Q1. Image Feature Extraction

→ We have increased the dataset by performing various pre-processing techniques as follows

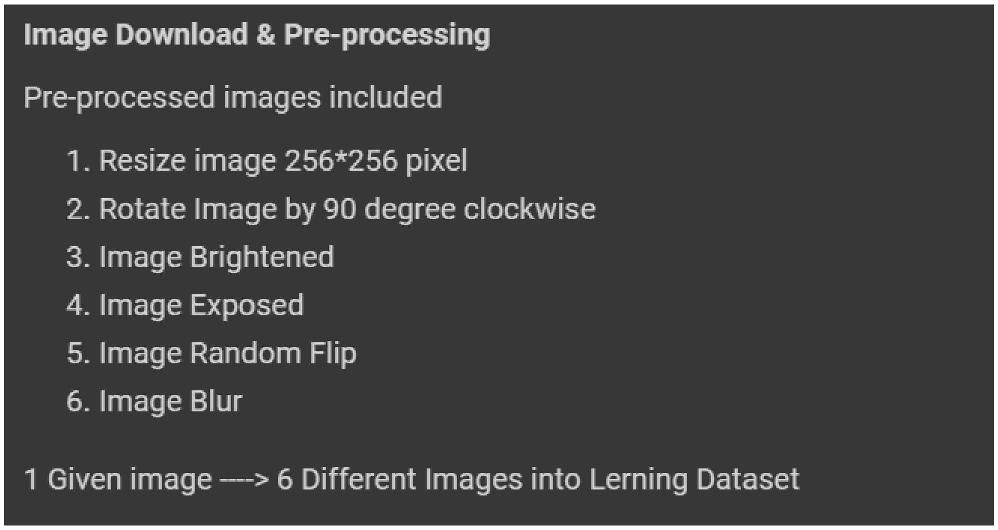


Image Download & Pre-processing

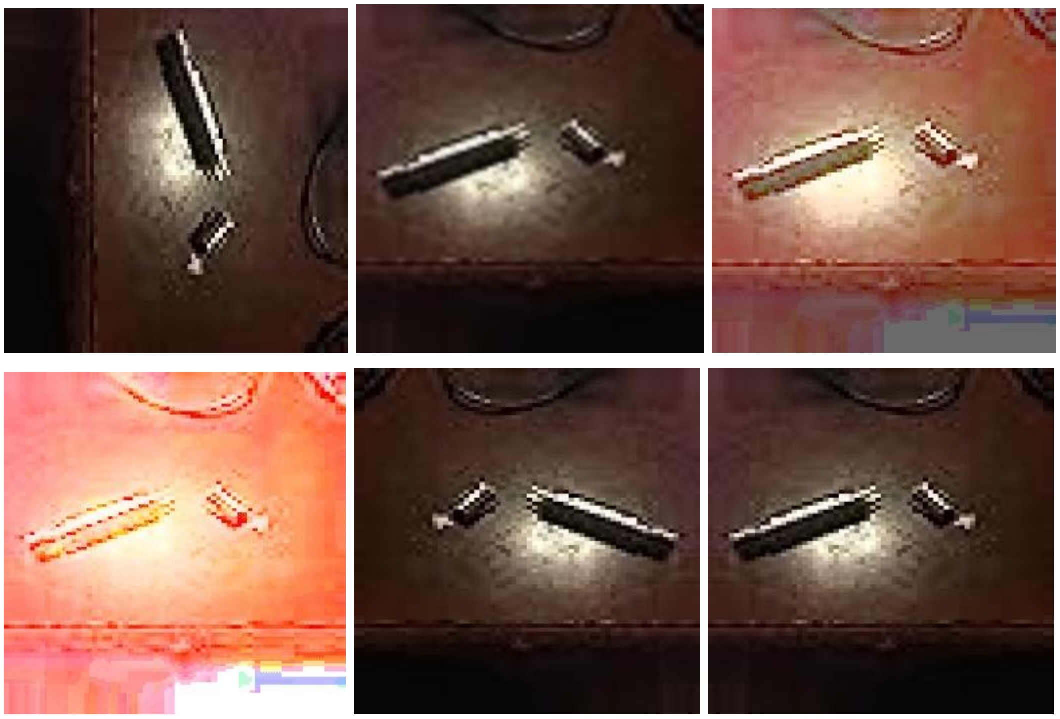
Pre-processed images included

1. Resize image 256*256 pixel
2. Rotate Image by 90 degree clockwise
3. Image Brightened
4. Image Exposed
5. Image Random Flip
6. Image Blur

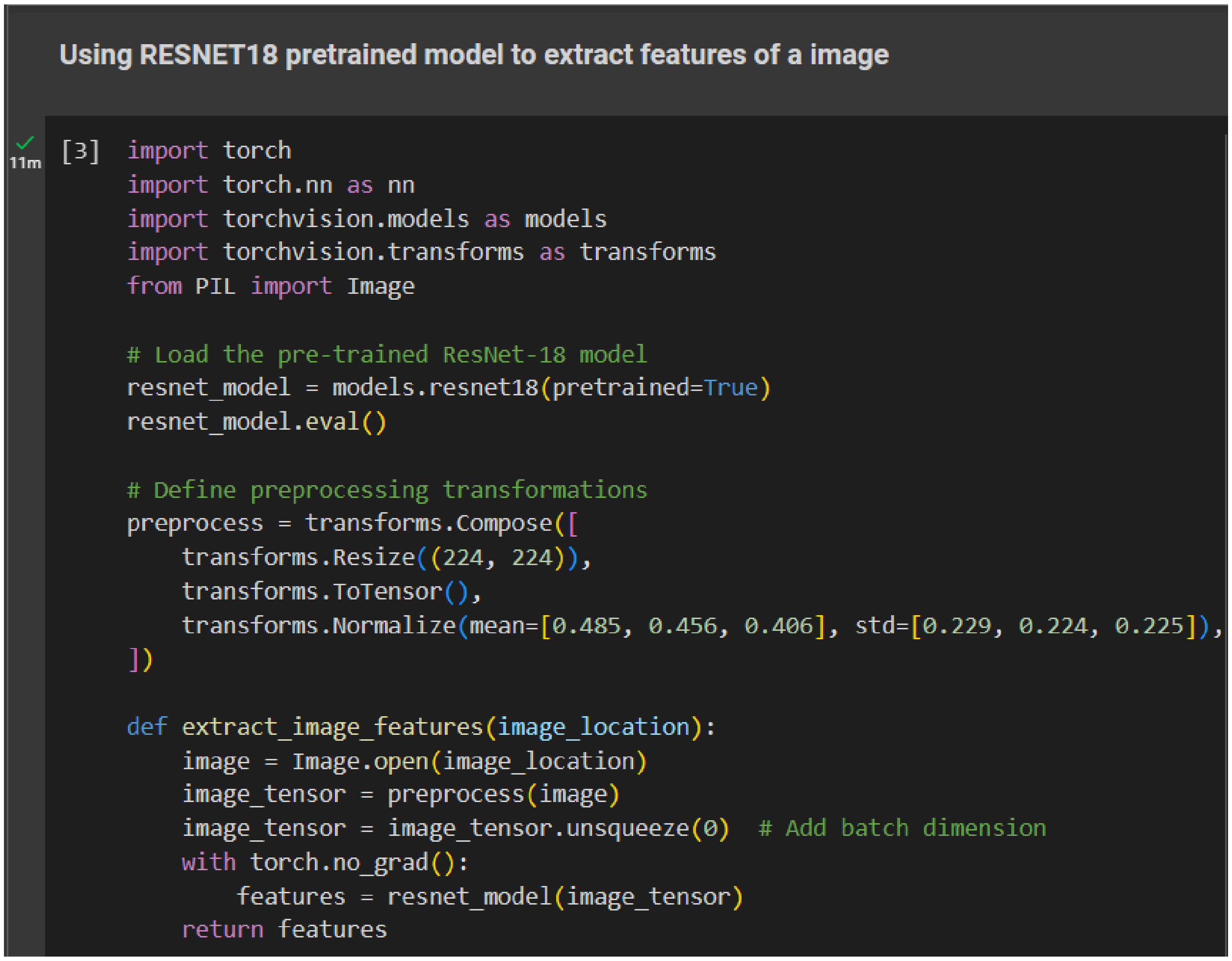
1 Given image ----> 6 Different Images into Lerning Dataset

Given Image:

Processed Images:



I have used the RESNET18 pre-trained model to extract features of an image & maintain in Python dictionary format where the key is image_name/ image_path, and the value is the matrix given by the model after extracting that image features.



Using RESNET18 pretrained model to extract features of a image

```
[3]  import torch
     import torch.nn as nn
     import torchvision.models as models
     import torchvision.transforms as transforms
     from PIL import Image

     # Load the pre-trained ResNet-18 model
     resnet_model = models.resnet18(pretrained=True)
     resnet_model.eval()

     # Define preprocessing transformations
     preprocess = transforms.Compose([
         transforms.Resize((224, 224)),
         transforms.ToTensor(),
         transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
     ])

     def extract_image_features(image_location):
         image = Image.open(image_location)
         image_tensor = preprocess(image)
         image_tensor = image_tensor.unsqueeze(0)  # Add batch dimension
         with torch.no_grad():
             features = resnet_model(image_tensor)
         return features
```

After Feature extraction → Normalization of the features corresponding to each image, and saved pickle file

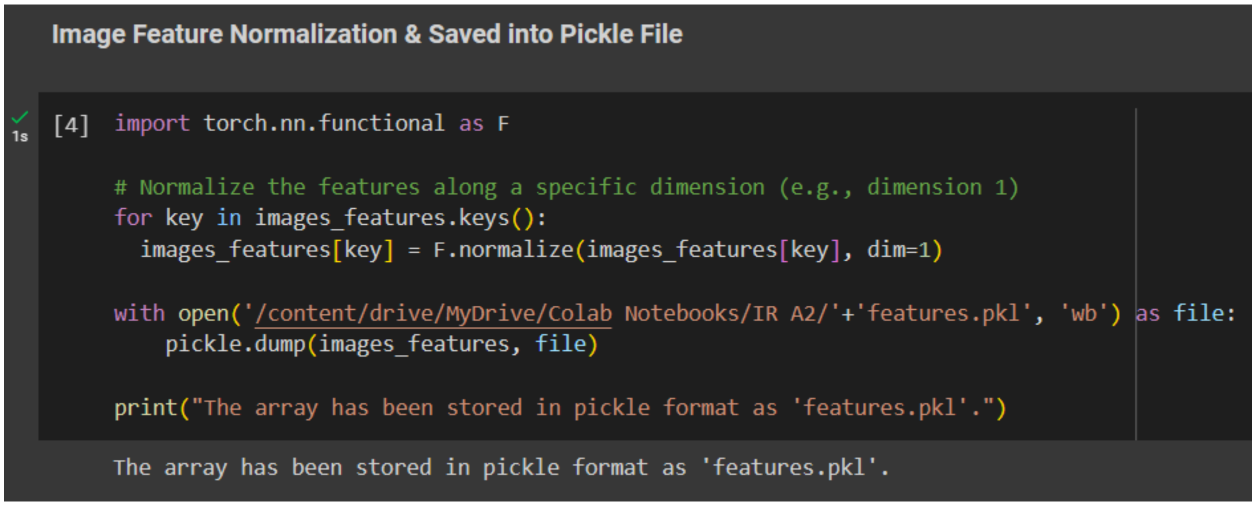


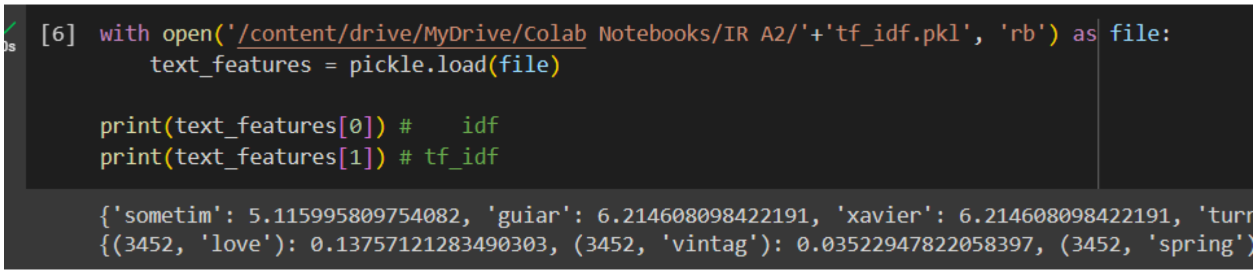
Image Feature Normalization & Saved into Pickle File

```
[4]  import torch.nn.functional as F

     # Normalize the features along a specific dimension (e.g., dimension 1)
     for key in images_features.keys():
       images_features[key] = F.normalize(images_features[key], dim=1)

     with open('/content/drive/MyDrive/Colab Notebooks/IR A2/'+'features.pkl', 'wb') as file:
         pickle.dump(images_features, file)

     print("The array has been stored in pickle format as 'features.pkl'.")
```

```
The array has been stored in pickle format as 'features.pkl'.
```

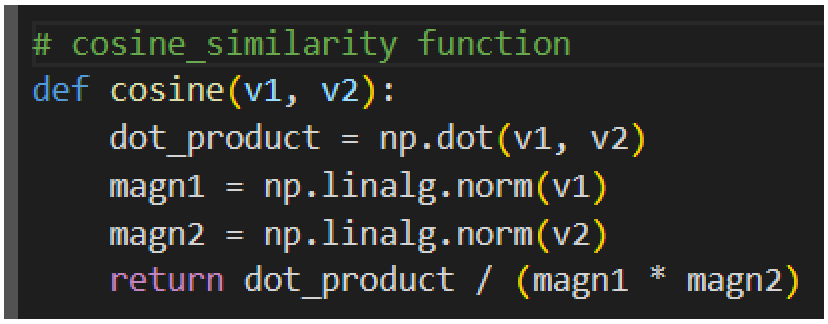
Q2. Text TF_IDF

1) Preprossing text Pipeline
2) Calculate the Term Frequency of each word of each document of the corpus
3) Calculate the Inverse Document Frequency of each unique word in the corpus
4) Calculated TF_IDF score for each word of each document of the corpus
5) Saved that TF_IDF score and IDF score in pickle file.



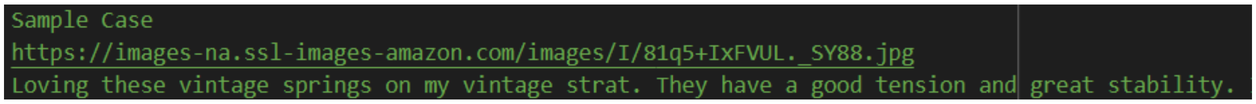
```
[6] with open('/content/drive/MyDrive/Colab Notebooks/IR A2/'+'tf_idf.pkl', 'rb') as file:
        text_features = pickle.load(file)

    print(text_features[0]) #    idf
    print(text_features[1]) # tf_idf

    {'sometim': 5.115995809754082, 'guiar': 6.214608098422191, 'xavier': 6.214608098422191, 'turn
    {(3452, 'love'): 0.13757121283490303, (3452, 'vintag'): 0.03522947822058397, (3452, 'spring')
```

Q3. Image Retrieval and Text Retrieval



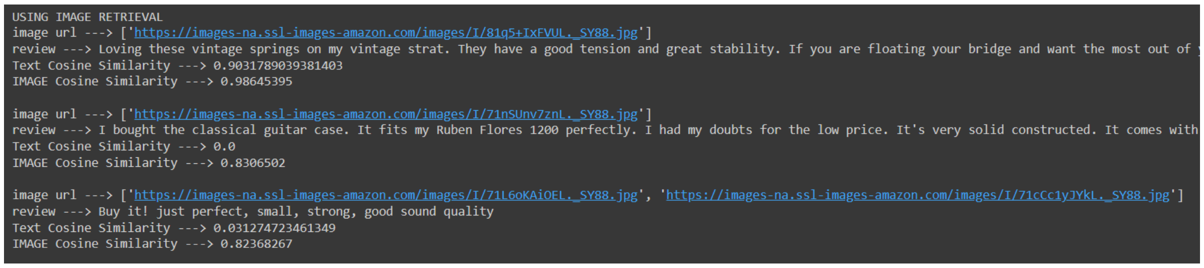
```
# cosine_similarity function
def cosine(v1, v2):
    dot_product = np.dot(v1, v2)
    magn1 = np.linalg.norm(v1)
    magn2 = np.linalg.norm(v2)
    return dot_product / (magn1 * magn2)
```

Query: Image_link & Review(Text)



```
Sample Case
https://images-na.ssl-images-amazon.com/images/I/81q5+IxFVUL._SY88.jpg
Loving these vintage springs on my vintage strat. They have a good tension and great stability.
```

a) Image:
   i) Extract the feature of query_image through the same model pipeline.
   ii) for each Dataset's Image feature, we compare we query_image_feature
   iii) Sort the by Cosine Similarity Score.

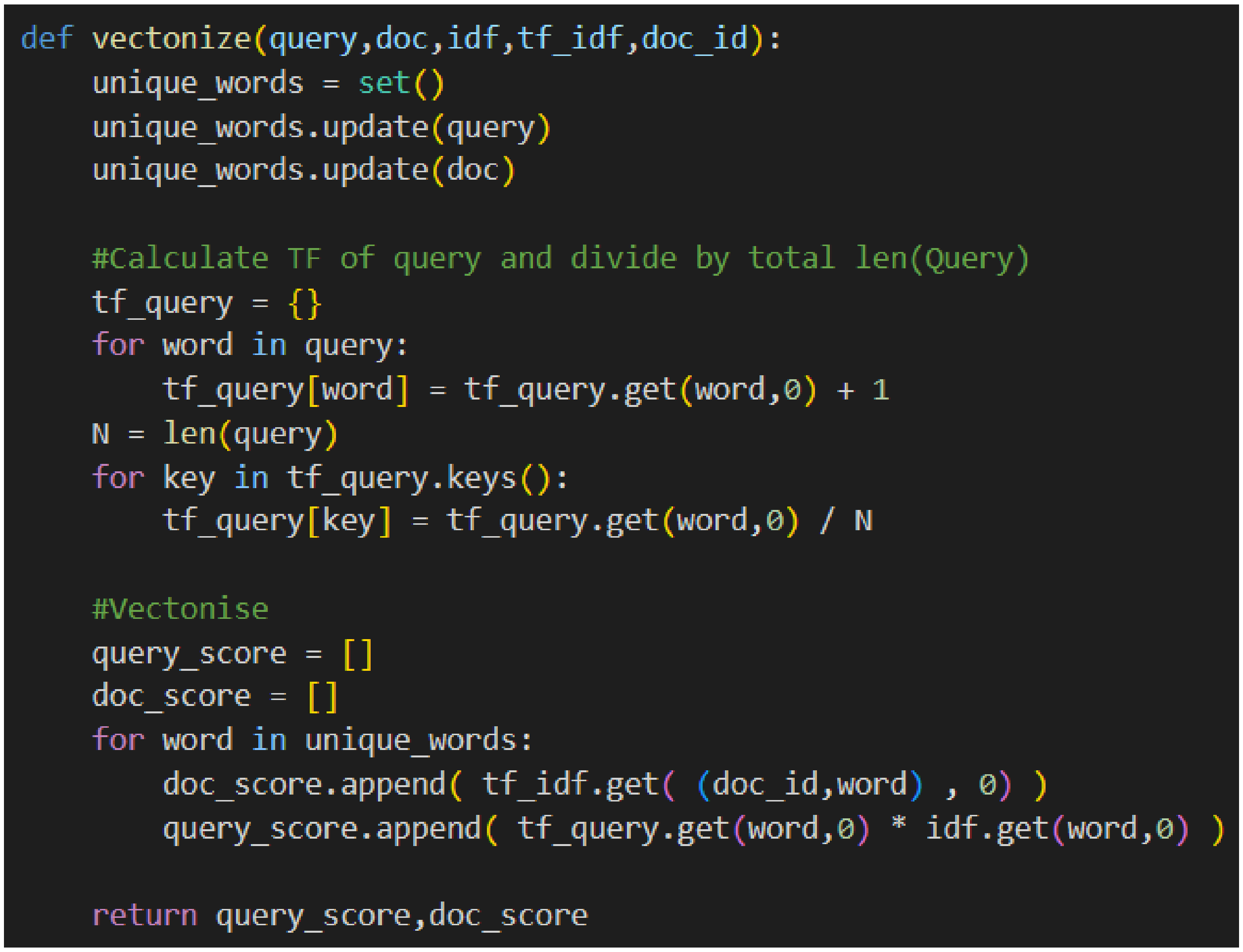


```
USING IMAGE RETRIEVAL
image url ---> ['https://images-na.ssl-images-amazon.com/images/I/81q5+IxFVUL._SY88.jpg']
review ---> Loving these vintage springs on my vintage strat. They have a good tension and great stability. If you are floating your bridge and want the most out of
Text Cosine Similarity ---> 0.9031789039381403
IMAGE Cosine Similarity ---> 0.98645395

image url ---> ['https://images-na.ssl-images-amazon.com/images/I/71nSUnv7znL._SY88.jpg']
review ---> I bought the classical guitar case. It fits my Ruben Flores 1200 perfectly. I had my doubts for the low price. It's very solid constructed. It comes with
Text Cosine Similarity ---> 0.0
IMAGE Cosine Similarity ---> 0.8306502

image url ---> ['https://images-na.ssl-images-amazon.com/images/I/71L6oKAiOEL._SY88.jpg', 'https://images-na.ssl-images-amazon.com/images/I/71cCc1yJYkL._SY88.jpg']
review ---> Buy it! just perfect, small, strong, good sound quality
Text Cosine Similarity ---> 0.031274723461349
IMAGE Cosine Similarity ---> 0.82368267
```

b) Text:

i) Preproceed the query_review_text through the same preprocessing model.

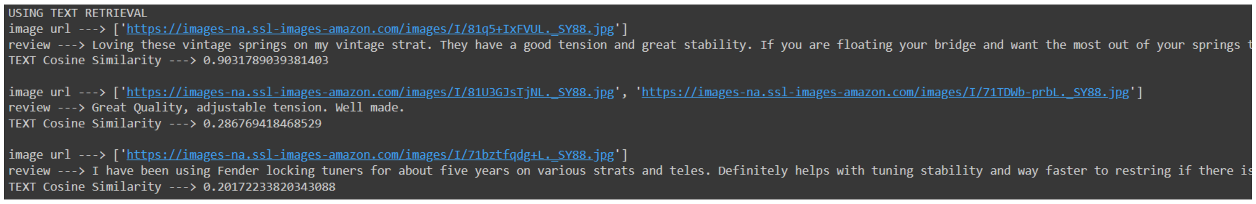
ii) Now, for each Dataset's review_text, first convert them into 2 vectors of their TF_IDF scores



```python
def vectonize(query,doc,idf,tf_idf,doc_id):
    unique_words = set()
    unique_words.update(query)
    unique_words.update(doc)

    #Calculate TF of query and divide by total len(Query)
    tf_query = {}
    for word in query:
        tf_query[word] = tf_query.get(word,0) + 1
    N = len(query)
    for key in tf_query.keys():
        tf_query[key] = tf_query.get(word,0) / N

    #Vectonise
    query_score = []
    doc_score = []
    for word in unique_words:
        doc_score.append( tf_idf.get( (doc_id,word) , 0) )
        query_score.append( tf_query.get(word,0) * idf.get(word,0) )

    return query_score,doc_score
```

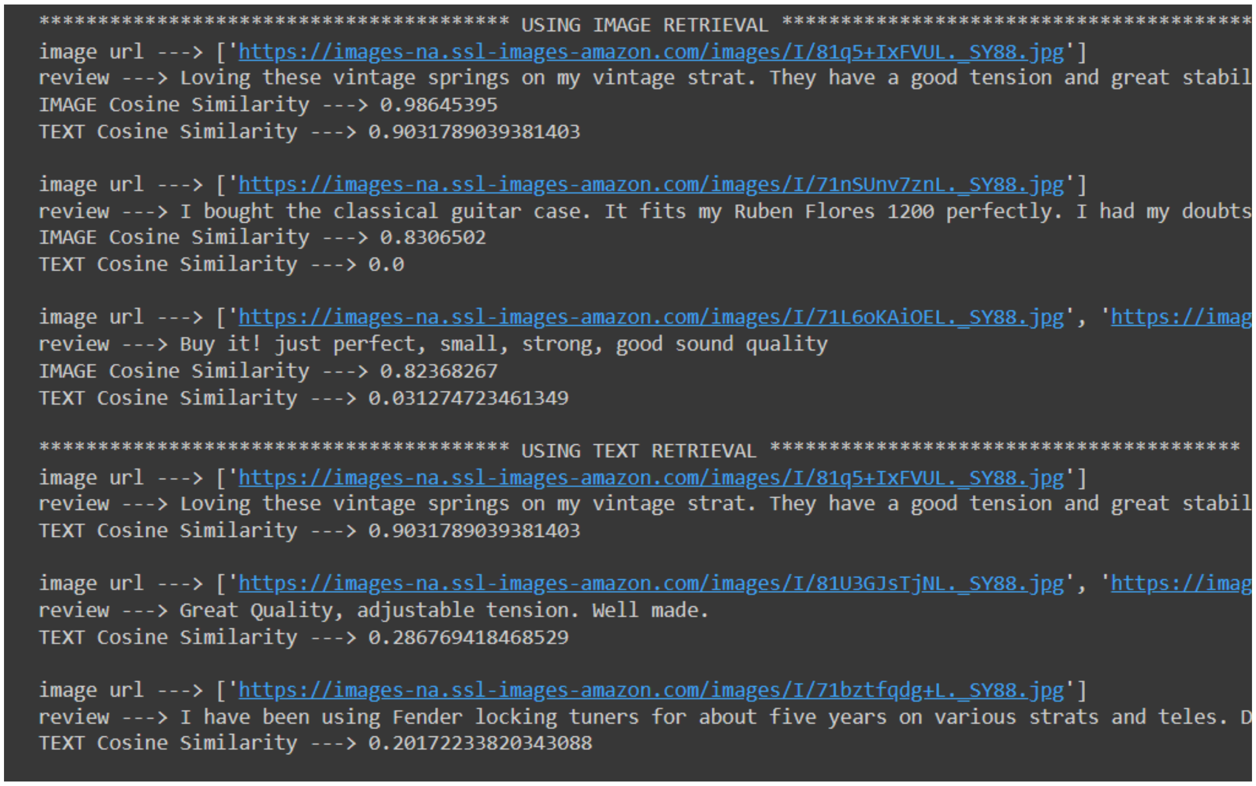
iii) Not get a Cosine Similarity score for each pair (Query_Text_Review, Doc[i]_Review)

iv) Sort the by Cosine Similarity Score.



```
USING TEXT RETRIEVAL
image url ---> ['https://images-na.ssl-images-amazon.com/images/I/81q5+IxFVUL._SY88.jpg']
review ---> Loving these vintage springs on my vintage strat. They have a good tension and great stability. If you are floating your bridge and want the most out of your springs t
TEXT Cosine Similarity ---> 0.9031789039381403

image url ---> ['https://images-na.ssl-images-amazon.com/images/I/81U3GJsTjNL._SY88.jpg', 'https://images-na.ssl-images-amazon.com/images/I/71TDWb-prbL._SY88.jpg']
review ---> Great Quality, adjustable tension. Well made.
TEXT Cosine Similarity ---> 0.286769418468529

image url ---> ['https://images-na.ssl-images-amazon.com/images/I/71bztfqdg+L._SY88.jpg']
review ---> I have been using Fender locking tuners for about five years on various strats and teles. Definitely helps with tuning stability and way faster to restring if there is
TEXT Cosine Similarity ---> 0.20172233820343088
```

Final Output for Q3)



```
**************************************** USING IMAGE RETRIEVAL ****************************************
image url ---> ['https://images-na.ssl-images-amazon.com/images/I/81q5+IxFVUL._SY88.jpg']
review ---> Loving these vintage springs on my vintage strat. They have a good tension and great stabil
IMAGE Cosine Similarity ---> 0.98645395
TEXT Cosine Similarity ---> 0.9031789039381403

image url ---> ['https://images-na.ssl-images-amazon.com/images/I/71nSUnv7znL._SY88.jpg']
review ---> I bought the classical guitar case. It fits my Ruben Flores 1200 perfectly. I had my doubts
IMAGE Cosine Similarity ---> 0.8306502
TEXT Cosine Similarity ---> 0.0

image url ---> ['https://images-na.ssl-images-amazon.com/images/I/71L6oKAiOEL._SY88.jpg', 'https://imag
review ---> Buy it! just perfect, small, strong, good sound quality
IMAGE Cosine Similarity ---> 0.82368267
TEXT Cosine Similarity ---> 0.031274723461349

**************************************** USING TEXT RETRIEVAL ****************************************
image url ---> ['https://images-na.ssl-images-amazon.com/images/I/81q5+IxFVUL._SY88.jpg']
review ---> Loving these vintage springs on my vintage strat. They have a good tension and great stabil
TEXT Cosine Similarity ---> 0.9031789039381403

image url ---> ['https://images-na.ssl-images-amazon.com/images/I/81U3GJsTjNL._SY88.jpg', 'https://imag
review ---> Great Quality, adjustable tension. Well made.
TEXT Cosine Similarity ---> 0.286769418468529

image url ---> ['https://images-na.ssl-images-amazon.com/images/I/71bztfqdg+L._SY88.jpg']
review ---> I have been using Fender locking tuners for about five years on various strats and teles. D
TEXT Cosine Similarity ---> 0.20172233820343088
```

Q4. Combined Retrieval (Text and Image)

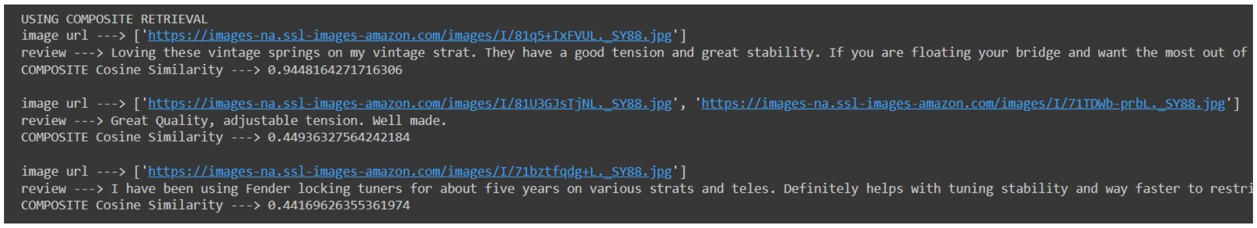
As we have Cosine Similarity for Images Individually and for Text individually from Q3) , Now we just have to combine them (average them)
Such that given a row in A2_Data.csv
For,

One Row/Entry → {Images: (abc.jpg, xyz.jpg), review: "------"}

We have extracted

Combined_Cosine = (*Max{*Cosine(abc.jpg) , Cosine(xyz.jpg)*}* + Cosine(review)) / 2

Now, perform this operation for each row/entry. And sort it



```
USING COMPOSITE RETRIEVAL
image url ---> ['https://images-na.ssl-images-amazon.com/images/I/81q5+IxFVUL._SY88.jpg']
review ---> Loving these vintage springs on my vintage strat. They have a good tension and great stability. If you are floating your bridge and want the most out of
COMPOSITE Cosine Similarity ---> 0.9448164271716306

image url ---> ['https://images-na.ssl-images-amazon.com/images/I/81U3GJsTjNL._SY88.jpg', 'https://images-na.ssl-images-amazon.com/images/I/71TDWb-prbL._SY88.jpg']
review ---> Great Quality, adjustable tension. Well made.
COMPOSITE Cosine Similarity ---> 0.44936327564242184

image url ---> ['https://images-na.ssl-images-amazon.com/images/I/71bztfqdg+L._SY88.jpg']
review ---> I have been using Fender locking tuners for about five years on various strats and teles. Definitely helps with tuning stability and way faster to restri
COMPOSITE Cosine Similarity ---> 0.44169626355361974
```

Q5. Results and Analysis

a. Present the top-ranked (image, review) pairs and the cosine similarity scores.

→

RANK ***1***: **IMAGE** RETRIEVAL
RANK ***2***: COMBINED (**IMAGE + TEXT**) RETRIEVAL
RANK ***3***: **TEXT** RETRIEVAL

b. Observe which of the two retrieval techniques gives a better similarity score and argue why.

→

The Image Technique provides the most efficient result as it maps 1-1 of 2 image features extracted from the pre-trained model and finds cosine similarity,
whereas the TF-IDF score is over corpus and cosine similarity of text's TF-IDF score has been influenced by Query_text, Current_Doc_text, and also by other documents of the corpus (IDF Score)

E.g. doc_id = 3452

IMAGE Cosine Similarity → 0.98645395
COMPOSITE Cosine Similarity → 0.9448164271716306
TEXT Cosine Similarity → 0.9031789039381403

Thus, even if we have an exact match, we can't get cosine similarity ~1 in Text retrieval.

c. Discuss the challenges faced and potential improvements in the retrieval process.

→

Challenges faced during retrieval:

- Semantic Gap: The gap between low-level features (e.g., image pixels) and high-level semantics (e.g., user intent) affects retrieval accuracy.
- Data Variability: Variations in lighting, angles, and image quality impact feature extraction.
- Feature Extraction: Efficient and robust feature extraction from images and text is crucial.

Potential improvements:

- Fine-tuning Models: Continuously train models on diverse data to improve feature extraction.
- Hybrid Approaches: Combine image and text features more effectively. Semantic Embeddings: Use embeddings that bridge the semantic gap.
- User Feedback: Incorporate user feedback to refine retrieval results.