Kmeans.py

```
import pandas as pd
from mpi4py import MPI
import numpy as np

# Define colors
RED = "\033[31m"
GREEN = "\033[32m"
YELLOW = "\033[33m"
RESET = "\033[0m"

#initialize communicator , rank to get rank of current processes and size for total number of processes
comm = MPI.COMM_WORLD
rank = comm.Get_rank()
size = comm.Get_size()


#load data from cluster_data csv on rank 0
if rank == 0:
    data = pd.read_csv("cluster_data.csv", header = 0, index_col=0)
else:
    # initialize data variable to None
    data = None

# Flatten data array & scatter the data evenly on all ranks
flat_data = np. array([]) if rank !=0 else data.values.flatten()
_data = np.empty(len(flat_data) // size, dtype=np.float64)
comm.Scatter(flat_data, _data , root=0)


k = 4 #number of clusters

 # initialize centroids randomly on rank 0
if rank == 0:
    centroids = np.random.rand(k, 2)
else:
    centroids = None

#broadcast centroids to all ranks
centroids = comm.bcast(centroids, root=0)

#print _data on each rank
print(f"{RED}Rank {rank} {RESET}")
print(f"{GREEN}centroids:{centroids} {RESET}")
print(f"{YELLOW}local data: {_data} {RESET}")


# compute the distances between each data point and the centroids
distances = np.zeros((_data.shape[0], k))
for i in range(k):
    distances[:, i] = np.linalg.norm(_data - centroids[i], axis=1)

#collect distances computed by all ranks onto each rank
all_distances = comm.allgather(distances)

#flatten and transpose all_distances into shape (n, k) where n is the total number of
```

```
    data points
    all_distances = np.concatenate(all_distances, axis=0)
    all_distances = np.transpose(all_distances)

    #print all_distances on rank 0
    if rank == 0:
        print(f"{RED}all distances{RESET}: {GREEN}{all_distances} {RESET}")
```