**CODE CONVERSION (R to Python)**

R> library("MSA")

R> data("mcdonalds", package = "MSA")

R> names(mcdonalds)

Python:

import pandas as pd

mcdonalds = pd.read\_csv('path\_to\_mcdonalds.csv')

print(mcdonalds.columns)

R> dim(mcdonalds)

Python:

dimensions = mcdonalds.shape

print(f"Dimensions: {dimensions}")

R> head(mcdonalds, 3)

P: print(mcdonalds.head(3))

R> MD.x <- as.matrix(mcdonalds[, 1:11])

R> MD.x <- (MD.x == "Yes") + 0

R> round(colMeans(MD.x), 2)

P: MD\_x = mcdonalds.iloc[:, 0:11].apply(lambda x: (x == "Yes").astype(int))

means = np.round(MD\_x.mean(), 2)

print(means)

R> MD.pca <- prcomp(MD.x)

R> summary(MD.pca)

MD\_x = mcdonalds.iloc[:, 0:11].apply(lambda x: (x == "Yes").astype(int))

pca = PCA() MD\_pca = pca.fit\_transform(MD\_x)

explained\_variance = pca.explained\_variance\_ratio\_

cumulative\_variance = np.cumsum(explained\_variance)

R> library("flexclust")

R> plot(predict(MD.pca), col = "grey")

R> projAxes(MD.pca)

MD\_x = mcdonalds.iloc[:, 0:11].apply(lambda x: (x == "Yes").astype(int))

pca = PCA()

MD\_pca = pca.fit\_transform(MD\_x)

plt.scatter(MD\_pca[:, 0], MD\_pca[:, 1], color='grey')

plt.xlabel('Principal Component 1')

plt.ylabel('Principal Component 2')

plt.title('PCA Projection')

for i, (length, vector) in enumerate(zip(pca.explained\_variance\_, pca.components\_)):

v = vector \* 3 \* np.sqrt(length) # Scaling factor for better visualization

plt.plot([0, v[0]], [0, v[1]], color='red', label=f'PC{i+1}')

plt.grid()

plt.legend()

plt.show()

R> set.seed(1234)

R> MD.km28 <- stepFlexclust(MD.x, 2:8, nrep = 10,

+ verbose = FALSE)

R> MD.km28 <- relabel(MD.km28)

P:

MD\_x = mcdonalds.iloc[:, 0:11].apply(lambda x: (x == "Yes").astype(int))

np.random.seed(1234)

best\_k = 0

best\_inertia = np.inf

best\_kmeans = None

best\_score = -1

for k in range(2, 9):

kmeans = KMeans(n\_clusters=k, n\_init=10, random\_state=1234, verbose=False)

clusters = kmeans.fit\_predict(MD\_x)

inertia = kmeans.inertia\_

score = silhouette\_score(MD\_x, clusters)

if score > best\_score:

best\_score = score

best\_k = k

best\_kmeans = kmeans

print(f'Best number of clusters: {best\_k}')

print(f'Silhouette Score: {best\_score}')

MD\_km28 = best\_kmeans.labels\_

R> set.seed(1234)

R> MD.b28 <- bootFlexclust(MD.x, 2:8, nrep = 10,+ nboot = 100)

R> plot(MD.b28, xlab = "number of segments",+ ylab = "adjusted Rand index")

P:

MD\_x = mcdonalds.iloc[:, 0:11].apply(lambda x: (x == "Yes").astype(int))

np.random.seed(1234)

ari\_values\_mean = []

ari\_values\_sd = []

for k in range(2, 9):

ari\_scores = []

for \_ in range(100):

sample\_indices = np.random.choice(len(MD\_x), size=len(MD\_x), replace=True)

bootstrap\_sample = MD\_x.iloc[sample\_indices]

kmeans = KMeans(n\_clusters=k, n\_init=10, random\_state=1234)

clusters = kmeans.fit\_predict(bootstrap\_sample)

true\_labels = kmeans.labels\_

ari = adjusted\_rand\_score(true\_labels, clusters)

ari\_scores.append(ari)

mean\_ari = np.mean(ari\_scores)

sd\_ari = np.std(ari\_scores)

ari\_values\_mean.append(mean\_ari)

ari\_values\_sd.append(sd\_ari)

plt.figure(figsize=(10, 6))

plt.errorbar(range(2, 9), ari\_values\_mean, yerr=ari\_values\_sd, marker='o', linestyle='--', color='b')

plt.xlabel('Number of Segments')

plt.ylabel('Adjusted Rand Index')

plt.title('Bootstrap K-means Clustering: ARI vs Number of Segments')

plt.xticks(range(2, 9))

plt.grid(True)

plt.show()

R> histogram(MD.km28[["4"]], data = MD.x, xlim = 0:1)

P:

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

plt.hist(MD\_x.iloc[:, 3][cluster\_4\_membership], bins=[0, 1], edgecolor='black', alpha=0.7) plt.xlabel('Variable Values')

plt.ylabel('Frequency')

plt.title('Histogram of Cluster 4 Membership')

plt.xlim(0, 1)

plt.grid(True)

plt.show()

R> library("flexmix")

R> set.seed(1234)

R> MD.m28 <- stepFlexmix(MD.x ~ 1, k = 2:8, nrep = 10,+ model = FLXMCmvbinary(), verbose = FALSE)

R> MD.m28

P:

MD\_x = mcdonalds.iloc[:, 0:11].apply(lambda x: (x == "Yes").astype(int))

scaler = StandardScaler()

MD\_x\_scaled = scaler.fit\_transform(MD\_x)

np.random.seed(1234)

bic\_values = []

for k in range(2, 9):

models = []

for \_ in range(10): # Number of repetitions

model = MixModel(k, dist=['mvbinary'], n\_init=10, verbose=False)

model.fit(MD\_x\_scaled)

models.append(model)

best\_bic = np.min([m.bic(MD\_x\_scaled) for m in models])

bic\_values.append(best\_bic)

print("BIC values for each number of clusters:")

for k, bic in zip(range(2, 9), bic\_values):

print(f"k = {k}: BIC = {bic}")

R> plot(MD.m28, + ylab = "value of information criteria (AIC, BIC, ICL)")

R> MD.m4 <- getModel(MD.m28, which = "4")

R> table(kmeans = clusters(MD.k4), + mixture = clusters(MD.m4))

np.random.seed(1234)

MD\_x = pd.DataFrame(np.random.randint(0, 2, size=(100, 11)), columns=[f'Var{i}' for i in range(1, 12)])

MD\_km\_labels = np.random.randint(0, 4, size=100) # Replace with actual K-means cluster labels

MD\_x = mcdonalds.iloc[:, 0:11].apply(lambda x: (x == "Yes").astype(int))

scaler = StandardScaler()

MD\_x\_scaled = scaler.fit\_transform(MD\_x)

aic\_values = []

bic\_values = []

icl\_values = []

for k in range(2, 9):

model = MixModel(k, dist=['mvbinary'], n\_init=10, verbose=False)

model.fit(MD\_x\_scaled)

aic\_values.append(model.aic(MD\_x\_scaled))

bic\_values.append(model.bic(MD\_x\_scaled))

icl\_values.append(model.icl(MD\_x\_scaled))

plt.figure(figsize=(10, 6))

plt.plot(range(2, 9), aic\_values, marker='o', linestyle='-', color='b', label='AIC')

plt.plot(range(2, 9), bic\_values, marker='o', linestyle='--', color='g', label='BIC')

plt.plot(range(2, 9), icl\_values, marker='o', linestyle='-.', color='r', label='ICL')

plt.xlabel('Number of Segments')

plt.ylabel('Value of Information Criteria')

plt.title('Model-Based Clustering: Information Criteria vs Number of Segments')

plt.xticks(range(2, 9))

plt.legend()

plt.grid(True)

plt.show()

MD\_m\_labels = model.labels\_

contingency\_table = pd.crosstab(MD\_km\_labels, MD\_m\_labels, rownames=['K-means'], colnames=['Model-based'])

print("\nContingency Table (K-means vs Model-based Clustering):")

print(contingency\_table)

R> MD.m4a <- flexmix(MD.x ~1, cluster = clusters(MD.k4), + model = FLXMCmvbinary())

R> table(kmeans = clusters(MD.k4), + mixture = clusters(MD.m4a))

P:

model = MixModel(len(np.unique(MD\_km\_labels)), dist=['mvbinary'], n\_init=10, verbose=False)

model.fit(MD\_x\_scaled, init=MD\_km\_labels)

MD\_m\_labels = model.labels\_

R> rev(table(mcdonalds$Like))

R> mcdonalds$Like.n <- 6 - as.numeric(mcdonalds$Like)

R> table(mcdonalds$Like.n)

R> f <- paste(names(mcdonalds)[1:11], collapse = "+")

R> f <- paste("Like.n ~ ", f, collapse = "")

R> f <- as.formula(f)

R> f

R> set.seed(1234)

R> MD.reg2 <- stepFlexmix(f, data = mcdonalds, k = 2,

+ nrep = 10, verbose = FALSE)

R> MD.reg2

P:

mcdonalds['Like.n'] = 6 - mcdonalds['Like'].replace({'Yes': 1, 'No': 0})

formula = 'Like.n ~ ' + ' + '.join(mcdonalds.columns[1:12])

model = MixModel(2, dist=['gaussian'], n\_init=10, verbose=False)

model.fit(mcdonalds[formula])

print(model.summary())

R> MD.vclust <- hclust(dist(t(MD.x)))

R> barchart(MD.k4, shade = TRUE,+ which = rev(MD.vclust$order))

R> plot(MD.k4, project = MD.pca, data = MD.x, + hull = FALSE, simlines = FALSE, + xlab = "principal component 1", + ylab = "principal component 2")

R> projAxes(MD.pca)

P:

MD\_x = pd.DataFrame(np.random.randint(0, 2, size=(100, 11)), columns=[f'Var{i}' for i in range(1, 12)])

MD\_vclust = linkage(MD\_x.transpose(), method='complete', metric='euclidean')

plt.figure(figsize=(12, 6))

dendrogram(MD\_vclust, labels=MD\_x.columns)

plt.title('Hierarchical Clustering Dendrogram')

plt.xlabel('Variables')

plt.ylabel('Distance')

plt.show()

R> k4 <- clusters(MD.k4)

R> mosaicplot(table(k4, mcdonalds$Like), shade = TRUE, + main = "", xlab = "segment number")

R> mosaicplot(table(k4, mcdonalds$Gender), shade = TRUE)

mcdonalds['Like'] = pd.Categorical(mcdonalds['Like'], categories=['Yes', 'No'])

mcdonalds['Gender'] = pd.Categorical(mcdonalds['Gender'], categories=['Male', 'Female'])

plt.figure(figsize=(10, 6))

sns.mosaicplot(data=mcdonalds, index=['Like', MD\_k4], title='Mosaic Plot of Like vs Clusters', labelizer=lambda x: '')

plt.xlabel('Cluster')

plt.ylabel('Like')

plt.show()

plt.figure(figsize=(10, 6))

sns.mosaicplot(data=mcdonalds, index=['Gender', MD\_k4], title='Mosaic Plot of Gender vs Clusters', labelizer=lambda x: '')

plt.xlabel('Cluster')

plt.ylabel('Gender')

plt.show()

R> library("partykit")

R> tree <- ctree(

+ factor(k4 == 3) ~ Like.n + Age +

+ VisitFrequency + Gender,

+ data = mcdonalds)

R> plot(tree)

P:

mcdonalds['Like'] = pd.Categorical(mcdonalds['Like'], categories=['Yes', 'No'])

mcdonalds['Gender'] = pd.Categorical(mcdonalds['Gender'], categories=['Male', 'Female'])

pandas2ri.activate()

ro.globalenv['mcdonalds'] = pandas2ri.py2rpy(mcdonalds)

partykit = importr('partykit')

formula = ro.Formula('factor(k4 == 3) ~ Like + Age + VisitFrequency + Gender')

tree = partykit.ctree(formula, data=ro.globalenv['mcdonalds'])

ro.r('''

library("partykit")

''')

ro.globalenv['tree'] = tree

ro.r('''

plot(tree)

''')

R> visit <- tapply(as.numeric(mcdonalds$VisitFrequency),+ k4, mean)

R> visit

R> like <- tapply(mcdonalds$Like.n, k4, mean)

R> like

R> female <- tapply((mcdonalds$Gender == "Female") + 0,+ k4, mean)

R> female

R> plot(visit, like, cex = 10 \* female,

+ xlim = c(2, 4.5), ylim = c(-3, 3))

R> text(visit, like, 1:4)

P:

MD\_k4 = np.random.randint(1, 5, size=100) # Replace with actual cluster labels

visit\_means = mcdonalds.groupby(MD\_k4)['VisitFrequency'].mean()

like\_means = mcdonalds.groupby(MD\_k4)['Like.n'].mean()

female\_means = mcdonalds.groupby(MD\_k4)['Gender'].apply(lambda x: np.mean(x == 'Female'))

plt.figure(figsize=(10, 8))

plt.scatter(visit\_means, like\_means, s=10 \* female\_means, alpha=0.6)

plt.xlim(2, 4.5)

plt.ylim(-3, 3)

plt.xlabel('Visit Frequency (Mean)')

plt.ylabel('Like (Mean)')

plt.title('Scatter Plot with Annotations')

for i, txt in enumerate(range(1, 5)):

plt.text(visit\_means[i], like\_means[i], txt, fontsize=12, ha='center', va='center')

plt.show()