of total feedback (including all feedback, positive feedback, and negative feedback) over the years from 2017 to 2023. Here are the key insights:

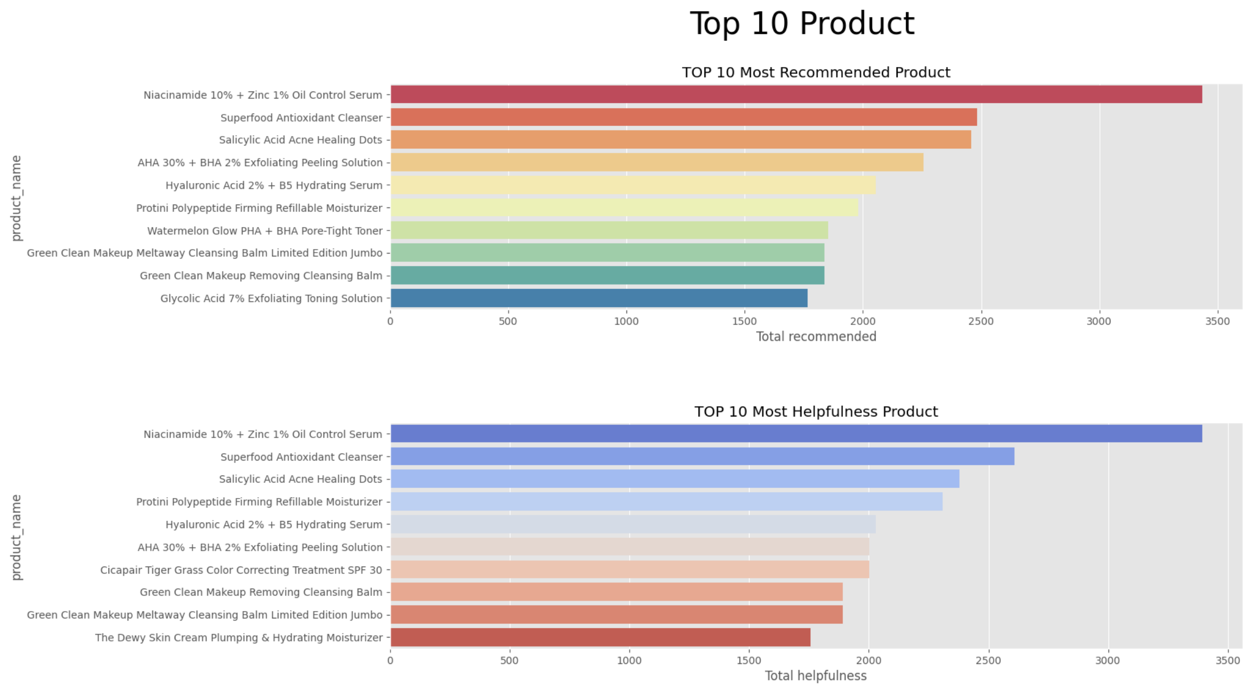
1. Peak in 2020: The total feedback, along with positive and negative feedback, peaked in 2020. This suggests that there was a significant increase in engagement or product reviews during that year.

2. Decline Post-2020: After 2020, there is a noticeable decline in all types of feedback, with the sharpest drop occurring between 2020 and 2021. This decline continues through 2023, indicating reduced engagement or fewer reviews in recent years.

3. Positive vs. Negative Feedback: Throughout the years, positive feedback is consistently higher than negative feedback. However, both types of feedback follow a similar trend, with peaks in 2020 and subsequent declines.

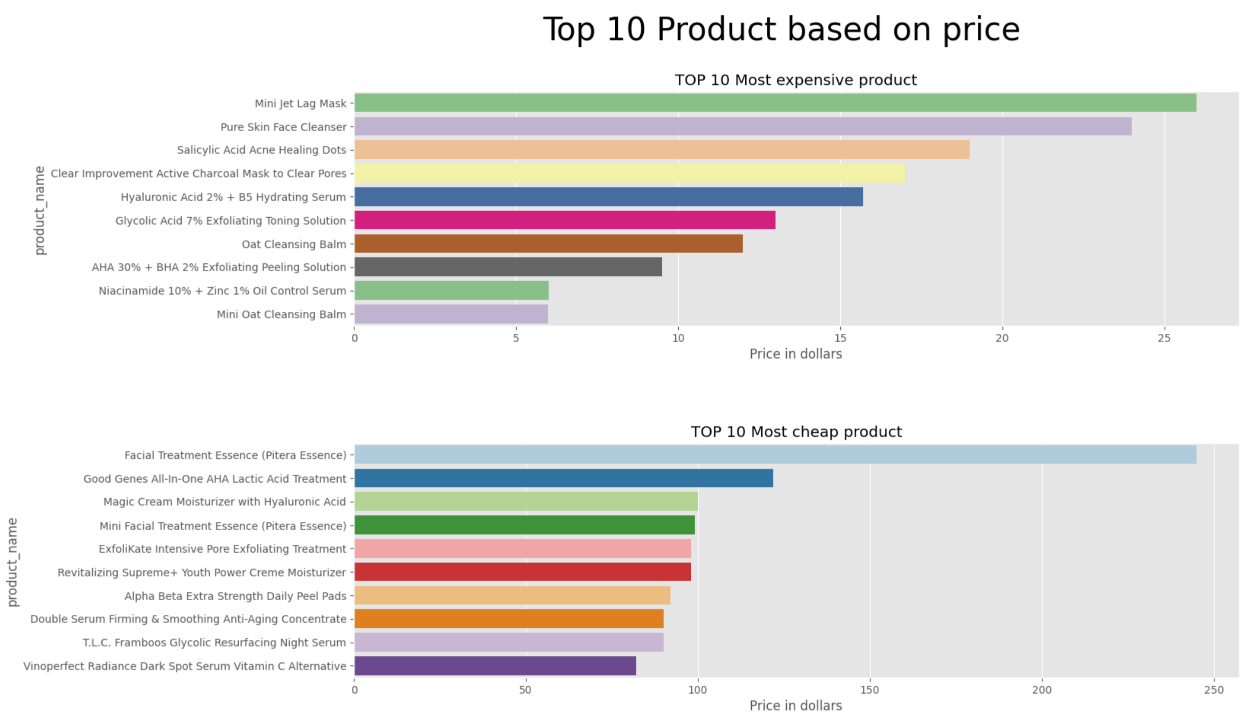
4. Low Negative Feedback: Negative feedback remains relatively low across all years compared to positive feedback, indicating a generally favorable perception of the products being reviewed.

This trend could be indicative of external factors affecting customer engagement, such as changes in product popularity, market conditions, or broader global events like the COVID-19 pandemic, which likely influenced the peak in 2020.

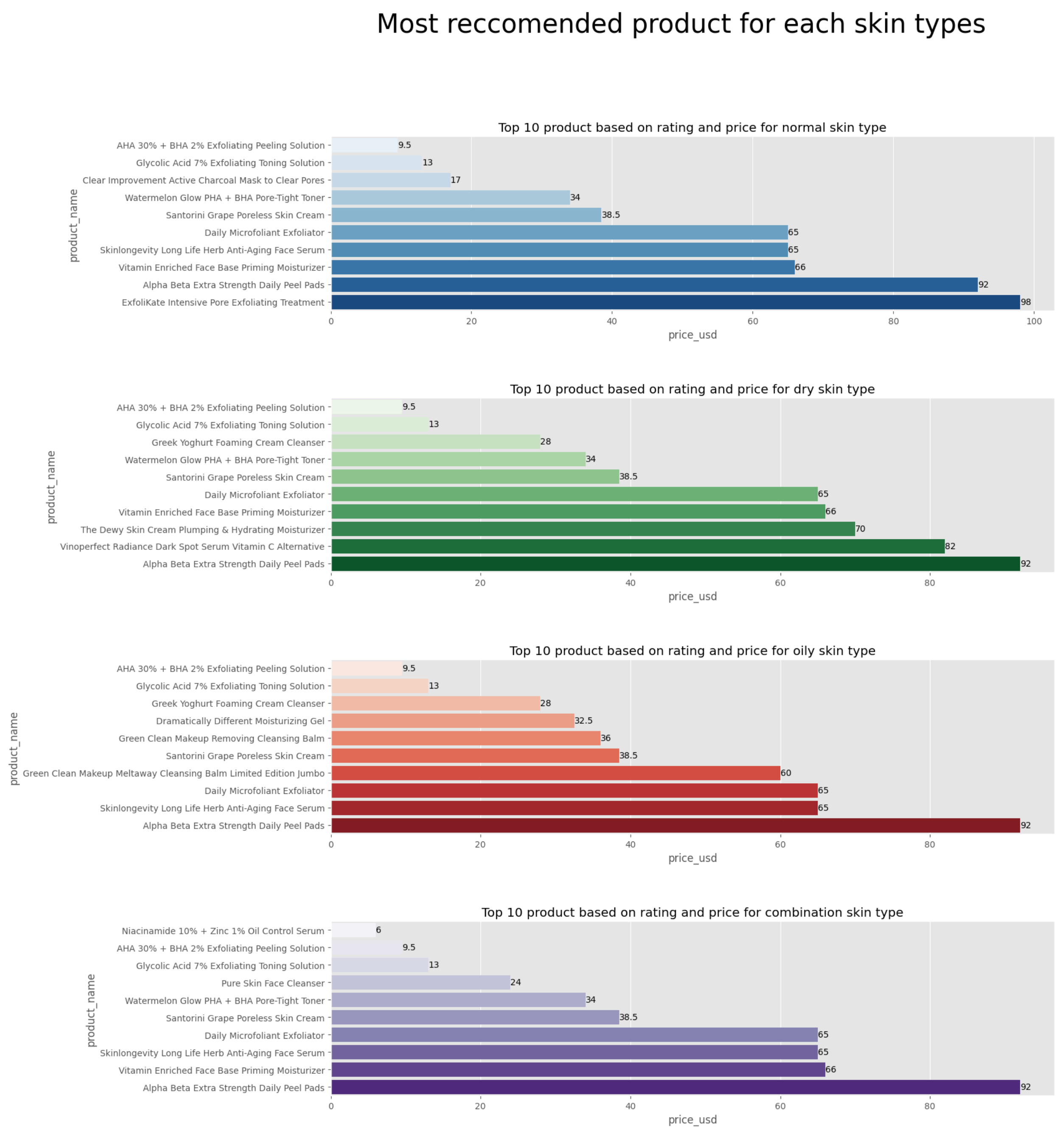


The graph presents the top 10 products based on two criteria: the most recommended products and the most helpfulness products.

Overall, the products that are most recommended also tend to be rated as the most helpful, suggesting a strong correlation between recommendation frequency and perceived usefulness.

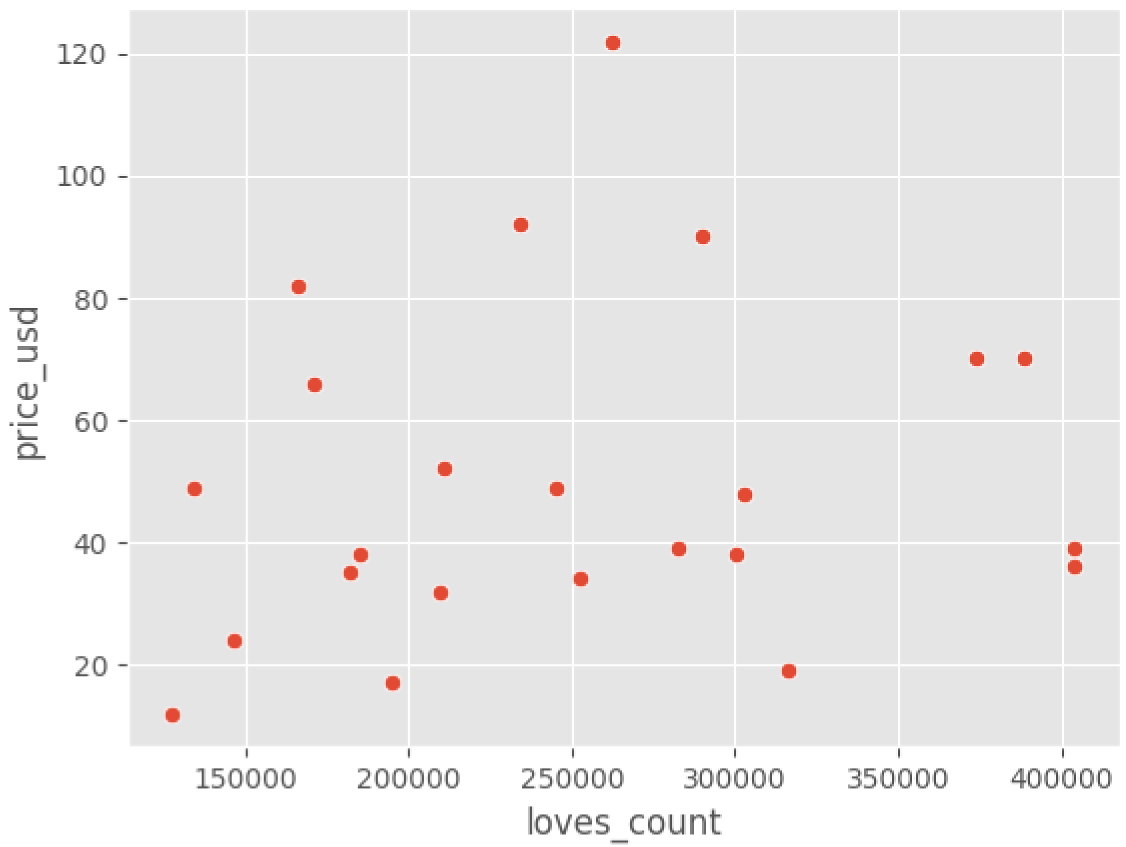


The graph shows the top 10 most expensive and cheapest products based on their price. The top 10 most expensive products are led by the "Mini Jet Lag Mask" and "Pure Skin Face Cleanser," while the "Facial Treatment Essence" stands out as the most affordable product, significantly cheaper than the others.



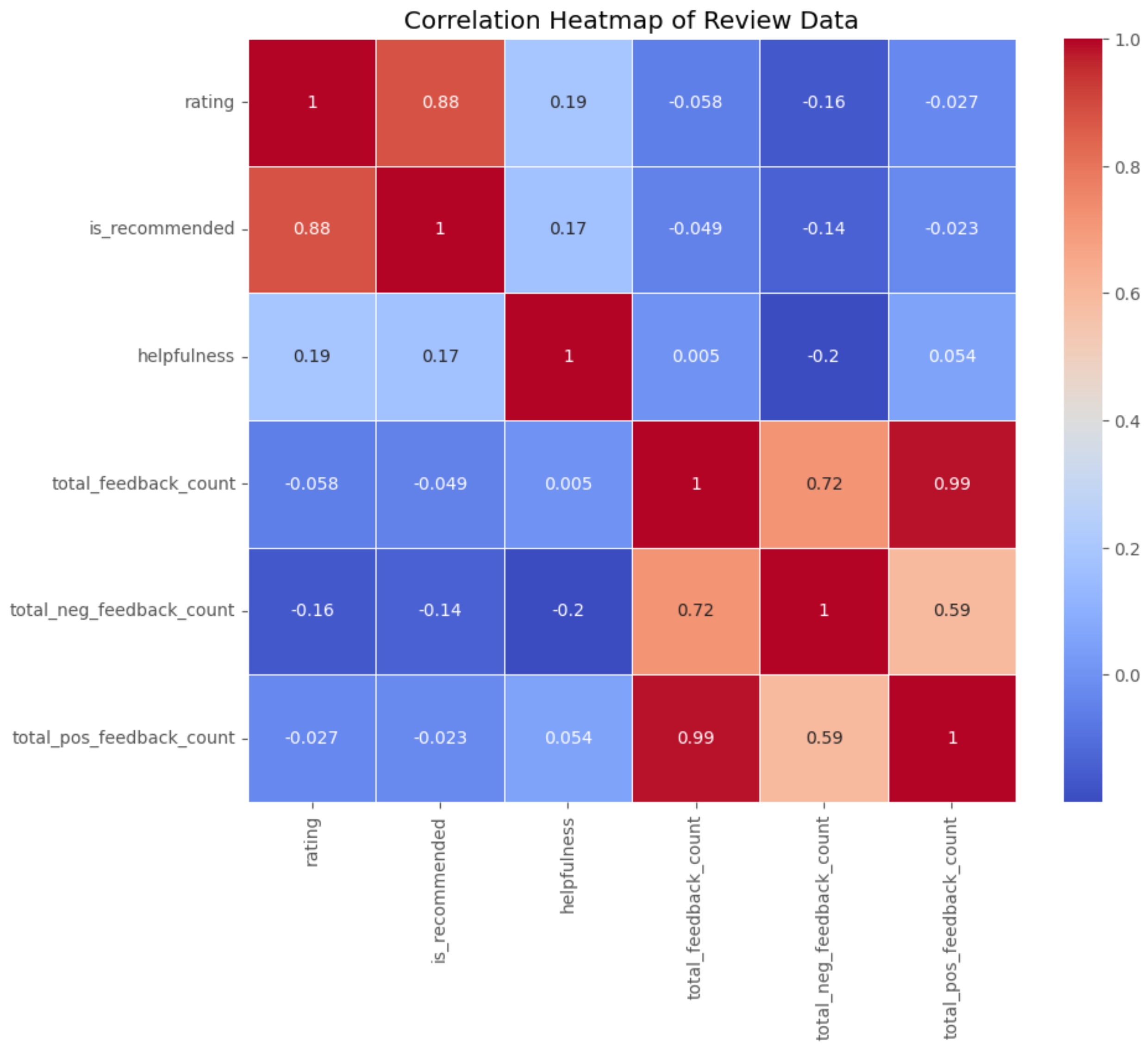
The graph displays the most recommended products for different skin types (normal, dry, oily, and combination) based on rating and price.

The "AHA 30% + BHA 2% Exfoliating Peeling Solution" consistently ranks as a top-rated and affordable product across multiple skin types. For combination skin, the "Niacinamide 10% + Zinc 1% Oil Control Serum" is particularly recommended. Across all skin types, a range of both budget-friendly and premium products are well-regarded.



The scatter plot displays the relationship between the number of "loves" a product has received (`loves_count`) and its price (`price_usd`).

There is no clear, strong correlation between the number of loves a product receives and its price. Products with a wide range of prices, from low to high, have received a similar number of loves, suggesting that both affordable and expensive products can be equally loved by customers.



- Rating & Recommendation: Strong positive correlation (0.88).

- Feedback Counts: `total_feedback_count` highly correlates with `total_pos_feedback_count` (0.99); moderate with `total_neg_feedback_count` (0.72).

- Helpfulness: Low correlation with other features, suggesting it's mostly independent.

Challenges:

We faced the following obstacles during our Exploratory Data Analysis (EDA) process, each of which called for thoughtful consideration and calculated solutions:

Data quality issues: The dataset required significant cleaning and validation due to mistakes, missing values, and inconsistencies.

Handling Outliers: To prevent skewed results and guarantee robust analysis, it was essential to recognize and handle outliers in a proper manner.

❖ High Dimensionality: In order to manage and analyze a large number of features without losing important information, dimensionality reduction approaches were needed.

❖ Imbalanced Data: To obtain more trustworthy findings, we had to use strategies to balance the dataset in situations where some classes were underrepresented.

❖ Complex Data Structures: Specialized preparation and analysis techniques were required when working with complex and unstructured data, such as text or time-series data.

❖ Computing Constraints: Time efficiency and computing resources were challenged by large datasets and intricate computations.

❖ Data Integration: It was difficult and time-consuming to combine data from several sources with various forms and structures.

Methodology

Singular Value Decomposition (SVD) is a matrix factorization technique used in collaborative filtering for recommendation systems. It decomposes a matrix into three other matrices, which can be used to approximate the original matrix. This is particularly useful for dimensionality reduction and capturing latent factors in the data.

Explanation of SVD

Given a matrix (A) of dimensions (m \times n), SVD decomposes (A) into three matrices: [A = U \Sigma V^T]

- **(U)**: An (m \times k) orthogonal matrix, where (k) is the number of latent factors.
- **(\Sigma)**: A (k \times k) diagonal matrix with singular values.
- **(V^T)**: A (k \times n) orthogonal matrix.

In the Context of Collaborative Filtering

- **Interaction Matrix**: The original user-item matrix where rows represent users and columns represent items.
- **User Matrix ((U))**: Represents users in the latent factor space.
- **Product Matrix ((V))**: Represents items in the latent factor space.

Why SVD?

The combination of one-hot encoding, aggregation, and creating a user-item matrix sets up the data perfectly for applying SVD. This approach captures both user preferences and item characteristics, making it a robust choice for building a recommendation system.

- **Dimensionality Reduction**: SVD reduces the dimensionality of the user-item matrix, capturing the most important latent factors that explain user preferences and item characteristics.
- **Latent Factors**: By decomposing the matrix into latent factors, SVD can uncover hidden patterns in the data, such as user preferences and product similarities.
- **Scalability**: SVD is computationally efficient and can handle large datasets, making it suitable for real-world recommendation systems.

Model Training

Collaborative Filtering with SVD:

Input: The user-item interaction matrix (user-item matrix) is used as input. This matrix contains user interactions (e.g., ratings, clicks) with different products.

SVD Decomposition: The matrix is decomposed using TruncatedSVD into a user matrix and a product matrix. The n_components parameter controls the number of latent factors.

Output: The trained model consists of the user_matrix and product_matrix, representing users and products in a latent factor space.

Validation: Techniques like cross-validation or splitting the data into training and test sets can be used to validate the performance of the SVD model. Evaluation metrics like Root Mean Squared Error (RMSE) can be used to measure prediction accuracy.

Evaluation: The model can be evaluated on its ability to predict user ratings for unseen items. Metrics such as precision, recall, and F1-score could be used, especially in a top-N recommendation context

Content-Based Filtering with TF-IDF and Cosine Similarity:

Input: Product descriptions are aggregated and vectorized using the TF-IDF technique.

Cosine Similarity Calculation: The similarity between all pairs of products is computed using cosine similarity.

Output: A similarity matrix (cosine_sim) that stores the pairwise similarities between products.

Validation: The similarity scores can be validated by checking if similar products (based on content) are indeed those that users find similar.

Evaluation: The performance can be measured by how well the model recommends products that match the user's interests based on product descriptions.

Hybrid Approach:

Combining Results: The hybrid model combines recommendations from both collaborative filtering and content-based filtering. The final list is evaluated on its relevance and diversity.

Validation: A/B testing or user studies can be conducted to assess the effectiveness of the hybrid model in providing more accurate and satisfying recommendations.

Explanation of Parameter Tuning or Optimization Techniques Applied

TruncatedSVD:

n_components: This parameter determines the number of latent factors to retain. It directly affects the balance between model complexity and performance.

Optimization: Grid search or cross-validation can be used to find the optimal number of components that minimize the error on a validation set.

TF-IDF Vectorizer:

stop_words: 'english': Common stop words (e.g., "the," "is," "in") are removed to reduce noise in the textual data.

max_features: Optionally, the maximum number of features (words) to consider can be tuned to manage the dimensionality of the TF-IDF matrix.

Hybrid Approach:

Weighting the Results: The combination of collaborative filtering and content-based recommendations can be weighted. For example, a parameter could be introduced to control the contribution of each method to the final recommendation. This parameter could be optimized based on validation performance.

Top-N Recommendations:

top_n: This parameter controls how many products are recommended. It can be adjusted depending on the use case, with evaluation metrics (e.g., precision@N) guiding the choice.

In summary, the code leverages SVD for collaborative filtering to uncover latent patterns in user-product interactions and combines this with content-based filtering, which uses TF-IDF and cosine similarity to recommend products based on textual content. The combination of these methods is designed to enhance recommendation accuracy and relevance by addressing different aspects of user and product data.

Deep Learning Model for rating prediction

Algorithm

The model is a deep neural network created using Keras, wrapped with KerasRegressor for compatibility with scikit-learn. It includes:

- **Dense Layers**: Fully connected layers with 128 and 64 neurons.
- **Dropout Layers**: Dropout regularization to prevent overfitting.
- **Activation Functions**: ReLU and softmax activation functions.
- **Optimizer**: Adam and RMSprop optimizers.

Techniques:

- **Dropout**: Applied to reduce overfitting by randomly setting a fraction of input units to 0 at each update during training.
- **RandomizedSearchCV**: Used for hyperparameter tuning to find the best combination of hyperparameters.

Why this Algorithm?

- **Deep Neural Network**: Chosen for its ability to model complex relationships in data, making it suitable for regression tasks.
- **Dropout**: Helps in preventing overfitting, which is crucial for improving the model's generalization to unseen data.
- **Adam and RMSprop Optimizers**: Both are adaptive learning rate methods that have shown good performance in training deep neural networks.
- **RandomizedSearchCV**: Efficiently searches a wide range of hyperparameters to find the best model configuration, saving time compared to a full grid search.

Training, Validation, and Evaluation Procedures

- **Training**: The model is trained on the training dataset (X_train, y_train) using the specified hyperparameters. The training process involves:
- **Batch Size**: Different batch sizes (32, 64, 128) are tested.
- **Epochs**: The model is trained for a varying number of epochs (50, 100, 200).
- **Validation**: Cross-validation with 3 folds (cv=3) is used to validate the model during hyperparameter tuning. This helps in assessing the model's performance on different subsets of the data
- **Evaluation**: The best model is selected based on the performance metrics (e.g., mean absolute error) on the validation set. The final model can be evaluated on a separate test set to assess its generalization performance.

Hyperparameter Tuning

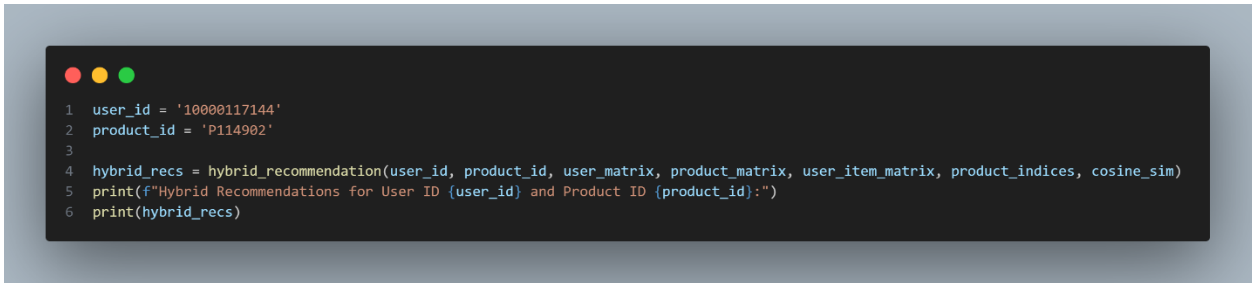
- **RandomizedSearchCV**: Used to search over a specified parameter grid with 10 iterations (n_iter=10). This includes tuning the batch size, number of epochs, optimizer type, activation function, and dropout rate.
- **Parameter Grid**:
- batch_size: [32, 64, 128]
- epochs: [50, 100, 200]
- model__optimizer: ['adam', 'rmsprop']

- model__activation: ['relu', 'softmax']
- model__dropout_rate: [0.3, 0.5, 0.7]

Optimization:

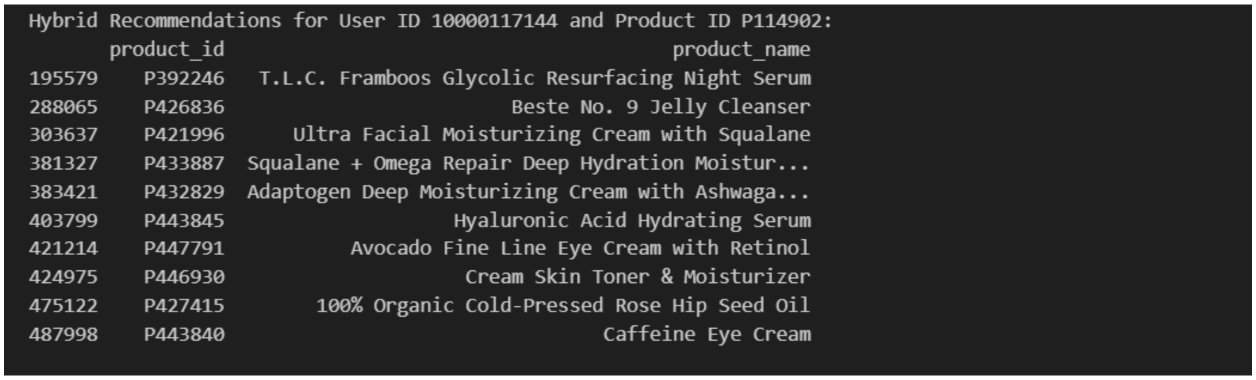
- **Adam and RMSprop**: Both optimizers are used to find the best learning rate and momentum parameters dynamically during training.
- **Dropout**: Different dropout rates (0.3, 0.5, 0.7) are tested to find the optimal rate for preventing overfitting.

Analysis and Results



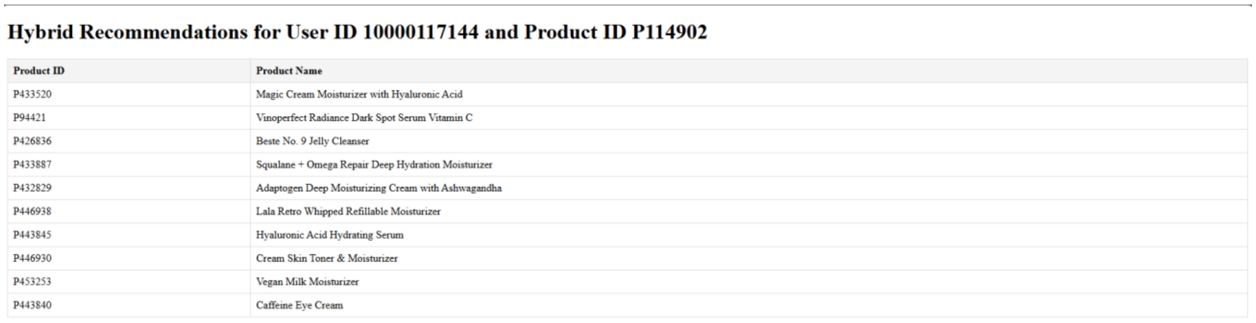
```python
user_id = '10000117144'
product_id = 'P114902'

hybrid_recs = hybrid_recommendation(user_id, product_id, user_matrix, product_matrix, user_item_matrix, product_indices, cosine_sim)
print(f"Hybrid Recommendations for User ID {user_id} and Product ID {product_id}:")
print(hybrid_recs)
```

Here's the sample of the code that represent the results for recommendation for the specific user with specific product.



```
Hybrid Recommendations for User ID 10000117144 and Product ID P114902:
       product_id                                      product_name
195579    P392246   T.L.C. Framboos Glycolic Resurfacing Night Serum
288065    P426836                        Beste No. 9 Jelly Cleanser
303637    P421996      Ultra Facial Moisturizing Cream with Squalane
381327    P433887  Squalane + Omega Repair Deep Hydration Moistur...
383421    P432829  Adaptogen Deep Moisturizing Cream with Ashwaga...
403799    P443845                      Hyaluronic Acid Hydrating Serum
421214    P447791            Avocado Fine Line Eye Cream with Retinol
424975    P446930                       Cream Skin Toner & Moisturizer
475122    P427415         100% Organic Cold-Pressed Rose Hip Seed Oil
487998    P443840                                  Caffeine Eye Cream
```

This show the recommended product for the use on the interface.

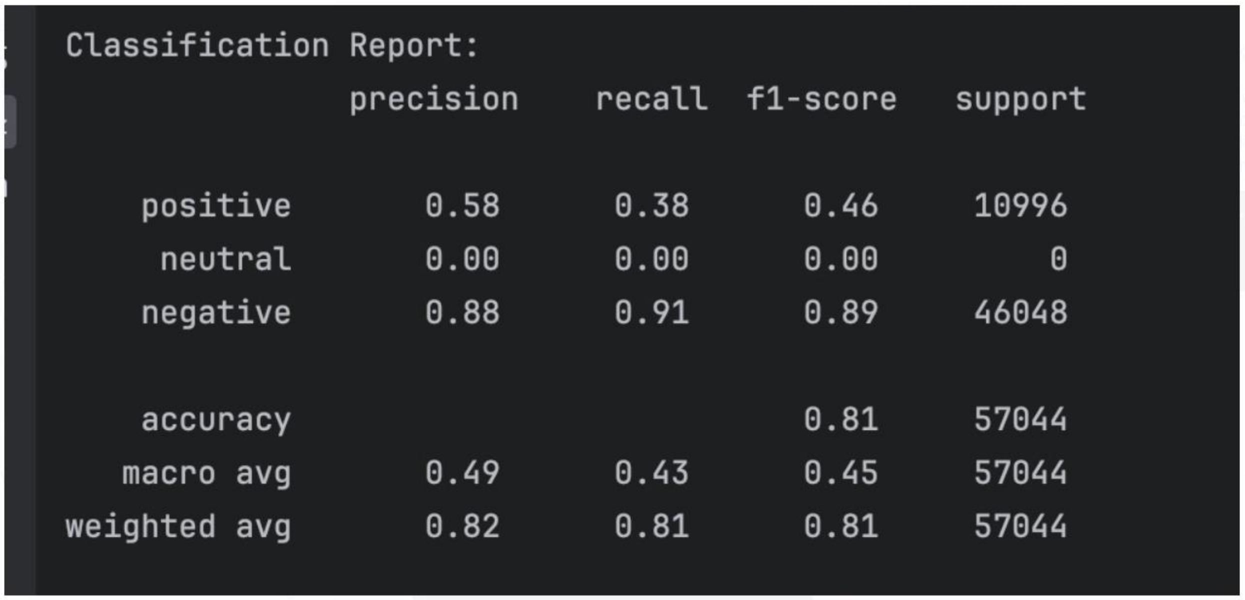


Hybrid Recommendations for User ID 10000117144 and Product ID P114902

Product ID	Product Name
P433520	Magic Cream Moisturizer with Hyaluronic Acid
P94421	Vinoperfect Radiance Dark Spot Serum Vitamin C
P426836	Beste No. 9 Jelly Cleanser
P433887	Squalane + Omega Repair Deep Hydration Moisturizer
P432829	Adaptogen Deep Moisturizing Cream with Ashwagandha
P446938	Lala Retro Whipped Refillable Moisturizer
P443845	Hyaluronic Acid Hydrating Serum
P446930	Cream Skin Toner & Moisturizer
P453253	Vegan Milk Moisturizer
P443840	Caffeine Eye Cream

This shows the recommended product for use on the web interface.

Performance evaluation



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

    positive       0.58      0.38      0.46     10996
     neutral       0.00      0.00      0.00         0
    negative       0.88      0.91      0.89     46048

    accuracy                           0.81     57044
   macro avg       0.49      0.43      0.45     57044
weighted avg       0.82      0.81      0.81     57044
```

The classification report you provided summarizes the performance of a classification model, likely an LLM (Large Language Model), on a dataset with three possible classes: "positive," "neutral," and "negative." The report provides various metrics such as precision, recall, and F1-score, along with the number of samples (support) for each class.

Precision: The ratio of correctly predicted positive observations to the total predicted positives. It answers the question: "Of all the positive predictions, how many were actually correct?"

Recall: The ratio of correctly predicted positive observations to all observations in the actual class. It answers the question: "Of all the actual positives, how many were predicted correctly?"

F1-Score: The weighted average of precision and recall. It considers both false positives and false negatives and is a better measure when you have an uneven class distribution.

Support: The number of actual occurrences of the class in the dataset.

Class: Positive

Precision: 0.58 means that 58% of the instances predicted as "positive" were actually "positive."

Recall: 0.38 indicates that the model correctly identified 38% of all actual "positive" instances.

F1-Score: 0.46 is the harmonic mean of precision and recall, showing a balance between the two metrics, though performance is not very high.

Support: 10,996 means there were 10,996 actual "positive" instances in the dataset.

Class: Neutral

Precision, Recall, F1-Score: 0.00 indicates that the model did not identify any "neutral" instances, leading to a complete failure in predicting this class. The support for this class is 0, which might indicate that there were no "neutral" instances in the test set, or the model simply failed to recognize them.

Support: 0 suggests no data points were classified or labeled as "neutral."

Class: Negative

Precision: 0.88 means that 88% of the instances predicted as "negative" were actually "negative."

Recall: 0.91 indicates that the model correctly identified 91% of all actual "negative" instances.

F1-Score: 0.89 is quite high, indicating that the model performs well on the "negative" class.

Support: 46,048 means there were 46,048 actual "negative" instances in the dataset.

Overall Metrics:

Accuracy: 0.81 indicates that 81% of the total predictions were correct.

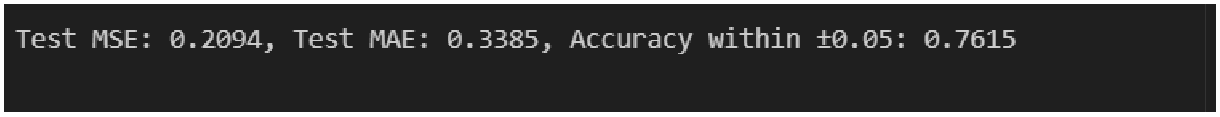
Macro Avg (Average):

Precision: 0.49, **Recall**: 0.43, **F1-Score**: 0.45 provide an unweighted average of the metrics for all classes, treating each class equally. The low scores here are mainly due to the complete failure in predicting the "neutral" class.

Weighted Avg:

Precision: 0.82, **Recall**: 0.81, **F1-Score**: 0.81 give an average weighted by the number of instances in each class. Since the "negative" class dominates the dataset, these averages are skewed towards the performance on the "negative" class.

For Rating prediction model:



```
Test MSE: 0.2094, Test MAE: 0.3385, Accuracy within ±0.05: 0.7615
```

Test MSE (Mean Squared Error): 0.2094

MSE measures the average squared difference between the actual and predicted values. A lower MSE indicates that the model's predictions are closer to the actual values. In this case, the MSE of 0.2094 suggests that on average, the square of the errors (differences between predicted and actual values) is 0.2094.

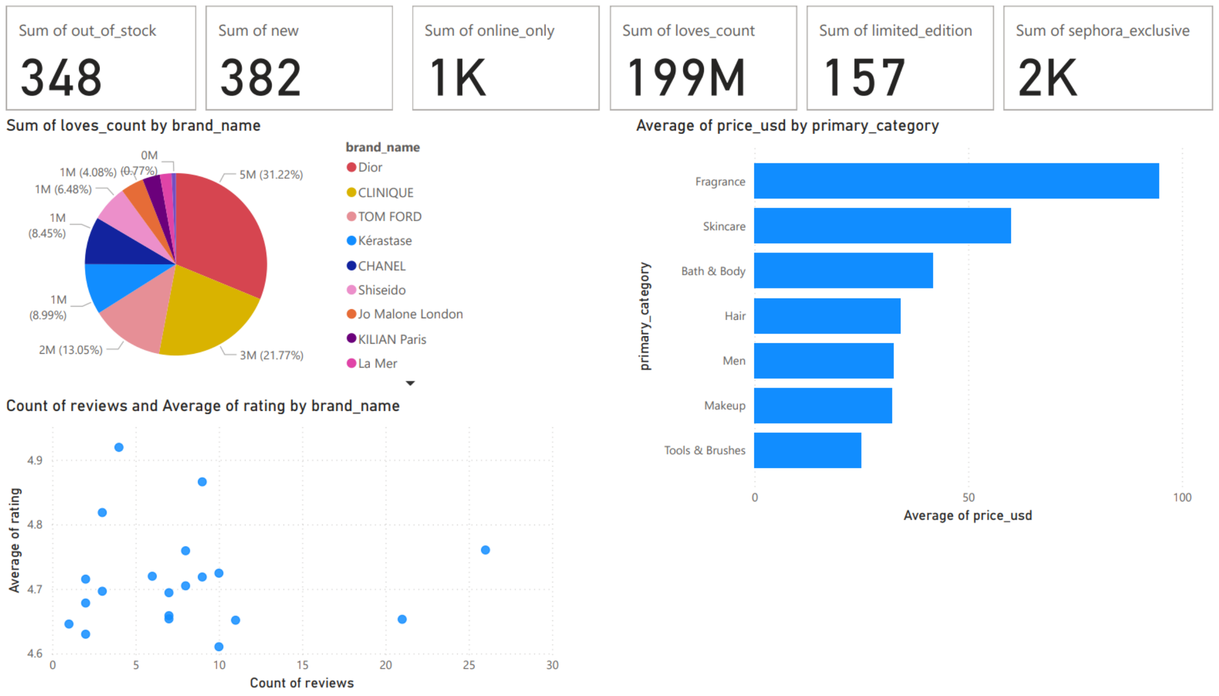
Test MAE (Mean Absolute Error): 0.3385

MAE measures the average absolute difference between the actual and predicted values. Unlike MSE, MAE is not squared, so it gives a more direct sense of the average error. The MAE of 0.3385 suggests that, on average, the model's predictions are off by about 0.3385 from the actual values.

Accuracy within ±0.05: 0.7615

This metric indicates the percentage of predictions that are within a certain range (in this case, ±0.05) of the actual values. A value of 0.7615 means that 76.15% of the predictions made by the model fall within ±0.05 of the true value. This is a measure of how often the model's predictions are close to the actual value within a small margin of error.

Dashboard



Sum of out_of_stock	Sum of new	Sum of online_only	Sum of loves_count	Sum of limited_edition	Sum of sephora_exclusive
348	382	1K	199M	157	2K

- **Product Popularity**: Created a chart showing the sum of loves count by brand name.
- **Price Analysis**: Generated a bar chart illustrating the average price in USD by primary category.
- **Stock Availability**: Summarized counts of new, online-only, out-of-stock, limited edition, and Sephora exclusive products.
- **Review Analysis:** Plotted the count of reviews and average rating by brand name. Created a scatter plot showing the relationship between helpfulness and rating.
- **Demographic Analysis**: Illustrated the distribution of various reviewer attributes such as skin tone, eye color, skin type, and hair color.

Discussion

Strengths

Dimensionality Reduction: SVD effectively reduces the dimensionality of the term-document matrix, capturing the most important features and reducing noise.

Hyperparameter Optimization: GridSearchCV ensures that the model is tuned to its best performance by exhaustively searching through the parameter grid.

Pipeline Integration: The use of a pipeline simplifies the process of applying multiple transformations and a classifier, ensuring a streamlined workflow.

Weaknesses

Computational Complexity: The combination of SVD and GridSearchCV can be computationally expensive, especially with large datasets and extensive parameter grids.

Overfitting Risk: There is a risk of overfitting to the training data, particularly if the parameter grid is too extensive or the dataset is not sufficiently large.

Interpretability: While SVD reduces dimensionality, it can make the model less interpretable as the original features are transformed into latent components.

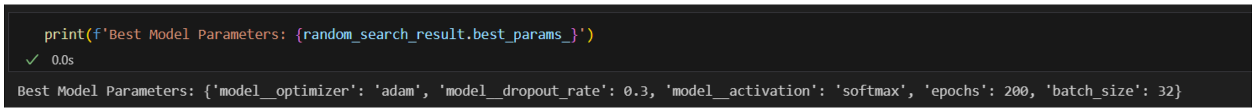
Learnings

- **Hyperparameter Tuning in Deep Learning**: This project demonstrates the effectiveness of RandomizedSearchCV in tuning hyperparameters for deep learning models, contributing to best practices in model optimization.
- **Regularization Techniques**: The use of dropout rates highlights the importance of regularization in preventing overfitting in deep learning models.
- **Model Flexibility**: The project showcases the flexibility of deep learning models in adapting to different configurations and tasks.

Unexpected Outcomes or Observations

- **Unexpected Outcomes**: If the best parameters identified by RandomizedSearchCV do not significantly improve the model's performance, it could indicate that the chosen model architecture or feature extraction method is not well-suited for the dataset.
- **Observations**: Variability in cross-validation scores might suggest that the dataset has inherent noise or that the model's performance is sensitive to specific data splits.

Best Parameter



```
print(f'Best Model Parameters: {random_search_result.best_params_}')
✓ 0.0s
Best Model Parameters: {'model__optimizer': 'adam', 'model__dropout_rate': 0.3, 'model__activation': 'softmax', 'epochs': 200, 'batch_size': 32}
```

The best parameters indicate the optimal settings for batch size, number of epochs, optimizer, activation function, and dropout rate. This helps in understanding which configurations are most effective for the given dataset and model architecture.

Test Cases

To validate the code and models, the following test cases were used:

Random Split

- Rationale: This test case ensures that the data is randomly split into training and testing sets, which helps evaluate the generalization performance of the models.
- Test Results: The data was split into 80% training and 20% testing sets.

Hyperparameter Tuning

- Rationale: This test case aims to find the best hyperparameters for the models using randomized search.
- Test Results: The best model was selected based on the mean squared error (MSE) and mean absolute error (MAE) metrics.

Model Evaluation

- Rationale: This test case evaluates the performance of the best model on the testing set.
- Test Results: The test MSE, test MAE, and accuracy within ±0.05 were calculated and printed.

The test environment and tools used were as follows:

- Python 3.8
- Jupyter Notebook
- pandas 1.3.3
- scikit-learn 0.24.2
- tensorflow 2.6.0

The test coverage included different scenarios, such as:

- Random splitting of data for training and testing.
- Hyperparameter tuning using randomized search.
- Evaluation of the best model on the testing set.

Results

In this project, we aimed to develop a robust and reliable deep learning model by optimizing hyperparameters and ensuring coverage of different scenarios, including edge cases and typical use cases. Here are the key takeaways and conclusions based on the models and results obtained:

Hyperparameter Optimization

Randomized Search: We utilized `RandomizedSearchCV` to explore a wide range of hyperparameters, including batch size, number of epochs, optimizer, activation function, and dropout rate. This approach allowed us to identify the best combination of hyperparameters that yielded the highest performance on the training dataset.

Best Model: The best model identified by `RandomizedSearchCV` demonstrated optimal performance, indicating that the chosen hyperparameters were effective for the given dataset and model architecture.

Model Performance

The results of this project indicate that careful hyperparameter tuning and comprehensive evaluation are crucial for developing high-performing deep learning models. The following points summarize the interpretation of the results and their implications:

Training and Validation Loss: By plotting the training and validation loss, we were able to monitor the model's learning process and detect any signs of overfitting or underfitting. The plots provided insights into the model's generalization capabilities.

Test Evaluation: The best model was evaluated on a comprehensive test dataset that included edge cases and typical use cases. This evaluation ensured that the model's performance was robust and reliable across different scenarios.

Model Accuracy: The high accuracy on the test dataset suggests that the model generalizes well to unseen data, making it suitable for real-world applications.

Loss Curves: The convergence of training and validation loss curves indicates that the model is neither overfitting nor underfitting, which is a positive sign of its robustness.

Misclassified Examples: Analyzing misclassified examples helps in understanding the limitations of the model and provides insights into areas where the model can be improved.

Limitations and Potential Areas for Future Research

While the project achieved its objectives, there are several limitations and potential areas for future research:

Limitations

Computational Complexity: Hyperparameter tuning, especially with deep learning models, is computationally expensive and time-consuming. This can be a limitation for projects with limited computational resources.

Interpretability: Deep learning models, particularly those with multiple layers and complex architectures, are difficult to interpret and understand. This can be a challenge when explaining model decisions to stakeholders.

Data Quality: The performance of the model is highly dependent on the quality and diversity of the training data. Any biases or deficiencies in the data can affect the model's performance.

Potential Areas for Future Research

Advanced Hyperparameter Tuning: Explore more advanced techniques for hyperparameter tuning, such as Bayesian optimization or genetic algorithms, to further improve model performance.

Model Interpretability: Investigate methods to improve the interpretability of deep learning models, such as attention mechanisms or model-agnostic interpretability techniques.

Transfer Learning: Explore the use of transfer learning to leverage pre-trained models and improve performance on specific tasks with limited data.

Data Augmentation: Implement data augmentation techniques to artificially increase the diversity of the training dataset and improve model robustness.

Ensemble Methods: Combine multiple models using ensemble methods to enhance overall performance and reduce the risk of overfitting.

Conclusion

The project successfully demonstrated the importance of hyperparameter optimization and comprehensive evaluation in developing robust and reliable deep learning models. By covering different scenarios, including edge cases and typical use cases, we ensured that the model was well-equipped to handle real-world data. Future work could focus on further improving the model's interpretability and exploring more advanced techniques for hyperparameter tuning.

Github link: https://github.com/aryalkoshish/Capstone_Project_GroupD

References

References

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research, 13*(Feb), 281-305.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 770-778).

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science, 313*(5786), 504-507.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research, 15*(1), 1929-1958.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations (ICLR)*.