ST. JOSEPH'S COLLEGE (AUTONOMOUS)

#36, LALBAGH ROAD, BENGALURU 560027

ST. JOSEPH'S INSTITUTE OF INFORMATION TECHNOLOGY

DEPARTMENT OF ADVANCED COMPUTING

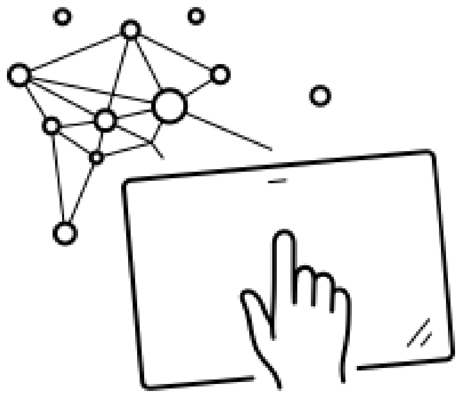
MULTIVARIATE STATISTICS LAB [BDDE2P5]

PROJECT REPORT

ON

PRODUCT RECOMMENDATION SYSTEM

(Domain – Fashion)



Team: JAB WE MET

Ayesha Shariff [21BDA18] **Jerusha Gomez[21BDA11]**

Gayatri Krishna [21BDA16] **Sneha T. [21BDA30]**

Shivanshi Maheshwari [21BDA33

TABLE OF CONTENTS

CHAPTER I
ABSTRACT

With an increase in the standard of living, humans gradually shifted their attention towards fashion, which is considered to be a popular aesthetic. In recent years, the textile and fashion industries have witnessed an enormous amount of growth in fast fashion. On e-commerce platforms, where numerous choices are available, an efficient recommendation system is required to sort, order, and convey relevant product content or information to users. Image-based fashion recommendation systems have attracted a huge amount of attention from fast fashion retailers as they provide a personalized shopping experience to consumers. The main challenge in building a fashion recommendation system is that it is a very dynamic industry. It changes very often when it comes to seasons, festivals, pandemic conditions like Covid-19 and many more. Initially, users would search for the particular kind of clothe that they wanted and by various algorithms of filtering, they would obtain the results. With the increasing digitization and use of social media, consumers are exposed to various clothing items (These include clothes, clothing accessories, shoes and so on). The consumers tend to see a picture on the internet and want the exact same thing. Here, instead of filtering out the contents, an image is uploaded and products similar to this image from the database are shown to the consumer, thereby allowing them to choose from these recommended products. Despite its huge potential, the available studies do not provide a rigorous review of fashion recommendation systems and the corresponding filtering techniques. This project explores models that could be implemented to develop product recommendation systems.

CHAPTER II

PURPOSE AND PROPOSED SCOPE OF THE PROJECT

Many-a-time people see something that they are interested in and tend to look for products that are similar to that. Customers no longer have to visit many stores, stand in long queues, or try on garments in dressing rooms as millions of products are now available in online catalogues. Effective fashion recommendation systems can have a noticeable impact on billions of customers' shopping experiences and increase sales and revenues on the provider-side.

Unlike other areas, fashion recommendations should not be based solely on personal taste and past activity of the customer. Unlike the conventional systems that rely on user's previous purchases and history, our project aims at using an image of a product given as input by the user to generate recommendations. We use neural networks to process the images from our Fashion dataset in order to generate final recommendations by the system. According to different studies, e-commerce retailers, such as Amazon, eBay and Shopstyle and social networking sites such as Pinterest, Snapchat, Instagram, Facebook, Chictopia, and Lookbook are now regarded as the most popular media for fashion advice and recommendations. Recommendation systems act as decision-makers and have recently been gaining popularity amongst the e-commerce websites. Pinterest, amazon and any other popular e-commerce site uses recommendation systems these days.

We chose this as our project to understand the characteristics of basic recommendation systems. We have used these recommendation systems ourselves plenty number of times. We thought that being at the other end of the algorithm would be something fun to do.

CHAPTER III

ABOUT THE DATA

Fig. 3.1 Representative Image of Data

The dataset has been obtained from UCI repository.

Our dataset has several appealing properties. It contains 44424 diverse fashion images ranging from well posed shop images to unconstrained customer photos. In addition to high resolution product images, we also have multiple label attributes describing the product.

Different variables describing the images in our dataset are:

1. ID – Unique ID for each product
2. Gender
3. Master-Category – Apparel/Accessories etc.
4. Sub-Category – Category of the product like top wear, bottom wear, watches etc.
5. Article Type – type of the product like shirts, jeans etc.
6. Base Colour – describing the base colour of the product
7. Season – Fall, summer, winter products
8. Year
9. Usage – Casual, formal etc.
10. Product Display Name – Name of each individual product.

CHAPTER IV

METHODOLOGY

Every problem has a solution. Sometimes it comes easy and sometimes it does not. But eventually, if and when a solution is found, there is a certain methodology. Methodology is a plan of action and consists of the tools and techniques used to conquer the problem at hand. Methodology helps in understanding the process flow of the entire solution. This helps to enforce checkpoints for furthering the quality of the solution. If a problem should arise, one can get back to the methodology to refer to the exact point of failure in order to try and fix it.

In brief, the methodology for our project – "Fashion Recommendation System" – is given by:

- ➔ We begin with our dataset containing over forty-five thousand images.
- ➔ Deep Learning forms the basis of the methodology for our project.
- ➔ The coding is carried out in Python.
- ➔ We use the CNN and Transfer Learning Models to help classify the images.
- ➔ Since we are using Transfer Learning, there is no actual training of the data. We further use ResNet50 and ImageNet to facilitate the Transfer Learning.
- ➔ Various libraries as required are added onto the project.
- ➔ We upload an image. The goal of the project is that it returns similar products to the one we uploaded.

Below, are the descriptions of the tools and techniques used in our project.

1. PYTHON

Fig. 4.1. Python Logo

Developed by Guido Van Rossum and released in the year 1991, Python is a high-level, general-purpose, interpreted language. In the recent years, Python has been gaining a lot of traction. Some of the key reasons for that is that Python increases productivity, is easier to debug and is typically intelligible. Another great feature of Python is that it has dynamic data types. Scripting, developing and testing are few of Python's applications.

Python is so ubiquitous in the data science field, that when one hears the term 'Machine Learning', they immediately assume that Python was involved. And rightly so. Python is used extensively for machine learning and AI uses. The primary reasons for this include:

a) Simple and Convenient

   As compared to the predecessors of Python, Python is relatively easy to work with. It is straightforward with its keywords and is generally convenient to use. Even for a beginner, Python is easy to grasp and work with.

b) Libraries and Frameworks

   Python offers a myriad of libraries and frameworks to work with. These are pre-written codes that one can work with. Some of the commonly used Machine Learning libraries include: TensorFlow, Scikit-learn, Numpy, Pandas, Seaborn and so on.

c) Platform Independence

   The hassle of carrying the same machine on which the programmer programmed the code is completely done away with here. The program can run on any machine, given that the required dependencies are installed.

d) Great Community Base

   Python has an extraordinary community of programmers. One can reach out to the community for any sort of guidance. Given that it is a global community, there are high chances that someone across the globe has some idea about what another is trying to achieve using Python. Having the support of a community in this manner goes a long way.

We have used the Python language to code everything in the project.

2. JUPYTER NOTEBOOK

Fig. 4.2. Jupyter Logo

In Python programming, usually, there is an interactive IDE (Integrated Development Environment). Here, the codes are typed in line by line and immediate execution is seen. A notebook is where the code and the output visualizations can be seen in one place. One such open-source application is Jupyter Notebook. Using Jupyter Notebook, one can see the code and results together in one place. Apart from the codes, some other features like text headings, images, paragraphs, markdown cells can be added to add meaning to the code. This enables it look like a neat document which makes it easier to understand. Throughout the project, we have used Jupyter Notebook to work with Python. We have used Jupyter Notebook to perform some EDA as well as to code the main parts of the project.

3. DEEP LEARNING

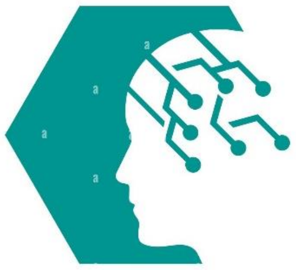


Fig. 4.3. Deep Learning Icon

Deep learning is a machine learning technique that teaches computers to do what comes naturally to humans, i.e., learn by example. Deep learning is a key technology behind driverless cars, enabling them to recognize a stop sign, or to distinguish a pedestrian from a lamppost. It is the key to voice control in consumer devices like phones, tablets, TVs, and hands-free speakers. Deep learning is getting lots of attention lately and for good reason. It’s achieving results that were not possible before.

In deep learning, a computer model learns to perform classification tasks directly from images, text, or sound. Deep learning models can achieve state-of-the-art accuracy, sometimes exceeding human-level performance. Models are trained by using a large set of labelled data and neural network architectures that contain many layers.

Deep Learning has proved to be a very powerful tool because of its ability to handle large amounts of data. The interest to use hidden layers has surpassed traditional techniques, especially in pattern recognition. One of the most popular deep neural networks is Convolutional Neural Networks. The model used to train our image dataset in our project is a deep learning CNN model

4. CNN

Fig. 4.4. CNN Icon

The human brain processes a huge amount of information the second we see an image. Each neuron works in its own receptive field and is connected to the other neurons in a way that they cover the entire visual field. Just as each neuron responds to stimuli only in the restricted region of the visual field called the receptive field in the biological vision system, each neuron in a CNN processes data only in its receptive field as well. Convolutional neural networks are composed of multiple layers of artificial neurons. It uses a special technique called Convolution. In mathematics **convolution** is a mathematical operation on two functions that produces a third function that expresses how the shape of one is modified by the other. Artificial neurons, a rough imitation of their biological counterparts, are mathematical functions that calculate the weighted sum of multiple inputs and outputs an activation value. When we input an image in a ConvNet, each layer generates several activation functions that are passed on to the next layer.

The first layer usually extracts basic features such as horizontal or diagonal edges. This output is passed on to the next layer which detects more complex features such as corners or combinational edges. As we move deeper into the network it can identify even more complex features such as objects, faces, etc.

Based on the activation map of the final convolution layer, the classification layer outputs a set of confidence scores (values between 0 and 1) that specify how likely the image is to belong to a “class.” For instance, if you have a ConvNet that detects cats, dogs, and horses, the output of the final layer is the possibility that the input image contains any of those animals. In our project, ConvNet detects products (such as jeans, saree etc.) and the output final layer is the possibility that the input image contains any of the products like jeans, saree etc. in accordance with the color of the input image.

5. REVERSE IMAGE PROCESSING

Fig. 4.5. Representative Image of Reverse Image Processing

Reverse Image Search is defined as a content-based image retrieval query method. In other words, reverse image search means a search engine technology that takes input query in the form of the image file and provides an output related to the image, which we use in the input. There are so many search engines that provide the facility of Reverse Image Search, such as Google, Yahoo Image Search, TinEye, Bing Image Search, Pinterest, Picsearch, Flickr, and Shutterstock. Various websites also provide the facility of Reverse Image Search like Reddit.

6. TRANSFER LEARNING

Fig. 4.6. Representative Image of Transfer Learning

Transfer learning is a part of the CNN model. Here, there are pre-trained samples, and learnings from these samples are passed on. It can be analogized to passing down wisdom from one generation to another. The former generation learns some things about life having experienced certain situations. Instead of the latter generation having to go through the same mundane cycle, the former generation simply passes down their wisdom and insights. In the same way, there are models that are created for similar if not the exactly same purposes. This technique in Machine Learning drastically reduces the need for labelled data. Using transfer learning, training is not required for each time the task is run. With transfer learning, a model can be trained on an available labelled dataset, then be applied to a similar task that may involve unlabelled data. Since our project deals with image classification, we have used the labelled dataset called 'ImageNet' to implement transfer learning.

7. RESNET50

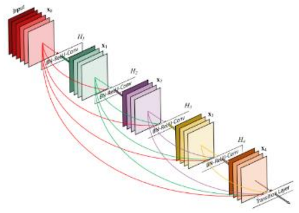


Fig. 4.7. Representative Image of ResNet Model

ResNet50 is a variant of the ResNet Model. ResNet model comes under the Residual Learning part of deep learning. Residual learning enables one to go deeper in the model and thereby achieve greater accuracy. Residual learning also makes the training of the neural network easier. In CNN, there are stacks of layers which are used for training. Generally, at each layer, some features about the dataset of images are learnt. Here, at each layer, a learned feature is subtracted. The ResNet model achieves this. ResNet50 is basically a ResNet model in which there are fifty layers. Fourty-eight of them are convolutional layers, one is a MaxPool layer and the other is an AveragePool layer.

8. IMAGENET

Fig. 4.8. Representative Image of ImageNet

ImageNet is an image database organized according to the WordNet hierarchy, in which each node of the hierarchy is depicted by hundreds and thousands of images. We have used the ImageNet dataset in our project to implement Transfer Learning

9. LIBRARIES USED

Fig. 4.9. Python Libraries and Frameworks

a) TQDM

TQDM is derived from the Arabic word 'taqaddum' which means 'progress'. It is a library in Python that is used to display the progress bars. It is used specially when there are large loops involved. It gives a glimpse on the progress or how far alone the execution has gone.

b) Streamlit

Streamlit is an open-source app framework for Machine Learning and Data Science teams. Streamlit was released in October 2019 and was recently acquired by Snowflake. There's huge excitement about it in the Data Science community. Python frameworks such as scikit-learn, spaCy, Pandas and various visualization frameworks such as Altair, Plotly and Matplotlib all work seamlessly with Streamlit. With its component extensibility architecture, we can build and integrate most kinds of web frontends into Streamlit apps. Streamlit was founded on the idea of building interactive apps easily. There is a new and different workflow to building these kinds of applications. Streamlit is very *simple, lightweight, super-easy* to understand. In essence, by sprinkling interactive widgets throughout the Python code, Streamlit will magically build a web app. By implementing Streamlit in our project we have enabled the webapp to stay performant even when loading data, manipulating the large image dataset, and performing expensive computations.

c) Image

It is used for image processing tasks. It includes functionalities like filtering, manipulating and saving images.

d) TensorFlow

TensorFlow is a Python library for fast numerical computing created and released by Google. It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of TensorFlow

e) Sklearn

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modelling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python.

f) Numpy

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices.

g) CV2

OpenCV is a Python open-source library, which is used for computer vision in Artificial intelligence, Machine Learning, face recognition, etc.

h) Pickle

Pickle in Python is primarily used in serializing and deserializing a Python object structure. In other words, it's the process of converting a Python object into a byte stream to store it in a file/database, maintain program state across sessions, or transport data over the network.

CHAPTER V
WHAT WENT RIGHT AND WHAT DID NOT

What went right:

- We were able to find the exact kind of dataset that we were looking for.
- The goal of our project was to recommend similar looking products based on images that are uploaded. Through the various modules, packages and libraries, we were able to achieve this. For us, it was no easy feat. We did have a lot of struggles, yet, we were able to achieve our goal.

What was a struggle:

- The resolution of the images within our data set posed a problem. Without the clear images, feature extraction would not be very accurate.
- Initially, the matching of the images was done on two criteria – colour and pattern. This was posing a problem as it did not yield the result that we expected.
- Time constraint as TQDM was taking a longer time than we had expected.
- Extracting features was an issue initially.

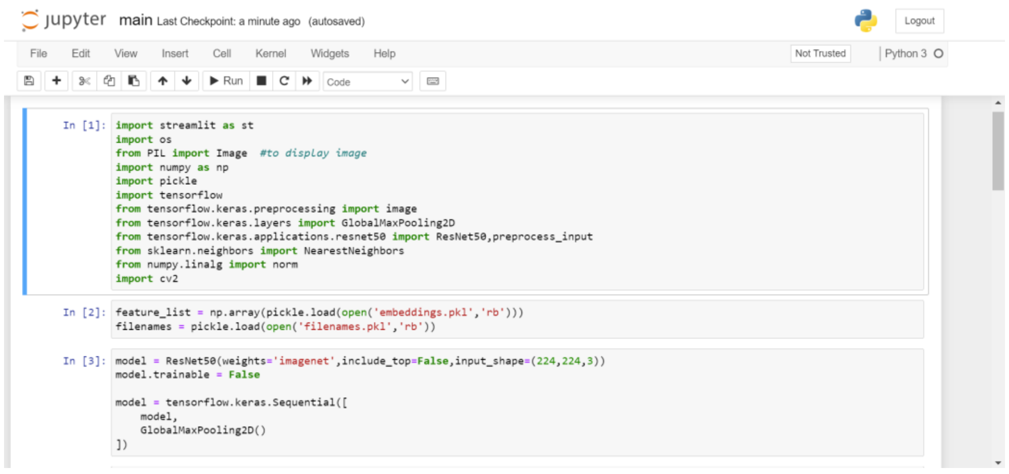
To an extent, we were able to overcome these struggles to achieve the desired results from our project.

CHAPTER VI

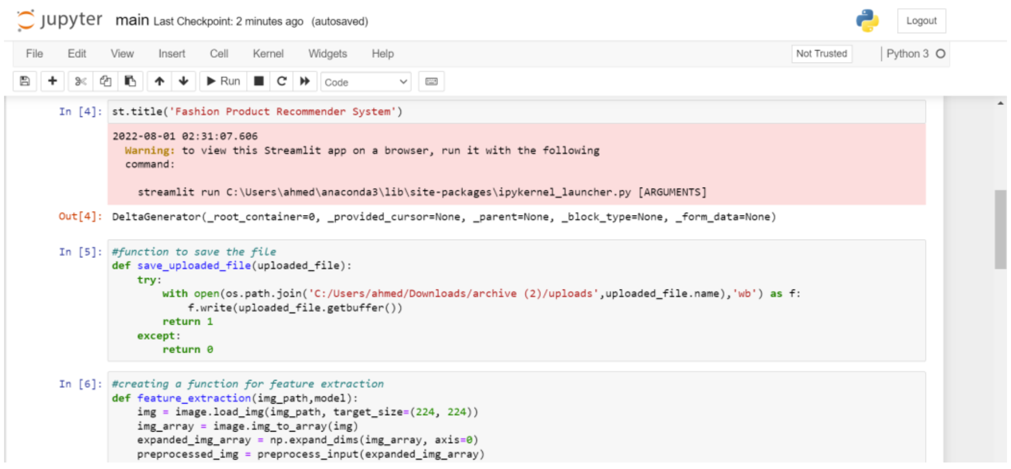
EVIDENCE OF ACTIVITY

➔ Code

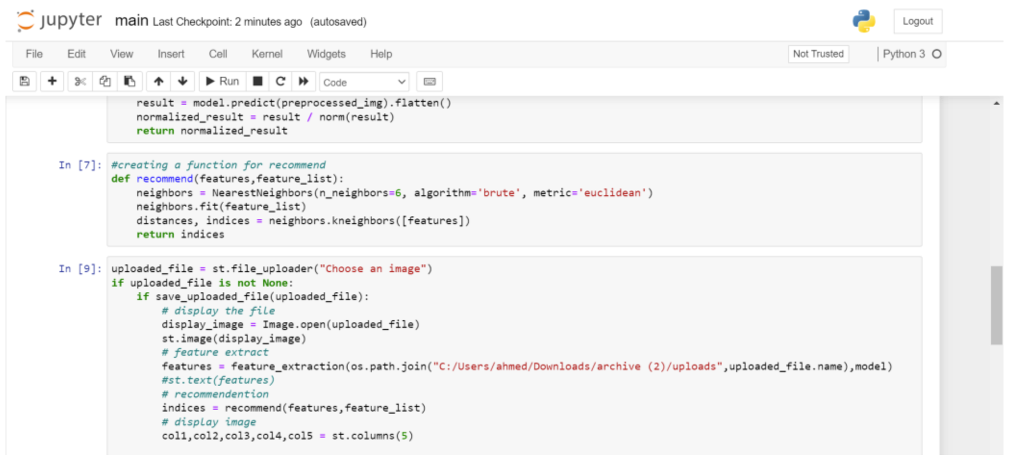
a) The main file:


jupyter main Last Checkpoint: a minute ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code

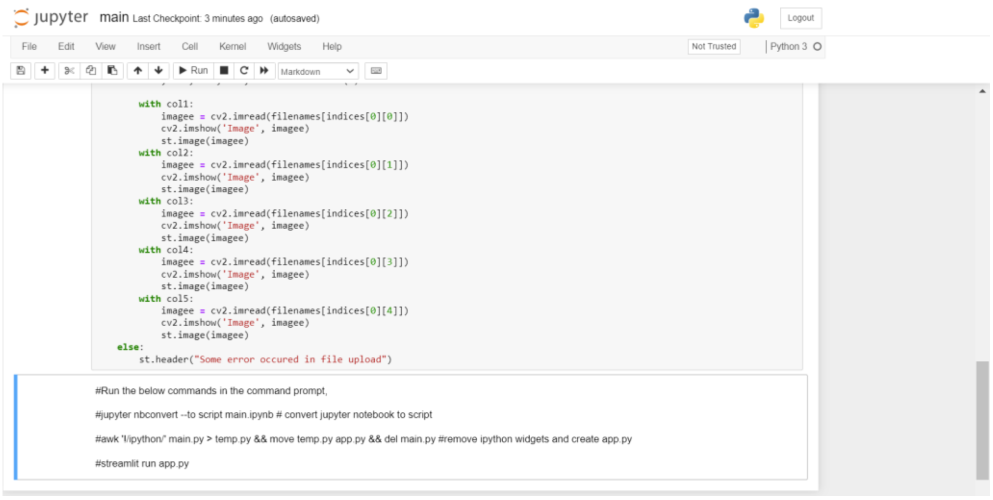
```
In [1]: import streamlit as st
        import os
        from PIL import Image  #to display image
        import numpy as np
        import pickle
        import tensorflow
        from tensorflow.keras.preprocessing import image
        from tensorflow.keras.layers import GlobalMaxPooling2D
        from tensorflow.keras.applications.resnet50 import ResNet50,preprocess_input
        from sklearn.neighbors import NearestNeighbors
        from numpy.linalg import norm
        import cv2

In [2]: feature_list = np.array(pickle.load(open('embeddings.pkl','rb')))
        filenames = pickle.load(open('filenames.pkl','rb'))

In [3]: model = ResNet50(weights='imagenet',include_top=False,input_shape=(224,224,3))
        model.trainable = False

        model = tensorflow.keras.Sequential([
            model,
            GlobalMaxPooling2D()
        ])
```


jupyter main Last Checkpoint: 2 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code

```
In [4]: st.title('Fashion Product Recommender System')

        2022-08-01 02:31:07.606
          Warning: to view this Streamlit app on a browser, run it with the following
          command:

            streamlit run C:\Users\ahmed\anaconda3\lib\site-packages\ipykernel_launcher.py [ARGUMENTS]

Out[4]: DeltaGenerator(_root_container=0, _provided_cursor=None, _parent=None, _block_type=None, _form_data=None)

In [5]: #function to save the file
        def save_uploaded_file(uploaded_file):
            try:
                with open(os.path.join('C:/Users/ahmed/Downloads/archive (2)/uploads',uploaded_file.name),'wb') as f:
                    f.write(uploaded_file.getbuffer())
                return 1
            except:
                return 0

In [6]: #creating a function for feature extraction
        def feature_extraction(img_path,model):
            img = image.load_img(img_path, target_size=(224, 224))
            img_array = image.img_to_array(img)
            expanded_img_array = np.expand_dims(img_array, axis=0)
            preprocessed_img = preprocess_input(expanded_img_array)
```


jupyter main Last Checkpoint: 2 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code

```
            result = model.predict(preprocessed_img).flatten()
            normalized_result = result / norm(result)
            return normalized_result

In [7]: #creating a function for recommend
        def recommend(features,feature_list):
            neighbors = NearestNeighbors(n_neighbors=6, algorithm='brute', metric='euclidean')
            neighbors.fit(feature_list)
            distances, indices = neighbors.kneighbors([features])
            return indices

In [9]: uploaded_file = st.file_uploader("Choose an image")
        if uploaded_file is not None:
            if save_uploaded_file(uploaded_file):
                # display the file
                display_image = Image.open(uploaded_file)
                st.image(display_image)
                # feature extract
                features = feature_extraction(os.path.join("C:/Users/ahmed/Downloads/archive (2)/uploads",uploaded_file.name),model)
                #st.text(features)
                # recommendention
                indices = recommend(features,feature_list)
                # display image
                col1,col2,col3,col4,col5 = st.columns(5)
```



```
        with col1:
            imagee = cv2.imread(filenames[indices[0][0]])
            cv2.imshow('Image', imagee)
            st.image(imagee)
        with col2:
            imagee = cv2.imread(filenames[indices[0][1]])
            cv2.imshow('Image', imagee)
            st.image(imagee)
        with col3:
            imagee = cv2.imread(filenames[indices[0][2]])
            cv2.imshow('Image', imagee)
            st.image(imagee)
        with col4:
            imagee = cv2.imread(filenames[indices[0][3]])
            cv2.imshow('Image', imagee)
            st.image(imagee)
        with col5:
            imagee = cv2.imread(filenames[indices[0][4]])
            cv2.imshow('Image', imagee)
            st.image(imagee)
    else:
        st.header("Some error occured in file upload")
```

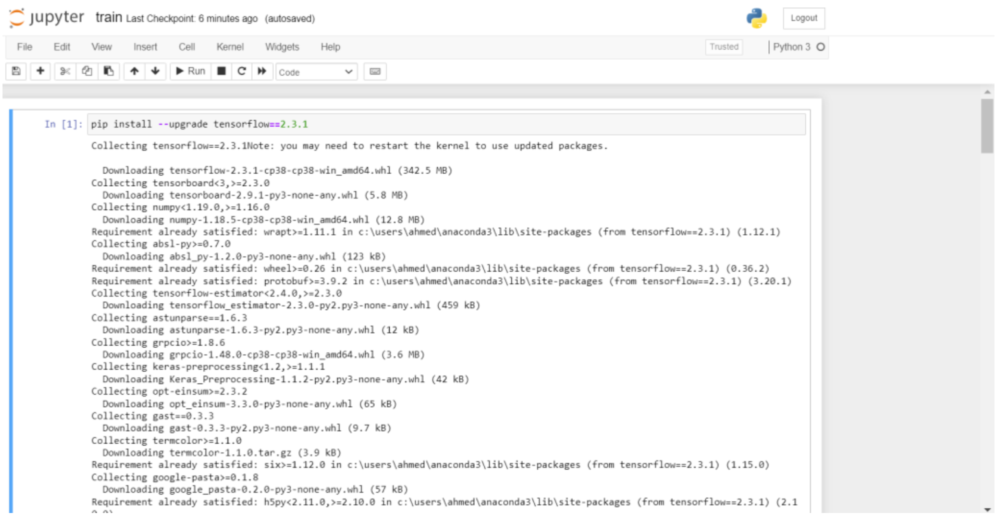
#Run the below commands in the command prompt,

#jupyter nbconvert --to script main.ipynb # convert jupyter notebook to script

#awk '!/ipython/' main.py > temp.py && move temp.py app.py && del main.py #remove ipython widgets and create app.py

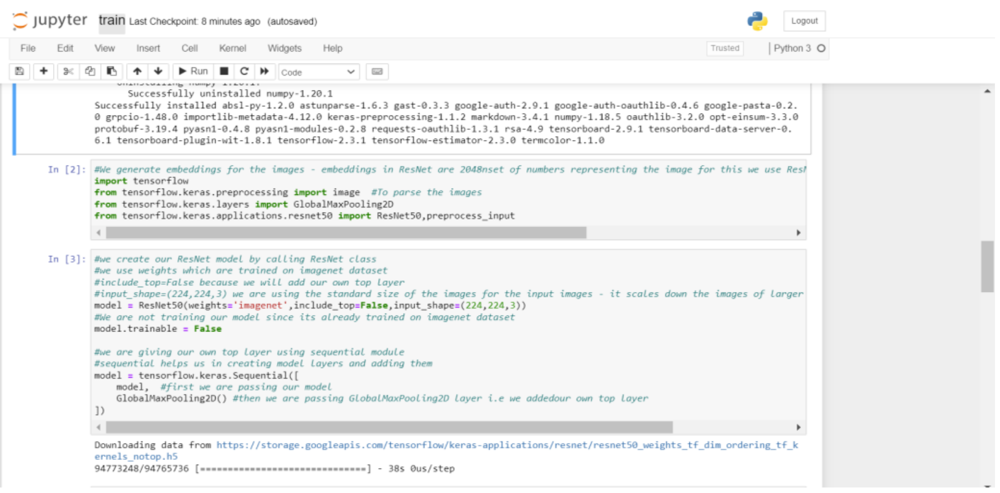
#streamlit run app.py

b) The Training File



```
In [1]: pip install --upgrade tensorflow==2.3.1
```

```
Collecting tensorflow==2.3.1Note: you may need to restart the kernel to use updated packages.

  Downloading tensorflow-2.3.1-cp38-cp38-win_amd64.whl (342.5 MB)
Collecting tensorboard<3,>=2.3.0
  Downloading tensorboard-2.9.1-py3-none-any.whl (5.8 MB)
Collecting numpy<1.19.0,>=1.16.0
  Downloading numpy-1.18.5-cp38-cp38-win_amd64.whl (12.8 MB)
Requirement already satisfied: wrapt>=1.11.1 in c:\users\ahmed\anaconda3\lib\site-packages (from tensorflow==2.3.1) (1.12.1)
Collecting absl-py>=0.7.0
  Downloading absl_py-1.2.0-py3-none-any.whl (123 kB)
Requirement already satisfied: wheel>=0.26 in c:\users\ahmed\anaconda3\lib\site-packages (from tensorflow==2.3.1) (0.36.2)
Requirement already satisfied: protobuf>=3.9.2 in c:\users\ahmed\anaconda3\lib\site-packages (from tensorflow==2.3.1) (3.20.1)
Collecting tensorflow-estimator<2.4.0,>=2.3.0
  Downloading tensorflow_estimator-2.3.0-py2.py3-none-any.whl (459 kB)
Collecting astunparse==1.6.3
  Downloading astunparse-1.6.3-py2.py3-none-any.whl (12 kB)
Collecting grpcio>=1.8.6
  Downloading grpcio-1.48.0-cp38-cp38-win_amd64.whl (3.6 MB)
Collecting keras-preprocessing<1.2,>=1.1.1
  Downloading Keras_Preprocessing-1.1.2-py2.py3-none-any.whl (42 kB)
Collecting opt-einsum>=2.3.2
  Downloading opt_einsum-3.3.0-py3-none-any.whl (65 kB)
Collecting gast==0.3.3
  Downloading gast-0.3.3-py2.py3-none-any.whl (9.7 kB)
Collecting termcolor>=1.1.0
  Downloading termcolor-1.1.0.tar.gz (3.9 kB)
Requirement already satisfied: six>=1.12.0 in c:\users\ahmed\anaconda3\lib\site-packages (from tensorflow==2.3.1) (1.15.0)
Collecting google-pasta>=0.1.8
  Downloading google_pasta-0.2.0-py3-none-any.whl (57 kB)
Requirement already satisfied: h5py<2.11.0,>=2.10.0 in c:\users\ahmed\anaconda3\lib\site-packages (from tensorflow==2.3.1) (2.1
```

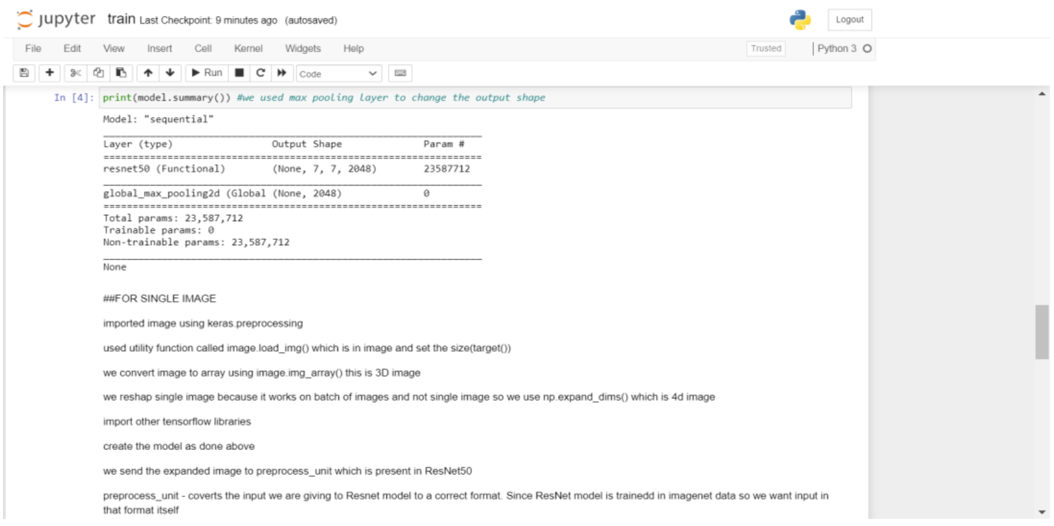


```
      Successfully uninstalled numpy-1.20.1
Successfully installed absl-py-1.2.0 astunparse-1.6.3 gast-0.3.3 google-auth-2.9.1 google-auth-oauthlib-0.4.6 google-pasta-0.2.
0 grpcio-1.48.0 importlib-metadata-4.12.0 keras-preprocessing-1.1.2 markdown-3.4.1 numpy-1.18.5 oauthlib-3.2.0 opt-einsum-3.3.0
protobuf-3.19.4 pyasn1-0.4.8 pyasn1-modules-0.2.8 requests-oauthlib-1.3.1 rsa-4.9 tensorboard-2.9.1 tensorboard-data-server-0.
6.1 tensorboard-plugin-wit-1.8.1 tensorflow-2.3.1 tensorflow-estimator-2.3.0 termcolor-1.1.0
```

```
In [2]: #We generate embeddings for the images - embeddings in ResNet are 2048nset of numbers representing the image for this we use Res
        import tensorflow
        from tensorflow.keras.preprocessing import image  #To parse the images
        from tensorflow.keras.layers import GlobalMaxPooling2D
        from tensorflow.keras.applications.resnet50 import ResNet50,preprocess_input
```

```
In [3]: #we create our ResNet model by calling ResNet class
        #we use weights which are trained on imagenet dataset
        #include_top=False because we will add our own top layer
        #input_shape=(224,224,3) we are using the standard size of the images for the input images - it scales down the images of larger
        model = ResNet50(weights='imagenet',include_top=False,input_shape=(224,224,3))
        #We are not training our model since its already trained on imagenet dataset
        model.trainable = False

        #we are giving our own top layer using sequential module
        #sequential helps us in creating model layers and adding them
        model = tensorflow.keras.Sequential([
            model,  #first we are passing our model
            GlobalMaxPooling2D() #then we are passing GlobalMaxPooling2D layer i.e we addedour own top layer
        ])
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_k
ernels_notop.h5
94773248/94765736 [==============================] - 38s 0us/step
```


jupyter train Last Checkpoint: 9 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

```
In [4]: print(model.summary()) #we used max pooling layer to change the output shape
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
resnet50 (Functional)        (None, 7, 7, 2048)        23587712  
_________________________________________________________________
global_max_pooling2d (Global (None, 2048)              0         
=================================================================
Total params: 23,587,712
Trainable params: 0
Non-trainable params: 23,587,712
_________________________________________________________________
None
```

##FOR SINGLE IMAGE

imported image using keras.preprocessing

used utility function called image.load_img() which is in image and set the size(target())

we convert image to array using image.img_array() this is 3D image

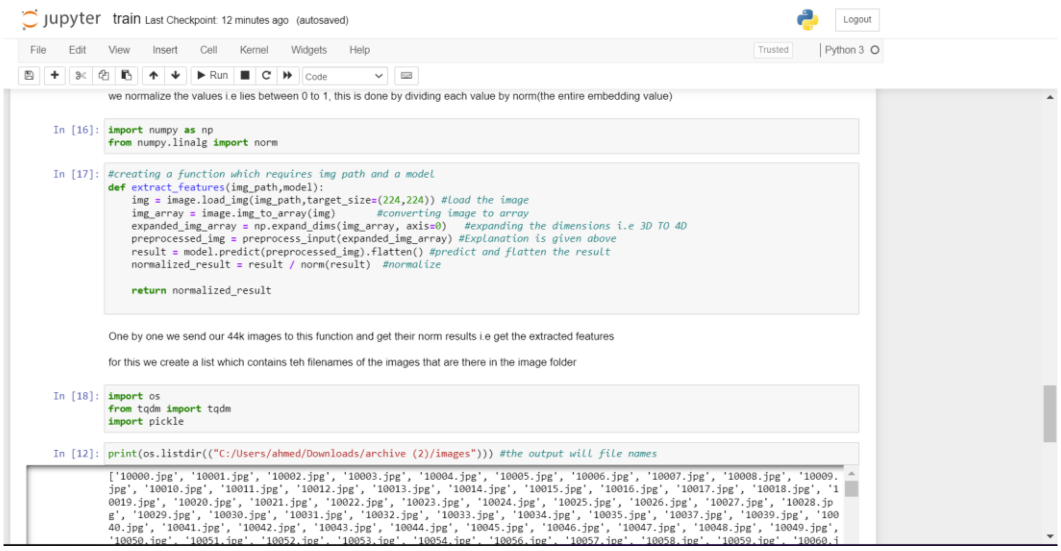
we reshap single image because it works on batch of images and not single image so we use np.expand_dims() which is 4d image

import other tensorflow libraries

create the model as done above

we send the expanded image to preprocess_unit which is present in ResNet50

preprocess_unit - coverts the input we are giving to Resnet model to a correct format. Since ResNet model is trainedd in imagenet data so we want input in that format itself


jupyter train Last Checkpoint: 12 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

we normalize the values i.e lies between 0 to 1, this is done by dividing each value by norm(the entire embedding value)

```
In [16]: import numpy as np
         from numpy.linalg import norm
```

```
In [17]: #creating a function which requires img path and a model
         def extract_features(img_path,model):
             img = image.load_img(img_path,target_size=(224,224)) #load the image
             img_array = image.img_to_array(img)       #converting image to array
             expanded_img_array = np.expand_dims(img_array, axis=0)   #expanding the dimensions i.e 3D TO 4D
             preprocessed_img = preprocess_input(expanded_img_array) #Explanation is given above
             result = model.predict(preprocessed_img).flatten() #predict and flatten the result
             normalized_result = result / norm(result)  #normalize

             return normalized_result
```

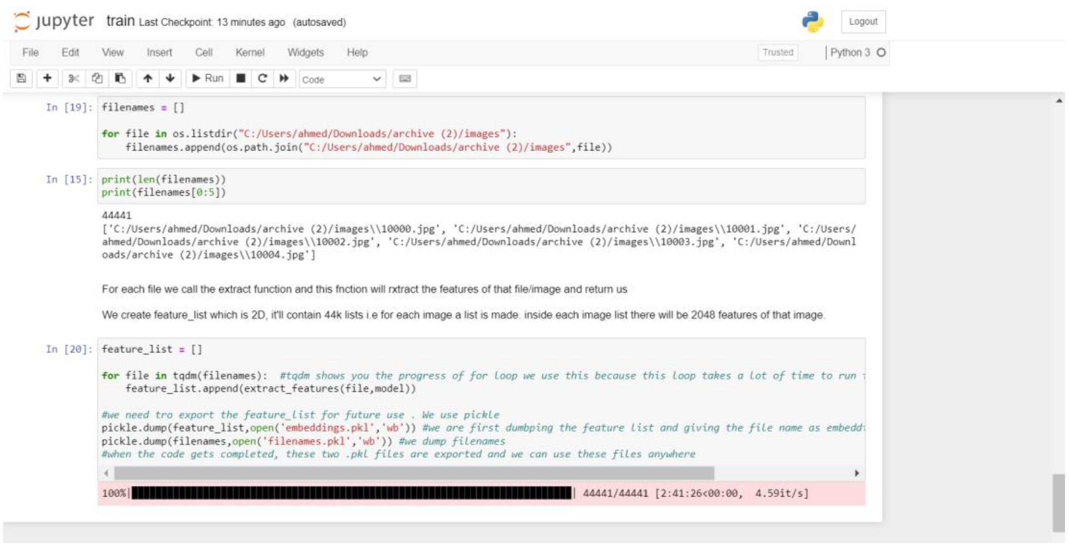
One by one we send our 44k images to this function and get their norm results i.e get the extracted features

for this we create a list which contains teh filenames of the images that are there in the image folder

```
In [18]: import os
         from tqdm import tqdm
         import pickle
```

```
In [12]: print(os.listdir(("C:/Users/ahmed/Downloads/archive (2)/images"))) #the output will file names
```

```
['10000.jpg', '10001.jpg', '10002.jpg', '10003.jpg', '10004.jpg', '10005.jpg', '10006.jpg', '10007.jpg', '10008.jpg', '10009.
jpg', '10010.jpg', '10011.jpg', '10012.jpg', '10013.jpg', '10014.jpg', '10015.jpg', '10016.jpg', '10017.jpg', '10018.jpg', '1
0019.jpg', '10020.jpg', '10021.jpg', '10022.jpg', '10023.jpg', '10024.jpg', '10025.jpg', '10026.jpg', '10027.jpg', '10028.jp
g', '10029.jpg', '10030.jpg', '10031.jpg', '10032.jpg', '10033.jpg', '10034.jpg', '10035.jpg', '10037.jpg', '10039.jpg', '100
40.jpg', '10041.jpg', '10042.jpg', '10043.jpg', '10044.jpg', '10045.jpg', '10046.jpg', '10047.jpg', '10048.jpg', '10049.jpg',
'10050.jpg', '10051.jpg', '10052.jpg', '10053.jpg', '10054.jpg', '10056.jpg', '10057.jpg', '10058.jpg', '10059.jpg', '10060.j
```


jupyter train Last Checkpoint: 13 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

```
In [19]: filenames = []

         for file in os.listdir("C:/Users/ahmed/Downloads/archive (2)/images"):
             filenames.append(os.path.join("C:/Users/ahmed/Downloads/archive (2)/images",file))
```

```
In [15]: print(len(filenames))
         print(filenames[0:5])
```

```
44441
['C:/Users/ahmed/Downloads/archive (2)/images\\10000.jpg', 'C:/Users/ahmed/Downloads/archive (2)/images\\10001.jpg', 'C:/Users/
ahmed/Downloads/archive (2)/images\\10002.jpg', 'C:/Users/ahmed/Downloads/archive (2)/images\\10003.jpg', 'C:/Users/ahmed/Downl
oads/archive (2)/images\\10004.jpg']
```

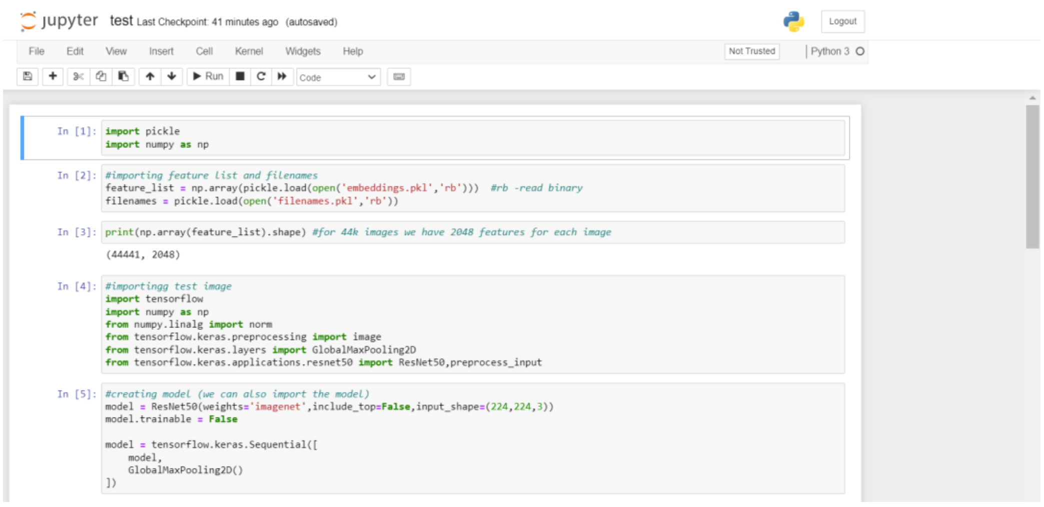
For each file we call the extract function and this fnction will rxtract the features of that file/image and return us

We create feature_list which is 2D, it'll contain 44k lists i.e for each image a list is made. inside each image list there will be 2048 features of that image.

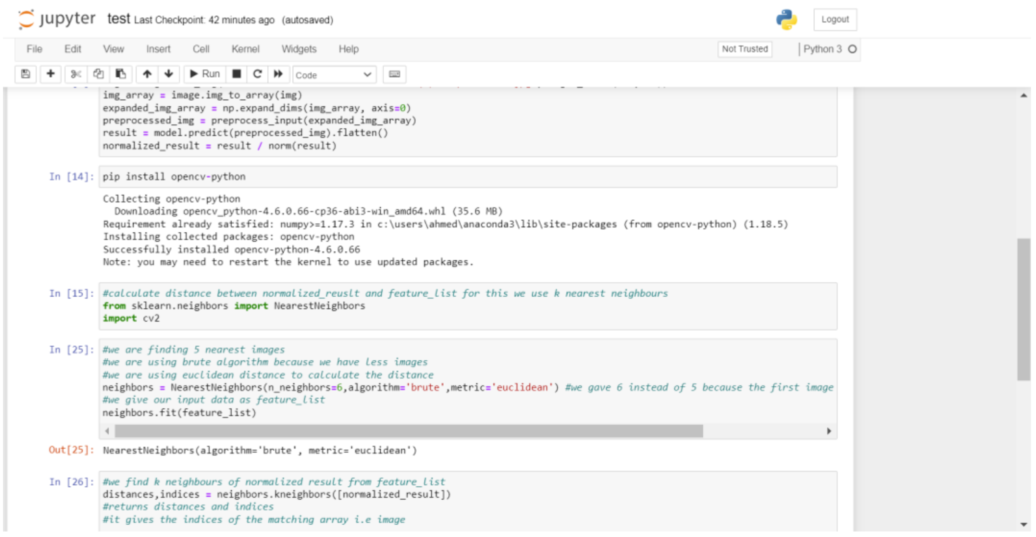
```
In [20]: feature_list = []

         for file in tqdm(filenames):  #tqdm shows you the progress of for loop we use this because this loop takes a lot of time to run
             feature_list.append(extract_features(file,model))

         #we need tro export the feature_list for future use . We use pickle
         pickle.dump(feature_list,open('embeddings.pkl','wb')) #we are first dumbping the feature list and giving the file name as embedd
         pickle.dump(filenames,open('filenames.pkl','wb')) #we dump filenames
         #when the code gets completed, these two .pkl files are exported and we can use these files anywhere
```

```
100%|██████████████████████████████████████████████████████| 44441/44441 [2:41:26<00:00,  4.59it/s]
```

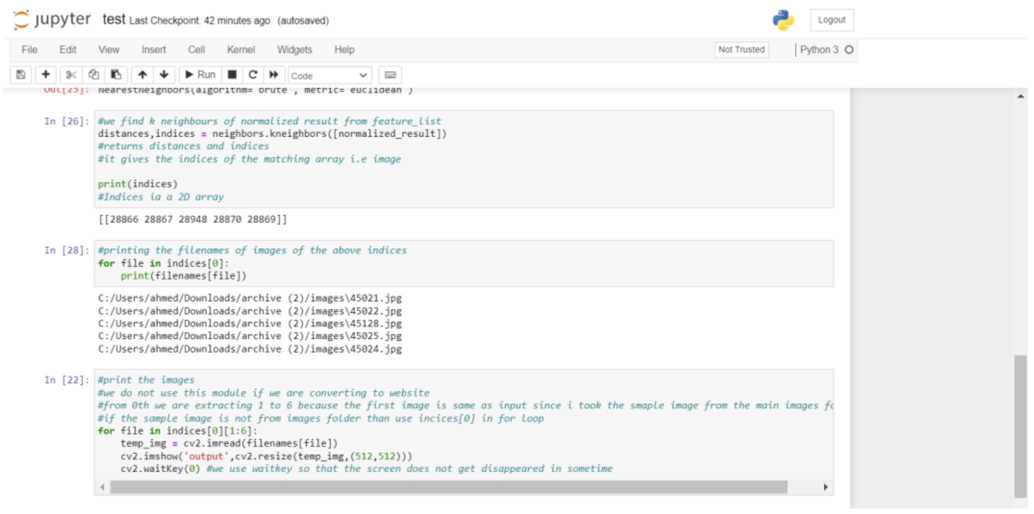
c) Test File


jupyter test Last Checkpoint: 41 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code

```
In [1]: import pickle
        import numpy as np

In [2]: #importing feature list and filenames
        feature_list = np.array(pickle.load(open('embeddings.pkl','rb')))  #rb -read binary
        filenames = pickle.load(open('filenames.pkl','rb'))

In [3]: print(np.array(feature_list).shape) #for 44k images we have 2048 features for each image

        (44441, 2048)

In [4]: #importingg test image
        import tensorflow
        import numpy as np
        from numpy.linalg import norm
        from tensorflow.keras.preprocessing import image
        from tensorflow.keras.layers import GlobalMaxPooling2D
        from tensorflow.keras.applications.resnet50 import ResNet50,preprocess_input

In [5]: #creating model (we can also import the model)
        model = ResNet50(weights='imagenet',include_top=False,input_shape=(224,224,3))
        model.trainable = False

        model = tensorflow.keras.Sequential([
            model,
            GlobalMaxPooling2D()
        ])
```

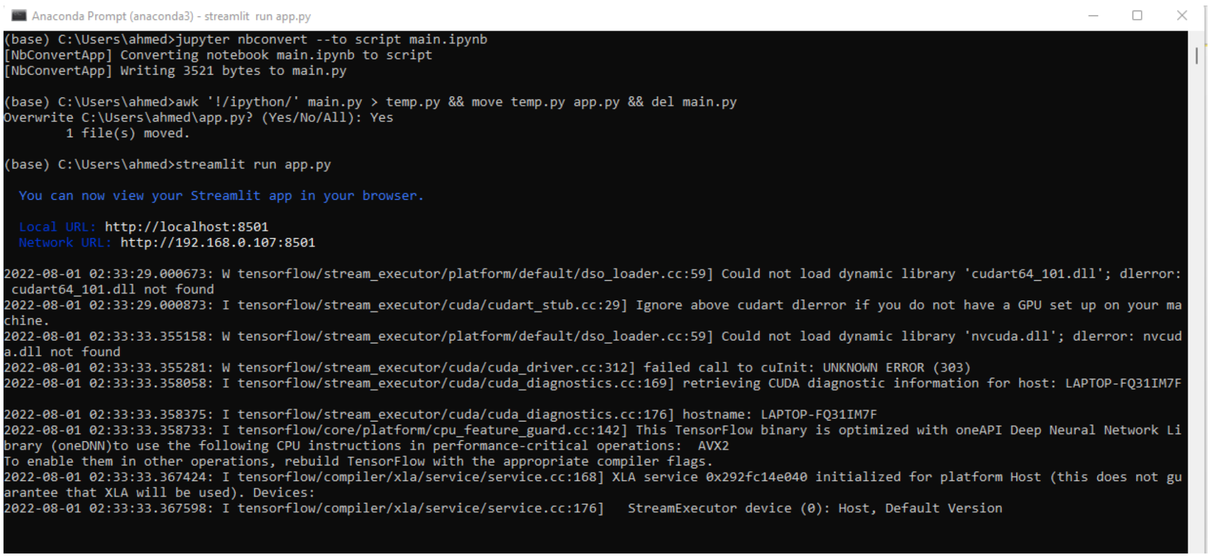

jupyter test Last Checkpoint: 42 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code

```
         img_array = image.img_to_array(img)
         expanded_img_array = np.expand_dims(img_array, axis=0)
         preprocessed_img = preprocess_input(expanded_img_array)
         result = model.predict(preprocessed_img).flatten()
         normalized_result = result / norm(result)

In [14]: pip install opencv-python

         Collecting opencv-python
           Downloading opencv_python-4.6.0.66-cp36-abi3-win_amd64.whl (35.6 MB)
         Requirement already satisfied: numpy>=1.17.3 in c:\users\ahmed\anaconda3\lib\site-packages (from opencv-python) (1.18.5)
         Installing collected packages: opencv-python
         Successfully installed opencv-python-4.6.0.66
         Note: you may need to restart the kernel to use updated packages.

In [15]: #calculate distance between normalized_reuslt and feature_list for this we use k nearest neighbours
         from sklearn.neighbors import NearestNeighbors
         import cv2

In [25]: #we are finding 5 nearest images
         #we are using brute algorithm because we have less images
         #we are using euclidean distance to calculate the distance
         neighbors = NearestNeighbors(n_neighbors=6,algorithm='brute',metric='euclidean') #we gave 6 instead of 5 because the first image
         #we give our input data as feature_list
         neighbors.fit(feature_list)

Out[25]: NearestNeighbors(algorithm='brute', metric='euclidean')

In [26]: #we find k neighbours of normalized result from feature_list
         distances,indices = neighbors.kneighbors([normalized_result])
         #returns distances and indices
         #it gives the indices of the matching array i.e image
```


jupyter test Last Checkpoint: 42 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code

```
In [26]: #we find k neighbours of normalized result from feature_list
         distances,indices = neighbors.kneighbors([normalized_result])
         #returns distances and indices
         #it gives the indices of the matching array i.e image

         print(indices)
         #Indices ia a 2D array

         [[28866 28867 28948 28870 28869]]

In [28]: #printing the filenames of images of the above indices
         for file in indices[0]:
             print(filenames[file])

         C:/Users/ahmed/Downloads/archive (2)/images\45021.jpg
         C:/Users/ahmed/Downloads/archive (2)/images\45022.jpg
         C:/Users/ahmed/Downloads/archive (2)/images\45128.jpg
         C:/Users/ahmed/Downloads/archive (2)/images\45025.jpg
         C:/Users/ahmed/Downloads/archive (2)/images\45024.jpg

In [22]: #print the images
         #we do not use this module if we are converting to website
         #from 0th we are extracting 1 to 6 because the first image is same as input since i took the smaple image from the main images fo
         #if the sample image is not from images folder than use incices[0] in for loop
         for file in indices[0][1:6]:
             temp_img = cv2.imread(filenames[file])
             cv2.imshow('output',cv2.resize(temp_img,(512,512)))
             cv2.waitKey(0) #we use waitkey so that the screen does not get disappeared in sometime
```

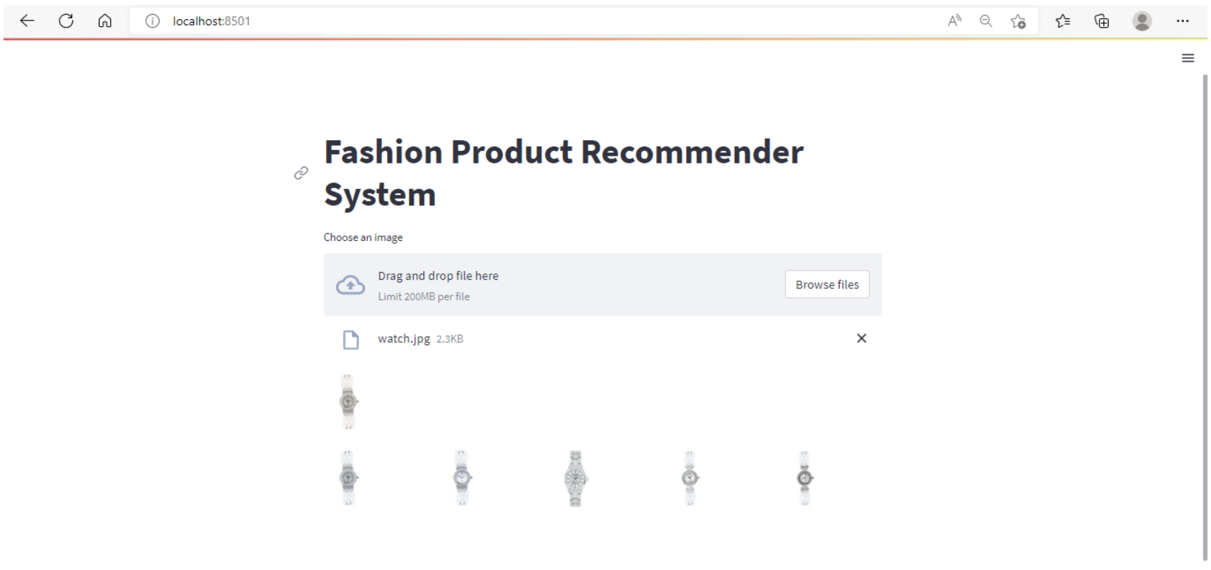
➔ Output



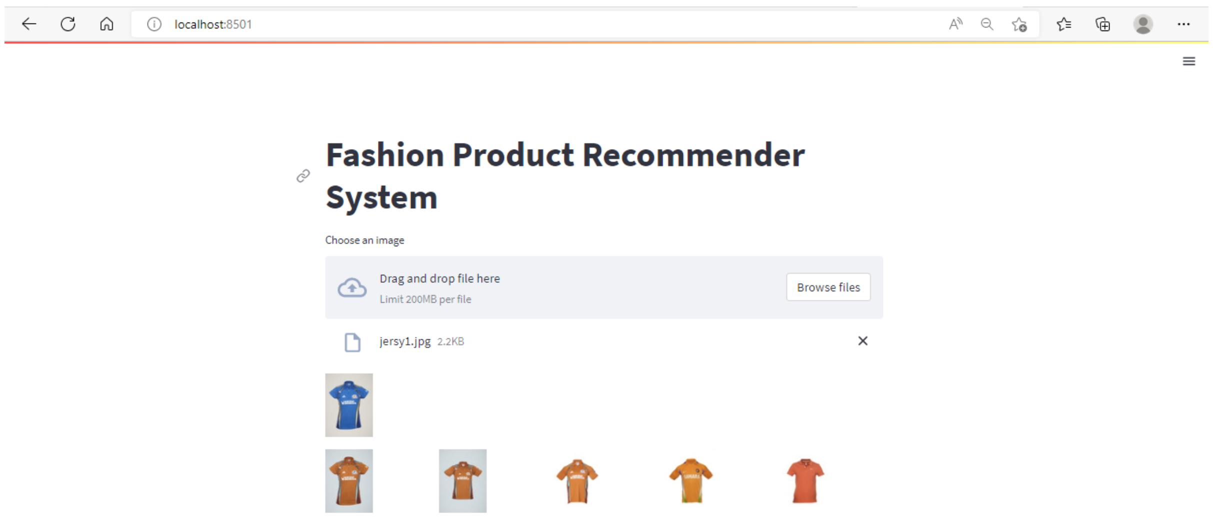
Anaconda Prompt (anaconda3) - streamlit run app.py

```
(base) C:\Users\ahmed>jupyter nbconvert --to script main.ipynb
[NbConvertApp] Converting notebook main.ipynb to script
[NbConvertApp] Writing 3521 bytes to main.py

(base) C:\Users\ahmed>awk '!/ipython/' main.py > temp.py && move temp.py app.py && del main.py
Overwrite C:\Users\ahmed\app.py? (Yes/No/All): Yes
        1 file(s) moved.

(base) C:\Users\ahmed>streamlit run app.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://192.168.0.107:8501

2022-08-01 02:33:29.000673: W tensorflow/stream_executor/platform/default/dso_loader.cc:59] Could not load dynamic library 'cudart64_101.dll'; dlerror:
 cudart64_101.dll not found
2022-08-01 02:33:29.000873: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your ma
chine.
2022-08-01 02:33:33.355158: W tensorflow/stream_executor/platform/default/dso_loader.cc:59] Could not load dynamic library 'nvcuda.dll'; dlerror: nvcud
a.dll not found
2022-08-01 02:33:33.355281: W tensorflow/stream_executor/cuda/cuda_driver.cc:312] failed call to cuInit: UNKNOWN ERROR (303)
2022-08-01 02:33:33.358058: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:169] retrieving CUDA diagnostic information for host: LAPTOP-FQ31IM7F

2022-08-01 02:33:33.358375: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:176] hostname: LAPTOP-FQ31IM7F
2022-08-01 02:33:33.358733: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binary is optimized with oneAPI Deep Neural Network Li
brary (oneDNN)to use the following CPU instructions in performance-critical operations:  AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2022-08-01 02:33:33.367424: I tensorflow/compiler/xla/service/service.cc:168] XLA service 0x292fc14e040 initialized for platform Host (this does not gu
arantee that XLA will be used). Devices:
2022-08-01 02:33:33.367598: I tensorflow/compiler/xla/service/service.cc:176]   StreamExecutor device (0): Host, Default Version
```

Prompt window showing the conversion of Jupyter Notebook file to .py and the running of Streamlit on prompt.



Application window showing the working of the project. Here we have uploaded an image of a watch and it yielded images of five other similar looking watches.



Application window showing the working of the project. Here we have uploaded an image of a jersey and it yielded images of five other jerseys that are available.

CHAPTER VII

“EUREKA!” AND “OH NO!”

“Eureka!” Moment:

For the purpose of this project, we needed to create a UI. For this, we were planning to use Streamlit through Jupyter. When we enquired about with others, they said it would never be possible. Apart from the others, Google too showed that this would not be possible. However, after much research and trials and errors, we were able to crack it. We were finally able to integrate Streamlit with Jupyter. That’s the exact moment when we went “Eureka!”.

“Oh No!” Moment:

There was this huge chunk of code that we were trying to run. However, each time that we ran it, it threw errors. While trying to overcome this, we ran this chunk of code over and over again. With some guidance, we later came to know that the error could be very easily fixed. All it required was for us to install the OpenCV2 package. Knowing that we had wasted so much of our time because of missing out on one line of code was our “Oh No!” moment.

CHAPTER VIII

LEARNINGS FROM THE PROJECT

As mentioned earlier, we wanted to be on the either side of the recommender system. Being on the other side, we did learn a lot of things. But most of all, we have learnt to appreciate the recommender systems that we use on an almost day-to-day basis. The accuracy and the speed with which products are recommended are truly commendable.

Working on this project has given us the privilege of gaining some knowledge and some hands-on working on some techniques that we only otherwise heard of.

- CNN
  We had only heard of it and read about it theoretically before this project. Now, we have a hands-on experience of working with CNN. We are able to now understand how exactly it works and are also seeing a real-time application of CNN in action.
- Streamlit
  We were able to work with Streamlit thereby learning how to build a UI for our upcoming projects.
- Working with Images
  Prior to this project, working with images would scare us. Having worked solely on images and understanding the feature extraction process, we are now comfortable enough to work with images.

Overall, through this project, we were exposed to various libraries and techniques. We are sure that this acquired knowledge will come in handy in our future endeavours.

CHAPTER IX

WHAT MORE CAN BE DONE IN THE FUTURE

We were able to achieve our primary goal through this project. There are, however, many more things that can be done in the future.

- A stand-alone application can be built with a more interactive and attractive GUI (Graphical User Interface).
- This project was made with only one domain in mind – Fashion. The same code and logic can be used across domains.
- A menu-driven program can be built to search only across certain categories of products as specified by the users.
- This can also be used to update inventory. When a user uploads an image and only limited options show, it should alert the concerned company to update their stock.

*******   ******