Product Recommendation System using K-Means Algorithm

(Word Count: 1997)

Team 1

Abstract: **This report aims to build a product recommendation system and design a Web-app interface to display recommendations. The goal is to provide similar products' recommendations, to one selected by the user. The dataset used contains 10,000 images of clothing products. We used a clustering method, grouping items based on similarity. For data preprocessing, t-SNE algorithm was utilised to reduce dimensionality. The K-Means algorithm was then implemented to separate the pre-processed data into clusters. Through a Silhouette score analysis, we found the optimal k value to be 8 clusters. Results show our model was moderately good at clustering the data into its true labels, with a homogeneity score of 0.62 and a completeness score of 0.69. Our conclusions are that this is mostly due to underfitting by having a number of clusters k=8, less than the true number of 10 clusters. Moreover, the limitations of the feature extraction method results in some similar true labels being clustered together, therefore reducing the homogeneity of the generated clusters.**

I. Introduction

A retail company is looking to enhance their online customer experience by creating a Web App which implements an image-based product recommendation system from products viewed. Online e-commerce businesses offer large varieties of products making it difficult for customers to find the right products, limiting the company's profit. Recommendation systems create value by offering a personalised shopping experience, increasing customer engagement, sales and revenue. In 2017, 35% of consumer purchases on Amazon came from recommended items [1]. For an effective system, accurate results are required in real-time to reach customers while they are still on the site [2]. This increases time spent on the website and the number of items purchased.

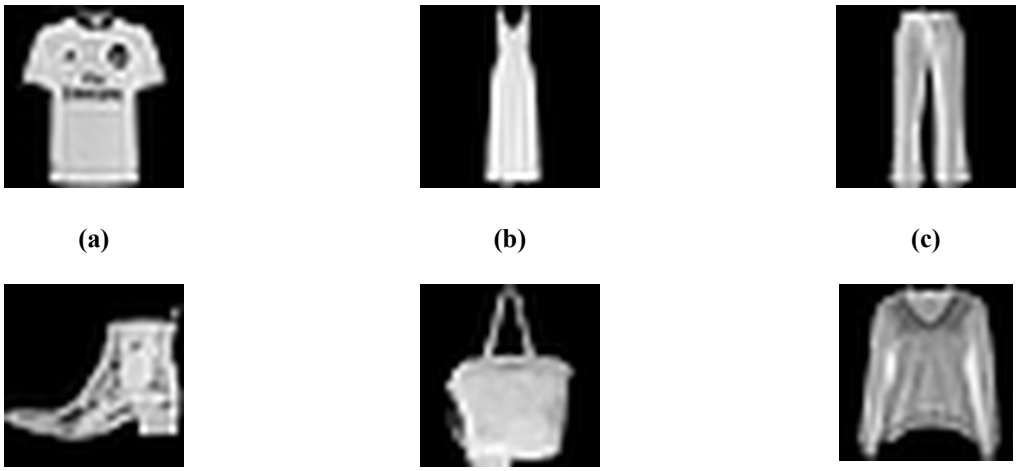
This report uses the K-Means algorithm to classify products to design a product recommendation system. The structure followed is visualising the data, evaluating the best methods for the algorithm, then implementing it to make predictions and assessing the accuracy of the results.

II. Data

A. Data Description

The dataset contains 10,000 black and white product images of clothing, each is 28 x 28 pixel. There are 784 dimensions where each row contains pixel data of images. Each pixel is associated with a pixel-value which indicates the lightness or darkness of that pixel. The pixel-value is an integer between 0 and 255. The lower the pixel number, the lighter the colour on a grey scale.

B. Data Visualisation



(a) **(b)** **(c)**

(d) **(e)** **(f)**

Fig. 1 Visualisations of row indexes 0, 49, 88, 749, 4180, 7896 respectively

III. Methods

A. K-means

K-means clustering is a method of unsupervised learning where data is grouped based on their similarity into k 'clusters'. The steps for the algorithm are:

1. Determine the number of clusters, k and randomly assign a centroid for each cluster
2. Each data point is assigned to a cluster in which its centroid is closest to.
3. After all points are assigned, positions of the centroids (mean value) of the clusters are recalculated.
4. Repeat steps 2 and 3 until the centroids no longer move.

$$\textbf{Eq.(1)} \quad : \qquad d(a,b) \;=\; ||\, b \;-\; a\,|| \;=\; \sqrt{\sum_{i=1}^{p} \left(X_{ai} \;-\; X_{bi} \right)^2}$$

At this point, the sum of squared Euclidean distances of points **Eq.(1)** in a cluster to its centroid and thus, the variance within each cluster is minimised and partitioning of points and their assignment to the clusters can be finalised.

Variables are assumed to be normally distributed, have the same variance; mean and standard deviation. Thus, we rescaled all variables to mean=0 and variance=1. K-means aims to create clusters with minimal inter cluster variation, ensuring that observations within each cluster are similar and different across clusters [3].

B. Approaches to Determine number of clusters, k

Approaches to determine optimal k includes: Silhouette Method, Elbow Method and Gap statistic. We have mainly used the silhouette method to select k. Calculating the silhouette coefficient multiple k-means model over different values of k allowed for selection of k with the largest silhouette coefficient. This is considered as the optimal number of clusters for the dataset as clusters are densely packed, cohesive and well segregated. [4]. We believe testing k from 2 to 14 was a suitable range since visualisations above showed at least 6 different types of clothing with variations in each of them.

C. Feature Extraction

Feature Extraction is used to find a reduced and statistically significant set of features which enhances the effectiveness of classification [5].

We attempted feature reduction with Principal Component Analysis (PCA) and t-Distributed Stochastic Neighbour Embedding (t-SNE). PCA linearly reduces dimensionality, converting correlated features in the high dimensional space into a series of uncorrelated features in the low dimensional space [6]. t-SNE is a Manifold Learning technique that uses a probability distribution to measure similarity between data points to reduce dimensionality while preserving the local structure of the data [7].

The original 784 features in data were reduced to 300 with PCA and 2 with t-SNE. Using data preprocessed with t-SNE consistently gave higher Silhouette scores than PCA. This is likely due to k-means better clustering data with fewer dimensions and t-SNE being able to capture non-linear relationships between variables. Hence, we selected t-SNE as the feature extraction method.

D. Cluster Quality Metrics

The metrics used to evaluate the clustering algorithm performance are Silhouette, Homogeneity and Completeness Scores.

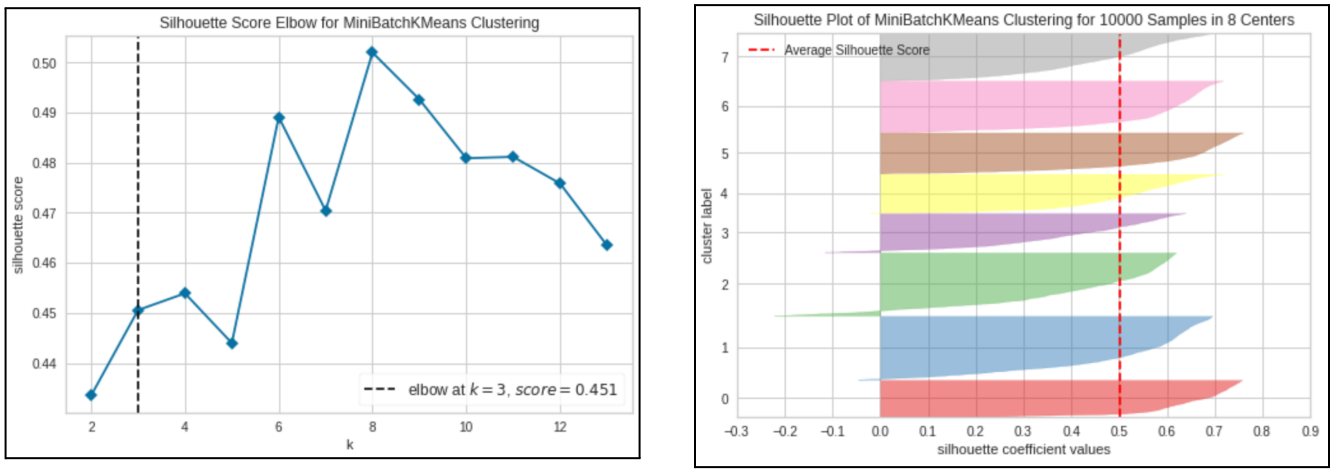
Silhouette score measures how distinguished the clusters are based on their cohesion and separation. It lies on a scale of -1 to 1 with higher scores indicating more distinguished clusters and more effective clustering.

Homogeneity Score measures the extent to which each cluster contains data points belonging to only a single class. It lies on the scale of 0 to 1 with 1 indicating perfect homogenous clusters while 0 indicates completely heterogeneous clusters [8].

Completeness Score measures the degree of which data points from one given class are assigned to the same cluster. It lies on the scale of 0 to 1 with 1 indicating that all data points from one class have been assigned to the same cluster while 0 indicates that all data points from one class have been assigned to completely different clusters [9].

IV. Implementation

A. Choosing number of clusters, k



(a) Silhouette Scores over different k

(b) Silhouette Plot for k=8

Fig. 2 Silhouette Analysis

We conducted silhouette analysis on k-means models fitted with t-SNE preprocessed data (Fig. 2a) and found that k=8 produced the highest mean silhouette score of 0.51. Approximately similar silhouettes for all clusters in Fig. 2b illustrates reasonable structure within clusters.

B. Arguments to k-means function

```
MiniBatchKMeans(n_clusters=8,init="k-means++",n_init=3,random_state=999)
```

MiniBatchKMeans was used as an alternative to KMeans to reduce computational cost and increase speed. The initialisation method k-means++ was used. It takes into account data points distribution and evenly spaced out centroids. This helps k-means reach local optimum faster and reduces the effect of initialisation on the final result. The number of clusters is 8, as found from silhouette analysis. The remaining parameters were default values.

C. Parameters to t-SNE

t-SNE takes two main parameters. Perplexity can be interpreted as a smooth measure of the effective number of neighbours which indicates the balance of attention between local and global aspects of the data [10]. Additionally, the number of iterations indicates the steps needed for the data to converge [7]. To choose the optimal parameter values, we tried using t-SNE with perplexity values 10-40 and number of iterations of 250-2000. We selected the combination **perplexity = 40** and **number of iterations = 1750**, producing a k-means model with the highest Silhouette score.

D. Flow of Code

Algo. 1 Steps in constructing K-means algorithm:

Data Preparation

1. Import data and extract pixel data as numpy array
2. Visualise sample of data
3. Equalise pixel exposure
4. Normalised and reshape pixel data

Preprocessing & Feature Extraction

PCA

1. Find number of components that explains 95% variance
2. Reduce dimensions using PCA

t-SNE

1. Try different combinations of perplexity and n_iter to find combination for optimal Silhouette score
2. Reduce dimensions using t-SNE

Comparing above methods

1. Find optimal number of clusters k for each method
2. Calculate silhouette score for each method at optimal

Building the model

1. Create k-means model with n_clusters=8, init_method="k-means++"
2. Fit the model with data preprocessed using t-SNE

V. Results

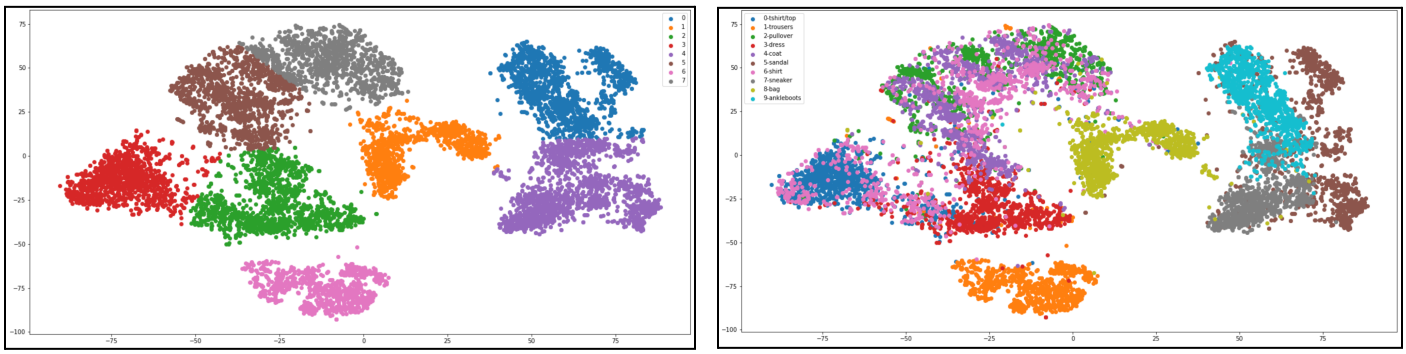
The cluster quality metrics evaluate the performance of our model and its effectiveness in clustering data points.

Table 1 K-Means Cluster Quality Metrics Scores for K=8 Clusters

Feature Extraction	Silhouette	Homogeneity	Completeness
t-SNE	0.50194	0.62140	0.69551
PCA	0.34142	0.50843	0.58024
None	0.33473	0.48004	0.55748

Extracting features using t-SNE resulted in a k-means model that performs best in all metrics. As such, feature extraction produced more homogeneous and complete clusters, improving classification of images. Higher Silhouette score shows clusters being more dense and better separated.

Scores of 0.62 for homogeneity and of 0.69 for completeness show moderately good clustering. The completeness scores may be negatively affected by our chosen value of k=8, less than the true number clusters (10). This means 10 classes are forcefully assigned to only 8 clusters and there is underfitting.



(a) Generated Clusters **(b) Generated Clusters with True Labels**

Fig. 3 Scatterplots of preprocessed data

By comparing clusters of generated labels and true labels in Fig.3 , some clusters are more homogeneous than others.

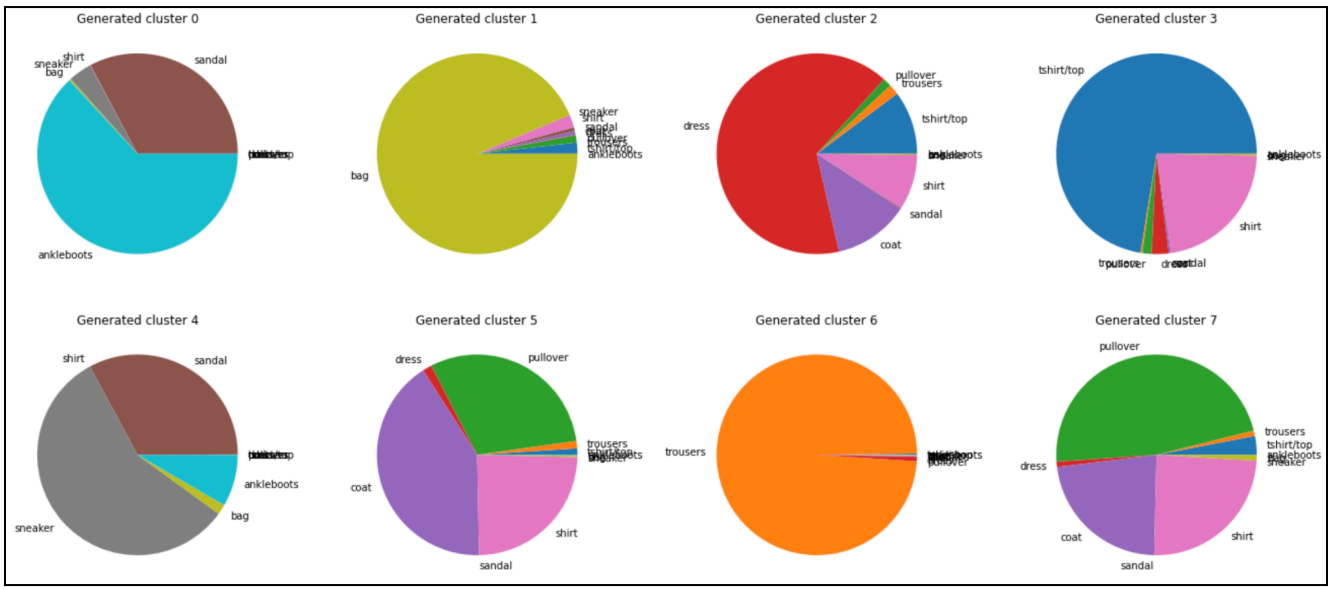


Fig. 4 Proportion of true labels within each cluster

In Fig. 4, clusters 1 and 6 are more homogenous while others are more heterogeneous. This may be a result of feature extraction performing differently on different classes. Classes such as bags and trousers are very complete while shirts and coats are very fragmented.

This may be as t-SNE reduces dimensions based on the outline of an image. For example, tops are clustered into long-sleeves and short-sleeves. Details in the images are also often lost as we see how sandals are spread across the two clusters of footwear. As such, clusters are not highly homogeneous and classes are not highly complete.

VI. Conclusion

A. Pseudo Code for Recommendation Algorithm

Algo. 2 Product Recommendation Algorithm

Input: Labels produced from K-means model, Index of sample product
Output: List of indexes of n similar products.

```
1  Get label of sample product
2  Get indexes of products with same label
3  FOR each product with same label
4        Calculate Euclidean distance from sample product
5  Sort products by Euclidean distance from sample products
6  RETURN  indexes of n products that have smallest Euclidean distance
```

B. Main Points

The report follows the steps in building an image-based product recommendation system. By visualising the data we understand the dataset consists of different categories of clothing images (shirts, trousers, bags..). We then established methods used: k-means algorithm for clustering, t-SNE algorithm for feature extraction, silhouette analysis to determine number of clusters, and homogeneity and completeness scores to evaluate the quality of our system. Next, we implemented our k-means model with k=8, fitting data preprocessed with t-SNE. Finally, we evaluated our model which has a homogeneity score of 0.62 and a completeness score of 0.69, therefore it is moderately good at clustering the data points into its true labels.

C. Key Findings and Drawbacks

K-means is sensitive to initialisation of centroids, cluster numbers and outliers which can skew centroid placements, leading to significantly different results. Moreover, k-means struggles with high dimensional data. This technique also assumes that data is normally distributed and preprocessed before clustering.

t-SNE well extracts the shape of products but struggles with details, for example, leading to mixed clusters of pullovers, shirts and t-shirts as they were classified by the general shape. This resulted in incomplete classes and non-homogeneous clusters. Regardless, it still performs better than PCA.

An alternative solution is Convolutional Neural Networks (CNN). CNN is a deep learning network architecture extracting and identifying complex image features. It reduces the dimensions without losing its features, handling high dimension images better than k-means [11]. CNN utilises layers to help filter images of increasing complexity based on details such as texture and shape, possibly making it more accurate than k-means with t-SNE [12].

Appendix

Appendix A: Hyperlink of Web-app containing Product Recommendation System

https://itthisakd-kmeans-product-recom-main-cc557b.streamlit.app/

Appendix B: Code for constructing K-Means Model – kmeans_final.py

```
# -*- coding: utf-8 -*-

"""kmeans_final.ipynb

Automatically generated by Colaboratory.

Original file is located at

   https://colab.research.google.com/drive/1uuLwj1ziwquKl5Z9U4il5ETRon9Ty0fc

# Mount to drive

"""

"""# Data"""

import pandas as pd

from PIL import Image

from IPython.display import display

import matplotlib.pyplot as plt

import numpy as np

df = pd.read_csv("product_images.csv")

data = df.to_numpy()

plt.figure(figsize = (10,9))

bottom = 0.2

for i in range(64):

 plt.subplots_adjust(bottom)

 plt.subplot(8,8,i+1)

 plt.imshow(np.reshape(data[:64], (64,28,28))[i])

"""# Utility Functions"""

import random

def show_group(group, g):

 plt.figure(figsize=(10,10))
```

```
 for i, v in enumerate(random.sample(group[g], 36)):
   plt.subplots_adjust(0.2)
   plt.subplot(6,6,i+1)
   plt.imshow(np.reshape(data, (10000, 28,28))[v])
def group_by_clusters(labels):
 map = {}
 for i in range(len(set(labels))):
   map[i]=[ k for k,v in enumerate(labels) if v == i]
  return map
def display_img(pixels):
 plt.figure(figsize=(1,1))
 plt.imshow(np.reshape(pixels, (28,28)))
def normalise_img(img):
 return(img-min(img))/(max(img)-min(img))
"""# Preprocessing"""
random_state=999
from skimage import exposure
from sklearn import preprocessing
# Conversion to float
X = data.astype('float64')
# Normalise
X = X/255.0
# Equaliee exposure
X = np.array([exposure.equalize_hist(each) for each in X])
# # Normalise
X = np.array([normalise_img(each) for each in X])
# Reshaping input data
X = X.reshape(len(data),-1)
# display preprocessed data
```

```
plt.figure(figsize = (10,9))

bottom = 0.2

for i in range(64):

 plt.subplots_adjust(bottom)

 plt.subplot(8,8,i+1)

 plt.imshow(np.reshape(X[:64], (64,28,28))[i])

from sklearn.cluster import MiniBatchKMeans

from sklearn.metrics import silhouette_score

from yellowbrick.cluster import KElbowVisualizer

from yellowbrick.cluster import SilhouetteVisualizer

"""## PCA"""

from sklearn.decomposition import PCA

pca = PCA().fit(X)

plt.plot(np.cumsum(pca.explained_variance_ratio_))

plt.xlabel('Number of components')

plt.xlim(right=400)

plt.ylabel('Explained variance ratio')

plt.axhline(0.95, color = 'r', linestyle = '-')

plt.axvline(x=300, color='gray', linestyle='--')

plt.show()

pca = PCA(n_components = 300)

pca.fit(X)

X_pca = pca.transform(X)

"""## t-SNE

From above, best perplexity=40, best n_iter=1750

"""

from sklearn import manifold

import numpy as np

# Perform feature extraction using t-SNE
```

```
tsne = manifold.TSNE(n_components=2, init='pca', random_state=0, perplexity=40,
                     n_iter=1750)
X_tsne = tsne.fit_transform(X)
# Print the shape of the new feature spaces
print("t-SNE feature space shape:", X_tsne.shape)
"""# Choosing feature extraction method"""
total_clusters = 10
scores = {}
def get_silelbow(X, total_clusters):
 kmeans = MiniBatchKMeans(n_clusters = total_clusters, random_state=random_state)
 kmeans.fit(X)
 # Import the KElbowVisualizer method
 visualiser = KElbowVisualizer(kmeans, k=(2,14), metric='silhouette', timings=False)
 # Fit the data and visualize
 visualiser.fit(X)
 visualiser.poof()
get_silelbow(X, total_clusters)
# choose
n = 7
kmeans = MiniBatchKMeans(n_clusters = n, random_state=random_state)
kmeans.fit(X)
scores["all_features"] = silhouette_score(X, kmeans.labels_)
get_silelbow(X_pca, total_clusters)
# choose
n = 8
kmeans = MiniBatchKMeans(n_clusters = n, random_state=random_state)
kmeans.fit(X_pca)
scores["pca"] = silhouette_score(X_pca, kmeans.labels_)
get_silelbow(X_tsne, total_clusters)
```

```
# choose

n = 8

kmeans = MiniBatchKMeans(n_clusters = n, random_state=random_state)

kmeans.fit(X_tsne)

scores["tsne"] = silhouette_score(X_tsne, kmeans.labels_)

scores

"""Best feature extraction method is t-SNE

# Choosing k

# Optimal Silhouette Score at k=8

"""

SilhouetteVisualizer(MiniBatchKMeans(n_clusters=8,
random_state=random_state)).fit(X_tsne).show()

"""# Build final model"""

kmeans = MiniBatchKMeans(n_clusters = 8, init="k-means++", n_init=3, random_state=999)

kmeans.fit(X_tsne)

x_vals = [each[0] for each in X_tsne]

y_vals = [each[1] for each in X_tsne]

labels = kmeans.labels_

centroids = kmeans.cluster_centers_

for k in np.unique(kmeans.labels_):

   x_vals = [X_tsne[i][0] for i in range(len(X_tsne)) if labels[i]==k]

   y_vals = [X_tsne[i][1] for i in range(len(X_tsne)) if labels[i]==k]

   plt.scatter(x_vals , y_vals, label = k)

plt.scatter(centroids[:,0] , centroids[:,1] , s = 80, color = 'black')

plt.legend()

plt.show()

group = group_by_clusters(kmeans.labels_)

show_group(group, 0)

show_group(group, 1)

show_group(group, 2)
```

```
show_group(group, 3)

show_group(group, 4)

show_group(group, 5)

show_group(group, 6)

show_group(group, 7)

"""# Saving model outputs"""

from numpy import save

from numpy import load

save('X_tsne.npy', X_tsne)

save('labels.npy', kmeans.labels_)

save('centres.npy', kmeans.cluster_centers_)

save('X.npy', X)

save('X_pca.npy', X_pca)
```

Appendix C: Code for Model Evaluation – model_evaluation.py

```
# -*- coding: utf-8 -*-

"""model_evaluation.ipynb

Automatically generated by Colaboratory.

Original file is located at

   https://colab.research.google.com/drive/1Rw9VCWxpiQ_i60WAnLqxLxQ6H-K44YGM

"""# Data"""

import pandas as pd

import numpy as np

from numpy import load

import matplotlib.pyplot as plt

data = pd.read_csv("product_images.csv").to_numpy()

our_labels=load('labels.npy')

X_tsne=load('X_tsne.npy')
```

```
X=load('X.npy')

X_pca=load('X_pca.npy')

true_labels = pd.read_csv("true_label.csv").to_numpy()

true_labels = np.array([each[0] for each in true_labels])

np.unique(our_labels)

np.unique(true_labels)

"""# Cluster plots"""

legend = ["tshirt/top", "trousers", "pullover", "dress", "coat", "sandal", "shirt",
"sneaker", "bag", "ankleboots"]

# our clusters

x_vals = [each[0] for each in X_tsne]

y_vals = [each[1] for each in X_tsne]

plt.figure(figsize=(20,10))

for k in np.unique(our_labels):

   x_vals = [X_tsne[i][0] for i in range(len(X_tsne)) if our_labels[i]==k]

   y_vals = [X_tsne[i][1] for i in range(len(X_tsne)) if our_labels[i]==k]

   plt.scatter(x_vals , y_vals, label = k)

plt.legend()

plt.show()

# true clusters

x_vals = [each[0] for each in X_tsne]

y_vals = [each[1] for each in X_tsne]

plt.figure(figsize=(20,10))

for k in np.unique(true_labels):

   x_vals = [X_tsne[i][0] for i in range(len(X_tsne)) if true_labels[i]==k]

   y_vals = [X_tsne[i][1] for i in range(len(X_tsne)) if true_labels[i]==k]

   plt.scatter(x_vals , y_vals, label = f"{k}-{legend[k]}")

plt.legend()

plt.show()

"""# Evaluation Metrics"""
```

```
from sklearn.cluster import MiniBatchKMeans

from sklearn.metrics import silhouette_score

from sklearn.metrics.cluster import homogeneity_score, completeness_score

"""K-means with t-SNE feature extraction"""

n=10

def get_scores(n, data, true_labels):

 kmeans = MiniBatchKMeans(n_clusters = n, init="k-means++", random_state=999)

 kmeans.fit(data)

 s_score=silhouette_score(X_tsne, kmeans.labels_)

 h_score=homogeneity_score(true_labels, kmeans.labels_)

 c_score=completeness_score(true_labels, kmeans.labels_)

 return [n, round(s_score, 5), round(h_score, 5), round(c_score, 5)]

scores_tsne=[get_scores(n, X_tsne, true_labels) for n in range (7,11)]

pd.DataFrame(scores_tsne, columns=["n", "s_score", "h_score", "c_score"])

"""K-means with PCA """

scores_pca=[get_scores(n, X_pca, true_labels) for n in range (7,11)]

pd.DataFrame(scores_pca, columns=["n", "s_score", "h_score", "c_score"])

"""K-means without feature extraction"""

scores_allfeatures=[get_scores(n, X, true_labels) for n in range (7,11)]

pd.DataFrame(scores_allfeatures, columns=["n", "s_score", "h_score", "c_score"])

"""# Count proportion of true labels in each of our labels"""

labels_df=pd.DataFrame(

   dict(

       our_labels=our_labels,

        true_labels=true_labels

   )

)

df = pd.DataFrame(

   dict(
```

```
       true_labels=list(set(true_labels))

   ),

   index=list(set(true_labels))

)

for n in list(set(our_labels)):

 counts =
labels_df[labels_df["our_labels"]==n].groupby("true_labels").count().rename(columns={"
our_labels":n})

 df = df.merge(counts, on="true_labels", how="outer")

df=df.drop(columns=["true_labels"])

df.fillna(0)

df_pct = df/df[df.columns].sum()*100

df_pct = df_pct.fillna(0).round(2)

df_pct["legend"] = legend

df_pct

i = 1

fig, axs = plt.subplots(2,4, figsize=(23,10))

n_counter=0

for x in range(2):

 for y in range(4):

   vals = df_pct[n_counter].values

   axs[x,y].pie(vals, labels = legend)

   axs[x,y].set_title(f"Generated cluster {n_counter}")

   n_counter+=1

fig.show()
```

Appendix D: Code for constructing Web App – main.py

```
import streamlit as st

import pandas as pd
```

```
import numpy as np

from numpy import load

import random

from sklearn.metrics import silhouette_score

from sklearn.metrics.cluster import homogeneity_score, completeness_score,
contingency_matrix

st.set_page_config(page_title="Product Recommendation", layout="wide",

                   initial_sidebar_state="expanded", menu_items=None,
page_icon="random")

@st.cache_data

def load_data():

   df = pd.read_csv("product_images.csv")

   images = df.to_numpy()

   X_tsne = load('X_tsne.npy')

   labels = load('labels.npy')

   centres = load('centres.npy')

   true_labels = pd.read_csv("true_label.csv").to_numpy()

   true_labels = np.array([each[0] for each in true_labels])

   return labels, centres, images, X_tsne, true_labels

@st.cache_data

def initial_display():

   ids = list(range(len(images)))

   random.shuffle(ids)

   return ids

@st.cache_data

def get_metrics():

   h_score=round(homogeneity_score(true_labels, labels), 4)

   c_score = round(completeness_score(true_labels, labels), 4)

   s_score = round(silhouette_score(X_tsne, labels), 4)

   return h_score, c_score, s_score
```

```
@st.cache_data
def group_by_clusters(labels):
   mapper = {}
   for i in range(len(set(labels))):
       mapper[i] = [k for k, v in enumerate(labels) if v == i]
   for k, v in mapper.items():
       random.shuffle(v)
   return mapper
@st.cache_data
def make_plots():
   legend = ["tshirt/top", "trousers", "pullover", "dress",
             "coat", "sandal", "shirt", "sneaker", "bag", "ankleboots"]
   our_clusters, z = plt.subplots(figsize=(20, 10))
   x_vals = [each[0] for each in X_tsne]
   y_vals = [each[1] for each in X_tsne]
   for k in np.unique(labels):
       x_vals = [X_tsne[i][0] for i in range(len(X_tsne)) if labels[i] == k]
       y_vals = [X_tsne[i][1] for i in range(len(X_tsne)) if labels[i] == k]
       z.scatter(x_vals, y_vals, label=k)
   z.legend()
   x_vals = [each[0] for each in X_tsne]
   y_vals = [each[1] for each in X_tsne]
   true_clusters, j = plt.subplots(figsize=(20, 10))
   for k in np.unique(true_labels):
       x_vals = [X_tsne[i][0]
                 for i in range(len(X_tsne)) if true_labels[i] == k]
       y_vals = [X_tsne[i][1]
                 for i in range(len(X_tsne)) if true_labels[i] == k]
       j.scatter(x_vals, y_vals, label=f"{k}-{legend[k]}")
```

```
    j.legend()
    return our_clusters, true_clusters
def distance(point1, point2):
    return np.sqrt(np.sum(np.square(point1 - point2)))
def get_similar(X, labels, groups, sample_index, n_similar):
    sample_label = labels[sample_index]
    distances = [
        (i, distance(X[sample_index], X[i])) for i in groups[sample_label]
    ]
    sorted_distances = sorted(distances, key=lambda x: x[1])
    n_nearest = [x[0] for x in sorted_distances[1:n_similar+1]]
    return n_nearest
# Session State also supports attribute based syntax
if 'page' not in st.session_state:
    st.session_state.page = 0
if 'sample_idx' not in st.session_state:
    st.session_state.sample_idx = -1
st.title('Image-based Product Recommendation')
# States
labels, centres, images, X_tsne, true_labels = load_data()
h_score, c_score, s_score = get_metrics()
initial_ids = initial_display()
groups = group_by_clusters(labels)
display_size = 20
def img(i):
    pixels = [255 - each for each in images[i]]
    return np.reshape(pixels, (28,28))


def get_similar_wrapped(sample_index, n_similar):
```

```
    return get_similar(X=X_tsne, labels=labels, groups=groups,
sample_index=sample_index, n_similar=n_similar)

tab1, tab2, tab3, tab4 = st.tabs(["Top Picks", "Browse by Category", "Model
Evaluation", "Acknowledgements"])

with tab1:

    # col1,col2=st.columns(2)

    st.subheader("Choose products to view similar recommendations")

    # with col1:

    button_layout = st.columns(7)

    with button_layout[0]:

        prevpage = st.button(label="< Prev Page", type="primary")

        if prevpage and st.session_state.page > 0:

            st.session_state.page -= display_size

    with button_layout[1]:

        nextpage = st.button(label="Next Page >", type="primary")

        if nextpage and st.session_state.page < 10000/display_size:

            st.session_state.page += display_size

    subcols = st.columns(4)

    col_height=5

    for i in range(4):

        with subcols[i]:

            for idx in
initial_ids[st.session_state.page:st.session_state.page+display_size][i*col_height:i*c
ol_height+col_height]:

                st.image(img(idx), width=100)

                sample_button1 = st.button(label="See more like this",

                                key=idx)

                if sample_button1:

                    st.session_state.sample_idx = idx

with tab2:

    st.header("View products by category")
```

```
   for j, title in enumerate(["Ankleboots & Sandals","Bags","Dresses & Tops","T-shirt
& Tops",

                              "Sneakers & Sandals","Coats, Pullovers &
Tops","Trousers","Pullovers & Tops"]):

       with st.expander(str(title)):

           subcols = st.columns(5)

           col_height = 4

           for i in range(5):

               with subcols[i]:

                   for idx in
groups[j][:display_size][st.session_state.page:st.session_state.page+display_size][i*c
ol_height:i*col_height+col_height]:

                       st.image(img(idx), width=100)

                       sample_button1 = st.button(label="See more like this",

                                                  key="cat"+str(idx))

                       if sample_button1:

                           st.session_state.sample_idx = idx

st.sidebar.markdown(

   "## Select a product to get recommendations")

recom_no = st.sidebar.slider('Number of Recommendations', 0, 50, 10, 5)

if st.session_state.sample_idx >= 0:

   st.sidebar.markdown("# If you like... ")

   st.sidebar.image(img(st.session_state.sample_idx), width=200)

   st.sidebar.markdown("# you might also like...")

   subcols = st.sidebar.columns(3)

   recom_lst = get_similar_wrapped(

       sample_index=st.session_state.sample_idx, n_similar=recom_no)

   col_content = [

       [v for i, v in enumerate(recom_lst) if i % 3 == 0],

       [v for i, v in enumerate(recom_lst) if (i-1) % 3 == 0],

       [v for i, v in enumerate(recom_lst) if (i-2) % 3 == 0]
```

```
    ]


    for i in range(3):

        with subcols[i]:

            for idx in col_content[i]:

                st.image(img(idx), width=100, caption=idx)

                sample_button2 = st.button(label="See more like this",

                                           key=idx+9999)

                if sample_button2:

                    st.session_state.sample_idx = idx

with tab3:

    st.subheader("Model")

    st.write("Feature Extraction method: t-SNE")

    st.write("Number of clusters k: 8")

    st.write("Initialisation Method: k-means++")

    st.subheader("Evaluation Metrics")

    st.markdown("### Homogeneity Score")

    st.write(h_score)

    st.write("- A clustering result satisfies homogeneity if all of its clusters
contain only data points which are members of a single class. \nScore between 0.0 and
1.0. 1.0 stands for perfectly homogeneous labeling.  \n")



    st.markdown("### Completeness Score")

    st.write(c_score)

    st.write("- A clustering result satisfies completeness if all the data points that
are members of a given class are elements of the same cluster. \n Score between 0.0
and 1.0. 1.0 stands for perfectly complete labeling. \n")



    st.markdown("### Silhouette Score")

    st.write(s_score)

    st.write("- A measure of how similar an object is to its own cluster(cohesion)
compared to other clusters(separation). \n The best value is 1 and the worst value is
```

```
-1. Values near 0 indicate overlapping clusters. Negative values generally indicate
that a sample has been assigned to the wrong cluster, as a different cluster is more
similar.")

with tab4:

   st.header("Acknowledgements")

   st.balloons()

   st.markdown("### *Thank you* Deyu!")

   st.write()

   # our_clusters, true_clusters = make_plots()

   # st.subheader("Clustering of pre-processed data")

   # st.pyplot(our_clusters)

   # st.subheader("Clustering using true labels")

   # st.pyplot(true_clusters)
```

References

[1] MacKenzie, I., Meyer, C. and Noble, S. (2018) *How retailers can keep up with consumers*, *McKinsey & Company*. McKinsey & Company. Available at: https://www.mckinsey.com/industries/retail/our-insights/how-retailers-can-keep-up-with-consumers (Accessed: November 25, 2022).

[2] Linden, G., Smith, B. and York, J. (2003) "Amazon.com recommendations: Item-to-item collaborative filtering," *IEEE Internet Computing*, 7(1), pp. 76–80. Available at: https://doi.org/10.1109/mic.2003.1167344.

[3] Smith, A. (2022) *Clustering.* UCL School of Management.

[4] scikit.learn (no date) *Selecting the number of clusters with silhouette analysis on kmeans clustering*, *scikit learn*. Available at: https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html#:~:text=Silhouette%20analysis%20can%20be%20used,like%20number%20of%20clusters%20visually. (Accessed: February 22, 2023).

[5] Subasi, A. (2019) "Feature extraction and dimension reduction," Practical Guide for Biomedical Signals Analysis Using Machine Learning Techniques, pp. 193–275. Available at: https://doi.org/10.1016/b978-0-12-817444-9.00004-0.

[6] Parte, K. (2020) *Dimensionality reduction: Principal Component Analysis*, *Medium*. Analytics Vidhya. Available at: https://medium.com/analytics-vidhya/dimensionality-reduction-principal-component-analysis-d1402b58feb1#:~:text=PCA%20is%20a%20linear%20dimensionality,are%20also%20called%20principal%20components (Accessed: February 24, 2023).

[7] skicit.learn (no date) *2.2. Manifold Learning*, *scikit.learn*. Available at: https://scikit-learn.org/stable/modules/manifold.html#tips-on-practical-use (Accessed: February 21, 2023).

[8] *Mitra, A. (2023) K-means clustering using sklearn - the security buddy, How to measure clustering using homogeneity score? the security buddy. Available at: https://www.thesecuritybuddy.com/unsupervised-machine-learning/k-means-clustering-using-sklearn/ (Accessed: February 22, 2023).*

[9] Mitra, A. (2023) *How to measure clustering performance using completeness score?*, *how to measure clustering using completeness score?* the security buddy. Available at: https://www.thesecuritybuddy.com/unsupervised-machine-learning/how-to-measure-clustering-performance-using-completeness-score/ (Accessed: February 22, 2023).

[10] Suárez, D. (2022) *Dimensionality reduction - TSNE*, *Apiumhub*. Available at: https://apiumhub.com/tech-blog-barcelona/dimensionality-reduction-tsne/ (Accessed: February 22, 2023).

[11] Mishra, P. (2019) *Why are convolutional neural networks good for image classification?*, *Medium*. DataDrivenInvestor. Available at: https://medium.datadriveninvestor.com/why-are-convolutional-neural-networks-good-for-image-classification-146ec6e865e8 (Accessed: February 23, 2023).

[12] Stewart, M. (2020) *Simple introduction to Convolutional Neural Networks*, *Medium*. Towards Data Science. Available at: https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac (Accessed: February 23, 2023).