Configuration Manual

MSc Research Project
MSCDAD_JAN23A_O

SAGAR MALIK
Student ID: x22176608

School of Computing
National College of Ireland

Supervisor: JORGE BASILIO

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Sagar Malik

Student ID: X22176608

Programme: MSCDAD_JAN23_O **Year:** 2023

Module: MSC Research Project

Lecturer: Jorge Basilio

Submission Due Date: 31/01/2024

Project Title: Deep Learning-based Recommendation System for Personalized Product Recommendations

Word Count: **Page Count:** 10

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Sagar Malik

Date: 31/01/2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

SAGAR MALIK
X22176608

1. Introduction

This configuration manual represents a step-by-step guide for make the implementation code for the deep learning-based recommendation system objective is to delivers the personalized product recommendations. The implementation is done out in Python using the Jupyter Notebook integrated development environment situated in the Anaconda. The next following sections discuss about the necessary configurations and required tools.

2. System Specification

The product recommendation system is developed and performed test through the following system:

- Process: Intel i10 12th generation
- Operating System: Windows 11 (Professional)
- Ram: 16 Gb
- Gpu: RTX 3080 (10 Gb)
- Storage: 1024GB (PCEI SSD)

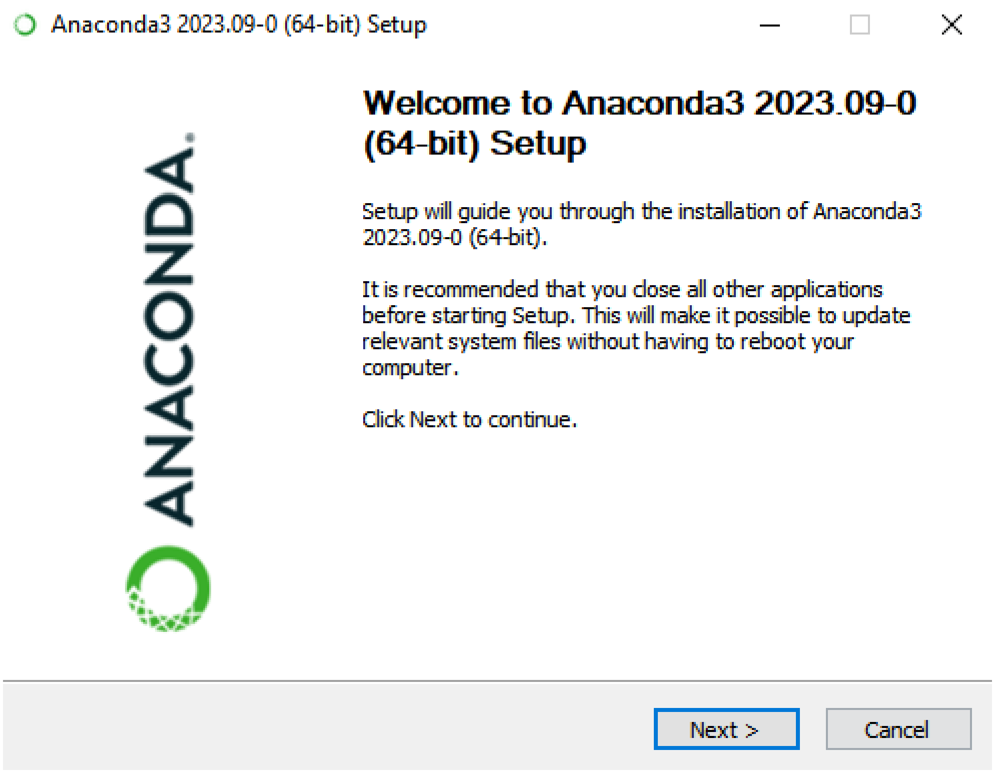
3. Softwares Used:

The following tools which are required to use and development for music recommendation system:

- ✓ Python
- ✓ Anaconda
- ✓ Jupyter

4. Installation of the Software:

- ✓ Install the Anaconda by downloading it from their open source website for the latest verison.



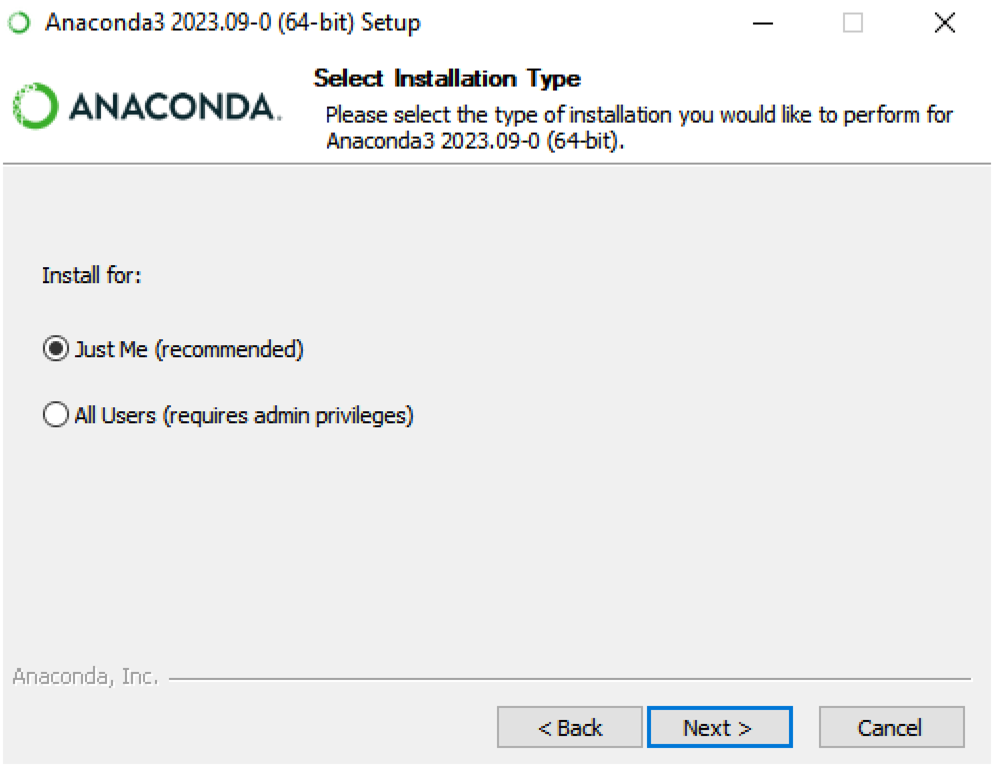
Anaconda3 2023.09-0 (64-bit) Setup

Welcome to Anaconda3 2023.09-0 (64-bit) Setup

Setup will guide you through the installation of Anaconda3 2023.09-0 (64-bit).

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.



Anaconda3 2023.09-0 (64-bit) Setup

Select Installation Type

Please select the type of installation you would like to perform for Anaconda3 2023.09-0 (64-bit).

Install for:

● Just Me (recommended)

○ All Users (requires admin privileges)

✓ After Installing the Anaconda install the requied packages which are not available in inbuilt installed anaconda kernel. By type the command “pip install package”.

5. Gather the Dataset:

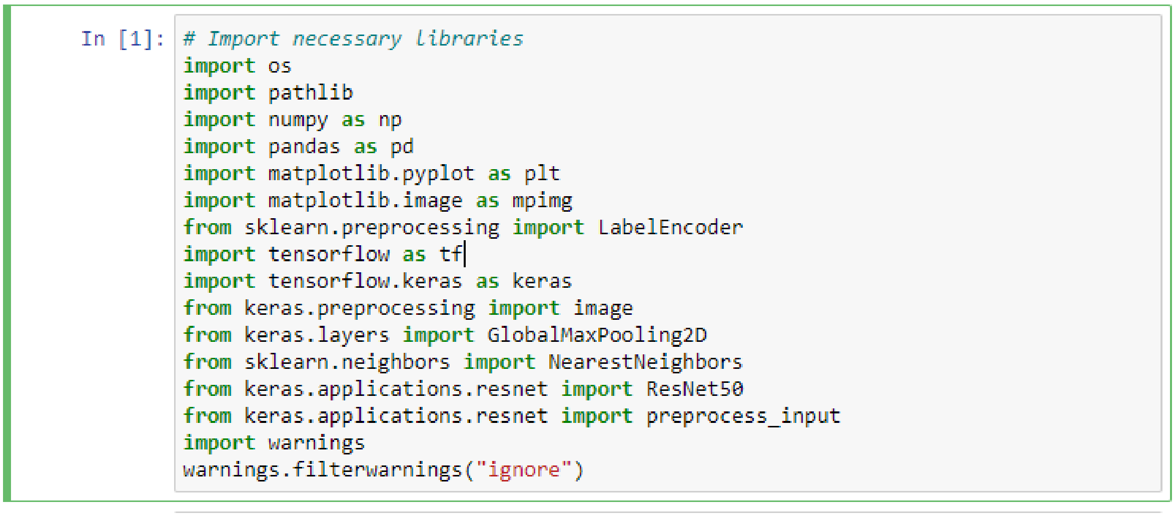
Get the appropriate dataset which would be the suitable for personalized product recommendations. E-commerce datasets from which are situated like Kaggle can be used. So I used the kaggle and obtained the dataset:

Dataset: https://www.kaggle.com/datasets/paramaggarwal/fashion-product-images-dataset

6. Application Execution

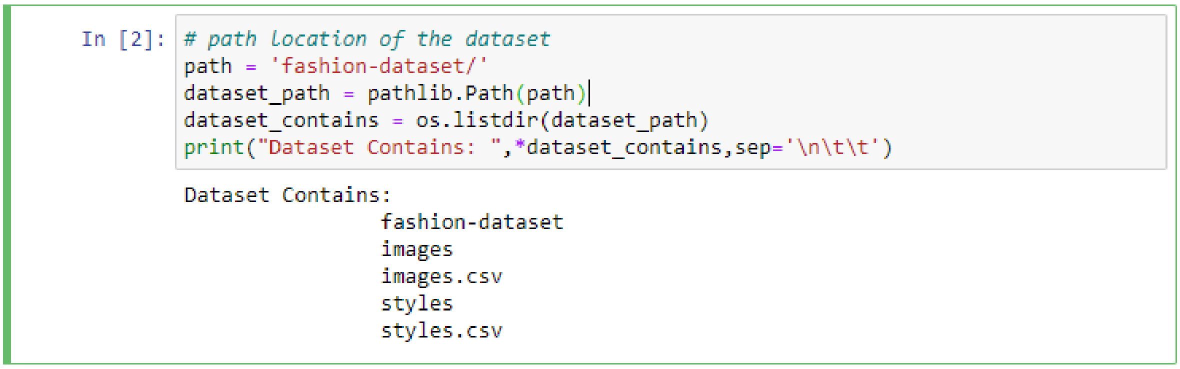
✓ Import the necessary libraries

✓



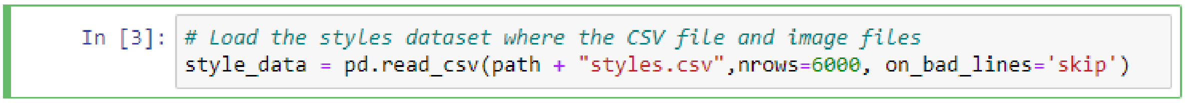
```
In [1]: # Import necessary libraries
        import os
        import pathlib
        import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import matplotlib.image as mpimg
        from sklearn.preprocessing import LabelEncoder
        import tensorflow as tf
        import tensorflow.keras as keras
        from keras.preprocessing import image
        from keras.layers import GlobalMaxPooling2D
        from sklearn.neighbors import NearestNeighbors
        from keras.applications.resnet import ResNet50
        from keras.applications.resnet import preprocess_input
        import warnings
        warnings.filterwarnings("ignore")
```

✓ Fetch the path of the dataset

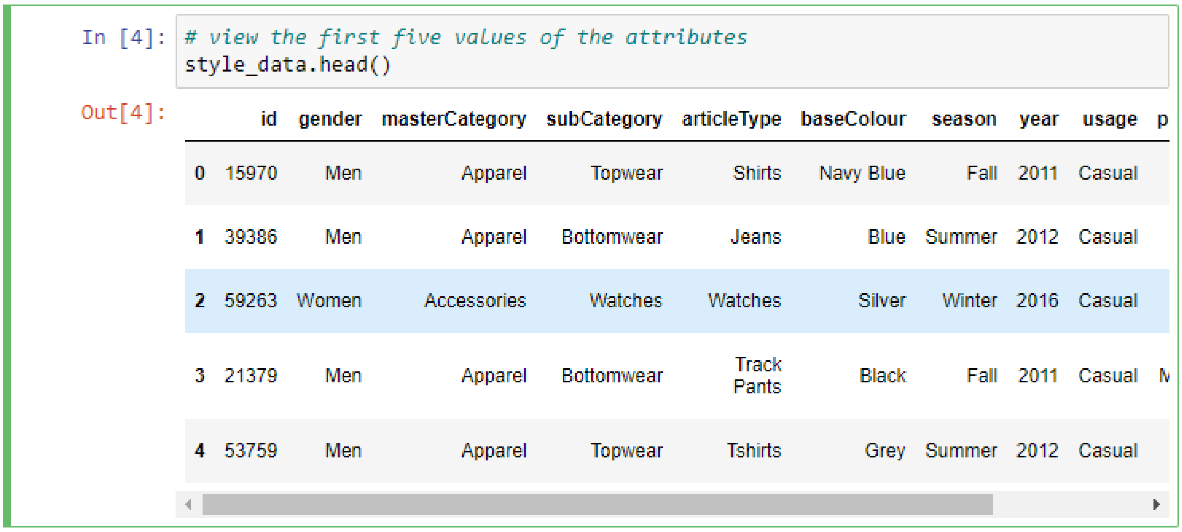


```
In [2]: # path location of the dataset
        path = 'fashion-dataset/'
        dataset_path = pathlib.Path(path)
        dataset_contains = os.listdir(dataset_path)
        print("Dataset Contains: ",*dataset_contains,sep='\n\t\t')

        Dataset Contains:
                        fashion-dataset
                        images
                        images.csv
                        styles
                        styles.csv
```

✓ Load the csv image dataset by fetched path



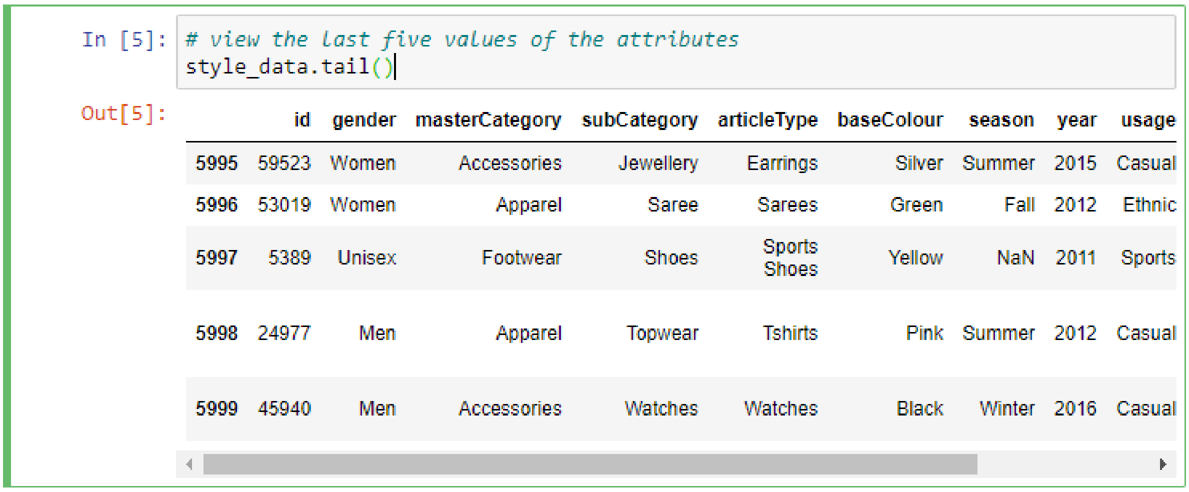
```
In [3]: # Load the styles dataset where the CSV file and image files
        style_data = pd.read_csv(path + "styles.csv",nrows=6000, on_bad_lines='skip')
```



```
In [4]: # view the first five values of the attributes
        style_data.head()
```

Out[4]:

	id	gender	masterCategory	subCategory	articleType	baseColour	season	year	usage	p
0	15970	Men	Apparel	Topwear	Shirts	Navy Blue	Fall	2011	Casual	
1	39386	Men	Apparel	Bottomwear	Jeans	Blue	Summer	2012	Casual	
2	59263	Women	Accessories	Watches	Watches	Silver	Winter	2016	Casual	
3	21379	Men	Apparel	Bottomwear	Track Pants	Black	Fall	2011	Casual	M
4	53759	Men	Apparel	Topwear	Tshirts	Grey	Summer	2012	Casual	

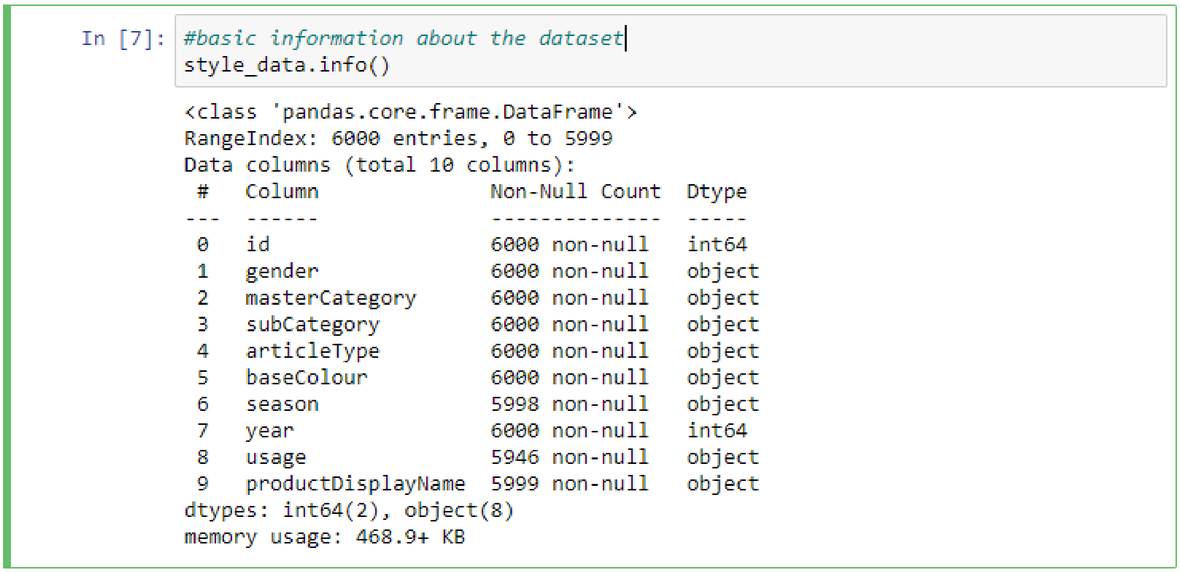


```
In [5]: # view the last five values of the attributes
        style_data.tail()
```

Out[5]:

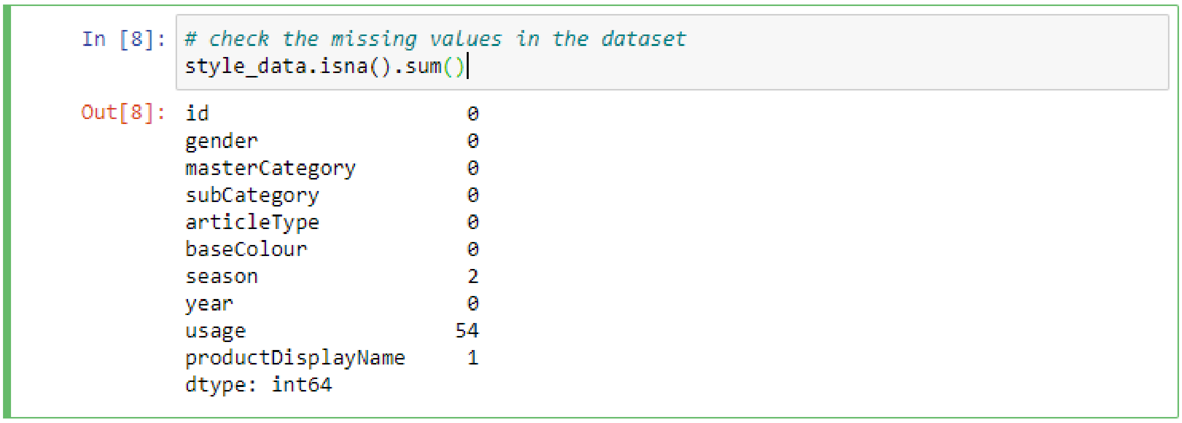
	id	gender	masterCategory	subCategory	articleType	baseColour	season	year	usage
5995	59523	Women	Accessories	Jewellery	Earrings	Silver	Summer	2015	Casual
5996	53019	Women	Apparel	Saree	Sarees	Green	Fall	2012	Ethnic
5997	5389	Unisex	Footwear	Shoes	Sports Shoes	Yellow	NaN	2011	Sports
5998	24977	Men	Apparel	Topwear	Tshirts	Pink	Summer	2012	Casual
5999	45940	Men	Accessories	Watches	Watches	Black	Winter	2016	Casual

- ✓ Exploration of the image dataset by fetched path

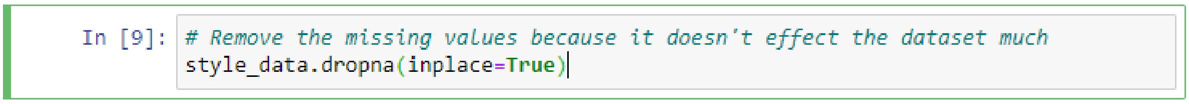


```
In [7]: #basic information about the dataset
        style_data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6000 entries, 0 to 5999
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   id                  6000 non-null   int64 
 1   gender              6000 non-null   object
 2   masterCategory      6000 non-null   object
 3   subCategory         6000 non-null   object
 4   articleType         6000 non-null   object
 5   baseColour          6000 non-null   object
 6   season              5998 non-null   object
 7   year                6000 non-null   int64 
 8   usage               5946 non-null   object
 9   productDisplayName  5999 non-null   object
dtypes: int64(2), object(8)
memory usage: 468.9+ KB
```

✓ Data Preprocessing to clean the dataset

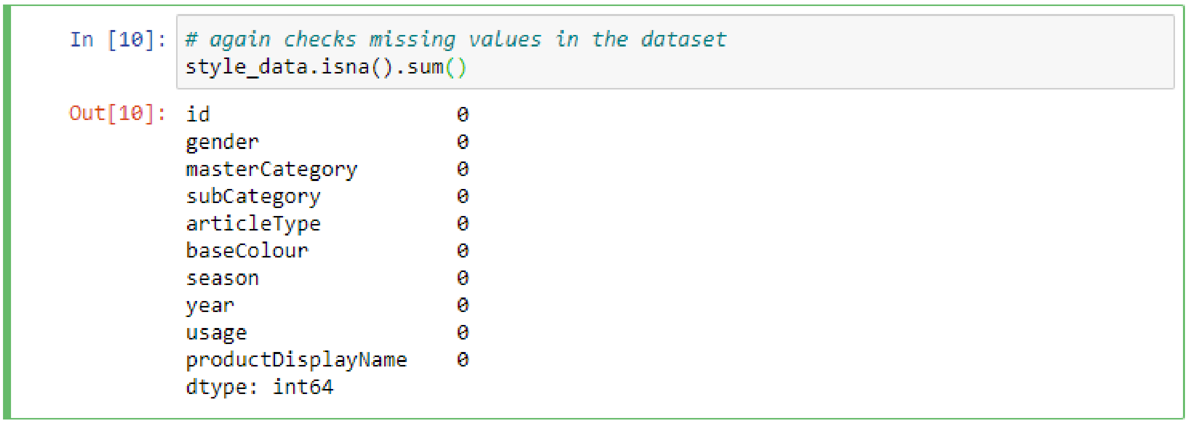


```
In [8]:  # check the missing values in the dataset
         style_data.isna().sum()

Out[8]:  id                     0
         gender                 0
         masterCategory         0
         subCategory            0
         articleType            0
         baseColour             0
         season                 2
         year                   0
         usage                 54
         productDisplayName     1
         dtype: int64
```



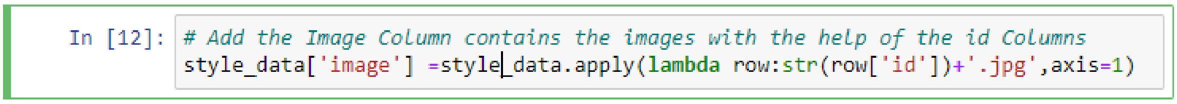
```
In [9]:  # Remove the missing values because it doesn't effect the dataset much
         style_data.dropna(inplace=True)
```

Again checks for any lefted missing value or not.

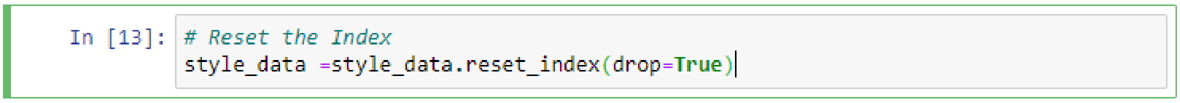


```
In [10]:  # again checks missing values in the dataset
          style_data.isna().sum()

Out[10]:  id                    0
          gender                0
          masterCategory        0
          subCategory           0
          articleType           0
          baseColour            0
          season                0
          year                  0
          usage                 0
          productDisplayName    0
          dtype: int64
```

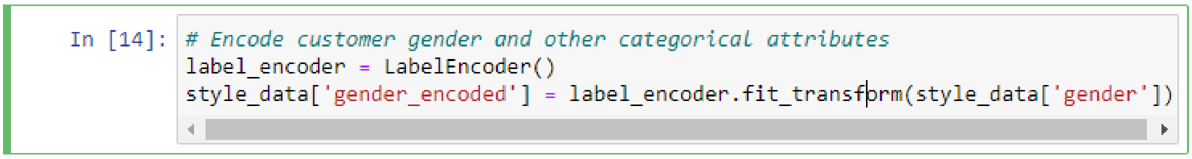
Add the column to derive or access any particular image



```
In [12]:  # Add the Image Column contains the images with the help of the id Columns
          style_data['image'] =style_data.apply(lambda row:str(row['id'])+'.jpg',axis=1)
```

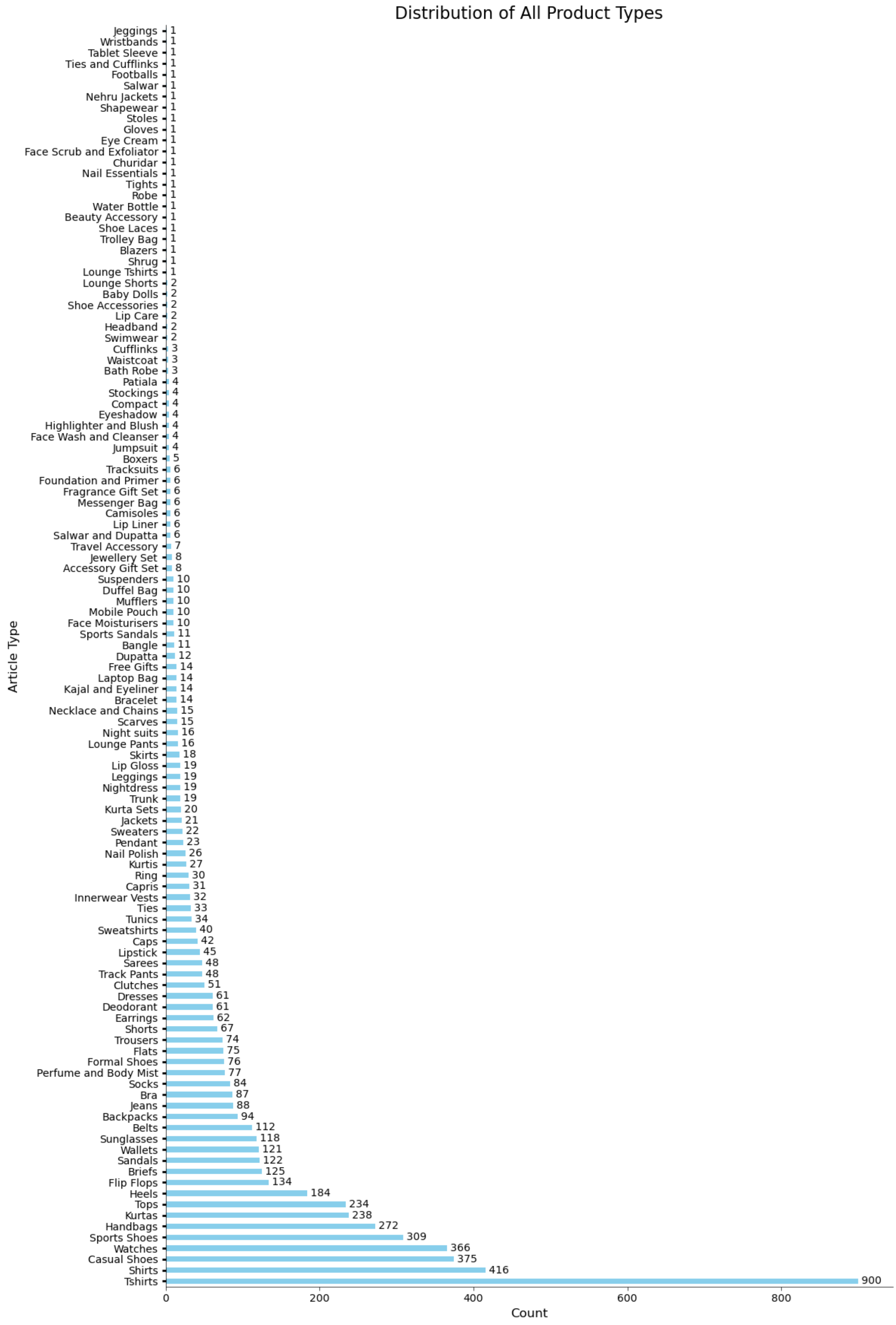


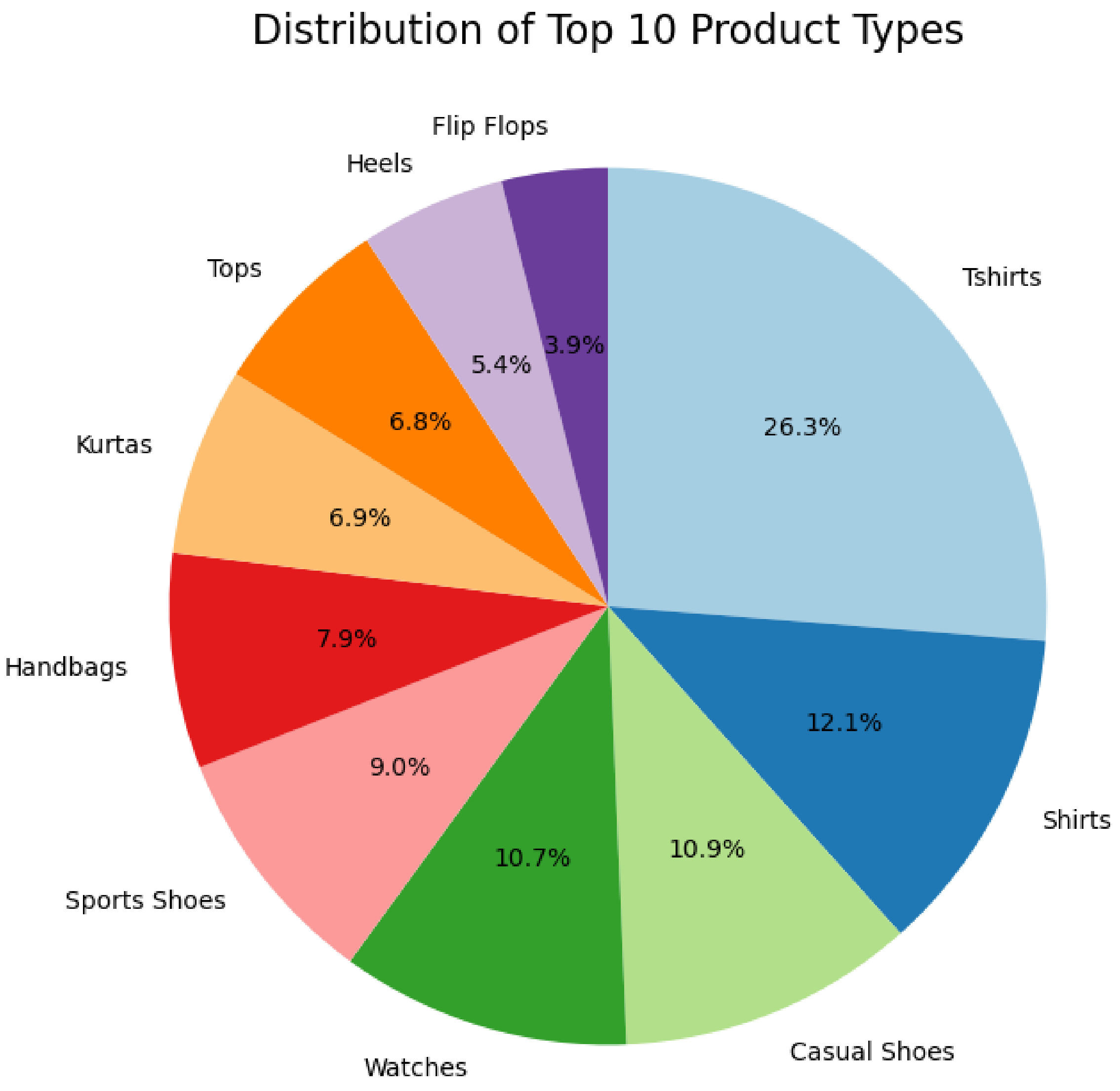
```
In [13]:  # Reset the Index
          style_data =style_data.reset_index(drop=True)
```



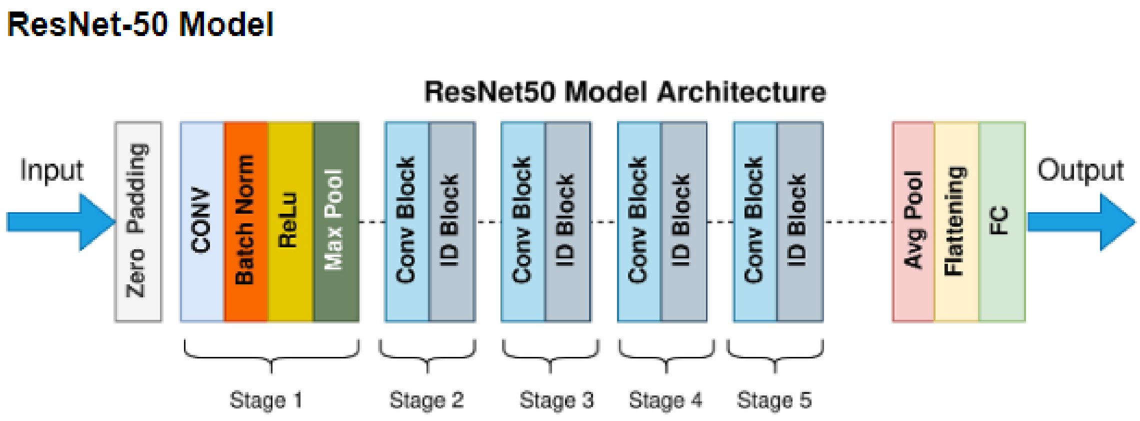
```
In [14]:  # Encode customer gender and other categorical attributes
          label_encoder = LabelEncoder()
          style_data['gender_encoded'] = label_encoder.fit_transform(style_data['gender'])
```

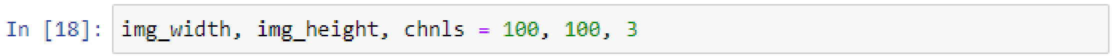
- ✓ Data Visualization from the cleaned and preprocessed dataset



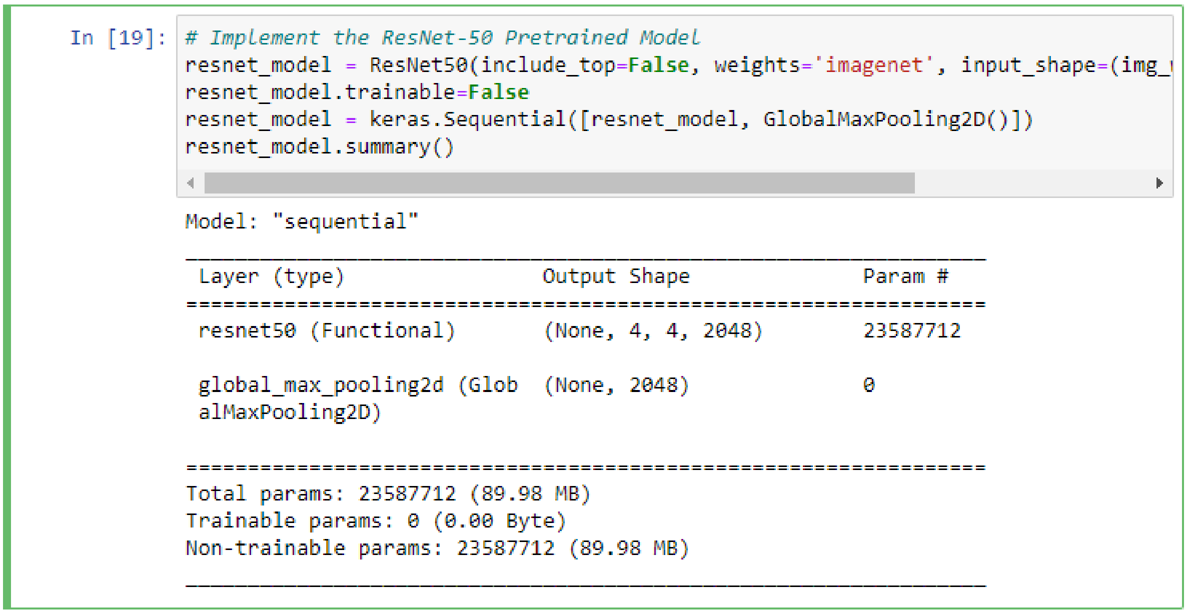


- ✓ Build and Integrate the ResNet50 Pre-trained Model

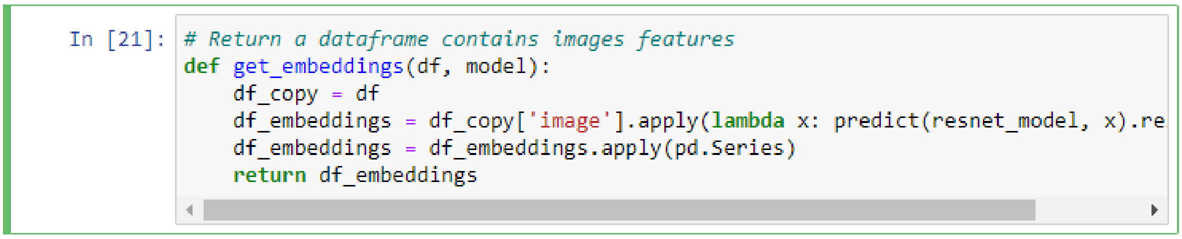




```
In [18]: img_width, img_height, chnls = 100, 100, 3
```

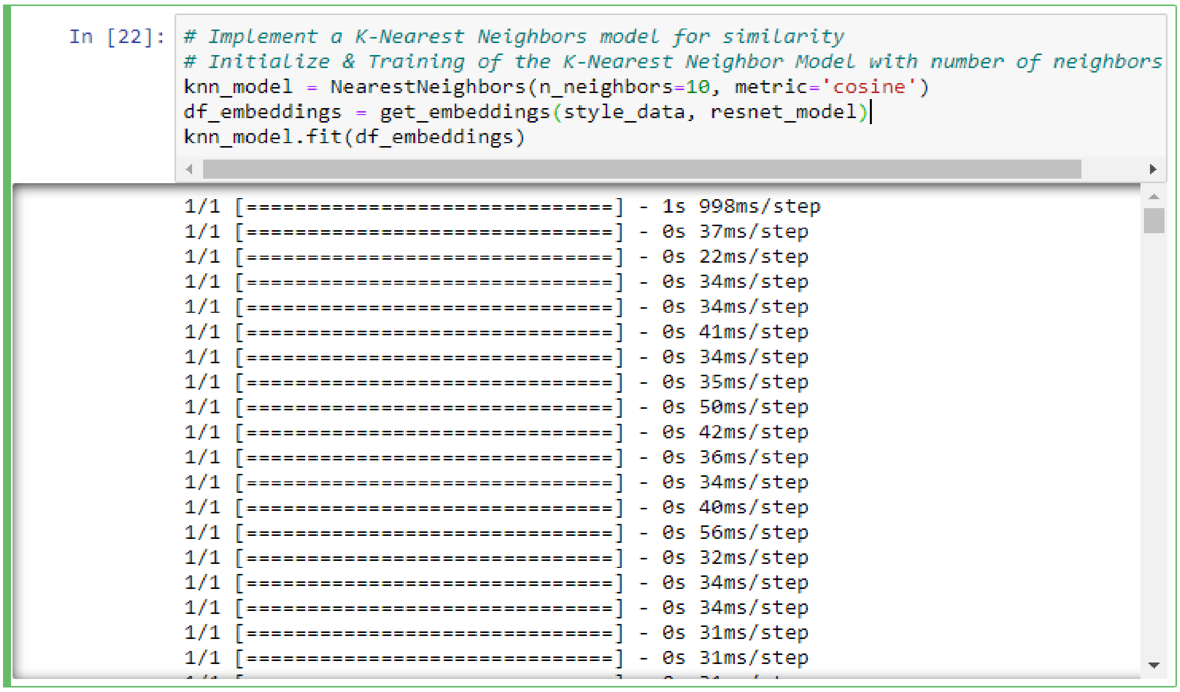


```
In [19]: # Implement the ResNet-50 Pretrained Model
         resnet_model = ResNet50(include_top=False, weights='imagenet', input_shape=(img_
         resnet_model.trainable=False
         resnet_model = keras.Sequential([resnet_model, GlobalMaxPooling2D()])
         resnet_model.summary()

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50 (Functional)       (None, 4, 4, 2048)        23587712

 global_max_pooling2d (Glob  (None, 2048)              0
 alMaxPooling2D)

=================================================================
Total params: 23587712 (89.98 MB)
Trainable params: 0 (0.00 Byte)
Non-trainable params: 23587712 (89.98 MB)
_________________________________________________________________
```



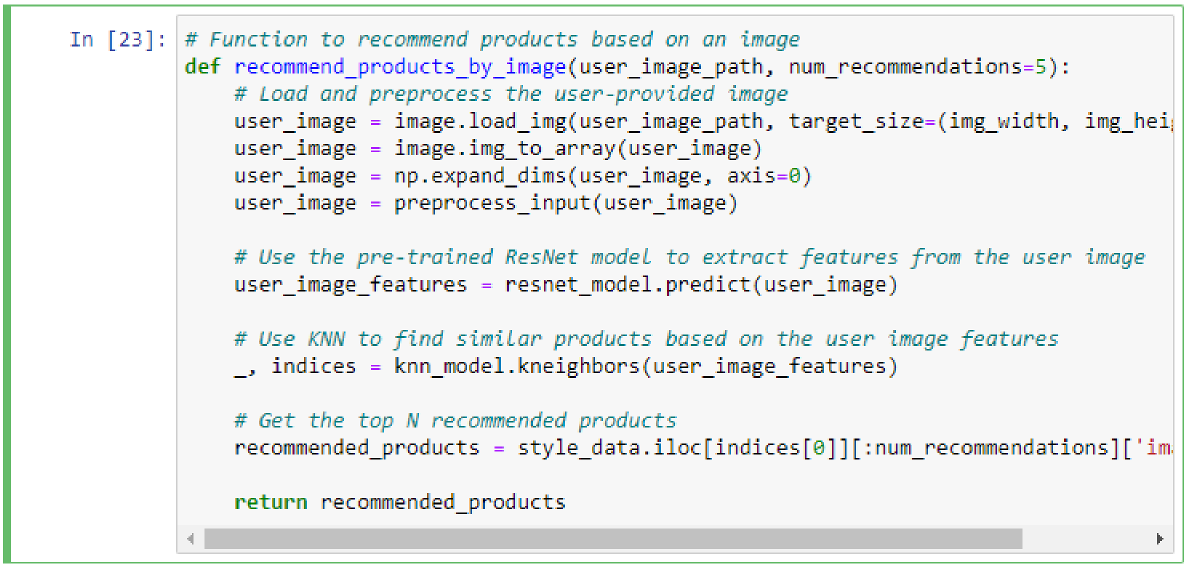
```
In [21]: # Return a dataframe contains images features
         def get_embeddings(df, model):
             df_copy = df
             df_embeddings = df_copy['image'].apply(lambda x: predict(resnet_model, x).re
             df_embeddings = df_embeddings.apply(pd.Series)
             return df_embeddings
```

- ✓ Generate the embedding through the ResNet50 Pre-trained Model with the KNN model for similarity



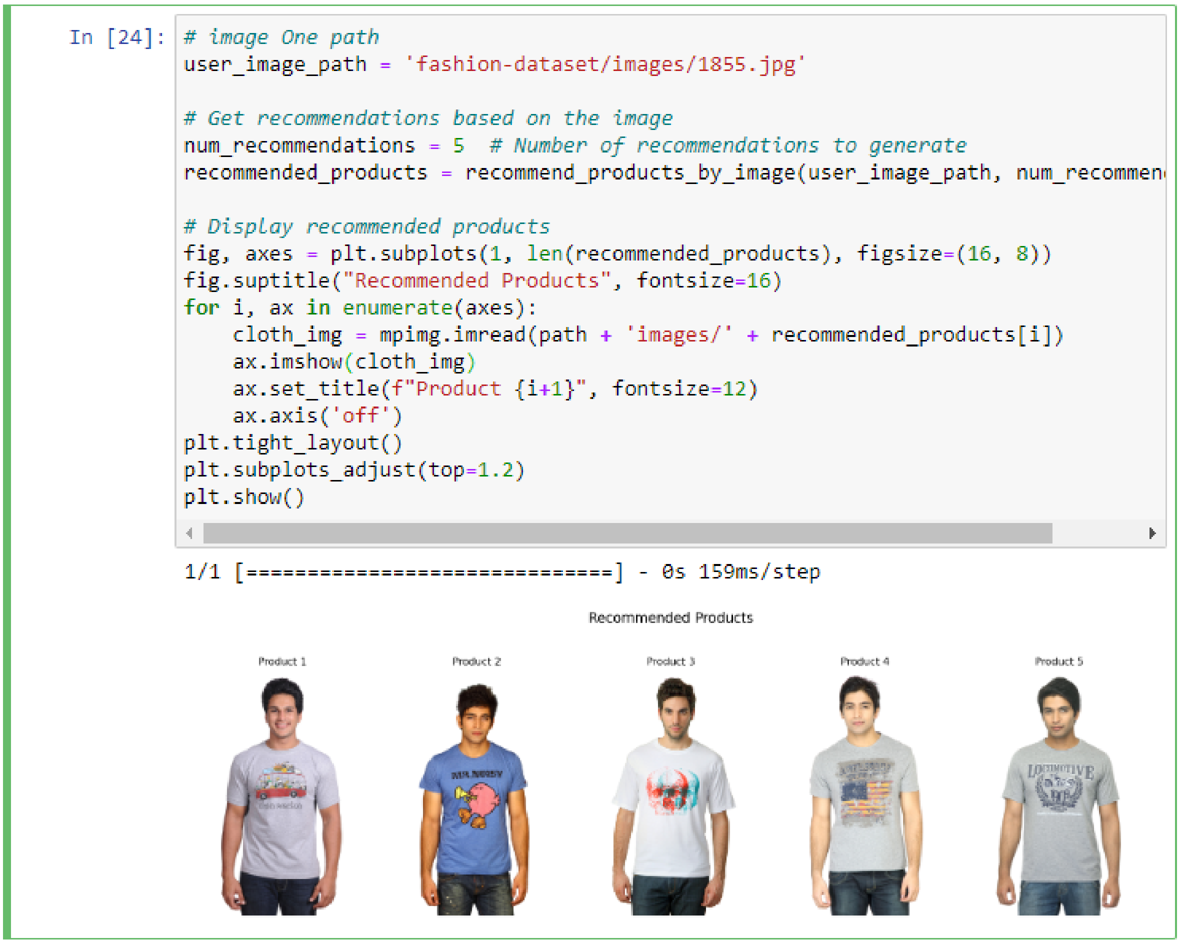
```
In [22]: # Implement a K-Nearest Neighbors model for similarity
         # Initialize & Training of the K-Nearest Neighbor Model with number of neighbors
         knn_model = NearestNeighbors(n_neighbors=10, metric='cosine')
         df_embeddings = get_embeddings(style_data, resnet_model)
         knn_model.fit(df_embeddings)

1/1 [==============================] - 1s 998ms/step
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 56ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 31ms/step
```

✓ Recommendation Function to recommend the product images



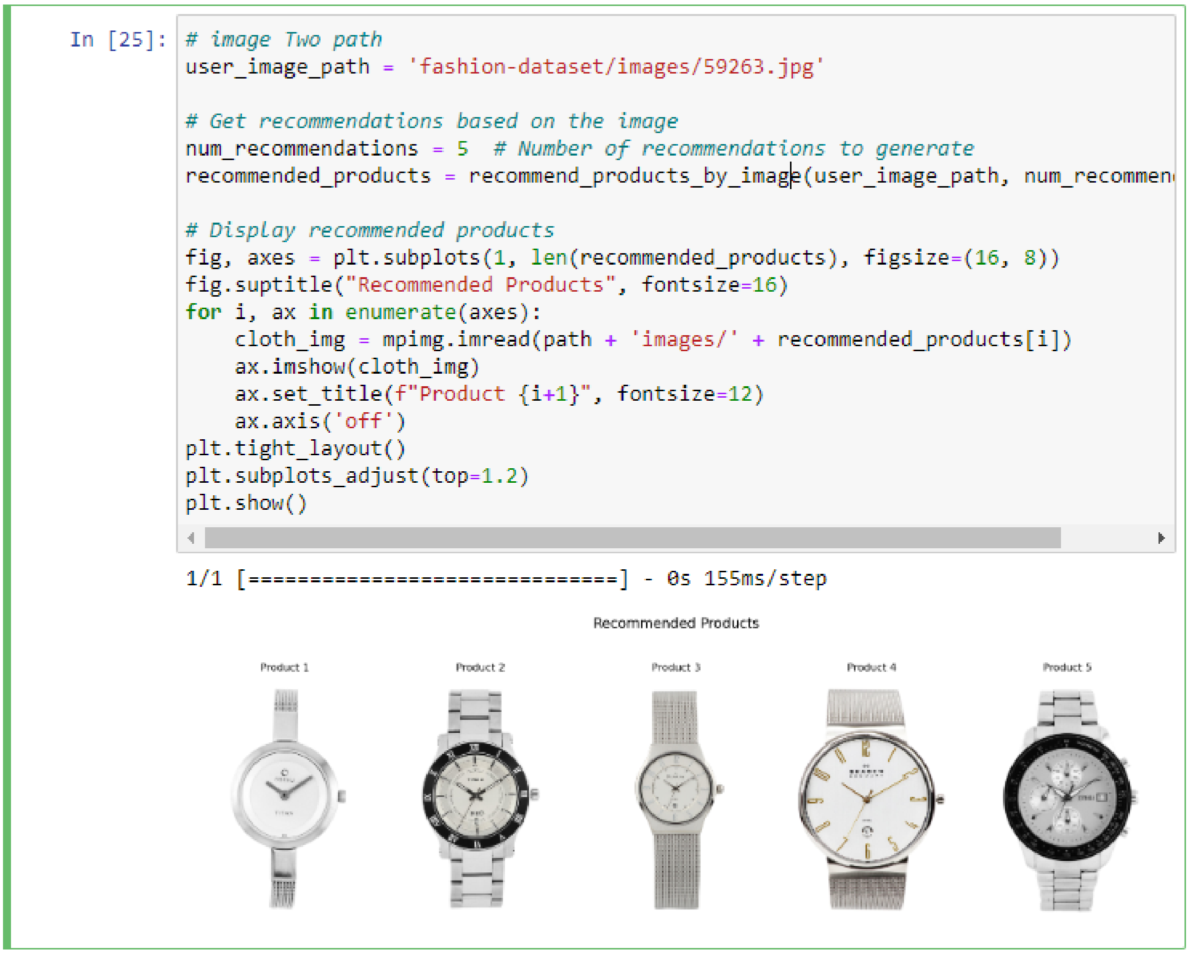
```
In [23]: # Function to recommend products based on an image
         def recommend_products_by_image(user_image_path, num_recommendations=5):
             # Load and preprocess the user-provided image
             user_image = image.load_img(user_image_path, target_size=(img_width, img_hei
             user_image = image.img_to_array(user_image)
             user_image = np.expand_dims(user_image, axis=0)
             user_image = preprocess_input(user_image)

             # Use the pre-trained ResNet model to extract features from the user image
             user_image_features = resnet_model.predict(user_image)

             # Use KNN to find similar products based on the user image features
             _, indices = knn_model.kneighbors(user_image_features)

             # Get the top N recommended products
             recommended_products = style_data.iloc[indices[0]][:num_recommendations]['im

             return recommended_products
```

✓ Applies the Recommendation Function to recommend the product images



```
In [24]: # image One path
         user_image_path = 'fashion-dataset/images/1855.jpg'

         # Get recommendations based on the image
         num_recommendations = 5  # Number of recommendations to generate
         recommended_products = recommend_products_by_image(user_image_path, num_recommen

         # Display recommended products
         fig, axes = plt.subplots(1, len(recommended_products), figsize=(16, 8))
         fig.suptitle("Recommended Products", fontsize=16)
         for i, ax in enumerate(axes):
             cloth_img = mpimg.imread(path + 'images/' + recommended_products[i])
             ax.imshow(cloth_img)
             ax.set_title(f"Product {i+1}", fontsize=12)
             ax.axis('off')
         plt.tight_layout()
         plt.subplots_adjust(top=1.2)
         plt.show()
```

```
1/1 [==============================] - 0s 159ms/step
```



```
In [25]: # image Two path
         user_image_path = 'fashion-dataset/images/59263.jpg'

         # Get recommendations based on the image
         num_recommendations = 5  # Number of recommendations to generate
         recommended_products = recommend_products_by_image(user_image_path, num_recommen

         # Display recommended products
         fig, axes = plt.subplots(1, len(recommended_products), figsize=(16, 8))
         fig.suptitle("Recommended Products", fontsize=16)
         for i, ax in enumerate(axes):
             cloth_img = mpimg.imread(path + 'images/' + recommended_products[i])
             ax.imshow(cloth_img)
             ax.set_title(f"Product {i+1}", fontsize=12)
             ax.axis('off')
         plt.tight_layout()
         plt.subplots_adjust(top=1.2)
         plt.show()
```

```
1/1 [==============================] - 0s 155ms/step
```

This configuration manual which contains the detailed process and description about the application of Personalized Product Recommendation System running and their requirements to make the study performed well.

References

Anaconda: https://docs.anaconda.com/free/anaconda/install/windows/
Dataset: https://www.kaggle.com/datasets/paramaggarwal/fashion-product-images-dataset
Kaggle: https://www.kaggle.com/