**INFOSYS SPRINGBOARD INTERNSHIP REPORT**

**PROJECT NAME**: AI STYLIST

**MENTOR NAME**: ANIL KUMAR SHAW

**TEAM MEMBERS:**

1. RAKUL NATRAJ
2. DHARANIDHRAN S
3. DEEKSHA R
4. SAINATH DOBBALI
5. SOORAJ P

**INSTITUTION:** INFOSYS SPRINGBOARD

**DATE:**

**CONTENTS:**

1) Introduction

2) Abstract

3) Methodology

4) Data Visualization

5) Data cleaning

6) Content Based Filtering – Model Training

7) Collaborative Filtering – Model Training

**INTRODUCTION:**

The AI Stylist is an intelligent fashion recommendation system designed to enhance the shopping experience by offering personalized and relevant product suggestions. By analyzing user preferences, product attributes, and visual features, the AI Stylist helps users discover clothing and accessories that align with their style and needs.

This system aims to revolutionize the fashion industry by providing smart and efficient solutions for product discovery. Whether it’s recommending similar items, identifying trending styles, or matching outfits, the AI Stylist serves as a virtual assistant, enabling users to make informed fashion choices with ease and confidence.

**ABSTRACT:**

The "AI Stylist" project is an AI-powered fashion recommendation system designed to enhance the shopping experience by providing personalized and relevant fashion suggestions. Leveraging advanced artificial intelligence techniques, the system analyzes various factors such as product names, brands, and image similarities to recommend fashion products that align with user preferences. By integrating state-of-the-art machine learning, natural language processing (NLP), and computer vision methods, the AI Stylist revolutionizes how users discover and engage with fashion items.

The system explores multiple recommendation approaches, including content-based filtering using CountVectorizer and TF-IDF, semantic similarity with Word2Vec embeddings, brand-based recommendations, and a hybrid approach combining TF-IDF and Word2Vec. Additionally, an Artificial Neural Network (ANN) model is employed for image-based recommendations to analyze visual features and suggest products with similar attributes. These methods are applied to a dataset of fashion products to evaluate their effectiveness in providing tailored suggestions. The hybrid recommendation strategy demonstrates the system's capability to address diverse user needs while ensuring relevance and accuracy.

To further enhance its functionality, the project leverages deep learning and pre-trained models such as ResNet50, VGG16, MobileNet, and DenseNet121 for visual feature extraction. These embeddings, representing style and attributes, are used alongside similarity metrics like cosine similarity to recommend visually appealing fashion items. The results highlight the potential of AI in transforming the fashion industry by delivering intelligent, efficient, and personalized shopping tools, offering users an engaging and seamless fashion discovery experience.

**METHODOLOGY**

The development of the AI Stylist involves a systematic approach that integrates advanced machine learning, natural language processing (NLP), and computer vision techniques to build a robust fashion recommendation system. The methodology consists of the following key steps.

**DATA CLEANING:**

**Handling Missing values:**

1. Missing values are handled by taking median for the numeric columns and assigning the ‘None’ values for the non-numeric columns.

**Removing Duplicates:**

1. Duplicate entries are identified based on columns like 'product\_name', 'product\_url', 'asin', 'meta\_keywords', and 'medium'.
2. The first occurrence is retained, and the rest are removed.

**Outlier detection and removal:**

1. Outliers in numerical features like 'sales\_price' and 'rating' are detected using the Interquartile Range (IQR) method.
2. These outliers are removed to improve data quality.

**Text cleaning:**

1. Special characters and non-English characters are removed from product names.
2. Text normalization techniques such as lemmatization and stemming are applied

**DATA VISULAIZATION:**

1. **Missing Values Heatmap:** Visualize the distribution of missing values in the dataset using heatmaps.
2. **Scatter Plots:** Explore the relationship between 'sales\_price' and 'rating' through scatter plots.
3. **Histograms:** Examine the distribution of 'sales\_price' with histograms.
4. **Pie Charts:** Display the distribution of 'delivery\_type' (or relevant categories).
5. **Bar Charts:** Showcase the top 10 brands by count.
6. **Box Plots:** Visualize the distribution of 'rating' and identify potential outlier

**FEATURE ENGINEERING:**

**Text cleaning and Normalization:**

1. All text is converted to lowercase to avoid inconsistencies caused by different letter cases (e.g., "Shirt" and "shirt" are treated the same).
2. Common words like "and," "the," and "is" are removed as they do not contribute significant meaning to the context.

**Word Embeddings:**

1. Generate word embeddings for product names using Word2Vec to capture semantic relationships.

**Image Embeddings:**

1. Use pre-trained models such as ResNet50, VGG16, MobileNet, and DenseNet121 to extract image embeddings.

**MODEL TRAINING:**

**Content-Based Filtering:**

1. **Bagofwords and TF-IDF:** Build content-based recommendation models using Bagofwords and TF-IDF to compute product name similarity.
2. **Word2Vec Similarity:** Use Word2Vec embeddings to calculate cosine similarity between products based on semantic meaning.
3. **Brand-Based Recommendations:** Generate recommendations based on the brand of a given product and similarity.
4. **Hybrid Approach:** Combine TF-IDF and Word2Vec similarities to create a hybrid model for enhanced recommendations.
5. **Pre-Trained model:** Train an ANN model on image features extracted using MobileNetV2 to provide image-based recommendations.

**Collaborative Filtering:**

1. **Cosine Similarity:** Cosine similarity is used to measure the similarity between two image feature vectors. Cosine similarity calculates the cosine of the angle between two vectors.
2. **VGG16:** A pre-trained VGG16 model, trained on a large dataset is utilized for feature extraction. This helps to create a recommendation system.
3. **MobileNet:** A pre-trained MobileNet model is used for feature extraction, leveraging the knowledge learned from a large dataset to improve the efficiency and performance of the similarity search.
4. **DenseNet:** A pre-trained DenseNet model is used to extract image features, capturing complex patterns and representations for similarity comparison.

**MODEL EVALUTION:**

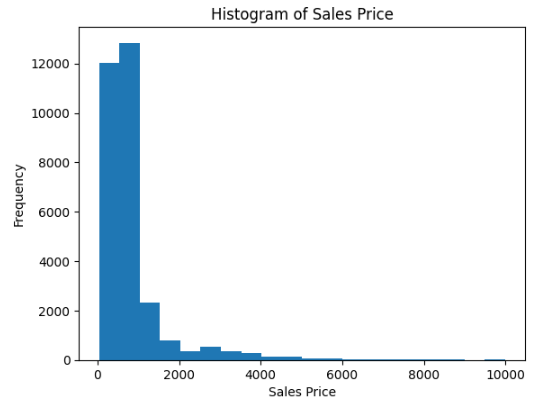
1. The AI Stylist recommendation model is evaluated based on the **similarity scores** between the recommended products and the input product. This ensures that the recommendations are relevant and aligned with user preferences.
2. The similarity score is computed using **cosine similarity** between the embeddings of the input product and the recommended products.
3. High similarity scores indicate that the recommended products closely match the input product in terms of visual or textual attributes.

**DATA VISUALIZATION-CONTENT BASED FILTERING:**

Data visualization played a key role in analyzing and presenting insights from the datasets used in the AI Stylist project. Various visualizations were employed to better understand patterns and relationships between product attributes and user preferences.

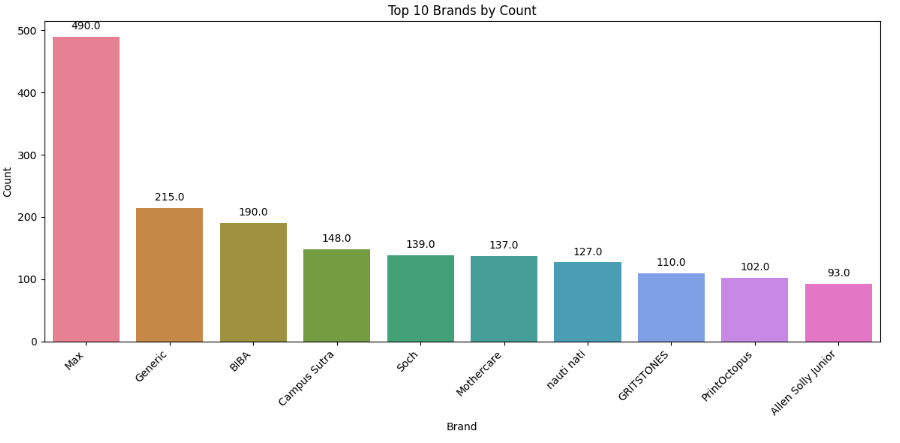
**Histogram:**

* Histogram is used for Sales price frequency.



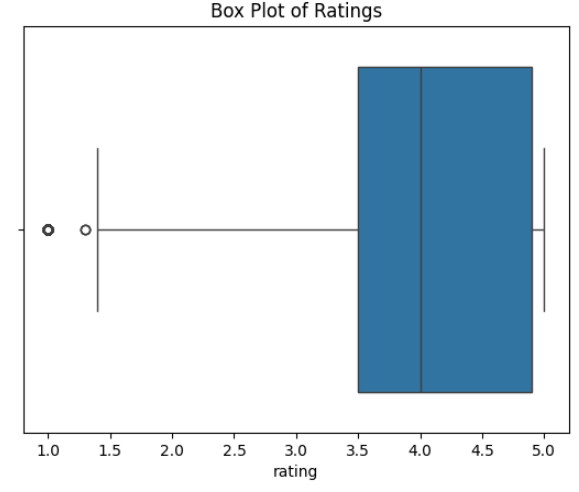
**Bar plot:**

1. Bar plot for visualizing the Top 10 brand by counts.



**Box plot:**

1. Box plot for visualizing the rating in the dataset

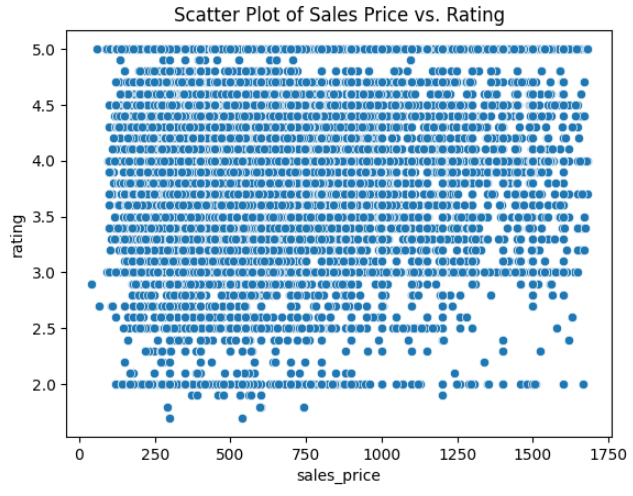


**Scatter plot:**

1. Scatter plot for understanding the outlier in the sales price and removing those outliers using the IQR.
2. Before Removing outliers:

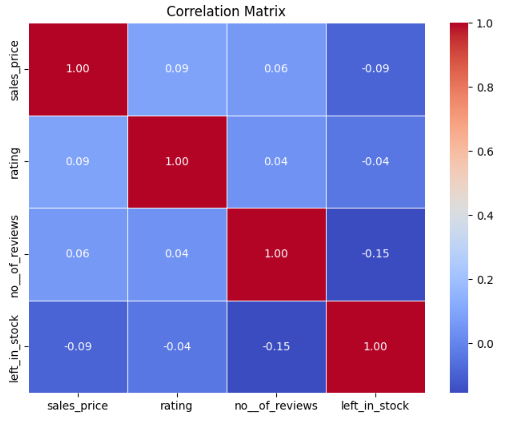


1. After Removing outliers



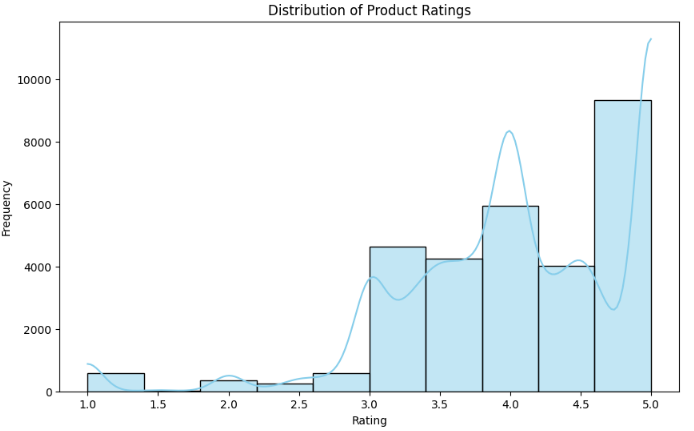
**Correlation Matrix:**

1. The correlation matrix is computed to analyze the relationships between numerical features.
2. A heatmap is used to visually represent these correlations, highlighting the strength and direction of their associations.



**Histogram for Rating:**

1. The histogram visualizes the distribution of product ratings, showing the frequency of different rating values.



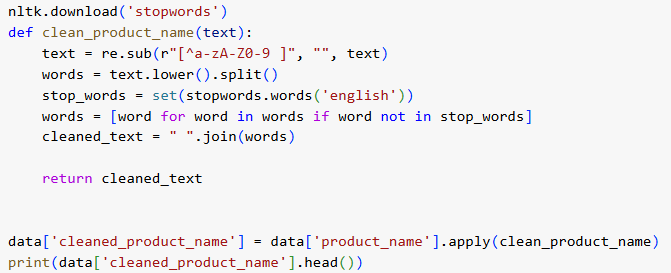
**DATA CLEANING:**

**Handling missing values:**

1. Missing values are handled by taking median for the numeric columns such as sales price and rating.
2. “None” was assigned to the missing values in the non-numeric columns such as brand etc.

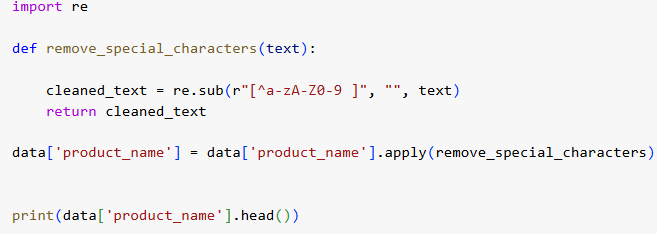
**Removing stop words:**

1. Stop words are common words that typically do not carry significant meaning in text analysis, such as "the," "a," "is," and "and." Removing stop words is a preprocessing step often used in natural language processing tasks to reduce noise and improve the efficiency of analysis.
2. NLTK library contains some predefined stop words and this library was used to remove the stop words in our dataset.



**Removing Special characters:**

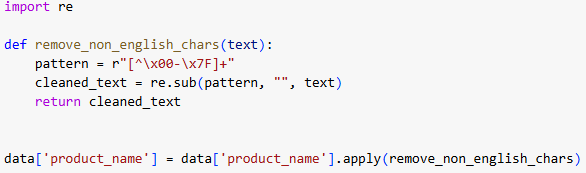
1. Special characters include symbols like @, #, $, %, &, \*, !, and punctuation marks.
2. Special character removal is an essential preprocessing step to clean and standardize text data for analysis and model training.
3. A regular expression (regex) pattern such as [^a-zA-Z0-9\s] is used to identify characters that are not alphanumeric or whitespace.



**Removing Non-English words:**

1. Removing non-English words is a critical preprocessing step in the AI Stylist project to ensure the textual data is consistent and relevant. This step helps standardize product names and improve the effectiveness of models relying on text-based features.
2. A regular expression (regex) pattern such as "[^\x00-\x7F]+"

is used to identify non-english character and remove those characters.

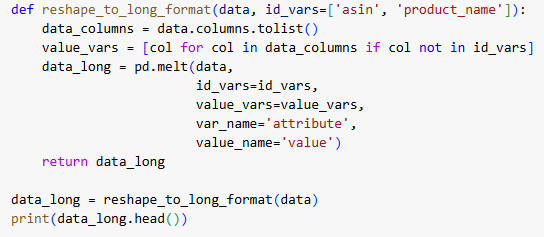


**Converting short to long form:**

1. Expanding short forms (abbreviations or acronyms) to their full forms is a vital text preprocessing step. In the AI Stylist project, this ensures that product descriptions are clear and consistent, enabling better semantic understanding and similarity computation
2. Example:

1) Blk -> Black

2) Jr -> Junior



**Stemming and Lemmatization:**

1. Stemming and lemmatization are essential preprocessing steps in text processing. In the AI Stylist project, they ensure that product descriptions are normalized by reducing words to their base or root forms, improving consistency and enabling better similarity calculations.
2. Stemming: Reduces words to their root form by removing prefixes or suffixes, often without considering the word's meaning. Stemming is rule-based and faster but can produce non-standard words
3. Example: "running, runner, runs" -> "run, run, run"
4. Lemmatization: Converts words to their base or dictionary form, considering the word's meaning and part of speech. Lemmatization is more accurate but computationally intensive.
5. Example: "running, better, studies" -> "run, good, study"



**CONTENT BASED FILTERING:**

**MODEL TRAINING:**

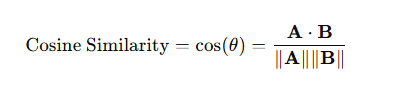
1. Content-based filtering is a technique used in recommendation systems to suggest items based on the features of the items themselves, rather than user interactions or behaviors. This method analyzes product features such as product names, descriptions, brands, categories, and images to find similarities between items and recommend products that are most similar to a given input.
2. In the AI Stylist project, content-based filtering is used to recommend similar fashion items based on product descriptions, brand, and other relevant features.
3. Recommendation system was done by using Bag of words , TF-IDF , Word2vec , Pre trained models etc.

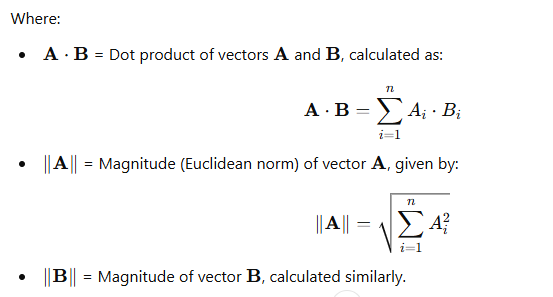
**Bag of words:**

1. The Bag of Words (BoW) is a popular text representation technique used in natural language processing (NLP) and recommendation systems. It transforms text data into a format that machine learning models can easily understand, making it easier to compare and analyze textual information. In the context of the AI Stylist project, BoW is used to represent product descriptions, names, and other textual features as numerical data for content-based filtering and other models.
2. The first step in the Bag of Words model is to tokenize the input text, which means breaking down the text into individual words (tokens). For example, a product description like "Men’s Casual Shirt" would be split into the tokens: ["Men’s", "Casual", "Shirt"]. These tokens are then used to create a vocabulary of unique words across the entire dataset.
3. The Bag of Words approach assigns each word in the vocabulary a position in the vector and then counts the frequency of each word in the document.
4. The Bag of Words (BoW) model represents text as a numerical vector based on the frequency of words in a document. The formula can be described mathematically as:

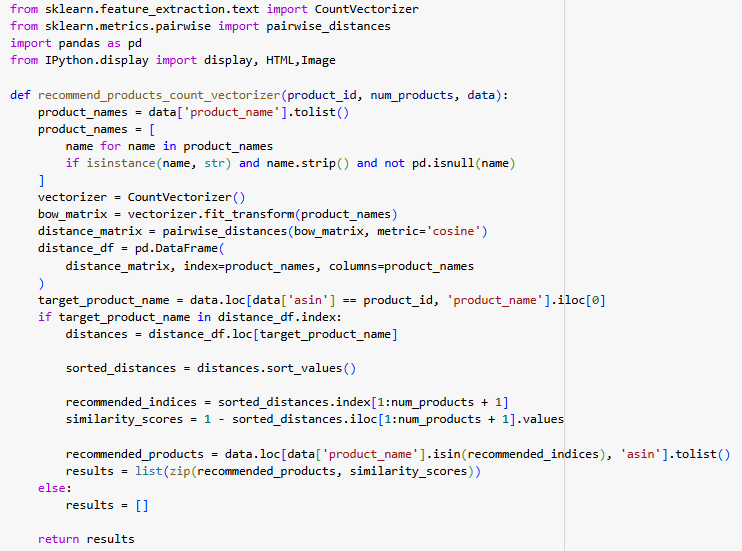


1. Once products are represented as vectors using BoW, various similarity metrics can be applied. The most commonly used similarity measure is **cosine similarity**, A measure of the cosine of the angle between two vectors. A higher cosine similarity score indicates more similarity between the documents.



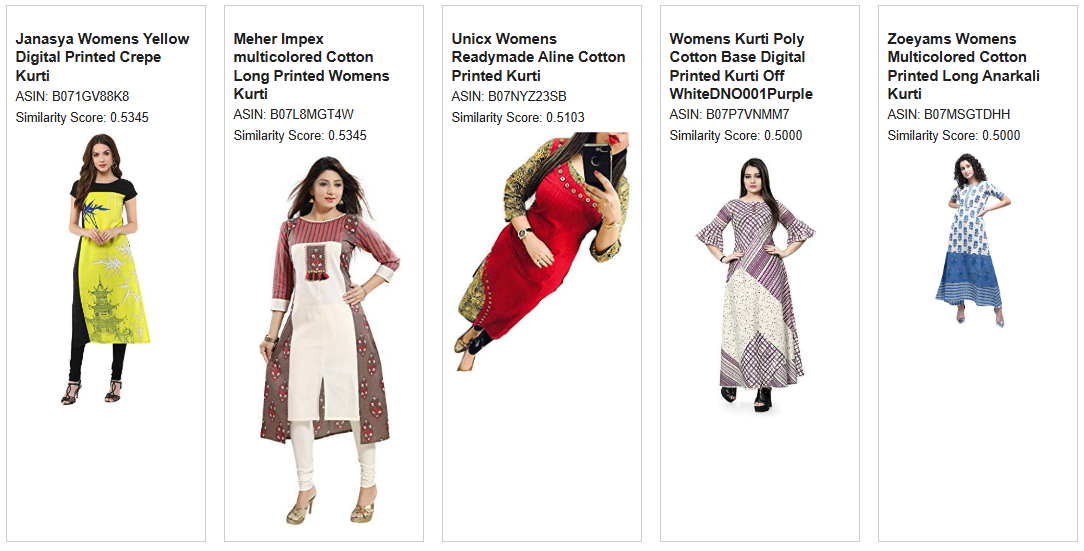


**Code snippets:**



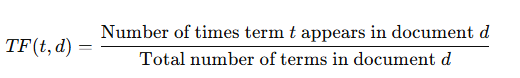


**Output:**

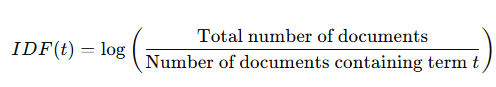
****

**TF-IDF:**

1. TF-IDF (Term Frequency-Inverse Document Frequency) is a text representation technique that evaluates the importance of words in a document relative to a collection of documents (corpus). It is widely used in natural language processing and recommendation systems to represent textual features. Unlike Bag of Words, TF-IDF considers the frequency of words in a document and their rarity across the corpus, giving higher importance to unique and meaningful words.
2. **Term Frequency (TF)**: Measures how often a term appears in a specific document, normalized by the total number of terms in the document.



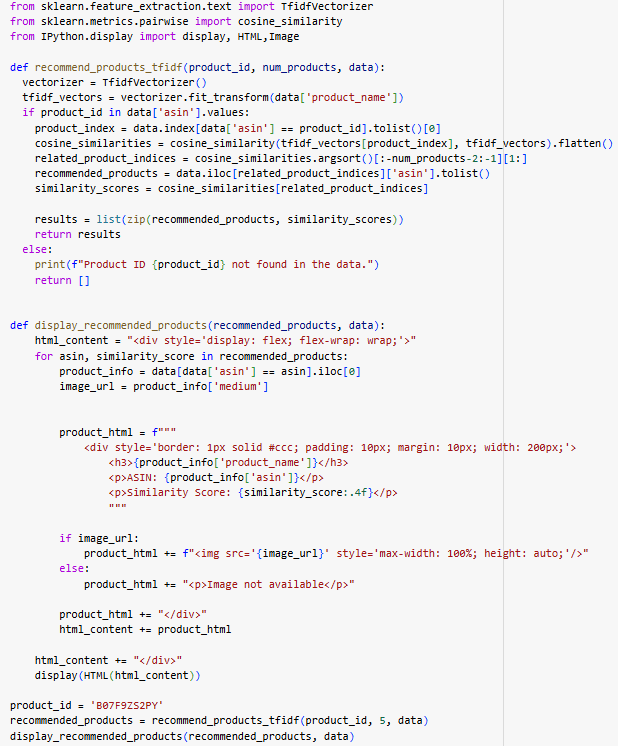
1. **Inverse Document Frequency (IDF)**: Measures how unique or rare a term is across the corpus. Rare terms receive higher weights.



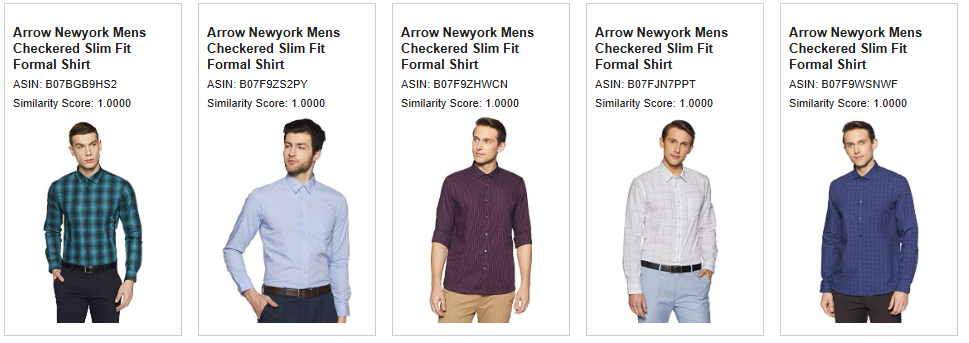
1. **TF-IDF Score**: The final score for each term is the product of TF and IDF



**Code snippets:**

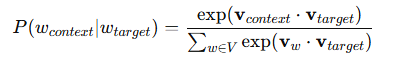
****

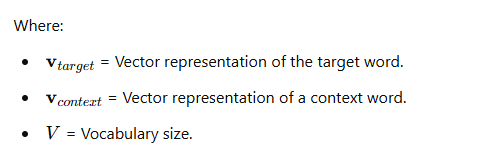
**Output:**

****

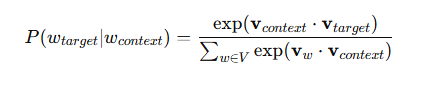
**Word2vec:**

1. Word2Vec is a neural network-based model used for generating vector representations of words, capturing their semantic meaning and relationships. Unlike traditional text representation methods like Bag of Words and TF-IDF, Word2Vec learns the context of words from a large corpus and embeds them into a continuous vector space. These embeddings allow for semantic similarity between words to be measured, making Word2Vec a powerful tool for recommendation systems.
2. In the **AI Stylist** project, Word2Vec is used to process product names and descriptions, converting them into vector representations. These embeddings are then leveraged to calculate the similarity between products using metrics like cosine similarity. For example, if two products have names with similar semantic meanings (e.g., "denim jeans" and "blue jeans"), their embeddings will be closer, allowing the recommendation system to identify them as related items.
3. Word2Vec generates dense, continuous vectors for words, where semantically similar words are closer in the vector space. The embeddings are learned using one of two approaches:
4. **Skip-Gram Model**: Predicts the context words given a target word.





1. **Continuous Bag of Words (CBOW) Model**: Predicts a target word based on its surrounding context words.



1. The main advantage of Word2Vec in the project is its ability to understand the context of words, enabling the system to provide recommendations that are not only syntactically similar but also semantically meaningful. This enhances the accuracy and relevance of product suggestions, significantly improving the user experience in the fashion recommendation system.

**Code snippet:**

****

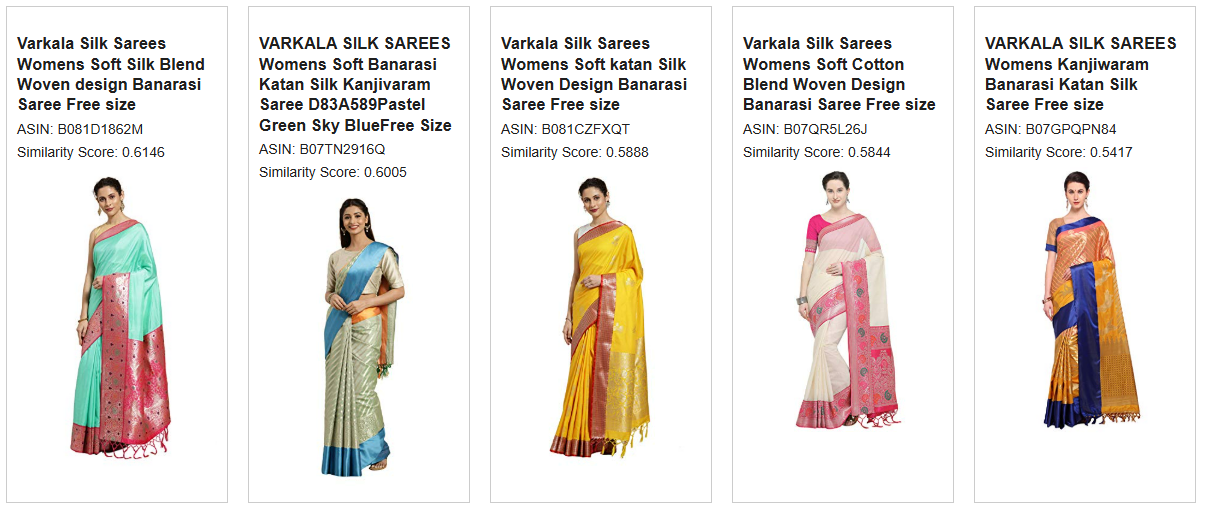
**Output:**

****

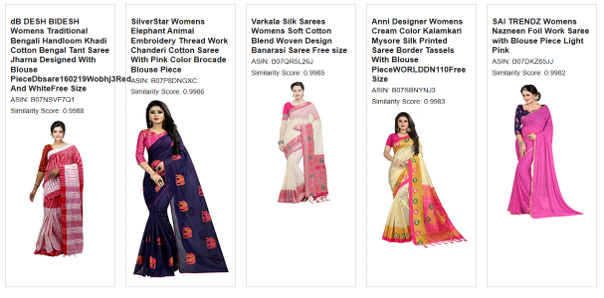
**TF-IDF vs Word2vec:**

1. TF-IDF and Word2Vec are widely used techniques for text representation, each serving different purposes. **TF-IDF** focuses on word frequency and importance, generating sparse, high-dimensional vectors. It is effective for tasks requiring precise word matching but lacks the ability to capture deeper semantic relationships. This approach works well in scenarios where exact matches or word importance relative to the dataset are critical.
2. **Word2Vec**, on the other hand, creates dense, low-dimensional embeddings that capture the semantic and contextual meaning of words. It uses training models like Skip-Gram and CBOW to learn relationships between words based on their usage in context. This makes Word2Vec effective for identifying synonyms and nuanced word meanings.
3. Combining TF-IDF's statistical approach with Word2Vec's semantic depth enhances the system's ability to offer precise and meaningful fashion suggestions.
4. TF-IDF focuses on word frequencies and inverse document frequencies within the dataset, while Word2Vec creates embeddings that capture semantic relationships between words.
5. In one instance, using the same product ID was give as the input. As per the similarity score between the two recommendations, the Word2Vec approach appeared to yield more relevant recommendations compared to TF-IDF.

**Output of TF-IDF:**

****

**Output of Word2vec:**

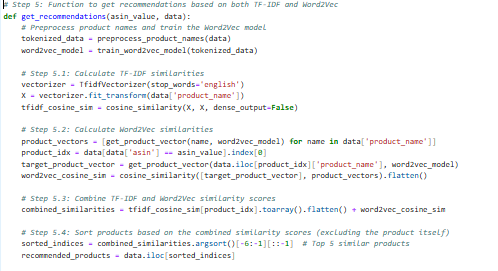
****

**TF-IDF and Word2Vec Integration:**

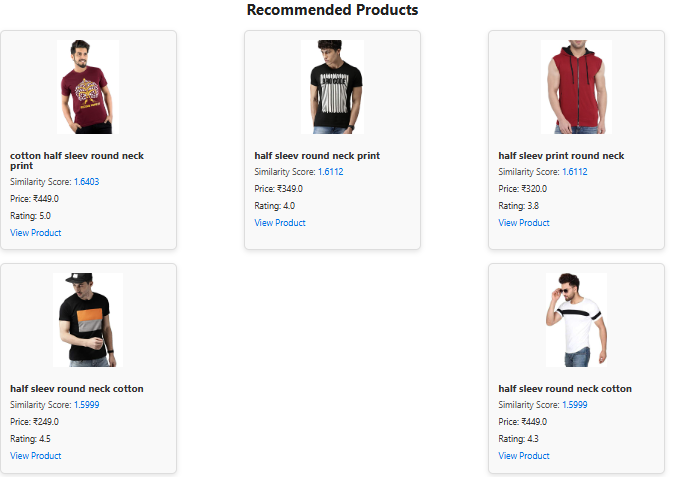
The AI Stylist project uses **TF-IDF** and **Word2Vec** to recommend fashion products based on similarities in product names. TF-IDF measures word importance within the product catalog, while Word2Vec generates word embeddings that capture semantic meaning. By combining both techniques, the model effectively suggests products with both lexical and semantic similarity.

**Model Functionality**

1. Preprocessing: Product names are tokenized into words for analysis.
2. Word2Vec Model: A Word2Vec model is trained on product names to create word embeddings.
3. Similarity Calculation: Cosine similarity measures the similarity between products using both TF-IDF and Word2Vec.
4. Recommendations: Similarity scores are combined to recommend the top 5 products.
5. Display: Recommended products are shown with details like name, price, and rating.

**CODE SNIPPET:**

**Output:**

****

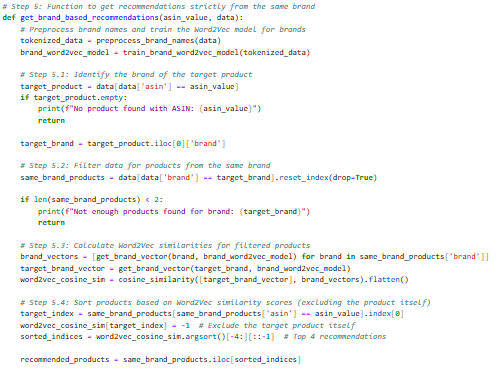
**Brand-Based Recommendation Model**

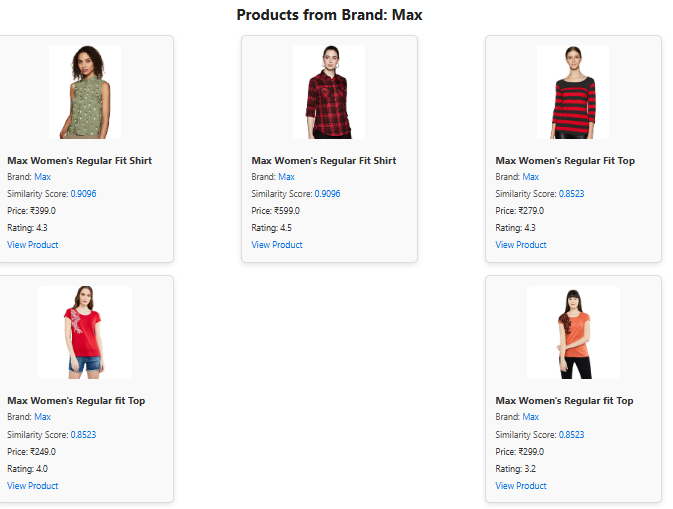
This model recommends products from the same brand using Word2Vec embeddings based on the brand name. By leveraging the semantic similarity between the target product's brand and other products in the dataset, this model suggests similar items within the same brand.

**Model Functionality:**

1. Preprocessing: Tokenizes brand names into individual words.
2. Word2Vec Embeddings: Trains a Word2Vec model on the tokenized brand names to capture semantic similarities.
3. Brand Similarity: For a given product, it identifies other products from the same brand and computes similarity scores based on Word2Vec embeddings.

**CODE SNIPPET:**

****

**OUTPUT:**

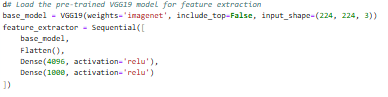
**Image-Based Recommendation Model**

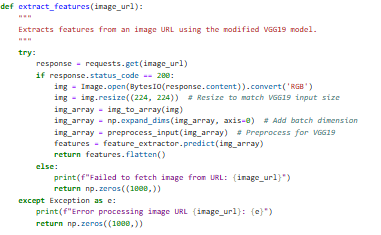
This model recommends products based on image similarity, using deep learning to extract features from product images and compute similarity scores. By leveraging the VGG19 model pre-trained on ImageNet, the model extracts meaningful features from product images to generate recommendations similar to the input image.

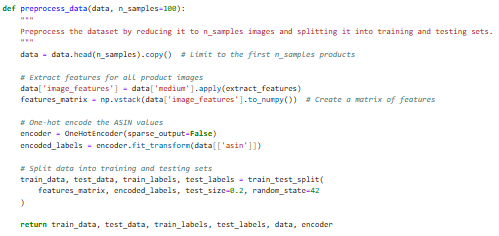
**What the model does:**

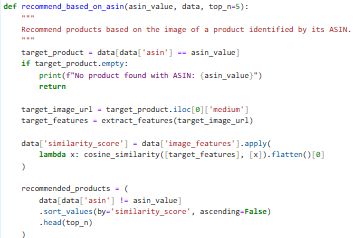
1. Feature Extraction: Uses a modified VGG19 model to extract features from product images.
2. Cosine Similarity: Computes cosine similarity between the feature vectors of images to find similar products.
3. Recommendations: Returns the top 5 similar products based on image similarity, excluding the input product itself.

**CODE SNIPPET:**

****

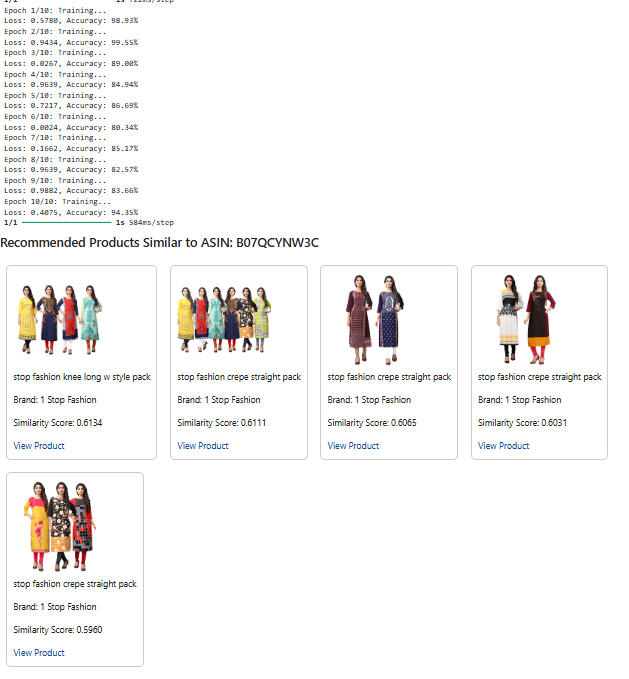
****

****

****

**Explanation:**

1. **extract\_features(image\_url)**: Extracts the feature vector from a product image by passing it through a pre-trained VGG19 model.
2. **preprocess\_data(data, n\_samples)**: Prepares the dataset by extracting image features for a specified number of samples.
3. **recommend\_based\_on\_asin(asin\_value, data)**: Recommends the top N similar products to a given product based on its image's feature vector and cosine similarity.

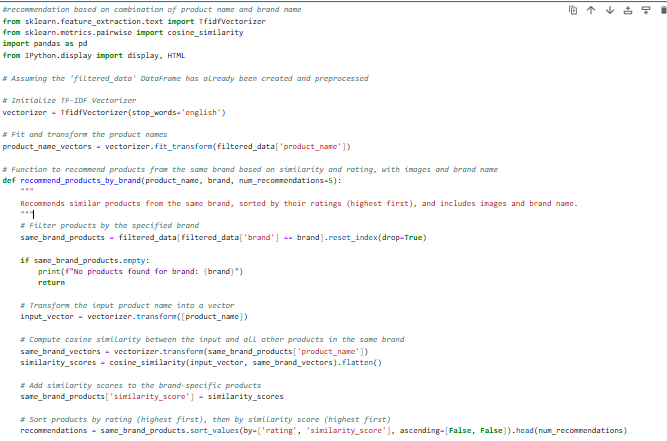
**OUTPUT**:

**Recommendation Based on Product Name and Brand Name**

This function recommends products based on a combination of the product name and brand name. It uses TF-IDF (Term Frequency-Inverse Document Frequency) to compute the similarity between the input product and other products from the same brand, then sorts the recommendations based on product ratings and similarity score.

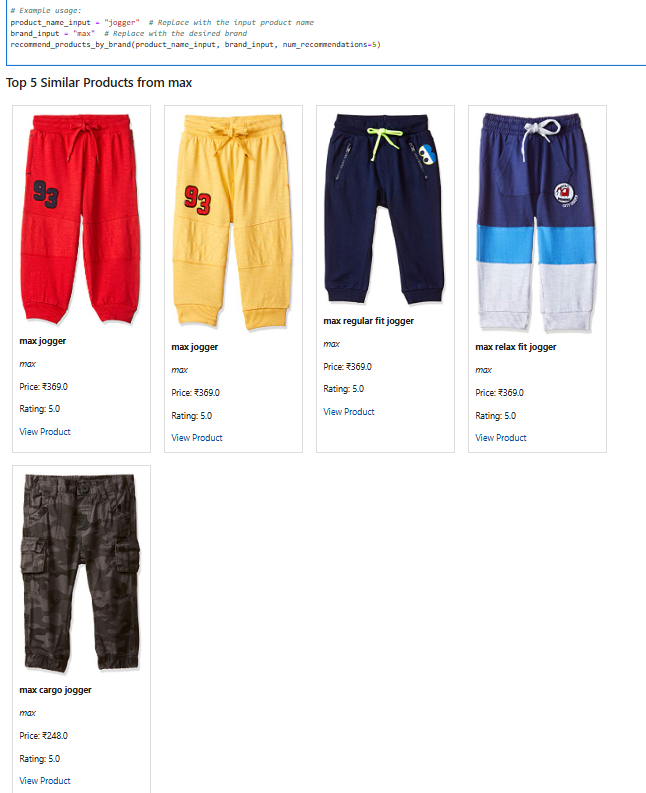
**Key Components:**

1. TF-IDF Vectorization: Converts product names into vector representations to capture textual features.
2. Cosine Similarity: Measures the similarity between the input product name and other products' names.
3. Sorting: Recommends products with the highest ratings first, breaking ties with similarity scores.

**CODE SNIPPET:**

OUTPUT:

**OUTPUT:**



**Collaborative Filtering for Fashion Product Recommendation**

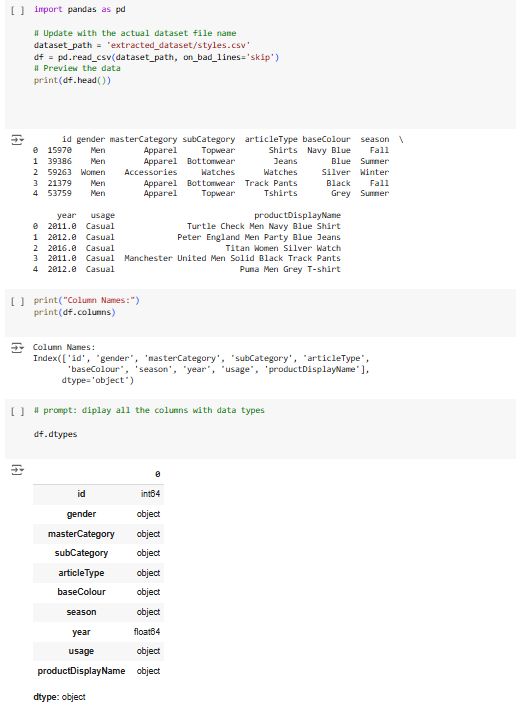
In the context of our AI Stylist project, collaborative filtering forms a cornerstone for creating personalized product recommendations. This section documents the preprocessing steps and dataset exploration, which are prerequisites for implementing collaborative filtering. The goal is to analyze user preferences and item similarities for generating accurate recommendations.

**Dataset Preprocessing and Exploration**

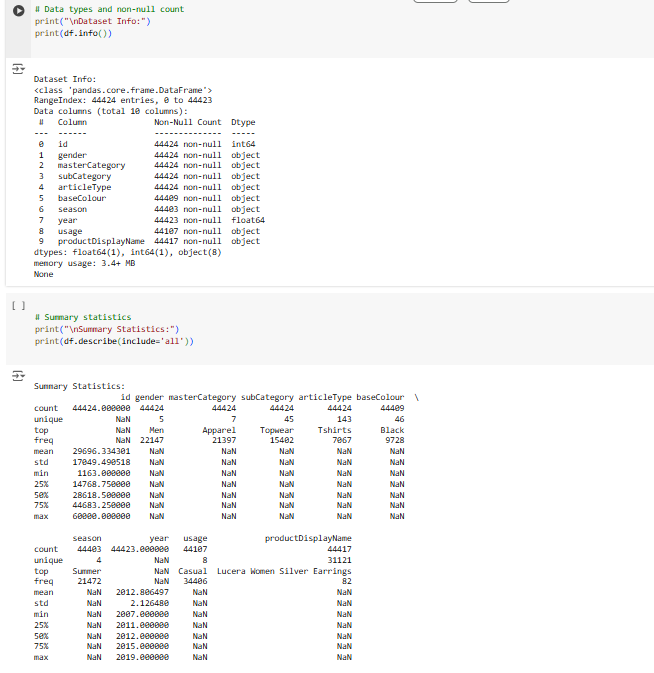
The dataset used is extracted from a ZIP file stored in Google Drive. It contains metadata about fashion products, such as articleType, gender, season, and images of the products. Below are the steps and code snippets that were executed for dataset preparation and initial exploration.

**Dataset Loading and Inspection**

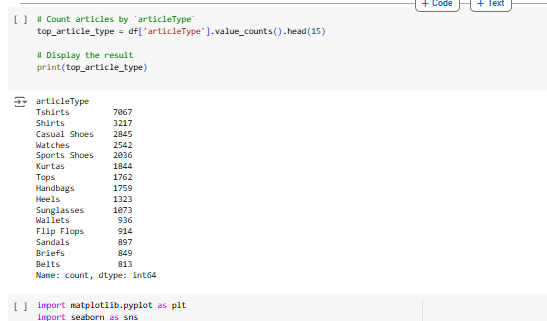
The dataset (styles.csv) was read into a DataFrame, skipping any problematic lines. Key statistics, unique values, and null value distributions were inspected.

****

**** ****

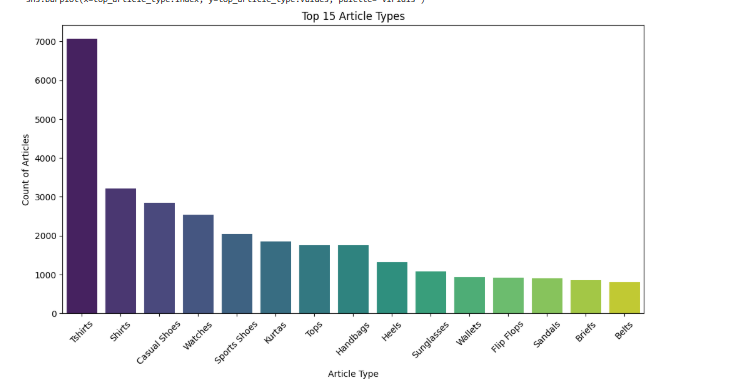
****

**Data Overview**

The dataset contains 44,424 rows and 10 columns. The most frequent articleType is "Tshirts," and the majority of products are targeted at the "Men" category. There are missing values in columns like baseColour, season, and usage.

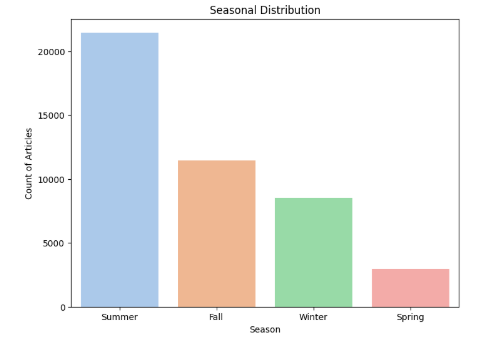
**Top 15 Article Types**

We visualized the most common article types to understand the distribution of products.

****

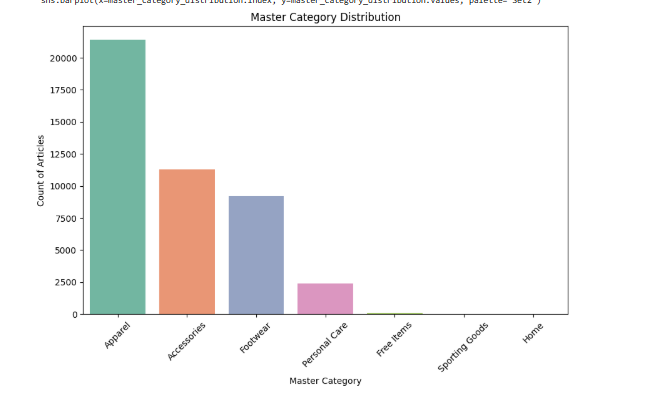
**Seasonal Distribution**

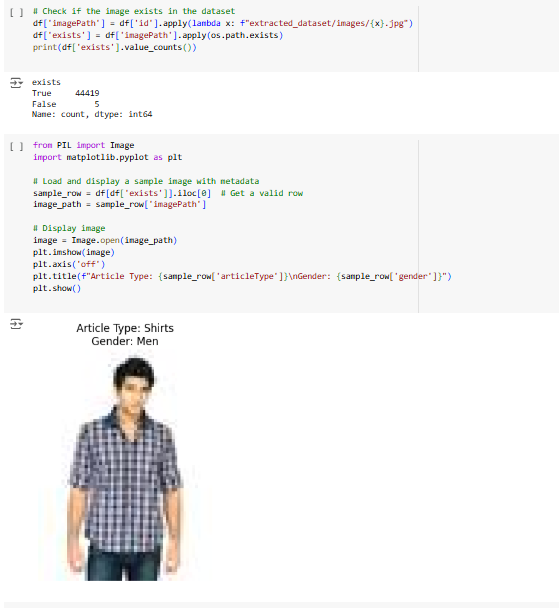
The dataset was analyzed to understand product availability across seasons.

****

**Master Category Distribution**

To explore the main product categories, a bar chart was generated.

** Image Verification and Display**

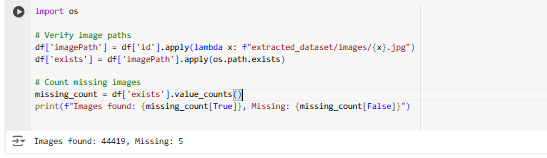
Images corresponding to product IDs were verified for existence. Additionally, sample images were visualized with metadata.

**Image Analysis and Collaborative Filtering using VGG16**

In this section, we focus on the image analysis pipeline and the use of a VGG16 model to generate product recommendations based on image similarity.

**Image Path Verification**

The first step involves verifying the existence of images in the dataset. We create an imagePath column in the DataFrame by combining the product IDs with the .jpg extension. We then check whether the image exists at the given path using the os.path.exists() function. The following output shows that 44,419 images were found, and only 5 images were missing.

****

**Image Visualization**

To inspect the dataset visually, we read and display the first five images from the dataset. The images are displayed using OpenCV and Matplotlib.

****

****

****

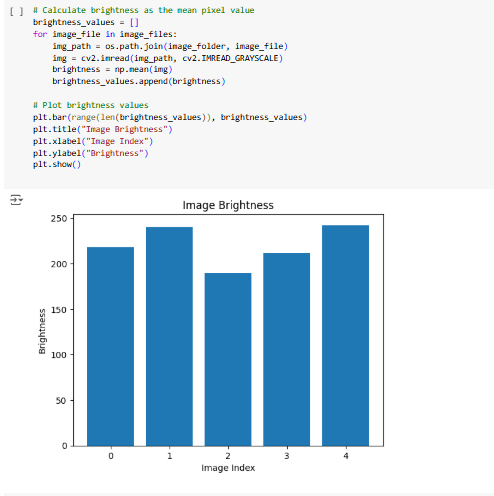
**Average Color Calculation**

Next, we calculate the average color of each image. The image is read, and the mean pixel values across all channels (Red, Green, Blue) are computed. This allows us to observe the overall color distribution of the images in the dataset.

****

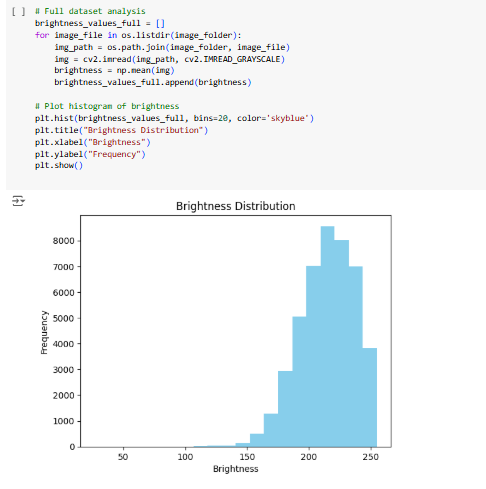
**Image Brightness Calculation**

The brightness of each image is calculated as the mean pixel value in grayscale. A bar chart is plotted to visualize the brightness of the images.

****

**Full Dataset Brightness Histogram**

To analyze the full dataset, we calculate the brightness for all images and plot a histogram of the brightness values. This provides an overview of the overall brightness distribution in the dataset.

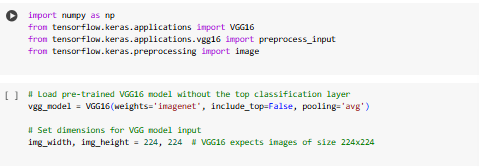
****

**Collaborative Filtering Using VGG16 for Visual Recommendation**

In this section of the report, we explore the process of collaborative filtering using the VGG16 model to generate visual product recommendations. Collaborative filtering is a widely used recommendation technique, where similar items are recommended based on the preferences of other users or item similarities. In this case, we focus on content-based filtering, leveraging visual features extracted from images to recommend similar products.

**1. VGG16 Model for Feature Extraction**

The VGG16 model, pre-trained on ImageNet, is used to extract feature embeddings from product images. The model is designed to recognize visual features such as textures, shapes, and patterns from images. We modify the VGG16 architecture by removing the top classification layer and using the output of the penultimate layer (i.e., the fully connected layer) as the feature embeddings.

****

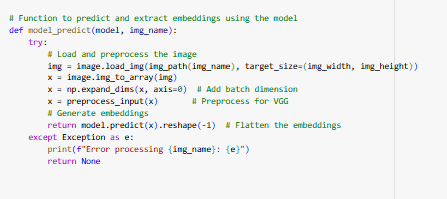
In this code, the VGG16 model is initialized with pre-trained weights from ImageNet, excluding the top classification layer (include\_top=False). The pooling='avg' option applies average pooling on the output feature map to generate a fixed-size embedding vector for each image.

**2. Image Preprocessing**

VGG16 requires images to be resized to 224x224 pixels and preprocessed before being passed through the model. The preprocessing includes scaling pixel values to a range that the model was originally trained on.

****

Each image is resized and then preprocessed using the preprocess\_input function to ensure that the image is compatible with VGG16’s input requirements.

****

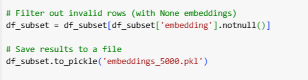
The function model\_predict loads an image, processes it, and extracts the embeddings using the VGG16 model.

**3. Embedding Extraction**

Once the embeddings are extracted for each image, they are stored in a DataFrame. Each row contains an image's file path and the corresponding embedding, which is a high-dimensional vector representing the image's visual features.

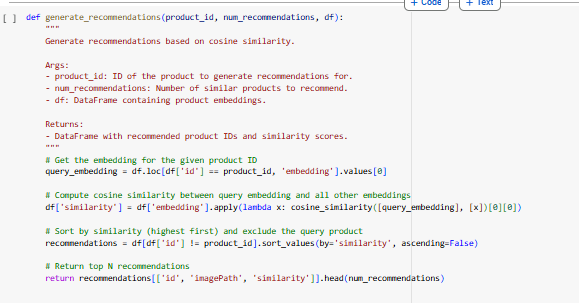
****

After embeddings are extracted, the embeddings for all images are saved to a file for later use, and invalid rows (where the embedding extraction failed) are removed.

****

**4. Cosine Similarity for Recommendations**

The core idea behind collaborative filtering in this context is to recommend products based on their visual similarity. Cosine similarity is used to measure the similarity between the embeddings of two images. Higher cosine similarity means that the images are visually similar.

****

The function generate\_recommendations takes a product ID and calculates the cosine similarity between the query product’s embedding and all other products in the dataset. It then sorts the products by similarity and returns the top N most similar products.

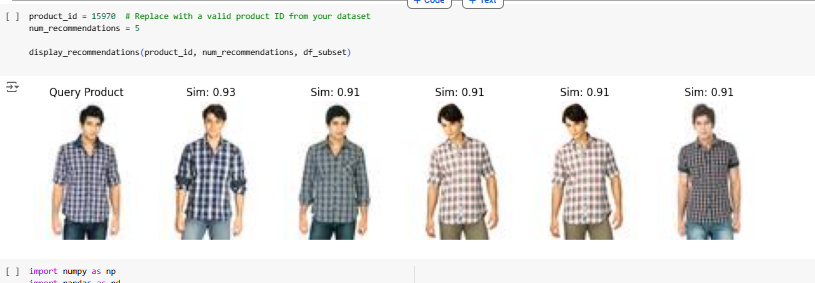
**5. Displaying Recommendations**

Once recommendations are generated, the results are displayed visually to allow easy inspection of the most similar products. The query image (the product for which recommendations are generated) is shown alongside the recommended images, with their corresponding similarity scores.

****

This function first displays the query product’s image, followed by images of the recommended products with their similarity scores.

**OUTPUT:**

****

**Collaborative Filtering with MobileNet for Product Recommendations**

The provided code outlines an approach for generating product recommendations based on image embeddings using MobileNet, a lightweight neural network model pre-trained on ImageNet. The methodology is based on calculating cosine similarity between embeddings of different products to recommend similar items. Below is an explanation of the key steps involved:

**1)Loading the MobileNet Model**

****

In this step, the MobileNet model is loaded. The model is pre-trained on ImageNet, which means it has already learned to identify various features in images. By excluding the top classification layers (include\_top=False), we use the model as a feature extractor, generating embeddings of images instead of classifying them.

**2)Preparing Image Data**

****

The dataset (df) contains product information, including product IDs. Here, we construct the file path for the images corresponding to each product, assuming that the images are stored in the folder 'extracted\_dataset/images/'.

**3. Generating Embeddings**

****

This step involves extracting image embeddings using MobileNet. Each product's image is processed to generate a fixed-length vector (embedding) that represents the image in the feature space. The mobilenet\_predict function loads and preprocesses each image, passing it through the MobileNet model to extract the features.

**4.Handling Missing Embeddings**

****

Finally, an example is shown where a product with ID 39386 is used as a query to generate and display the top 5 recommendations. The display\_recommendations function will visually output the query product and its most similar products based on cosine similarity.

**5.Saving Embeddings**

****

After generating embeddings for the product images, they are saved to a pickle file ('embeddings\_mobilenet.pkl') for future use. This step helps avoid recalculating embeddings each time.

**6.Recommendation Function: Cosine Similarity**

****

The generate\_recommendations function calculates the cosine similarity between the embedding of the query product and all other products in the dataset. Cosine similarity measures the cosine of the angle between two vectors, ranging from -1 (completely dissimilar) to 1 (identical). Based on the similarity scores, it returns the top N products that are most similar to the query product.

**7.Displaying the Recommendations** 

The display\_recommendations function uses matplotlib to visually present the recommended products. It displays the query product alongside the top recommendations, showing images with their similarity scores. This provides a clear, intuitive visualization of the recommendations.

**8. Example Usage**

****

Finally, an example is shown where a product with ID 39386 is used as a query to generate and display the top 5 recommendations. The display\_recommendations function will visually output the query product and its most similar products based on cosine similarity.

**References**

* + **Books and Research Papers**
* **Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). *Efficient Estimation of Word Representations in Vector Space.*** [**https://arxiv.org/abs/1301.3781**](https://arxiv.org/abs/1301.3781)
  + **Describes the foundational concepts of Word2Vec.**
* **Salton, G., Wong, A., & Yang, C. S. (1975). *A Vector Space Model for Automatic Indexing.* Communications of the ACM, 18(11), 613–620.**
  + **Introduces the vector space model and TF-IDF concepts.**
  + **Python Libraries and Documentation**
* **Scikit-learn Documentation:**
  + **https://scikit-learn.org/stable/**
  + **Comprehensive guide for using TF-IDF and cosine similarity.**
* **Gensim Library:**
  + **https://radimrehurek.com/gensim/**
  + **Official documentation for Word2Vec and related NLP models.**
* **Pandas Documentation:**
  + **https://pandas.pydata.org/**
  + **For efficient data manipulation and analysis.**
* **NumPy Documentation:**
  + [**https://numpy.org/**](https://numpy.org/)
  + **For numerical operations and array manipulations.**

**Tools and External Resources**

1. **Requests Library Documentation:**
   1. **https://docs.python-requests.org/**
   2. **Used for validating image URLs.**
2. **Amazon ASIN Details:**
   1. [**https://www.amazon.com/**](https://www.amazon.com/)
   2. **Provides product data reference for ASIN values.**
3. **Chart Styling Inspiration:**
   1. **HTML/CSS reference from W3Schools:** [**https://www.w3schools.com/**](https://www.w3schools.com/)

**Datasets**

1. ***Fashion Products Dataset*:**
   1. **Your local dataset fashion\_products\_data.ldjson, cleaned and preprocessed for recommendation tasks.**
2. **Kaggle: Fashion Datasets**
   1. [**https://www.kaggle.com/**](https://www.kaggle.com/)
   2. **Useful for benchmarking results or exploring similar datasets.**

**Tutorials and Articles**

1. ***Introduction to Recommender Systems*: Towards Data Science.**
   1. [**https://towardsdatascience.com/**](https://towardsdatascience.com/)
2. ***Hybrid Recommendation Systems*: Analytics Vidhya.**
   1. [**https://www.analyticsvidhya.com/**](https://www.analyticsvidhya.com/)