CHAPTER - I

INTRODUCTION

The emergence of E-Commerce has transformed the way consumers engage with products, making visually appealing recommendations crucial. With a special focus on visual similarity in product photos, this project offers a recommendation system based on visual similarity. The system uses deep learning and Python to recognise features in images and generate personalised product recommendations.

This method concentrates on visual content and goes beyond conventional collaborative filtering. The principal aim is to develop and implement a recommendation system that leverages Python programming and deep learning methodologies to improve the precision and pertinence of suggestions. In the quickly changing world of E-Commerce, the system seeks to improve the overall user experience by offering users visually solid and personalised suggestions. The scope includes every stage of a visual similarity-based recommendation system's development.

Important stages of the project include pre-processing the data, extracting visual features, training the model, and conducting a thorough evaluation. The recommendation model is a content-based system that provides consumers with individualised and appealing product recommendations. It emphasises visual similarity. Cosine similarity metrics facilitate evaluation, providing insight into how well the system recognises and recommends visually comparable products.

1.1 Methodology

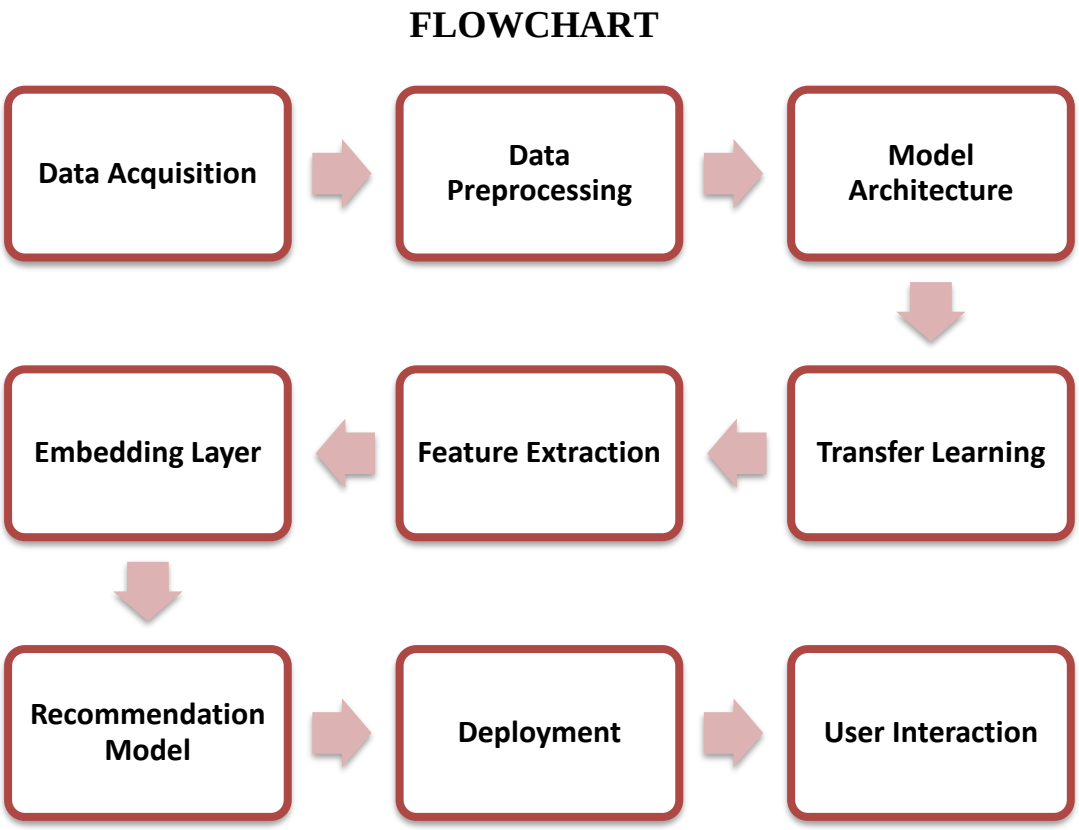

Fig. 1 Workflow of Methodology Used

CHAPTER-II

METHODOLOGY

2.1 PYTHON:

Python is a **versatile, high-level programming language** known for its simplicity, readability, and extensive ecosystem. Created by **Guido van Rossum** in the late 1980s, Python has gained widespread popularity across various domains, making it one of the most widely used programming languages. Python's syntax emphasizes code readability, using indentation for block structure. Its interpreted nature allows for interactive development and rapid prototyping. Python excels in data science and machine learning, supported by libraries such as **NumPy, Pandas, Matplotlib, and TensorFlow**, making it a go-to language for researchers and analysts. Python's use cases extend to **automation, scripting, scientific computing, game development, and desktop applications**.

Pandas for Data Handling:

- Pandas is used for data manipulation and analysis. It is employed to load and manipulate the fashion dataset.

TensorFlow and Keras for Deep Learning:

- TensorFlow and Keras are used to build and train a deep-learning model for fashion image embeddings.

Scikit-learn for Machine Learning:

- Scikit-learn is used for calculating cosine similarity and pairwise distances.

Matplotlib and Plotly for Data Visualization:

- Matplotlib and Plotly are used for data visualization, such as displaying reference and recommended images.

OpenCV for Image Processing:

- OpenCV is used for image loading, resizing, and colour conversion.

NumPy and SciPy for Numerical Computing:

- NumPy and SciPy are used for numerical computing tasks, including handling arrays and calculating cosine similarity.

Streamlit for Web Application:

- Streamlit is used to create the web application interface for the Fashion Recommender System. It simplifies the process of turning data scripts into shareable web apps.

2.2 DATA ACQUISITION:

For the Visual Recommendation project, data acquisition involved obtaining the Fashion Product Images Dataset from Kaggle. The dataset, curated by Param Aggarwal, contains a diverse collection of fashion product images along with corresponding metadata. The dataset was downloaded directly within the Colab environment using the Kaggle API. The primary dataset file, "styles.csv," was utilized to extract information about fashion products, such as product IDs and category labels. This structured data, combined with the actual product images, formed the basis for training the recommendation model. The dataset's richness and variety contribute to the model's ability to understand and recommend diverse fashion products.

2.3 DATA PRE-PROCESSING:

2.3.1 CSV File:

DATA PRE-PROCESSING OF CSV DATA

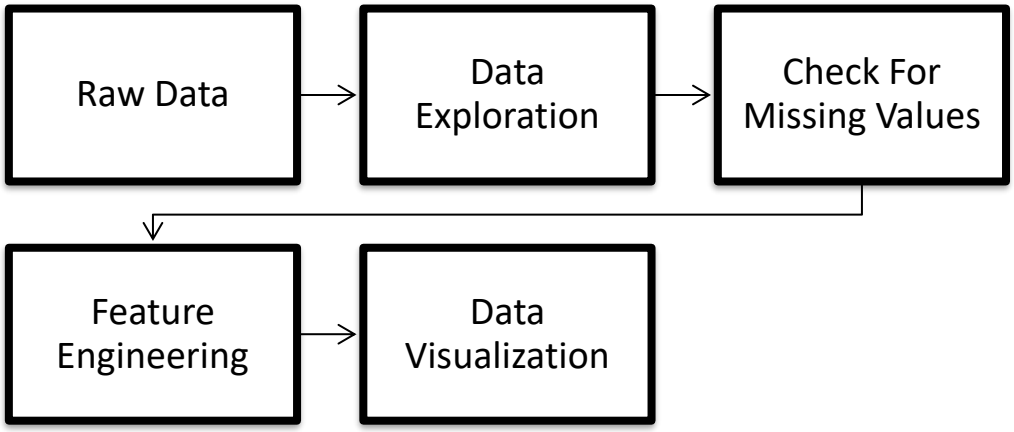

Fig 2 Work Flow of Pre-Processing

Data Exploration:

The loaded data frame is inspected to get an overview of the dataset's structure. This includes checking the first few rows using **df. head ()** to understand the column names and data types. Basic statistics, such as the number of rows and columns, are obtained using **df. Shape** function. Information about missing values, data types, and memory usage is obtained using the **df. info ()** function.

```
df = pd.read_csv("/content/fashion-dataset/styles.csv", nrows=7000, on_bad_lines='skip')
df.head()
```

	id	gender	masterCategory	subCategory	articleType	baseColour	season	year	usage	productDisplayName
0	15970	Men	Apparel	Topwear	Shirts	Navy Blue	Fall	2011	Casual	Turtle Check Men Navy Blue Shirt
1	39386	Men	Apparel	Bottomwear	Jeans	Blue	Summer	2012	Casual	Peter England Men Party Blue Jeans
2	59263	Women	Accessories	Watches	Watches	Silver	Winter	2016	Casual	Titan Women Silver Watch
3	21379	Men	Apparel	Bottomwear	Track Pants	Black	Fall	2011	Casual	Manchester United Men Solid Black Track Pants
4	53759	Men	Apparel	Topwear	Tshirts	Grey	Summer	2012	Casual	Puma Men Grey T-shirt

Fig. 3 First 5 rows in a data frame

Check for Missing Values:

Check for missing values in the dataset using **df.isnull().sum()** function. Depending on the nature and extent of missing values, choose an appropriate strategy. This may involve dropping rows with missing values or imputing values based on statistical measures.

Feature Engineering:

Extract relevant features from existing columns or create new features that might be useful for the recommendation model.

```
df['image'] = df.apply(lambda row: str(row['id']) + ".jpg", axis=1)
df = df.reset_index(drop=True)
df.head(10)
```

	id	gender	masterCategory	subCategory	articleType	baseColour	season	year	usage	productDisplayName	image
0	15970	Men	Apparel	Topwear	Shirts	Navy Blue	Fall	2011	Casual	Turtle Check Men Navy Blue Shirt	15970.jpg
1	39386	Men	Apparel	Bottomwear	Jeans	Blue	Summer	2012	Casual	Peter England Men Party Blue Jeans	39386.jpg
2	59263	Women	Accessories	Watches	Watches	Silver	Winter	2016	Casual	Titan Women Silver Watch	59263.jpg
3	21379	Men	Apparel	Bottomwear	Track Pants	Black	Fall	2011	Casual	Manchester United Men Solid Black Track Pants	21379.jpg
4	53759	Men	Apparel	Topwear	Tshirts	Grey	Summer	2012	Casual	Puma Men Grey T-shirt	53759.jpg
5	1855	Men	Apparel	Topwear	Tshirts	Grey	Summer	2011	Casual	Inkfruit Mens Chain Reaction T-shirt	1855.jpg
6	30805	Men	Apparel	Topwear	Shirts	Green	Summer	2012	Ethnic	Fabindia Men Striped Green Shirt	30805.jpg
7	26960	Women	Apparel	Topwear	Shirts	Purple	Summer	2012	Casual	Jealous 21 Women Purple Shirt	26960.jpg
8	29114	Men	Accessories	Socks	Socks	Navy Blue	Summer	2012	Casual	Puma Men Pack of 3 Socks	29114.jpg
9	30039	Men	Accessories	Watches	Watches	Black	Winter	2016	Casual	Skagen Men Black Watch	30039.jpg

Fig. 4 Creating a New Column from an Existing Column

Data Visualization – Target Columns:

Key visualizations include **bar charts** illustrating the distribution of products across master categories, sub-categories, usage categories, and gender. These charts offer insights into the dataset's composition and can aid in identifying potential biases or imbalances.

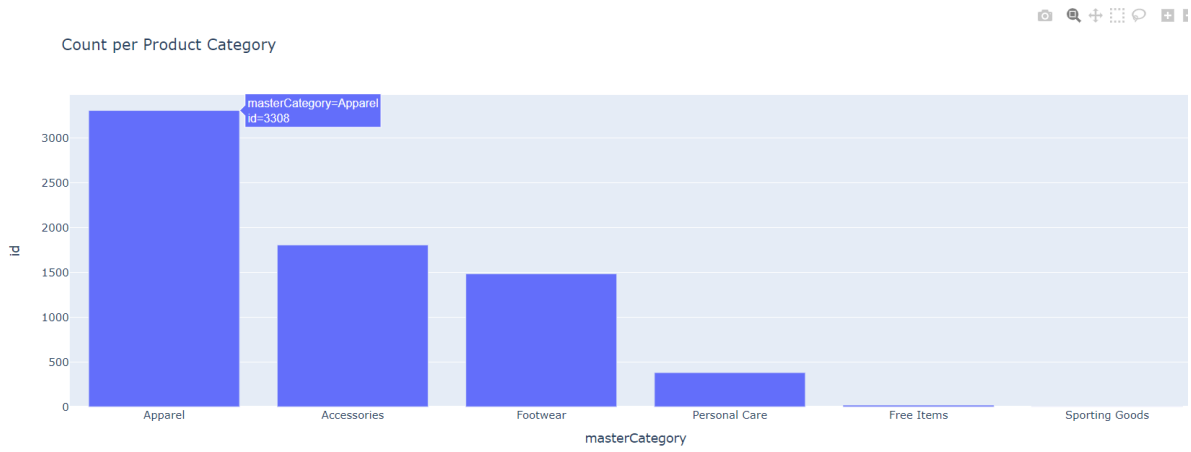

Fig. 5 Bar plot for Product Category

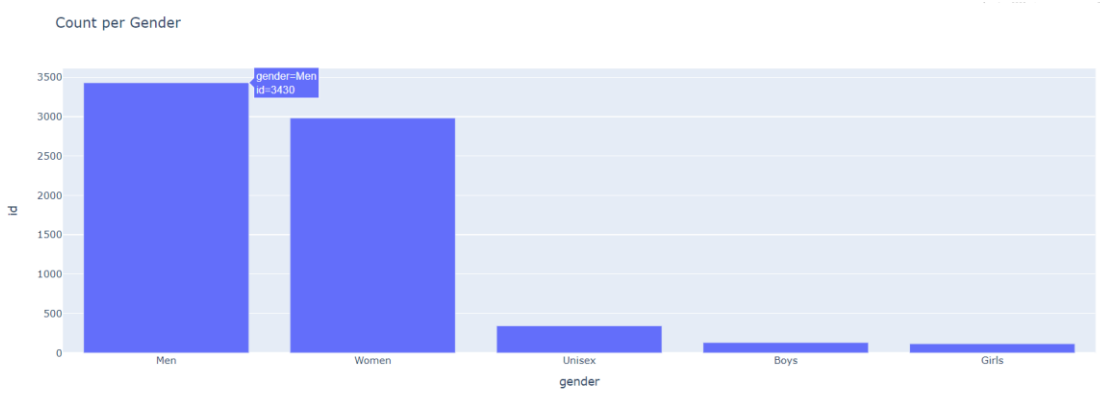

Fig. 6 Bar plot for Gender

2.3.2 IMAGE DATA:

Image pre-processing is a crucial step in many Computer vision and image analysis projects. It involves various techniques to enhance the quality of the input images and make them suitable for further analysis or machine learning tasks. Images are loaded using OpenCV's **cv2.imread** function from their file paths. The **load_image** function includes resizing to ensure uniformity in image dimensions. The resizing factor is applied to control the size of the images. A mechanism is implemented to detect and skip duplicate images before processing. This ensures that identical images are processed only once, reducing redundancy in the embedding generation.

Fig. 7 Pre-Processed Images

2.4 MODEL ARCHITECTURE:

The ResNet50 model is loaded with weights pre-trained on the ImageNet dataset. The weights of the pre-trained layers in the ResNet50 model are frozen to prevent them from being updated during training. This is done to retain the knowledge learned from the ImageNet dataset. A Global Max Pooling 2D layer is added on top of the ResNet50 model. This layer reduces the spatial dimensions of the output and captures the most important features from each feature map. The resulting architecture is a sequential model where the ResNet50 base model serves as a feature extractor, and the GlobalMaxPooling2D layer aggregates features for each image. The overall model architecture leverages the powerful feature extraction capabilities of the ResNet50 model while adapting it to the specific task of fashion product recommendation by using global max pooling. The resulting model can efficiently generate embeddings for fashion product images, which are then used for similarity comparison and recommendation.

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50 (Functional)       (None, 7, 7, 2048)        23587712

 global_max_pooling2d_1 (Glo  (None, 2048)             0
 balMaxPooling2D)

=================================================================
Total params: 23,587,712
Trainable params: 0
Non-trainable params: 23,587,712
_________________________________________________________________
```

Fig. 8 Summary of Model Architecture

2.5 TRANSFER LEARNING:

Transfer learning involves taking advantage of the pre-trained ResNet50 model's knowledge of general image features and adapting it to the fashion recommendation task by adding custom layers and, optionally, fine-tuning specific parts of the model.

2.6 FEATURE EXTRACTION:

Feature extraction is a crucial step that involves obtaining numerical representations (features) from the input images using a pre-trained ResNet50 model. A Global Max Pooling 2D layer is added on top of the pre-trained ResNet50 model. This layer aggregates the most important features from each feature map, reducing the spatial dimensions. The result is a model that takes images as input and outputs a fixed-size feature vector, capturing the essential characteristics of each image. The feature extraction process involves leveraging the ResNet50 model's ability to capture hierarchical features from images and customizing it for the specific task of fashion recommendation by adding a global max pooling layer. The resulting features are used to calculate cosine similarity for image recommendations

CHAPTER - III

VISUAL RECOMMENDATION SYSTEM

A visual recommendation system is a type of Content-based recommendation system that suggests items (Images) to users based on the content or features of the items themselves. It relies on the inherent characteristics and visual properties of the items rather than user behaviour or preferences.

3.1 SIMILARITY METRICS:

In visual-based recommendation systems, where the recommendations are made based on visual content such as images, the choice of similarity metric is crucial for effective performance.

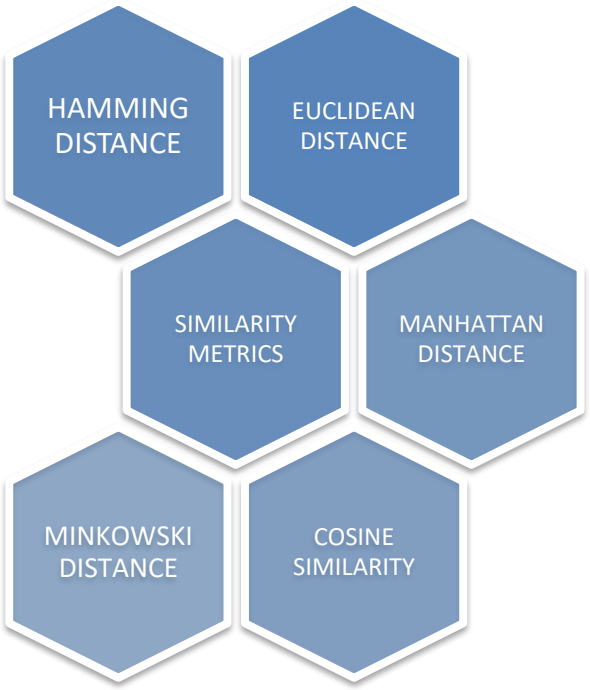

Fig. 9 Similarity Metrics

The choice of similarity metrics often depends on the nature of your data and the specific characteristics of your recommendation system. However, two commonly used similarity metrics suitable for Visual Recommendation systems are **Cosine similarity** and **Euclidean Distance.**

3.2 COSINE SIMILARITY:

Cosine similarity is a metric used to measure the similarity between two vectors, regardless of their magnitudes. It is widely used in various applications, including natural language processing, information retrieval, and content-based recommendation systems. Cosine

similarity measures the cosine of the angle between the vectors. The closer the cosine similarity is to 1, the more similar the vectors are; the closer it is to -1, the more dissimilar they are. In content-based recommendation systems, cosine similarity is often used to compare the feature vectors (embeddings) of items. Cosine similarity is employed to measure the similarity between fashion product images based on their ResNet50-derived embeddings.

```
# Convert df_embs to a sparse matrix
sparse_embs = csr_matrix(df_embs.values)

# Calculate Cosine Similarity
cosine_sim = cosine_similarity(sparse_embs)

# The resulting cosine_sim matrix contains cosine similarity scores.
cosine_sim[:4, :4]
```

```
array([[1.0000006 , 0.71280503, 0.45375037, 0.6793725 ],
       [0.71280503, 1.0000006 , 0.4151534 , 0.83330894],
       [0.45375037, 0.4151534 , 0.99999934, 0.44629493],
       [0.6793725 , 0.83330894, 0.44629493, 0.99999976]], dtype=float32)
```

Fig. 10 Cosine Similarity Metrics

3.3 EUCLIDEAN DISTANCE:

The Euclidean distance between two points (vectors) in an Euclidean space is a measure of the straight-line distance between those two points. It is the most common method of measuring the distance between two points in a Cartesian coordinate system. In the context of a recommendation system with embeddings or feature vectors representing items or products, the Euclidean distance is often used to measure the similarity between items. Items with smaller Euclidean distances are considered more similar to each other. In a product recommendation system, Euclidean distance can be used to measure the similarity between items or products. The lower the Euclidean distance between two items, the more similar they are.

```
#USING EUCLIDEAN DISTANCE

from sklearn.metrics.pairwise import euclidean_distances

# Calculate Euclidean Distance
euclidean_dist = euclidean_distances(sparse_embs)

# The resulting euclidean_dist matrix contains Euclidean distance scores.
euclidean_dist[:4, :4]
```

```
array([[  0.     , 204.3869 , 297.21036, 216.76057],
       [204.3869 ,   0.     , 311.15686, 158.48582],
       [297.21036, 311.15686,   0.     , 303.69812],
       [216.76057, 158.48582, 303.69812,   0.     ]], dtype=float32)
```

Fig. 11 Euclidean Distance metrics

3.4 INDICES MAPPING:

"Indices mapping" typically refers to the process or concept of associating elements with their corresponding indices in a data structure. In programming and data manipulation, indices are often used to identify and access uniquely elements within an array, list, or any other indexed data structure.

An "Indices Mapping" or "Index Mapping" in the context of recommendation systems refers to the creation of a relationship between the indices of items in the original dataset and their corresponding indices in a similarity or distance matrix.

3.5 RECOMMENDATION FOR UNSEEN DATA:

Users can upload an image through the web interface. The uploaded image undergoes pre-processing using the load_image function. This function reads the image, resizes it to a predefined size (224x224 pixels), and performs other necessary pre-processing steps. The pre-trained ResNet50 model is then used to process the pre-processed image and extract high-level features from it. This is achieved by passing the image through the layers of the ResNet50 model up to the GlobalMaxPooling2D layer. The embeddings of the uploaded image are compared with the embeddings of images in the existing dataset. This is done using cosine similarity. The cosine similarity between the embeddings of the uploaded image and the embeddings of each image in the dataset is calculated. The top N images (where N is determined by the user) that have the highest cosine similarity to the uploaded image are selected. These top N images are then recommended as visually similar fashion items to the user.

```
# Image processing code for the unseen image
input_img_path = '/content/sweat_shirt.jpg'
input_img = image.load_img(input_img_path, target_size=(224, 224))
input_img_array = image.img_to_array(input_img)
expand_input_img = np.expand_dims(input_img_array, axis=0)
preprocessed_input_img = preprocess_input(expand_input_img)
result_to_resnet_input = model.predict(preprocessed_input_img)
flatten_result_input = result_to_resnet_input.flatten()

# Convert df_embs to a sparse matrix
sparse_embs = csr_matrix(df_embs.values)

# Calculate Cosine Similarity with the input image features
cosine_sim_input = cosine_similarity(flatten_result_input.reshape(1, -1), sparse_embs)

# Get indices of top similar images
top_similar_indices = np.argsort(cosine_sim_input[0])[::-1][:5]

# Display reference image (unseen image)
plt.imshow(cv2.cvtColor(load_image(input_img_path), cv2.COLOR_BGR2RGB))
plt.title('Unseen Image (Reference)')
plt.axis('off')
plt.show()

# Display recommended images
display_images(top_similar_indices, df['image'], top_n=5)
```

Fig. 12 Recommendation for Unseen Data

CHAPTER – IV

RESULT AND DISCUSSION

A recommendation system is a content-based model used in E-Commerce platforms to suggest items based on their intrinsic features, particularly for fashion items. This model leverages **cosine similarity** to quantify the likeness between items and provide personalized recommendations. The core of the recommendation system is the **get_recommender** function, which uses indices mapping to identify the corresponding index in the similarity matrix **(cosine_sim).** The function then sorts these scores in descending order and selects the top recommendations, excluding the reference item itself. The recommendation model also includes a visualization component, displaying the reference image associated with the item and a dictionary storing the images of the recommended items.

The Recommendation system Covers broader categories of Products which include:

Apparel:

- Shirts, Jeans, Track Pants, T-shirts, Socks, Casual Shoes, Sweatshirts, Formal Shoes, Sports Shoes, Shorts, Kurtas, Waistcoat, Heels, Flats, Kurta Sets, Skirts, Night suits, Blazers, Trunk, Lounge Pants, Jackets, Bath Robe, Mufflers, Tunics, Sweaters, Tracksuits, Swimwear, Leggings, Jumpsuit, Salwar and Dupatta, Churidar, Gloves

Accessories:

- Watches, Belts, Handbags, Shoe Accessories, Bracelet, Sunglasses, Pendant, Earrings, Boxers, Jewellery Set, Lip Gloss, Scarves, Clutches, Shrug, Backpacks, Caps, Trousers, Mobile Pouch, Messenger Bag, Headband, Wristbands

Beauty and Personal Care:

- Deodorant, Lipstick, Nail Polish, Perfume and Body Mist, Foundation and Primer, Highlighter and Blush, Compact, Eye Cream, Lip Care, Kajal and Eyeliner, Eyeshadow, Face Scrub and Exfoliator, Mask and Peel

Bags and Luggage:

- Laptop Bag, Duffel Bag, Trolley Bag

Jewellery:

- Ring, Necklace and Chains, Bangle

Health and Wellness:

- Face Wash and Cleanser, Water Bottle

Fig. 13 Heels Recommendation

Fig. 14 Tops Recommendation

Fig. 15 Bag Recommendation

Fig. 16 Shoe Recommendation

Fig. 17 Jean Recommendation

Fig. 18 Lipstick Recommendation

Fig. 19 Women's T-Shirt Recommendation

Fig. 20 Watch Recommendation

Fig. 21 Nail Polish Recommendation

Fig. 22 Shirt Recommendation

Fig. 23 Sunglass Recommendation

Fig. 24 Shorts Recommendation

Fig. 25 Men's T-Shirt Recommendation

Fig. 26 Cap Recommendation

Fig. 27 Belt Recommendation

DEPLOYMENT:

We saved the Features Extracted from the Images as a CSV file and deployed the model as a web application interface.

FACTORS INFLUENCE THE PERFORMANCE OF CONTENT-BASED RECOMMENDATION MODEL:

- Feature Representation
- Image Quality and Variety
- Embedding Dimensionality
- Similarity Metric
- Model Complexity

LIMITATIONS:

- Limited Diversity
- Lack of Fine-Grained Recommendation
- Limited Exploration of Feature Space

FUTURE WORK:

- Explore the entire dataset instead of just the first 7000 rows to capture a more comprehensive understanding of the data.
- Combine content-based and collaborative filtering approaches to create a hybrid recommender system, taking advantage of both image embeddings and user behaviour.
- Implement a real-time recommendation system, considering user interactions and preferences dynamically.
- Explore multi-modal approaches by combining image data with other types of data (text, attributes) for a more comprehensive understanding of products.

CHAPTER – V

MODEL DEPLOYMENT

Deploying a machine learning model, known as model deployment, means integrating a machine learning model and integrate it into an existing production environment where it can take in an input and return an output. The purpose of deploying your model is so that you can make the predictions from a trained ML model available to others, whether that be users, management, or other systems.

There are various sources to deploy the model that we developed. Some of the easiest deployment sources from Python are **Streamlit, Flask API** etc. ML models will usually be deployed in an offline or local environment, so will need to be deployed with live data.

STREAMLIT:

Streamlit is an open-source Python library that simplifies the process of turning data scripts into interactive web applications. It is designed to be easy to use and allows data scientists and developers to create web applications for data analysis, machine learning, and visualization with minimal effort. Streamlit provides a high-level API that enables the creation of web applications using a few lines of Python code. With Streamlit, you can quickly prototype and iterate on your data applications. The library abstracts away much of the complexity involved in web development, allowing users to focus on the logic and visualization aspects of their applications.

Run the Python file in the terminal using the streamlit run command. This will generate a link. You can click the link which will take you to the web app.

You can now view your Streamlit app in your browser.

Local URL: http://localhost:8501

Network URL: http://192.168.29.251:8501

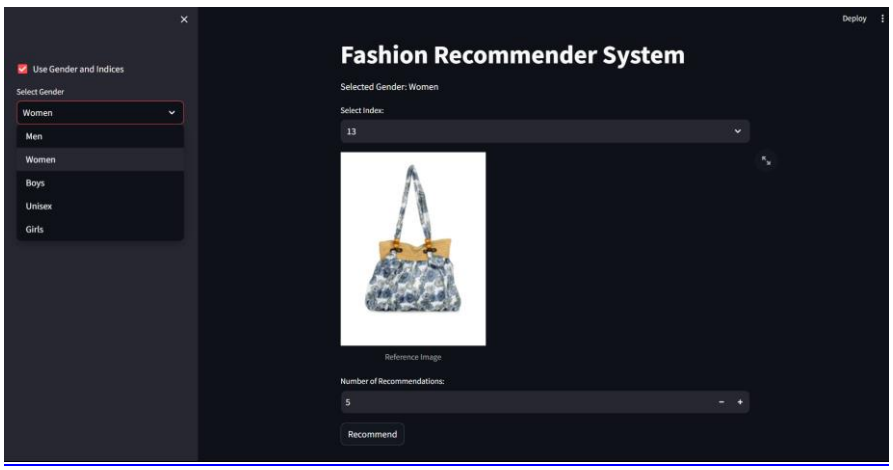

Fig. 28 Web interface for selecting indices

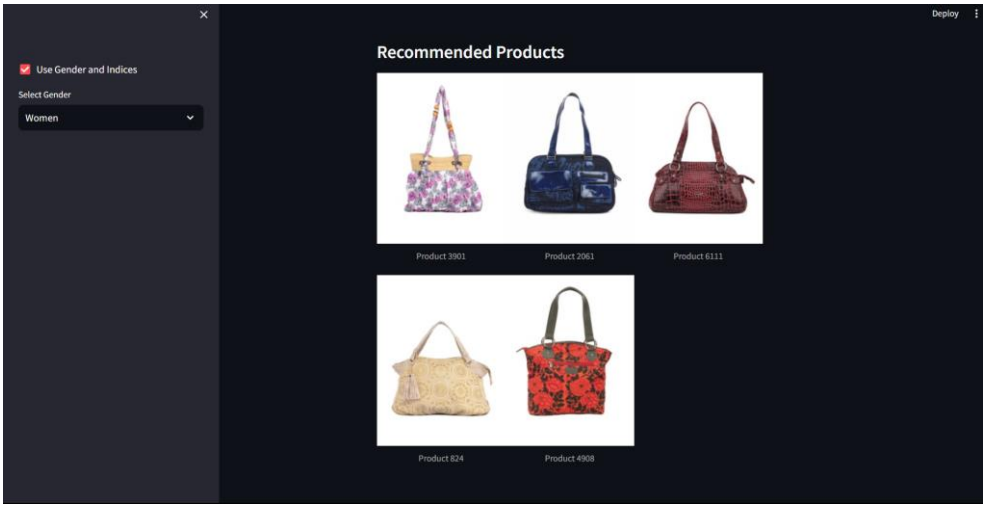

Fig. 29 Recommended Products for selected index

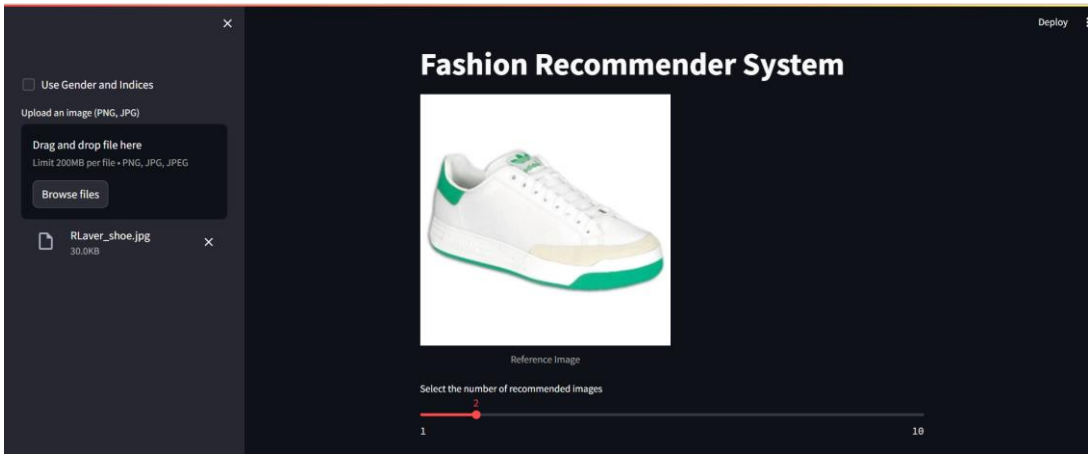

Fig. 30 Interface for uploading image

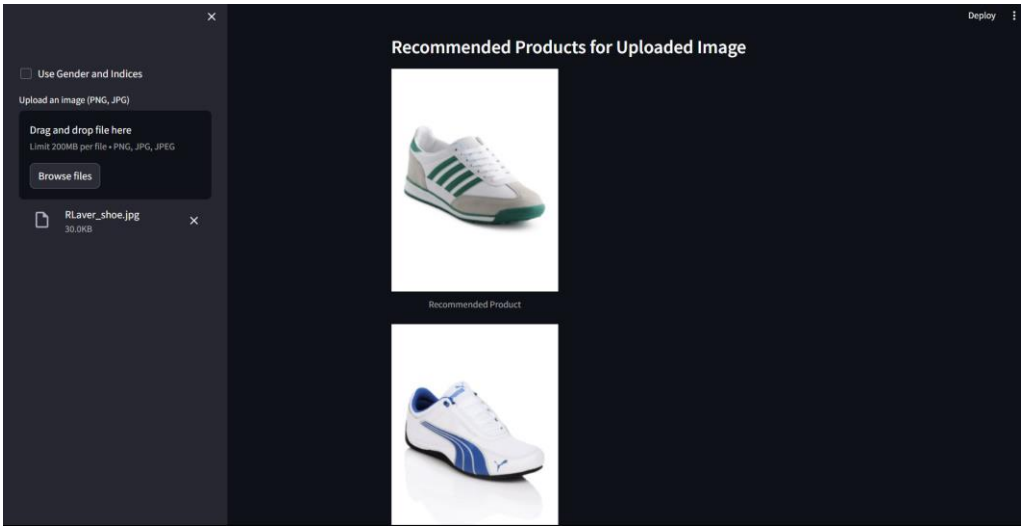

Fig. 31 Recommended Products for uploaded image

CHAPTER – VI
CONCLUSION

This project lays the groundwork for a Visual Recommendation System that not only meets the current demands of online shopping but also sets the stage for continuous improvement. By focusing on visual appeal and leveraging advanced deep learning techniques, the system aims to provide users with personalized and visually coherent product recommendations, contributing to an enriched online shopping experience. The project successfully navigated key stages, encompassing data pre-processing, visual feature extraction, model training, and comprehensive evaluation. Leveraging a pre-trained ResNet50 model for extracting intricate visual features, the recommendation system was crafted as a content-based model, focusing on enhancing user experience through visual similarity. The utilization of cosine similarity matrices in the evaluation process shed light on the system's proficiency in discerning and suggesting visually analogous products. The achieved milestones, including the successful extraction of visual features and the implementation of a content-based recommendation system, validate the effectiveness of the proposed approach.

BIBLIOGRAPHY

1. K.V Demochkin and A.V Savchenko (2019) Visual product recommendation using neural aggregation network and context gating.
2. Devashish Shankar, Sujay Narumanchi, Ananya H A, Pramod Kompalli, Krishnendu Chaudhury (2017) Deep Learning based Large Scale Visual Recommendation and Search for E-Commerce
3. Yanming Guo, Yu Liu, Ard Oerlemans, Songyang Lao, Song Wu, Michael S. Lew (2015) Deep learning for visual understanding: A review
4. https://medium.com/geekculture/building-a-visual-similarity-based-recommendation-system-using-python-872a5bea568e
5. https://www.databricks.com/blog/2022/03/01/building-a-similarity-based-image-recommendation-system-for-e-commerce.html
6. https://www.analyticsvidhya.com/blog/2022/12/streamlit-tutorial-building-web-apps-with-code-examples/
7. https://www.geeksforgeeks.org/a-beginners-guide-to-streamlit/
8. https://github.com/sonu275981/Fashion-Recommender-system