**Submitted By Junaid Asif / M.Khizar Baig**

**Roll No BSAI-144 / BSAI-156**

**Class BSAI - 3rd Sem**

**Date Dec 31, 2023**

**Submitted To Ms. Humaira Batool**

**Final Project - Lab Report**

**Code:**

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

from sklearn.neighbors import KNeighborsClassifier

from sklearn.preprocessing import StandardScaler

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score

# Load your dataset

file\_path = r"E:\Sem3-GH\Sem3Lab\AI\final\_project\AI-FP\Video\_Games.csv"

df = pd.read\_csv(file\_path)

# Check the dataset and handle missing values if any

print(df.head()) # Check the structure of the dataset

print(df.isnull().sum()) # Check for missing values

# Handling missing values - replace 'tbd' in 'User\_Score' with NaN

df['User\_Score'] = pd.to\_numeric(df['User\_Score'], errors='coerce')

# Assuming 'Genre' is the target variable

x = df[['Critic\_Score', 'Critic\_Count', 'User\_Score', 'User\_Count']]

y = df['Genre']

# Drop rows with NaN values in the features

x.dropna(inplace=True)

y = y.iloc[x.index] # Update y based on the filtered x indices

# Applying KNN

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.20, random\_state=42)

knn = KNeighborsClassifier(n\_neighbors=3)

knn.fit(x\_train, y\_train)

predictions = knn.predict(x\_test)

accuracy = accuracy\_score(y\_test, predictions)

print(f"KNN on data accuracy: {accuracy}")

# Applying KMeans clustering

numeric\_columns = ['Critic\_Score', 'Critic\_Count', 'User\_Score', 'User\_Count']

x = df[numeric\_columns].dropna() # Drop rows with NaN values

# Standardize the data

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(x)

# KMeans clustering

kmeans = KMeans(n\_clusters=3, n\_init=10, random\_state=42)

kmeans.fit(X\_scaled)

# Plotting clusters

cluster\_labels = kmeans.labels\_

cluster\_centers = kmeans.cluster\_centers\_

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

plt.scatter(X\_scaled[:, 0], X\_scaled[:, 1], c=cluster\_labels, cmap='viridis', edgecolor='k', label='Data Points')

plt.scatter(cluster\_centers[:, 0], cluster\_centers[:, 1], marker='\*', s=100, c='red', label='Cluster Centers')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

# Adjust x-axis and y-axis limits to show more data points

plt.xlim(X\_scaled[:, 0].min() - 0.5, X\_scaled[:, 0].max() + 0.5)

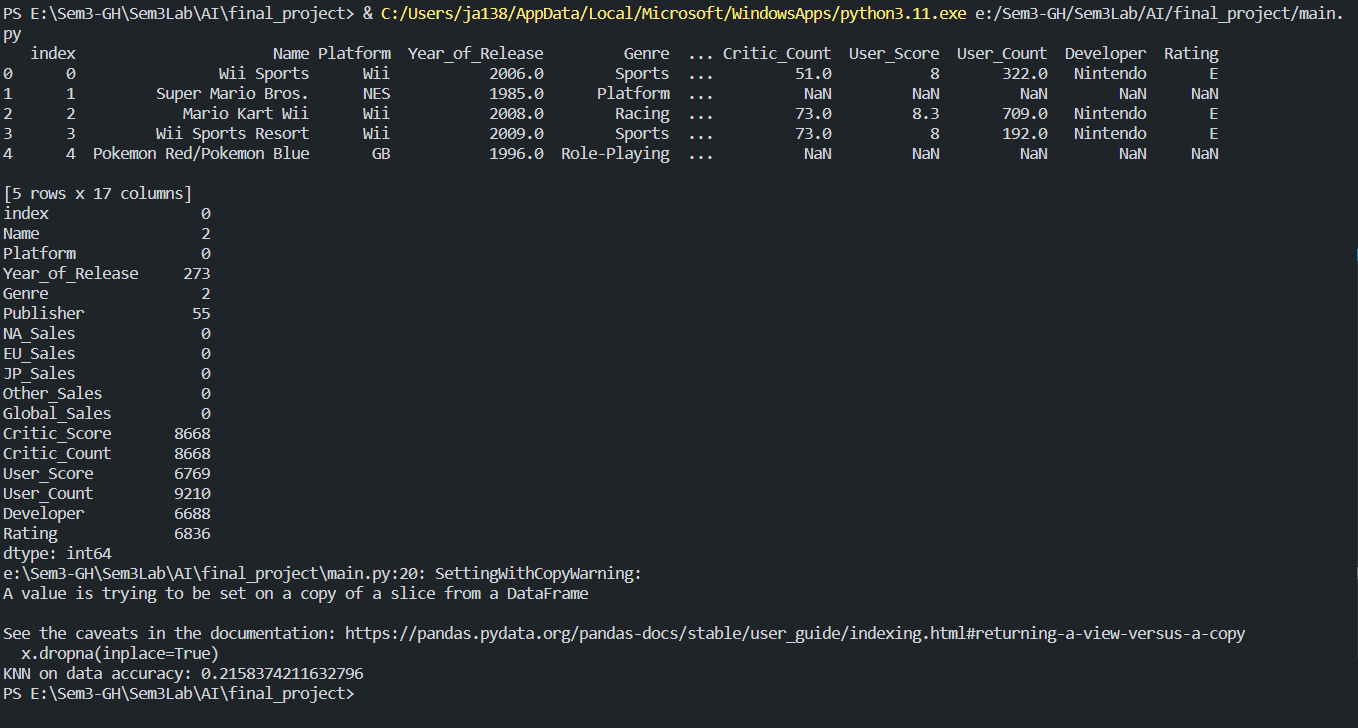
plt.ylim(X\_scaled[:, 1].min() - 0.5, X\_scaled[:, 1].max() + 0.5)

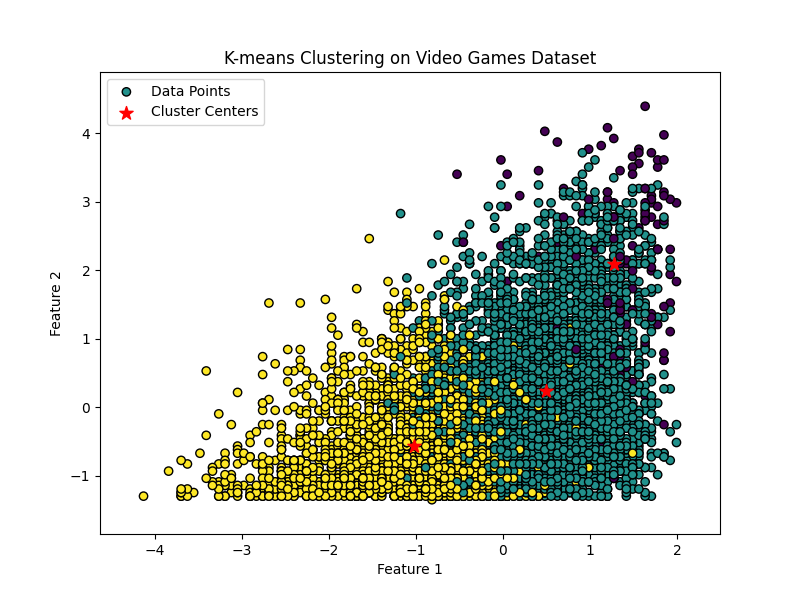
plt.title('K-means Clustering on Video Games Dataset')

plt.legend()

plt.show()

**Output:**





**Functionality:**

This Python code performs a few machine learning tasks using the scikit-learn library on a dataset of video games.

**Data Loading and Preprocessing:**

* It starts by loading a dataset from a CSV file using Pandas.
* It inspects the dataset by printing the first few rows and checking for missing values (NaN) using isnull().sum().

**Handling Missing Values:**

* It handles missing values in the 'User\_Score' column by converting 'tbd' values to NaN using pd.to\_numeric().

**Supervised Learning - K-Nearest Neighbors (KNN):**

* It prepares the features (x) and the target variable (y) for KNN, using columns 'Critic\_Score', 'Critic\_Count', 'User\_Score', and 'User\_Count'.
* Rows with missing values in features are dropped, and the target variable is updated accordingly.
* Splits the data into training and testing sets (x\_train, x\_test, y\_train, y\_test) using train\_test\_split().
* Trains a KNN classifier with 3 neighbors (n\_neighbors=3) using KNeighborsClassifier().
* Evaluates the model's accuracy on the test set using accuracy\_score().

**Unsupervised Learning - KMeans Clustering:**

* It focuses on columns 'Critic\_Score', 'Critic\_Count', 'User\_Score', and 'User\_Count' for clustering.
* Drops rows with missing values in these columns and standardizes the data using StandardScaler().
* Performs KMeans clustering with 3 clusters (n\_clusters=3) on the standardized data.

**Visualization:**

* Plots the clusters formed by KMeans on a 2D scatter plot.
* Data points are colored based on their assigned cluster, while cluster centers are marked with red asterisks.
* Axes labels, title, and legend are added for better understanding.
* The plot is displayed using Matplotlib.

The code essentially handles missing values, applies two different machine learning techniques—supervised (KNN for classification) and unsupervised (KMeans for clustering), and provides a visual representation of the clusters formed by KMeans in a 2D space.

