**Machine Learning**

1. **Create a K-means clustering algorithm to group customers of a retail store based on their purchase history:-**

import pandas as pd

from sklearn.cluster import KMeans

from sklearn.preprocessing import StandardScaler

# Load customer data (replace 'data.csv' with your actual file path)

data = pd.read\_csv('data.csv')

# Select relevant features from purchase history (e.g., total spent, categories purchased)

purchase\_features = ['feature1', 'feature2', 'feature3', ...] # Replace with your feature names

customer\_data = data[purchase\_features]

# Standardize features (optional, but recommended for K-means)

scaler = StandardScaler()

scaled\_data = scaler.fit\_transform(customer\_data)

# Define the number of clusters (k) based on domain knowledge or experimentation

k = 3 # Replace with the desired number of customer groups

# Create and train the K-means model

kmeans = KMeans(n\_clusters=k, random\_state=42)

kmeans.fit(scaled\_data)

# Assign cluster labels to each customer

customer\_data['cluster'] = kmeans.labels\_

# Analyze the clusters (optional)

# You can explore the characteristics of each cluster based on their purchase behavior

print("Customer data with cluster labels:")

print(customer\_data.head())

1. **Implement a support vector machine (SVM) to classify images of cats and dogs from the Kaggle dataset:-**

from sklearn.datasets import load\_files

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

from sklearn.preprocessing import StandardScaler

from sklearn.svm import SVC

from sklearn.metrics import accuracy\_score

# Load the cat vs dog dataset from Kaggle (download and replace path)

data = load\_files(container\_path='path/to/cats\_vs\_dogs', shuffle=True, random\_state=42)

# Separate features (images) and target labels (cat/dog)

X = data.data

y = data.target

# Split data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Reshape images (assuming grayscale images)

X\_train = X\_train.reshape(-1, 64 \* 64) # Replace with image size if different

X\_test = X\_test.reshape(-1, 64 \* 64)

# Standardize pixel intensities (optional but recommended)

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

# Create and train the SVM model

svm\_model = SVC(kernel='linear') # Experiment with different kernels

svm\_model.fit(X\_train, y\_train)

# Make predictions on the test set

y\_pred = svm\_model.predict(X\_test)

# Evaluate model accuracy

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

print(f"Model Accuracy: {accuracy:.4f}")

# You can further use the model to predict on new images

new\_image = ... # Load your new image data (reshape if necessary)

new\_prediction = svm\_model.predict(new\_image.reshape(1, -1)) # Predict for single image

print(f"Predicted Class: {new\_prediction[0]} (Cat: 0, Dog: 1)")