# **Algorithmic Digital Marketing Assignment 3**

## **IMPLEMENTING VISUAL SEARCH**

Marketing Analytical Solutions

|  |  |
| --- | --- |
| **Summary** | Implementing similarity search at scale and various searching algorithms, including search using Elastic Search to create a streamlit application and deploy it on Heroku |
| **Data Source** | <https://www.kaggle.com/c/cdiscount-image-classification-challenge> |
| **Category** | Web |
| **Author** | Anurag Rachcha, Gauri Verma, Shabham Mahajan |

# 

**GOAL:**

* Implement various recommendation systems
* Help users see items related to the one they recently viewed
* Enhance user online shopping experience

# INTRODUCTION

**Similarity search** is the most general term used for a range of mechanisms which share the principle of searching (typically, very large) spaces of objects where the only available comparator is the similarity between any pair of objects. Nearest neighbor sear[ch](https://en.wikipedia.org/wiki/Nearest_neighbor_search) and range queries are important subclasses of similarity search, and a number of solutions exist. The most general approach to similarity search relies upon the mathematical notion of metric space, which allows the construction of efficient index structures in order to achieve scalability in the search domain.

We will be implementing following three methods for similarity search of images-

1. **Cosine Similarity:** This algorithm contains both procedures and functions to calculate similarity between sets of data. The function is best used when calculating the similarity between small numbers of sets. The procedures parallelize the computation and are therefore more appropriate for computing similarities on bigger datasets. Cosine similarity is the cosine of the angle between two *n*-dimensional vectors in an *n*-dimensional space.
2. **Facebook Artificial Intelligence Similarity Search (FAISS):** A library that allows us to quickly search for multimedia documents that are similar to each other — a challenge where traditional query search engines fall short. Facebook has built nearest-neighbor search implementations for billion-scale data sets that are some 8.5x faster than the previous reported state-of-the-art, along with the fastest k-selection algorithm on the GPU known in the literature.
3. **Spotify Annoy (Approximate Nearest Neighbors Oh Yeah):** Annoy is a C++ library with Python bindings to search for points in space that are close to a given query point. It also creates large read-only file-based data structures that are mmapped into memory so that many processes may share the same data. Annoy has the ability to use static files as indexes which allows us to share indexes across processes.

# ABOUT THE DATASET

We will be using the Cdiscount dataset for our image similarity search. Cdiscount.com is France’s largest non-food e-commerce company. While the company already sells everything from TVs to trampolines, the list of products is still rapidly growing. Cdiscount.com has over 30 million products up for sale. Below is a brief overview of the data used in our analysis-

* train.bson - (Size: 58.2 GB) Contains a list of 7,069,896 dictionaries, one per product. Each dictionary contains a product id (key: \_id), the category id of the product (key: category\_id), and between 1-4 images, stored in a list (key: imgs). Each image list contains a single dictionary per image, which uses the format: {'picture': b'...binary string...'}. The binary string corresponds to a binary representation of the image in JPEG format.
* train\_example.bson - Contains the first 100 records of train.bson for exploring the dataset before downloading the entire file.
* test.bson - (Size: 14.5 GB) Contains a list of 1,768,182 products in the same format as train.bson, except there is no category\_id included. The objective of the competition is to predict the correct category\_id from the picture(s) of each product id (\_id). The category\_ids that are present in Private Test split are also all present in the Public Test split.
* category\_names.csv - Shows the hierarchy of product classification. Each category\_id has a corresponding level1, level2, and level3 name, in French. The category\_id corresponds to the category tree down to its lowest level.

# INGESTION AND PRE-PROCESSING

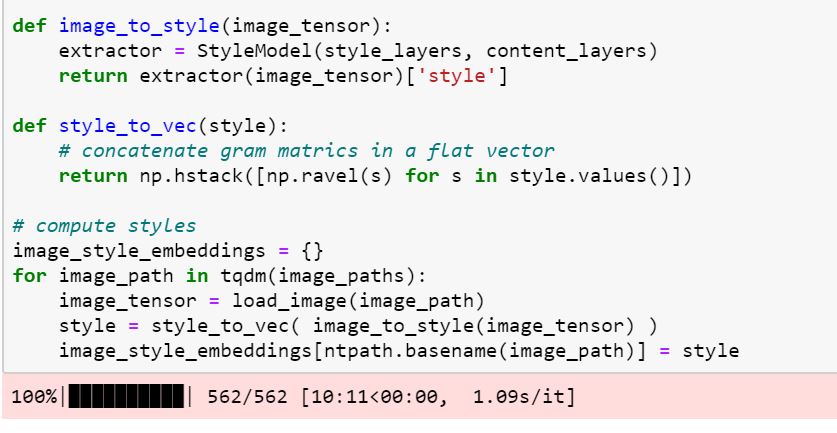
Pre-processing of the bson file was done using the code ‘preprocessing.ipynb’ to stream the images and extract images for implementation of the algorithms.

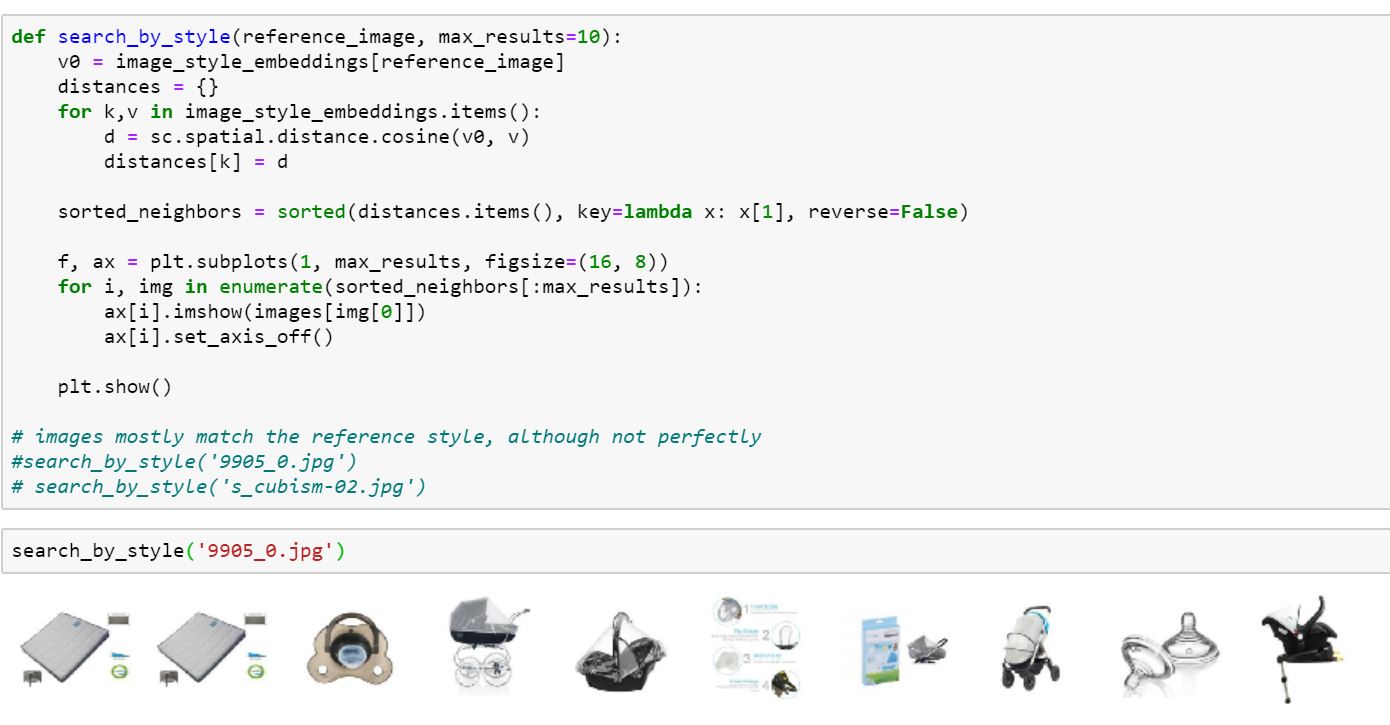
Due to the format of the data , we processed the data using python. Python helped us slice and stream the data as it iterated over each data frame in the dataset

# SIMILARITY SEARCH ( Version 1 )

* It uses tensorflow and keras model to train the pool of image that we have.
* Used the concept of gram matrix and style embedding is computed as a concatenation of gram matrices of the style layers.
* Cosine values were evaluated for nearest neighbors

The following screenshots depict the training and implementation in jupyter notebook:

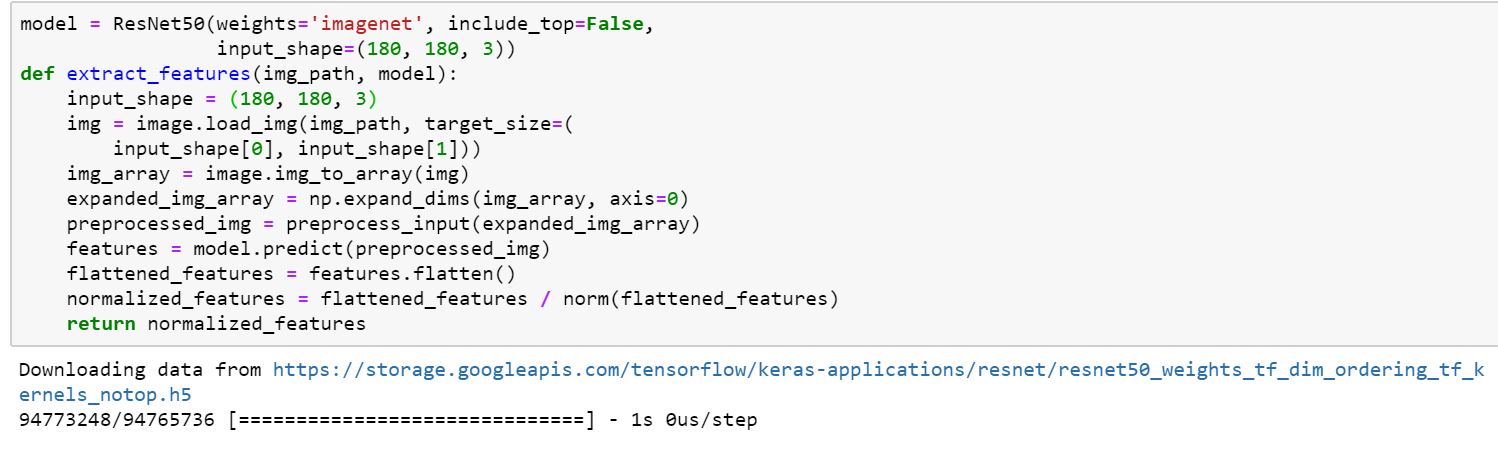




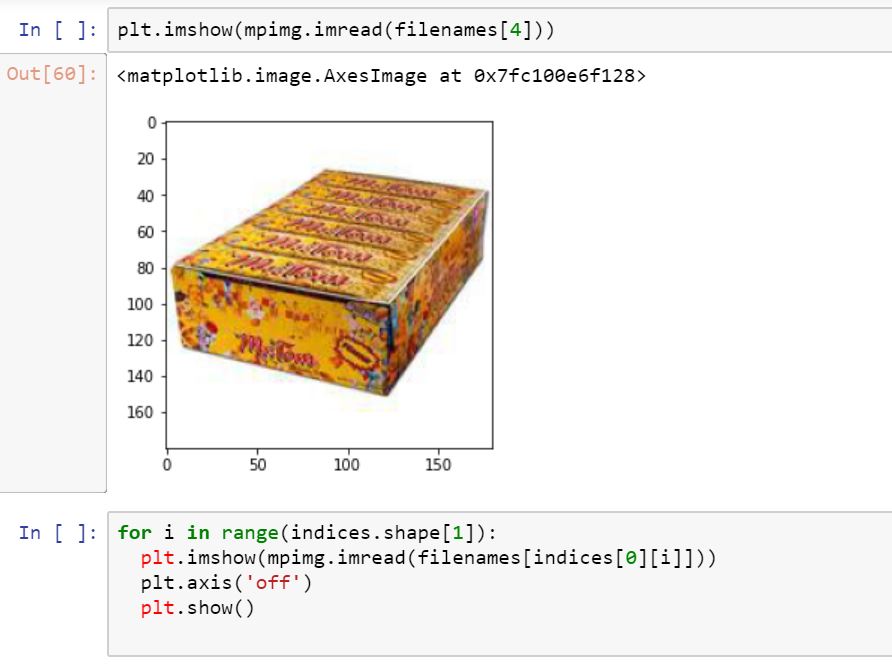
# SIMILARITY SEARCH ( Version 2 )

Facebook AI Similarity Search (FAISS)

* The algorithm helps convert visual information into a powerful visual search engine.
* It is a learning model that maps images into numeric vectors where more similar images are mapped closer to each other in a vector space.
* The representation learning model enables us to find images that are similar to a query image by searching for images whose vectors are close to the query’s vector.
* Uploading and image and extracting features:



* Passing an image



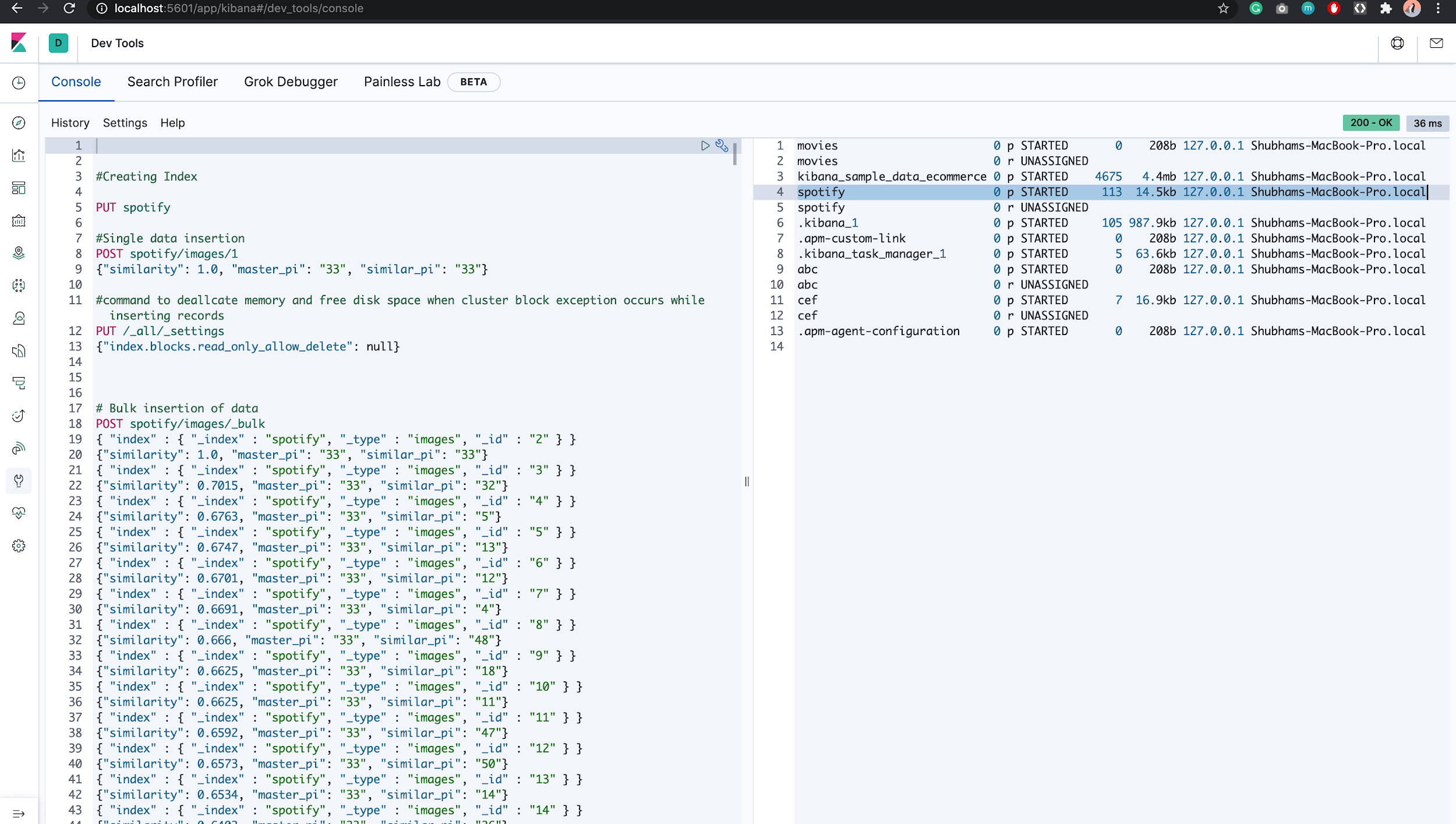
* Displaying nearest neighbors

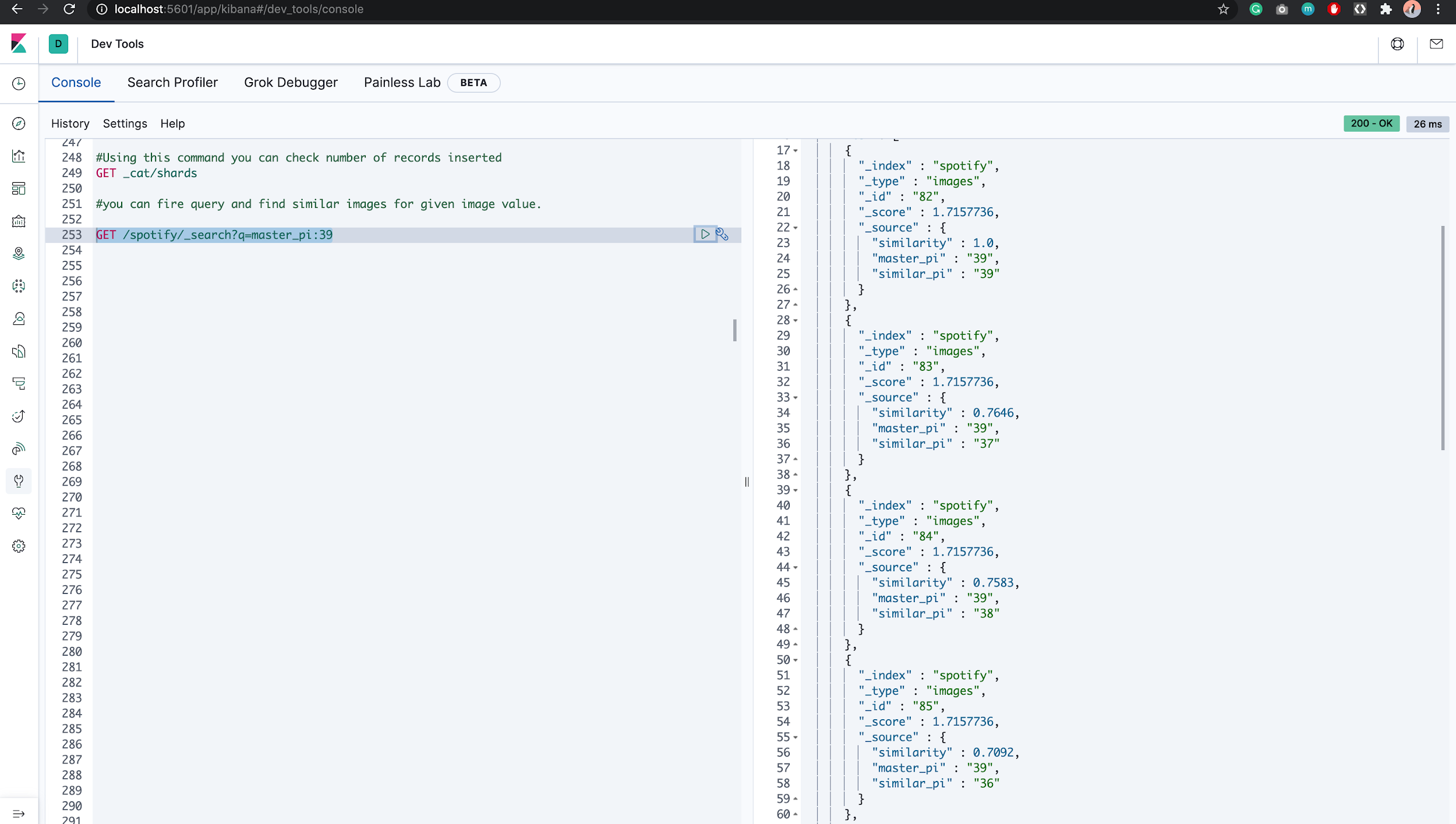


# ELASTIC SEARCH ON Version 2

**What is Elastic Search and indexing?**

* Elasticsearch is a distributed, open source search and analytics engine for all types of data, including textual, numerical, geospatial, structured, and unstructured
* An Elasticsearch index is a collection of documents that are related to each other. Elasticsearch stores data as JSON documents. Each document correlates a set of keys (names of fields or properties) with their corresponding values (strings, numbers, Booleans, dates, arrays of values, geolocations, or other types of data).
* Implementation



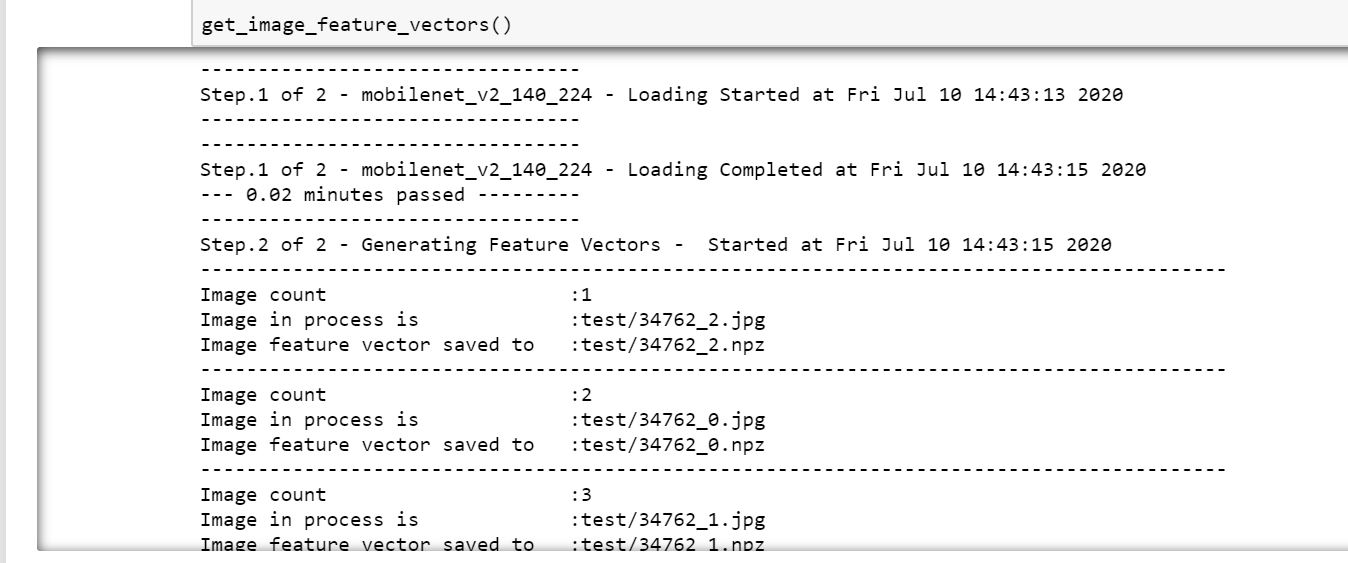


# SIMILARITY SEARCH ( Version 3 )

Spotify- Annoy Method

* Spotify Annoy is implemented using Tensorflow 2.0 and Tensorflow Hub
* Spotify Annoy method uses the concept of image feature vector. An image feature vector is a list of numbers that represents a whole image, typically used for image similarity calculations or image classification tasks
* The main purpose of this script is to generate image feature vectors by reading image files located in a local folder.
* It has two functions: load\_img() and get\_image\_feature\_vectors().
* load\_img(path) gets file names which are provided as an argument of the function. Then loads and pre-process the images so that we can use them in our MobilenetV2 CNN model.
* Annoy (Approximate Nearest Neighbor Oh Yeah), is an open-sourced library for approximate nearest neighbor implementation.We will use it to find the image feature vectors in a given set that is closest (or most similar) to a given feature vector.
* Implementation:

The output of the code resulted in creating vectors for the images in npz format.



# REFERENCE APP USING STREAMLIT

**What is Streamlit?**

* Streamlit is an open-source Python library that makes it easy to build beautiful custom web-apps for machine learning and data science.
* To use it, just pip install streamlit , then import it, write a couple lines of code, and run your script with streamlit run [filename] .
* Streamlit watches for changes on each save and updates the app live while you're coding. Code runs from top to bottom, always from a clean state, and with no need for callbacks.
* The application can be viewed on <http://localhost:8501/>

**Installation steps**

$ pip install streamlit

$ streamlit hello

**Imports**

import pandas as pd

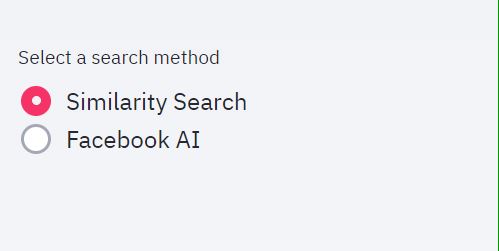
import streamlit as st

**Run Strealit app**

$ streamlit run app.py

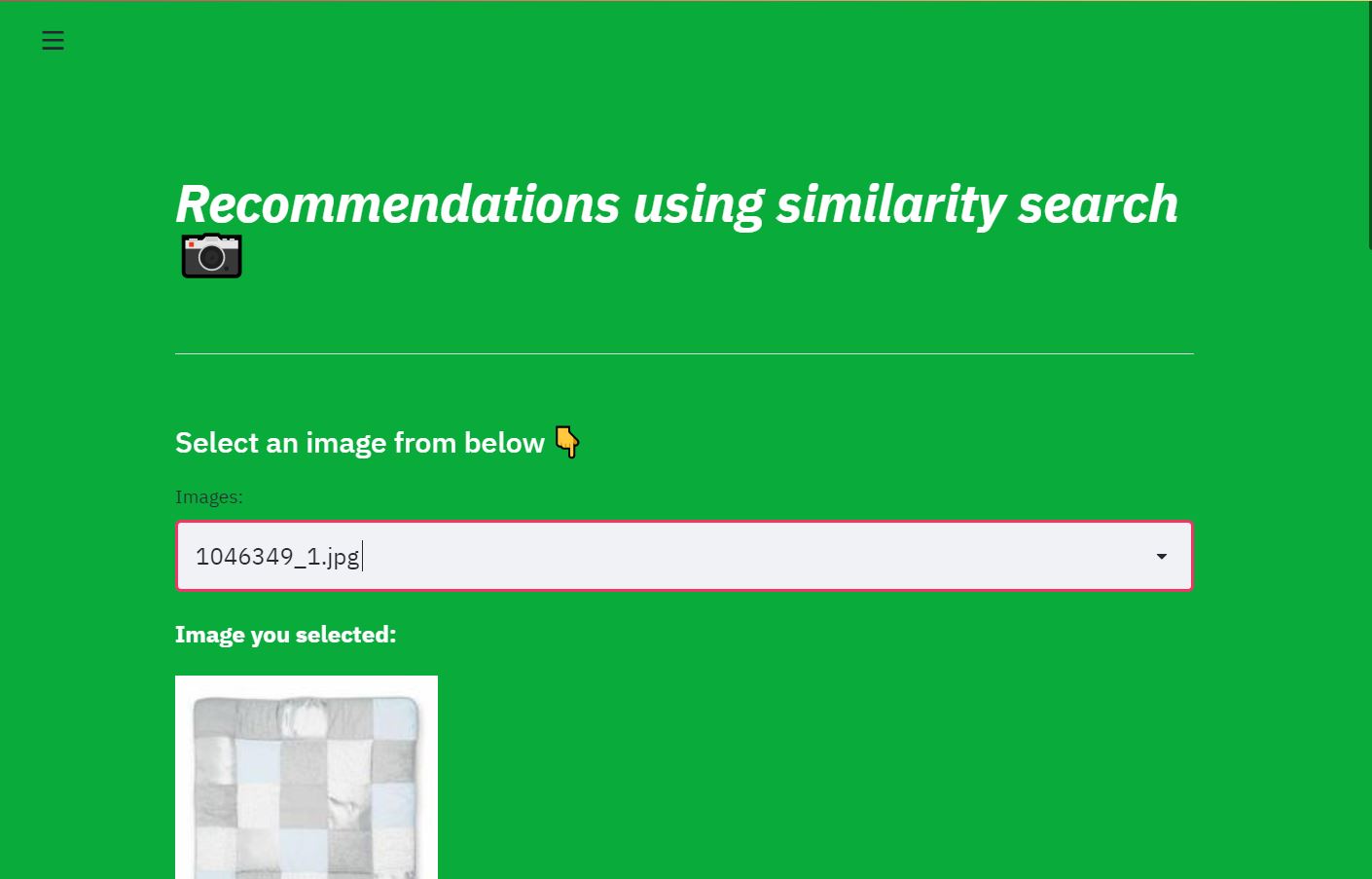
Implementation of Version 1 and Version 2 on streamlit:

* A radio button implementation to select from the algorithms



* User Interface of the application. Emoji’s, animations, and drop down menus are used to enhance user experience.

The background color and styling is implemented using style.css files.



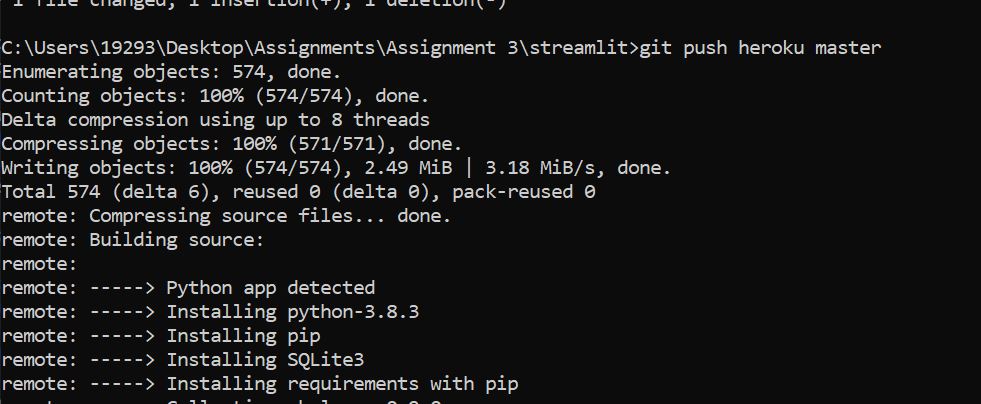
* Output returns the similar images from the one selected in the drop down menu.



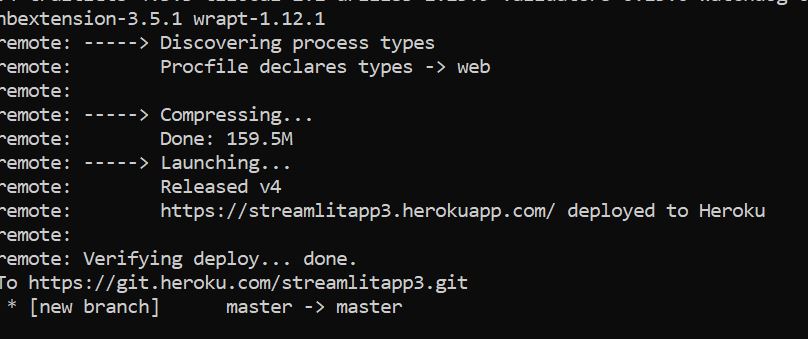
# DEPLOYMENT OF STREAMLIT ON HEROKU

**What is Heroku?**

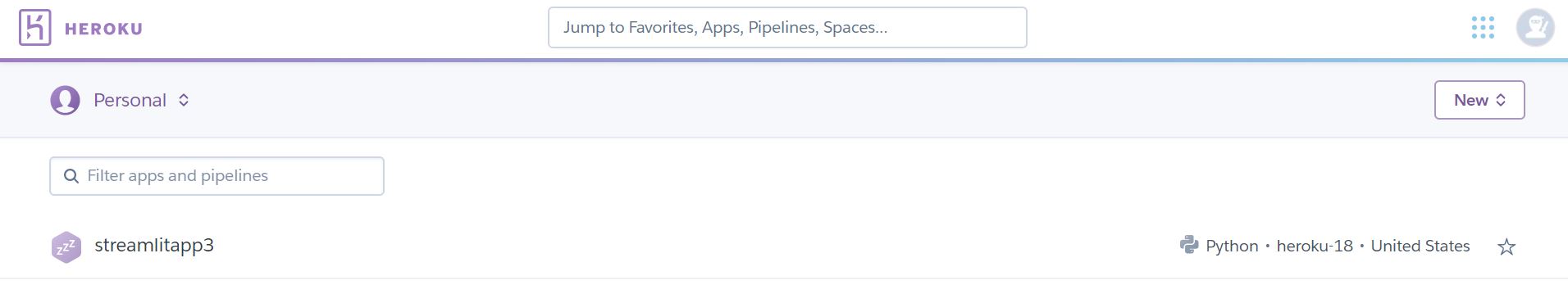
* Heroku is a container-based cloud Platform as a Service (PaaS).
* Developers use Heroku to deploy, manage, and scale modern apps
* The platform is elegant, flexible, and easy to use, offering developers the simplest path to getting their apps to market.
* After installation of Heroku CLI, the command line is used for deployment.
* Simple commands like ‘git push heroku master’ help in deployment after git setup



* Verification of deployment



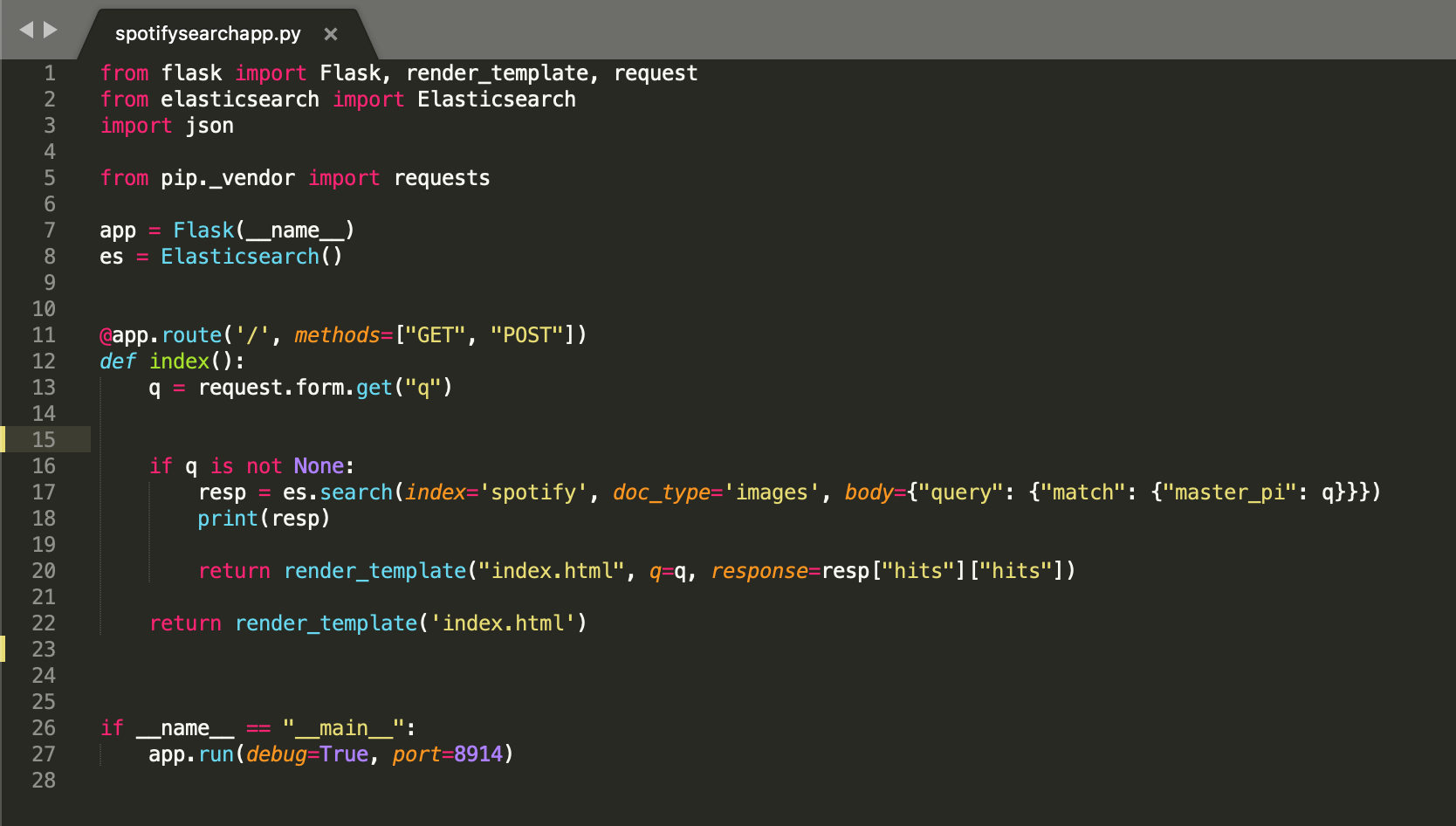
* Application can be viewed on heroku or on <https://streamlitapp3.herokuapp.com/>

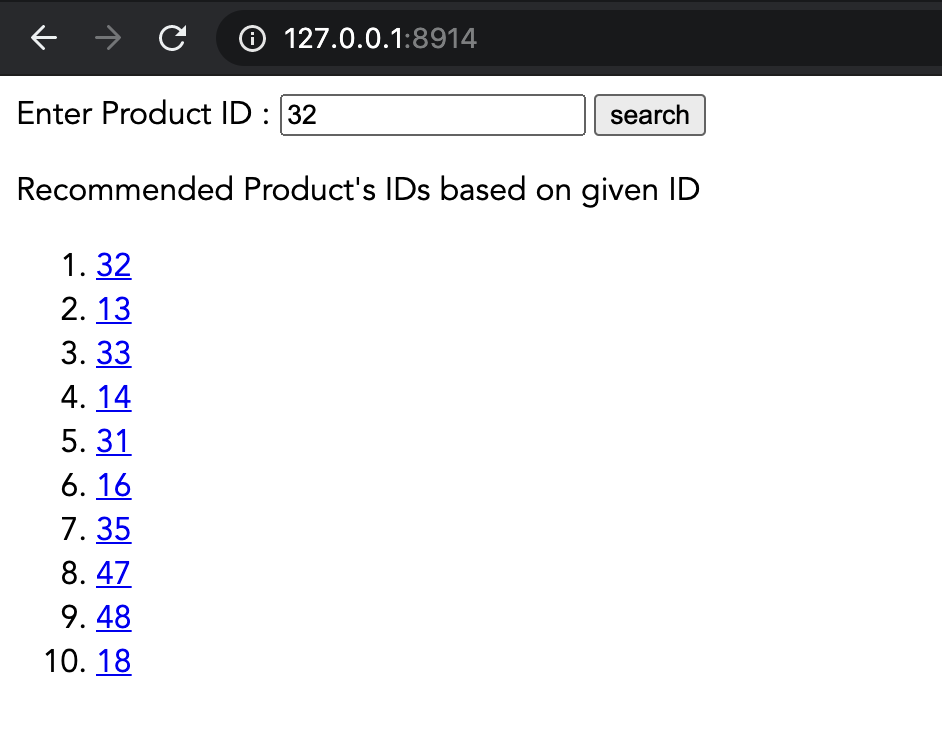


# REFERENCE APP USING FLASK

**What is Flask?**

* Flask is a micro web framework written in Python.
* It is classified as a microframework because it does not require particular tools or libraries.
* It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.





# CONCLUSION

* Implementation of 3 similarity search algorithms was carried out.
* Visual search results in helping customers in what they wish to buy, by giving them a visual representation of the products that suit their choices.
* It enhances user experience and helps attract more user engagement
* It helps brands increase their reach which eventually helps them in higher profit margins.

# ISSUES FACED

1. With the size of the json file, we struggled for a very long time to run it on our laptops with the given CPU and GPU compatibilities.
2. We could not stream the entire dataset, due to which we eventually interrupted the code after around 12000 iterations to work on ahead.
3. Wasted 9 hours running one code!
4. After spending days on one algorithm, moved to the next to a totally new tool: Elastic Search. Elastic search needed a different format of the json and the json we were able to create did not match it. Due to this, we ended up manually making the json file to be able to implement elastic search on few images.
5. MongoDB was another sudden addition to extract the json file.
6. We ‘assumed’ streamlit would be easy! While running streamlit, the training model kept running in the background which was very inefficient. With the help from TA, we were able to make a csv to use in streamlit for faster retrieval.
7. Heroku deployment gave us a lot of errors: from requirements.txt file to setup.sh - all of it had minor errors which were new to us.
8. While trying to use Flask, we have been only able to retrieve the image ID with its nearest neighbours and not the visual representation of the image itself.

# LESSONS FROM THIS ASSIGNMENT:

1. Always, ALWAYS, start working with sample data.
2. In a team, it is best to divide work and then share it with each other rather than the entire team working on one issue
3. If given another week for the assignment, we would go step by step rather than going all over the place and trying multiple things together. Pre planning the entire assignment along with keeping some margin time before the deadline is very important.
4. We as a team would like to learn more about Elastic Search as this is where we felt we had no clue about how it really works.
5. Patience, team- work and time management is very important.