Detailed Report:

Input dataset contain some rows of images with corrupt data and as suggested these are removed from the dataset(703 ,525, 112, 69, 700, 858, 933).

Introduction:

This code aims to provide a retrieval system for image and text data. It leverages deep learning for image feature extraction and natural language processing techniques for text preprocessing and similarity calculation.

Methodology:

The code follows a two-step process: image feature extraction using VGG16 and text feature extraction using TF-IDF. Cosine similarity is then used to calculate similarity scores between text and image features with inputs.

Implementation Details:

Firstly image preprocessing is done by doing resizing and brightness, contrast changing then Image features are extracted using the VGG16 model and normalized. **During feature extraction as we know that some image rows contain many urls in same row so I have implemented functionality like to select best feature from all of them and store that for using in making best result.**

Text data is preprocessed to remove stopwords, punctuation, and perform stemming and lemmatization.

TF-IDF scores are calculated for text documents by creating function from scratch.

Cosine similarity is computed between input text and image features for retrieving similar result.

Below are functions details how are these above functionality is achieved.

1. **adjust_contrast_brightness(img, contrast_factor=1.2, brightness_factor=1.2):**
    - This function adjusts the contrast and brightness of an image.
    - Parameters:
        - `img`: Input PIL image object.
        - `contrast_factor`: Factor by which to adjust the contrast. Default is 1.2.
        - `brightness_factor`: Factor by which to adjust the brightness. Default is 1.2.
    - Returns:
        - Modified PIL image object with adjusted contrast and brightness.
2. **load_image_from_url(url, target_size=(224, 224), contrast_factor=1.2, brightness_factor=1.2):**
    - This function loads an image from a URL, adjusts its contrast and brightness, resizes it to a target size, and preprocesses it for VGG16 model input.
    - Parameters:
        - `url`: URL of the image to be loaded.
        - `target_size`: Size to which the image should be resized. Default is (224, 224).
        - `contrast_factor`: Factor by which to adjust the contrast. Default is 1.2.
        - `brightness_factor`: Factor by which to adjust the brightness. Default is 1.2.
    - Returns:
        - Preprocessed image array suitable for VGG16 model input.
3. **extract_features_from_urls(image_urls):**
    - This function extracts features from a list of image URLs using the VGG16 model.
    - Parameters:
        - `image_urls`: List of lists containing image URLs.
    - Returns:
        - List of extracted image features.
4. **preprocessing_text_mod(text):**
    - This function preprocesses text data by converting it to lowercase, tokenizing, removing stopwords and punctuation, and performing stemming and lemmatization.
    - Parameters:

        - `text`: Input text data.
    - Returns:
        - Preprocessed text string.
5. **calculate_tf(word_list):**
    - This function calculates the term frequency (TF) of words in a document.
    - Parameters:
        - `word_list`: List of words in the document.
    - Returns:
        - Dictionary containing word frequencies.
6. **calculate_idf(documents):**
    - This function calculates the inverse document frequency (IDF) of words in a collection of documents.
    - Parameters:
        - `documents`: List of lists containing tokenized documents.
    - Returns:
        - Dictionary containing IDF scores for words.
7. **calculate_tf_idf(tf, idf):**
    - This function calculates the TF-IDF score of words based on their term frequency and inverse document frequency.
    - Parameters:
        - `tf`: Dictionary containing term frequency scores.
        - `idf`: Dictionary containing inverse document frequency scores.
    - Returns:
        - Dictionary containing TF-IDF scores.
8. **cosine_similarity_fresh(vec1, vec2):**
    - This function calculates the cosine similarity between two vectors.
    - Parameters:
        - `vec1`: First vector.
        - `vec2`: Second vector.
    - Returns:
        - Cosine similarity score between the two vectors.
9. **calculate_similarity(input_review):**
    - This function calculates the cosine similarity between the input text review and each review in the dataset.
    - Parameters:
        - `input_review`: Input text review.
    - Returns:
        - List of cosine similarity scores between the input review and dataset reviews.
10. **calculate_cosine_similarity(input_features, target_features):**

- This function calculates the cosine similarity between two sets of features.
- Parameters:
  - `input_features`: Features of the input image.
  - `target_features`: Features of the target images.
- Returns:
  - Array of cosine similarity scores.

11. **calculate_image_similarity(user_url):**
    - This function calculates the cosine similarity between the input image and images in the dataset.
    - Parameters:
      - `user_url`: URL of the input image.
    - Returns:
      - List of cosine similarity scores between the input image and dataset images.
12. **takingReview(review_input):**
    - This function calculates the similarity scores for the input text review and adds them to the DataFrame.
13. **takingUrl(url_input):**
    - This function calculates the similarity scores for the input image URL and adds them to the DataFrame.
14. **print_dataframe_info(dataframe):**
    - This function prints information about the DataFrame, including image URL, review text, cosine similarity scores, and composite scores.

These functions collectively perform image and text retrieval tasks by extracting features, calculating similarity scores, and updating the DataFrame with the results.

Results:

The code outputs a ranked list of images Url list and corresponding text review based on their similarity to the input text review and input image Url. It also provides results over composite scores combining image and text similarities.

Below are some screenshots of results:

```
[7] input_txt=input("Enter input review")#"guitar is very"
    input_img_url=input("enter your input url")#"https://images-na.ssl-images-amazon.com/images/I/71Isri9SEaL._SY88.jpg"
```

```
Enter input reviewguitar is very
enter your input urlhttps://images-na.ssl-images-amazon.com/images/I/71Isri9SEaL._SY88.jpg

Image based retrival
Image URL: ['https://images-na.ssl-images-amazon.com/images/I/71Isri9SEaL._SY88.jpg']
Review: Great price and good quality.  It didn't quite match the radius of my sound hole but it was close enough.
Cosine similarity of images - 0.9774
Cosine similarity of text - 0.000
Composite score - 0.489

Image URL: ['https://images-na.ssl-images-amazon.com/images/I/71Isri9SEaL._SY88.jpg']
Review: Worked Great in my kit build.  Had to enlarge the holes a tiny bit as they are ever so slightly larger then the st
Cosine similarity of images - 0.9774
Cosine similarity of text - 0.000
Composite score - 0.489

Image URL: ['https://images-na.ssl-images-amazon.com/images/I/71P0NZnb61L._SY88.jpg', 'https://images-na.ssl-images-amazon
Review: Perfect match up for my Musicman Sterling Ray34 Bass Guitar. Goes with the Abalone Pickguard and Knobs.
Cosine similarity of images - 0.5525
Cosine similarity of text - 0.086
Composite score - 0.319
```

```
Text based retrival
Image URL: ['https://images-na.ssl-images-amazon.com/images/I/7151UsSyCoL._SY88.jpg']
Review: Great as always, have these on most of my guitars
Cosine similarity of images - 0.4163
Cosine similarity of text - 0.326
Composite score - 0.371

Image URL: ['https://images-na.ssl-images-amazon.com/images/I/71sAoJv-OEL._SY88.jpg', 'https://images-na.ssl-images-amazon
Review: We used this one in the middle as seen in the picture holding the white guitar. BIG MISTAKE! The guitar almost was
Cosine similarity of images - 0.2735
Cosine similarity of text - 0.302
Composite score - 0.288

Image URL: ['https://images-na.ssl-images-amazon.com/images/I/61vqoKQ4S7L._SY88.jpg', 'https://images-na.ssl-images-amazon
Review: Works, and my guitar looks like a new one!
Cosine similarity of images - 0.3331
Cosine similarity of text - 0.299
Composite score - 0.316
```

```
print("composite score based retrival")
print_dataframe_info(df_sorted_on_composite)
```

```
composite score based retrival
Image URL: ['https://images-na.ssl-images-amazon.com/images/I/71Isri9SEaL._SY88.jpg']
Review: Worked Great in my kit build.  Had to enlarge the holes a tiny bit as they are ever so slightly larger then the st
Cosine similarity of images - 0.9774
Cosine similarity of text - 0.000
Composite score - 0.489

Image URL: ['https://images-na.ssl-images-amazon.com/images/I/71Isri9SEaL._SY88.jpg']
Review: Great price and good quality.  It didn't quite match the radius of my sound hole but it was close enough.
Cosine similarity of images - 0.9774
Cosine similarity of text - 0.000
Composite score - 0.489

Image URL: ['https://images-na.ssl-images-amazon.com/images/I/7151UsSyCoL._SY88.jpg']
Review: Great as always, have these on most of my guitars
Cosine similarity of images - 0.4163
Cosine similarity of text - 0.326
Composite score - 0.371
```

Conclusion:

The retrieval system shows promising results in finding relevant images and text based on user input. Further optimization and evaluation could enhance its performance and applicability in real-world scenarios.