A red circle with white text and a book

Description automatically generated

**AI and ML**

**Assignment 1**

|  |  |
| --- | --- |
| Tamara Ghatasha | 2035311 |
| Leen Alqasim | 2034562 |
| Rama Gayada | 2036789 |

Contents

[1. Assumption 2](#_Toc164454678)

[2. K-means algorithm 3](#_Toc164454679)

[3. How we deal with the data 4](#_Toc164454680)

[4. Explanation of our code 5](#_Toc164454681)

[5. Map 16](#_Toc164454682)

[6. Plots 17](#_Toc164454683)

[7. Conclusion 18](#_Toc164454684)

# 1. Assumption

Our main goal is to sort animals into groups based on their traits. This involves using K-means clustering, which helps us find natural groupings in our data. To do this, we first prepare our data by scaling numerical features and encoding categorical ones.

We're also using a technique called Principal Component Analysis (PCA) to simplify our data before clustering. This makes it easier to understand and visualize

Once we've clustered the animals, we'll visualize the results using scatter plots. Each cluster will have its own set of animals, and we'll mark the center of each cluster. This helps us see which animals are similar to each other based on the traits we've considered.

To make sure we're clustering our animals into the right number of groups, we're using something called the Elbow Method. This helps us find the optimal number of clusters by looking at how much variation there is within each cluster.

# 2. K-means algorithm

**K-means** is a popular unsupervised machine learning algorithm that groups unlabeled data points into clusters. Unsupervised learning means the algorithm doesn't rely on pre-labeled data.

**And here’s how it work :**

First, you randomly pick some points on the map as the starting centers of your clusters. These points are like the heart of each cluster.

Next, you look at each point on the map and decide which cluster it belongs to based on its distance to the cluster centers. You assign each point to the nearest cluster.

After sorting all the points, you take a look at each cluster and find the new center point for it. This new center is calculated by averaging the positions of all the points in that cluster. It's like finding the heart of the cluster again, but this time it might move a bit

Then You keep going back and forth between sorting the points into clusters and moving the cluster centers until things settle down. If the cluster centers don't move much anymore, it means the clusters are pretty well defined, and you're done.

The "K" in K-means is just the number of clusters you want to find. You decide how many groups you want before starting the process.

# 3. How we deal with the data

## 1.One Hot Encoding:

One hot encoding helps you translate categories into numbers in a way that makes sense. For each category, you create a new column where you put a 1 if the item belongs to that category and a 0 if it doesn't. So, for example, if your list has the color "blue," you'd create a column called "blue" and put a 1 for items that are blue and 0 for items that aren't. This way, the computer can understand and work with your categories more easily.

## 2.Label Encoding:

Label encoding is another way to turn categories into numbers, but it's a bit simpler. Instead of creating new columns like in one hot encoding, you just assign a unique number to each category. So, using the color example again, you might assign red=1, blue=2, and green=3. This way, each color is represented by a number, and the computer can still understand and process the data. It's like giving each category a name tag with its own number so the computer knows who's who.

## 3.PCA (Principal Component Analysis):

PCA is like looking at the puzzle and trying to find the biggest patterns or shapes that capture most of the information. It helps you simplify the puzzle by focusing on the pieces that matter most and ignoring the ones that don't contribute much. So, instead of dealing with every single piece individually, you can work with just a few big shapes that tell you most of what you need to know. This makes it easier to understand and analyze your data, especially when you have lots of pieces to deal with.

# 4. Explanation of our code

**First, we Import Libraries that we need:**

1. Numpy “Numerical Python”
2. Pandas “Panel Data"
3. Sklearn “Scikit-Learn”
   * 1. MinMaxScaler
     2. OneHotEncoder
     3. PCA
4. matplotlib.pyplot

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.preprocessing import MinMaxScaler

from sklearn.preprocessing import OneHotEncoder

from sklearn.decomposition import PCA

**Then: we read the data from excel file “animals” and normalize the numerical data using Min max scaler**

data = pd.read\_csv("animals.csv")

# Extract numerical features ( columns 2 to 4 contain numerical data)

numerical\_features = data.iloc[:, 2:5].values

# Normalize numerical features using MinMaxScaler

scaler = MinMaxScaler()

numerical\_features\_normalized = scaler.fit\_transform(numerical\_features)

# Create lists to store animal data

Habit = []

Diet = []

socialBehavior = []

Names = []

# Access animal names and other data

for index, row in data.iterrows():

Names.append(row.iloc[1]) #Names

Habit.append(row.iloc[-1]) #Habits

Diet.append(row.iloc[6]) #Diet

socialBehavior.append(row.iloc[5]) #socialBehavior

**Dealing with Categorical** **data by one hot encoding to the habit and label encoding to the diet and social behavior then normalize the data:**

# Categorical data

# One Hot Encoding

data\_series = pd.Series(Habit)

encodedHabit = pd.get\_dummies(data\_series)

encodedHabitInt = encodedHabit.astype(int)

le\_habit = np.argmax(encodedHabitInt, axis=1)

scaler = MinMaxScaler()

nurmlizeHabit = scaler.fit\_transform(np.array(le\_habit).reshape(-1, 1)).flatten()

###################################################################

## label encoding function

def sentiment\_label(text\_data):

"""

Encodes sentiment labels based on keywords in the text.

Args:

text\_data (str): The text to analyze.

Returns:

int: The encoded sentiment label (1 for Solitary/Herbivore, 2 for Semi-social/Omnivore, 3 for others).

"""

if "Social" in text\_data or "Herbivore" in text\_data:

return 1

elif "Semi-social" in text\_data or "Omnivore" in text\_data:

return 2

else:

return 3

encodedDitelabels = [sentiment\_label(text) for text in Diet]

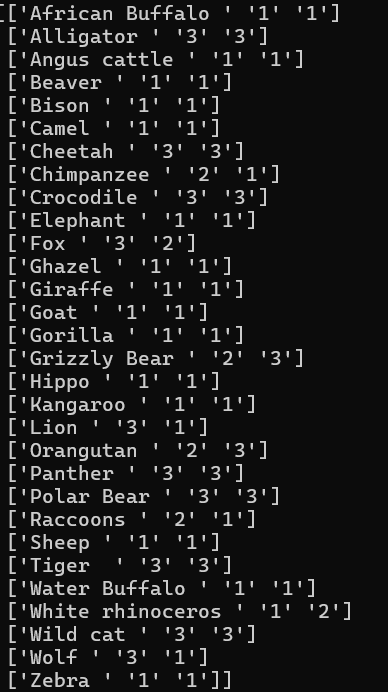
nurmlizeDiet = scaler.fit\_transform(np.array(encodedDitelabels).reshape(-1, 1)).flatten()

encodedsocialBehaviorlabels = [sentiment\_label(text) for text in socialBehavior]

nurmlizeSocialBehaviorlabels = scaler.fit\_transform(np.array(encodedsocialBehaviorlabels).reshape(-1, 1)).flatten()

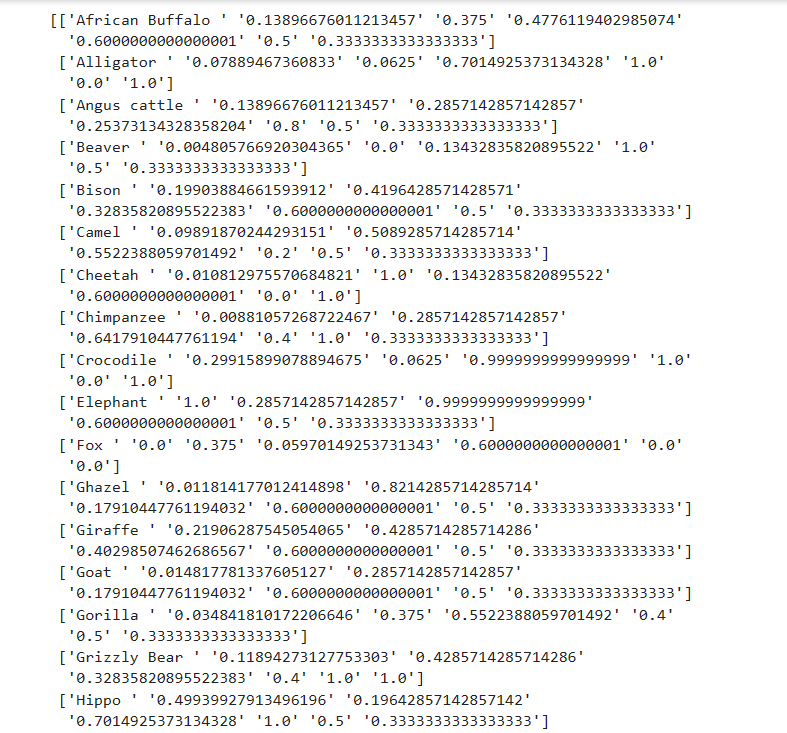
**Encoded (label) of the features**

**(social behavior and diet):**

****A screenshot of a computer screen

Description automatically generated**We use the one hot encoding library and we wrote a function for label encoding the labels is as follow:**

**Data after normalization:**



A screenshot of a computer

Description automatically generated

**Combine the numerical data then use PCA to minimize the number of feature as well combine the data with the animals names after:**

# Combine features

combined\_array = np.column\_stack([numerical\_features\_normalized, nurmlizeHabit, nurmlizeDiet, nurmlizeSocialBehaviorlabels])

combined\_array\_withnames=np.column\_stack([Names, combined\_array])

#########################################################

# PCA (exclude names)

pca = PCA(n\_components=2)

principalComponents = pca.fit\_transform(combined\_array)

# Combine principal components with names (for printing cluster membership)

principalComponents\_withnames = np.column\_stack([Names, principalComponents])

**Data after applied PCA :**

****

**K-mean function**

# K-means

def k\_means(data, k):

"""

K-Means clustering .

Args:

data (numpy.ndarray): The data to be clustered.

k (int): The number of clusters.

Returns:

tuple: A tuple containing the centroids and labels.

"""

# Initialize centroids randomly

centroids = data[np.random.choice(data.shape[0], k, replace=False)]

# Initialize labels with all -1 (un assigned)

labels = np.full(data.shape[0], -1)

# Main K-Means loop: Continue iterating until centroids don't change significantly

old\_centroids = None

while not np.array\_equal(old\_centroids, centroids):

old\_centroids = centroids.copy()

# Assign data points to the closest centroid

for i in range(data.shape[0]):

distances = np.linalg.norm(data[i] - centroids, axis=1)

labels[i] = np.argmin(distances)

# Recompute centroids based on assigned data points

for j in range(k):

data\_points\_in\_cluster = data[labels == j]

if data\_points\_in\_cluster.any(): # Check if there are any points in the cluster

centroids[j] = np.mean(data\_points\_in\_cluster, axis=0)

return centroids, labels

# K-Means with k=6

k = 5

centroids, labels = k\_means(principalComponents, k)

**Plot the clusters :**

# Plot clusters

plt.figure(figsize=(8, 6))

for i in range(k):

# Select data points belonging to cluster i

cluster\_data = principalComponents[labels == i]

plt.scatter(cluster\_data[:, 0], cluster\_data[:, 1], s=50, label=f'Cluster {i+1}')

# Plot clusters (continued)

plt.scatter(centroids[:, 0], centroids[:, 1], c='black', s=150, marker='x', label='Centroids') # Plot centroids

plt.title('Clusters of Animals (Custom K-Means, k={})'.format(k))

plt.xlabel('Principal Component 1')

plt.ylabel('Principal Component 2')

plt.legend()

plt.show()

**Print out the animals and there clusters sorted**

# to print the animals and there clusters

# Create a dictionary to store animal names for each cluster

clustered\_animals = {i: [] for i in range(k)}

# Populate the dictionary with animal names

for i in range(len(labels)):

animal\_name = principalComponents\_withnames[i][0]

cluster\_number = labels[i]

clustered\_animals[cluster\_number].append(animal\_name)

# Sort the animal names within each cluster

for cluster\_number, animals in clustered\_animals.items():

clustered\_animals[cluster\_number] = sorted(animals)

# Print sorted animal names for each cluster

for cluster\_number, animals in clustered\_animals.items():

print(f"Cluster {cluster\_number + 1} Animals:")

for animal in animals:

print(animal)

print() # Add a newline for clarity between clusters

**Elbow graph and WCSS**

# Elbow graph

def plot\_elbow\_method(data, max\_k):

"""

Plots the elbow graph to determine the optimal number of clusters.

Args:

data (numpy.ndarray): The data for clustering.

max\_k (int): The maximum number of clusters to consider.

"""

WCSS = []

for k in range(1, max\_k + 1):

centroids, labels = k\_means(data, k)

WCSS.append(np.sum((data - centroids[labels]) \*\* 2)) # Calculate within-cluster sum of squares

# Plot elbow graph

plt.figure(figsize=(8, 4))

plt.plot(range(1, max\_k + 1), WCSS , marker='o')

plt.title('Elbow Method for Optimal k')

plt.xlabel('Number of Clusters (k)')

plt.ylabel('Within-Cluster Sum of Squares (WCSS)')

plt.show()

# Define the maximum value of k to consider

max\_k = 10

# Plot the elbow graph

plot\_elbow\_method(principalComponents, max\_k)

**Extracting unique values from Diet ,Habit ,social Behavior then print the encoded to each one**

uniqueDiet = pd.Series(Diet).unique()

encodedDitelabels = [sentiment\_label(text) for text in uniqueDiet]

combined\_lableDiet=np.column\_stack([uniqueDiet, encodedDitelabels])

print(combined\_lableDiet)

uniqueHabit = pd.Series(Habit).unique()

encodedHabit = pd.get\_dummies(uniqueHabit)

encodedHabitInt = encodedHabit.astype(int)

combined\_lableHabit=np.column\_stack([uniqueHabit, encodedHabitInt])

print(combined\_lableHabit)

uniquesocialBehavior = pd.Series(socialBehavior).unique()

encodedsocialBehavior = [sentiment\_label(text) for text in uniquesocialBehavior]

combined\_lablesocialBehavior=np.column\_stack([uniquesocialBehavior, encodedsocialBehavior])

print(combined\_lablesocialBehavior)



Then we can reach our goal to visualize a dataset containing information about animals and grouping animals based on their characteristics and behavior patterns, providing insights into their similarities and differences.

# 5. Map

A group of animals in a zoo pen

Description automatically generated

A group of wild animals in a cage

Description automatically generated

A group of animals in a fenced area

Description automatically generated

A cartoon of a person in a crib

Description automatically generated

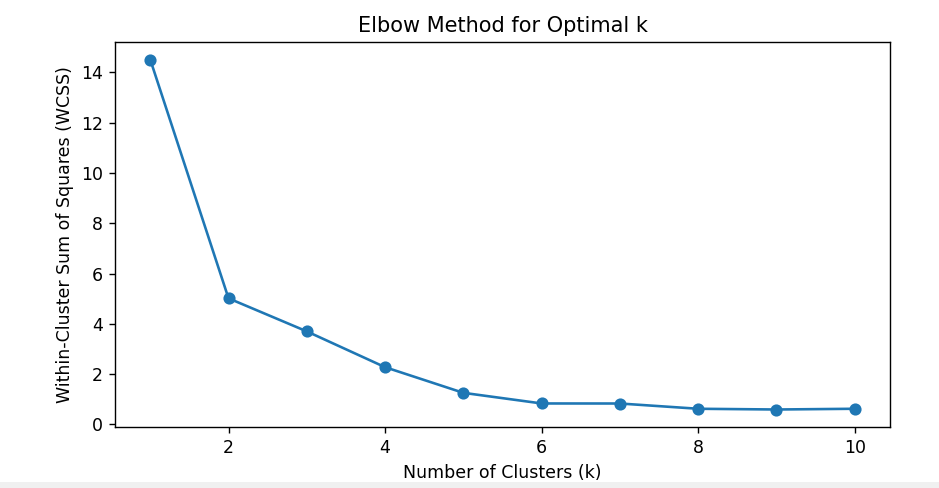
Elephants in a fence

Description automatically generated

A group of wild animals in a cage

Description automatically generated

# 6. Plots



# 7. Conclusion

* The code makes certain assumptions about the structure and content of the input data.
* It generates informative plots, including a cluster scatter plot and an elbow method plot, to analyze and visualize the clustering results.
* The cluster scatter plot serves as a map, illustrating the grouping of animals based on their features and behaviors.
* The elbow plot estimates the optimal value of number of clustering , in our code it has 5 clustering of animals.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Cluster 1**  Kangaroo  Raccoons  Ghazel  Sheep  Goat | **Cluster 2**  White rhinoceros  Elephant  Hippo | **Cluster 3**  Orangutan  Alligator  Crocodile | **Cluster 4**  Grizzly Bear  Polar Bear  Wild cat  Cheetah  Panther  Tiger  Lion  Wolf  Fox | **Cluster 5**  African Buffalo  Water Buffalo  Chimpanzee  Angus cattle  Beaver  Bison  Camel  Giraffe  Gorilla  Zebra |

The Map show the clustering of these 30 animals into 5 groups as :