Logbook

Pixels To Predication

Ayush Chalishajar
Dr. E.P. Scarlett High School
Grade 12 (2025)

Note: A written entry book will be provided in person at CYSF; this logbook provides a comprehensive workflow and outlines all of my steps and code.

Background/Intro

Problem:
Hypertrophic cardiomyopathy (HCM) is the most common genetic heart muscle disease, affecting approximately 1 in 500 people. This condition causes increased heart muscle thickness, leading to obstruction of blood flow, heart failure, and life-threatening heart rhythm abnormalities. Current risk scores for predicting sudden death rely on a single, manually performed 2D measurement of the maximum wall thickness, which is known to be inaccurate and hence has been widely considered an inaccurate risk assessment model.

What has been done:
Current clinical practice guidelines still recommend the use of cardiac MRI to assess wall thickness and other heart muscle disease features. Despite improved visualization, these assessments remain subjective to the doctor who performs them and still rely on the single 2D measurement. VERY LITTLE HAS BEEN DONE.

What is new:
These 3D models, paired with deep learning algorithms, are hypothesized to improve the accuracy and reproducibility of maximal wall thickness measurements and provide unique 3D shape features that can be used by unsupervised machine learning techniques, such as phenotype clustering, to classify high-risk patients.

Workflow

• **Patient Selection and Data Acquisition**
– Identify adult HCM patients in the CIROC Registry who have undergone baseline CMR and meet inclusion criteria (e.g., no prior myectomy/ablation).
– Collect demographic information and follow-up data on cardiac outcomes.

• **CMR Image Segmentation and 3D Mesh Generation**
– Transfer short-axis (SAX) and long-axis (LAX) cine images to a dedicated cloud environment.
– Manually contour endocardial and epicardial borders at end-diastole to accurately segment the left ventricle.
– Convert segmentation contours into a bi-layer 3D mesh model with standard elemental definitions (e.g., 456 hexahedral elements).
– Extract local wall thickness values from each mesh element to produce a high-dimensional thickness dataset.

• **Data Cleaning and Preparation**
– Remove patients or elements with incomplete or invalid measurements.
– Filter out rows with non-physiological (negative) thickness values.
– Ensure consistent labeling of patients, elements, and any clinical variables (e.g., cluster assignments).

• **Dimensionality Reduction (Python)**
– Normalize or standardize the 3D mesh thickness dataset as needed.
– Apply Principal Component Analysis (PCA) to reduce dimensionality while retaining ~80% of data variance (commonly ~8 principal components).

• **Unsupervised Clustering (Python)**
– Evaluate multiple clustering algorithms (e.g., K-Means, hierarchical, HDBSCAN) on the PCA scores.
– Use silhouette analysis or similar metrics (e.g., elbow plot) to determine the optimal cluster count.
– Assign each patient to a final cluster (e.g., two distinct HCM phenotypes).

• **Statistical Analysis (R)**
– Compare baseline demographic and clinical variables between clusters (t-tests or Wilcoxon rank-sum for continuous variables; chi-square for categorical).
– Summarize cluster characteristics in descriptive tables (e.g., Table 1).
– Use survival analysis techniques to assess cluster associations with adverse events:
▪ Construct Kaplan-Meier curves to visualize event-free survival stratified by cluster.
▪ Build multivariable Cox proportional hazards models, adjusting for key covariates (e.g., age, sex, LV mass, LA volume, and fibrosis burden).

• **Results Interpretation and Visualization**
– Inspect and interpret major findings:
▪ Verify that the identified clusters are clinically and statistically distinct.
▪ Confirm that one cluster correlates with higher risk of adverse outcomes.
– Present 3D scatter plots of PCA clustering, Kaplan-Meier survival curves, and forest plots of hazard ratios.

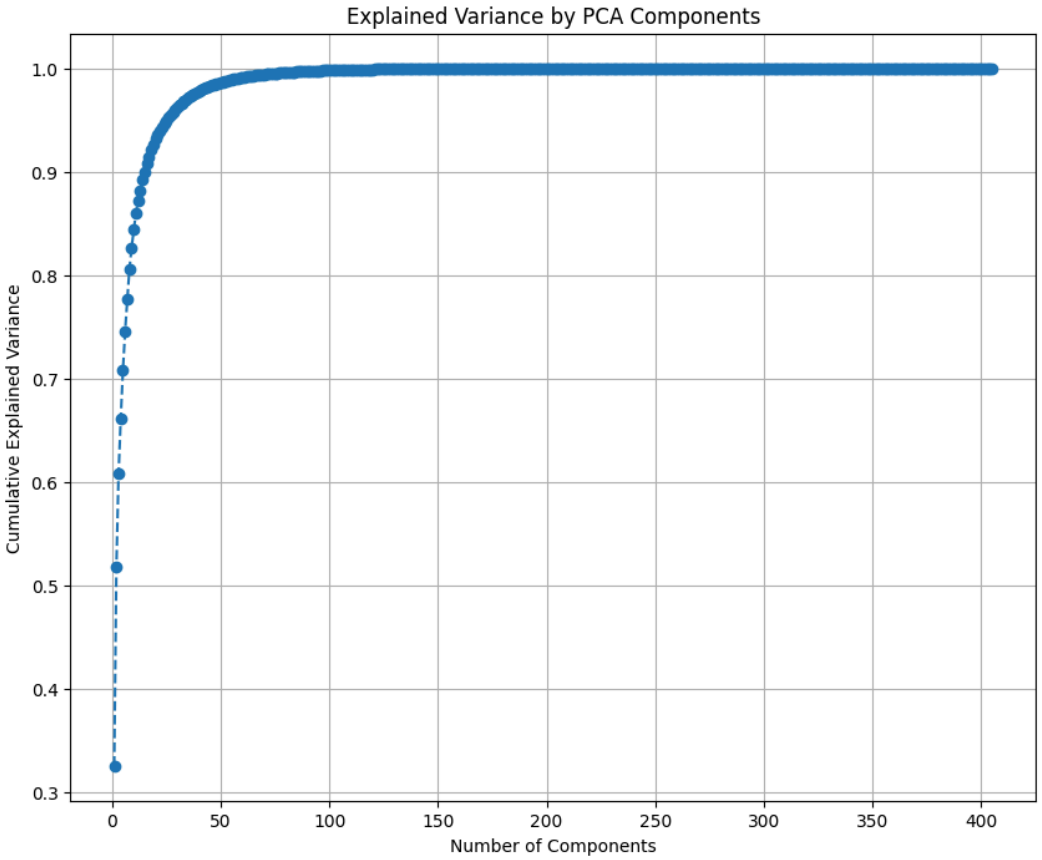
• **Conclusions and Future Directions**
– Conclude whether 3D mesh-based clusters add independent prognostic value over traditional single-segment wall thickness measures.
– Propose further validation studies to refine and integrate these 3D metrics into HCM risk stratification guidelines.

CLUSTERING CODE

```
------------------------------------------------------------
1) INSTALL REQUIRED PACKAGES
------------------------------------------------------------
```

```
!pip install pandas openpyxl scikit-learn matplotlib hdbscan
!pip install plotly==5.13.1

 ------------------------------------------------------------
 2) IMPORT LIBRARIES
 ------------------------------------------------------------
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
import matplotlib.pyplot as plt
import plotly.express as px
from google.colab import files
from itertools import repeat
import plotly.graph_objects as go
import hdbscan

 ------------------------------------------------------------
 3) UPLOAD EXCEL FILE FROM LOCAL (FOR GOOGLE COLAB → WHERE CODE WAS
WRITTEN FOR CLUSTERING)
 ------------------------------------------------------------
uploaded = files.upload()

# Use your filename here
file_path = 'hcm_3d_mesh_projects_final_16AUG2024_ayush.xlsx'
data = pd.read_excel(file_path)

 ------------------------------------------------------------
4) FILTER DATASET (EXAMPLE: KEEP ROWS WITH NONNEG. THICKNESS)
 ------------------------------------------------------------
data_clust = data.query('AHA_1_max_thickness_mm >= 0')




 ------------------------------------------------------------
 5) SELECT ONLY ELEMENT THICKNESS COLUMNS
 ------------------------------------------------------------
elem_cols = [col for col in data_clust.columns if 'e_' in col and '_thickness' in col]
elem_data = data_clust[elem_cols]
```

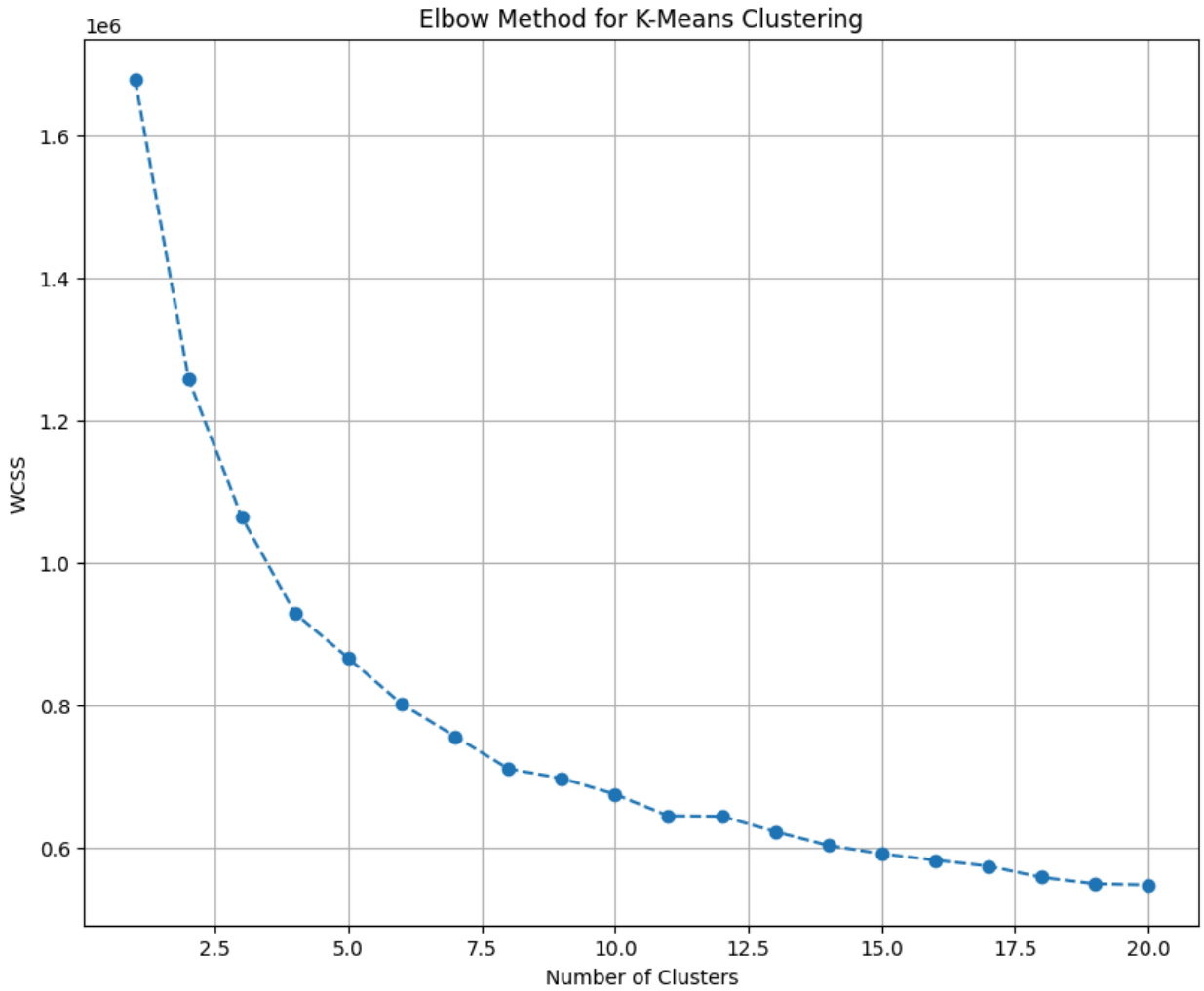
```
------------------------------------------------------------
6) PCA SETUP: INSPECT EXPLAINED VARIANCE
------------------------------------------------------------
pca_full = PCA()
pca_full.fit(elem_data)

plt.figure(figsize=(10, 8))
plt.plot(
    range(1, len(pca_full.explained_variance_ratio_) + 1),
    pca_full.explained_variance_ratio_.cumsum(),
    marker='o', linestyle='--'
)
plt.title('Explained Variance by PCA Components')
plt.xlabel('Number of Components')
plt.ylabel('Cumulative Explained Variance')
plt.grid(True)
plt.show()
```




```
# Based on the plot (e.g., ~80% variance), select 8 components
pca = PCA(n_components=8)
pca.fit(elem_data)
scores_pca = pca.transform(elem_data)
```

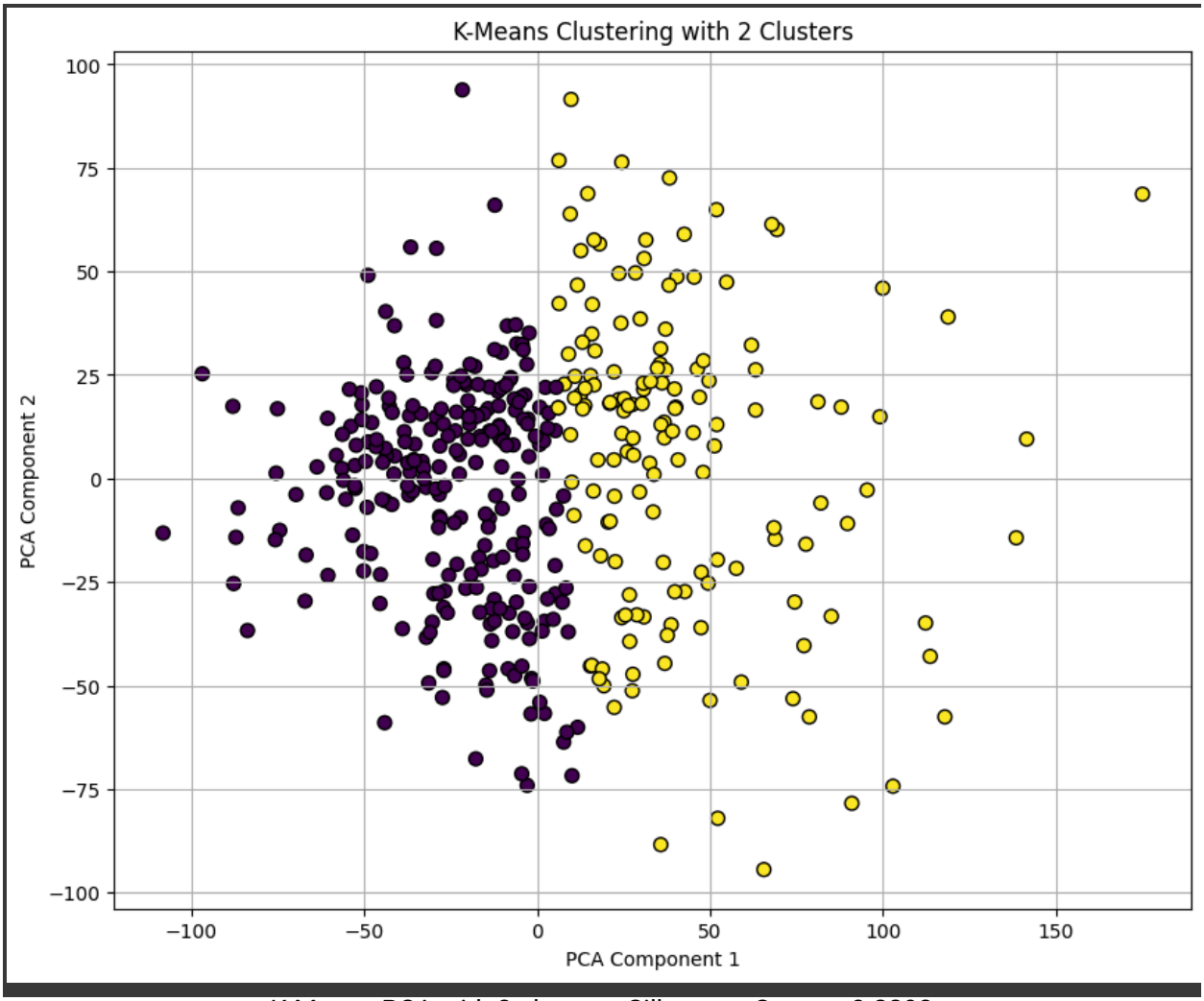
---
7) ELBOW METHOD FOR K-MEANS (WCSS)

```
wcss = []
for i in range(1, 21):
    km = KMeans(n_clusters=i, init='k-means++', random_state=42)
    km.fit(scores_pca)
    wcss.append(km.inertia_)
plt.figure(figsize=(10, 8))
plt.plot(range(1, 21), wcss, marker='o', linestyle='--')
plt.title('Elbow Method for K-Means Clustering')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.grid(True)
plt.show()
```

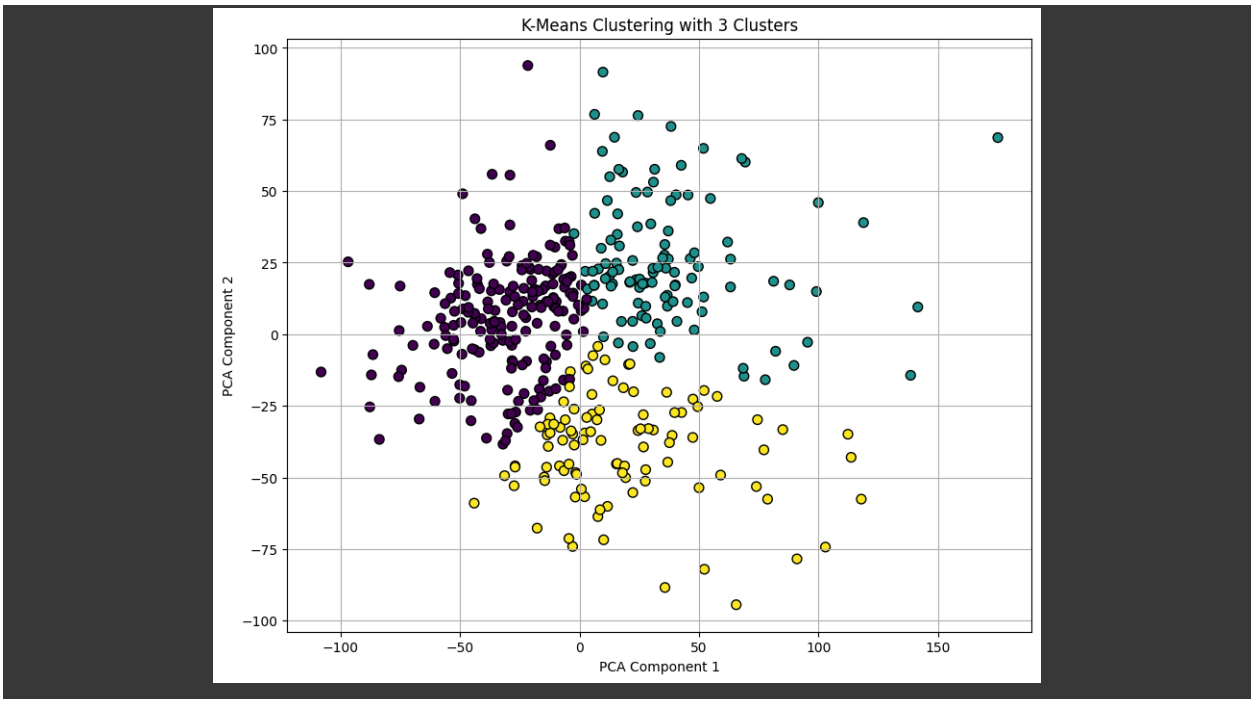


```
 8) K-MEANS (VARIOUS CLUSTER COUNTS) & SILHOUETTE SCORE
 ------------------------------------------------------------
clusters = [2, 3, 4, 5, 6, 7]
sil_scores = []
kmeans_models = {}

for n_clusters in clusters:
    km = KMeans(n_clusters=n_clusters, init='k-means++', random_state=42)
    km.fit(scores_pca)
    data_clust[f'cluster_{n_clusters}_pca'] = km.labels_
    kmeans_models[n_clusters] = km

    sil = silhouette_score(scores_pca, km.labels_, metric='euclidean')
    sil_scores.append(sil)
    print(f'K-Means PCA with {n_clusters} clusters: Silhouette Score = {sil:.4f}')

    # 2D Scatter Plot for visualization
    plt.figure(figsize=(10, 8))
    plt.scatter(
        scores_pca[:, 0],
        scores_pca[:, 1],
        c=km.labels_,
        cmap='viridis',
        marker='o',
        edgecolor='k',
        s=50
    )
    plt.title(f'K-Means Clustering with {n_clusters} Clusters')
    plt.xlabel('PCA Component 1')
    plt.ylabel('PCA Component 2')
    plt.grid(True)
    plt.show()
```

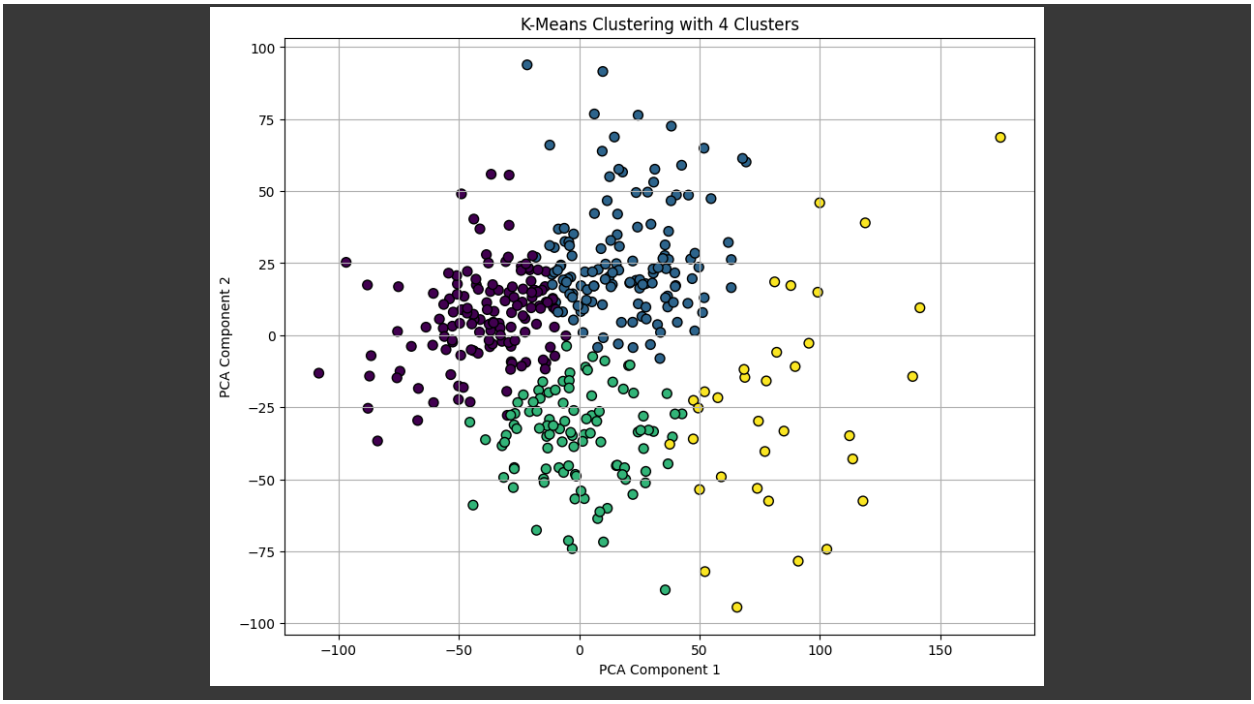
K-Means PCA with 2 clusters: Silhouette Score = 0.2523



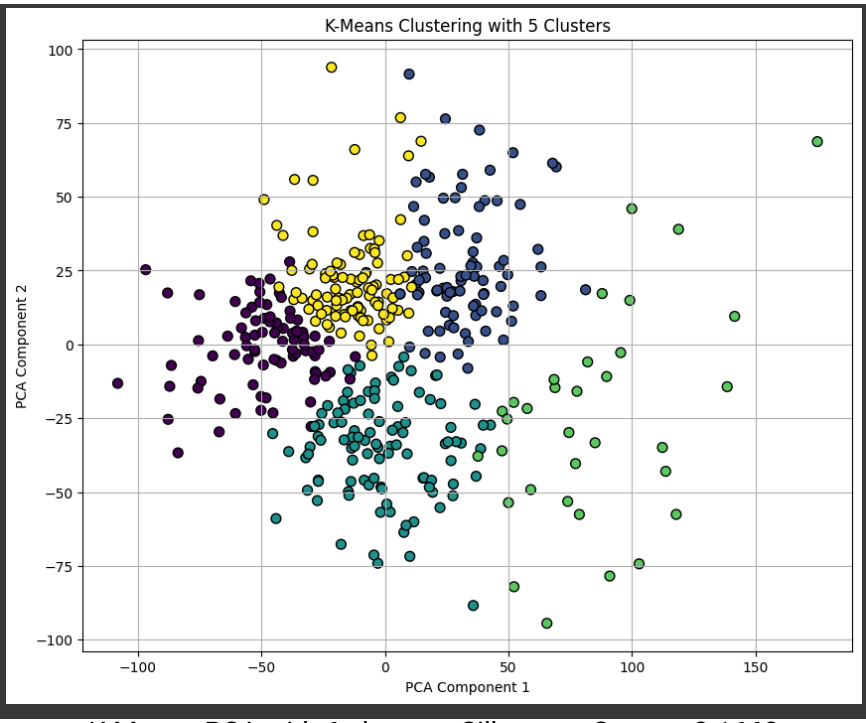
K-Means PCA with 3 clusters: Silhouette Score = 0.2293



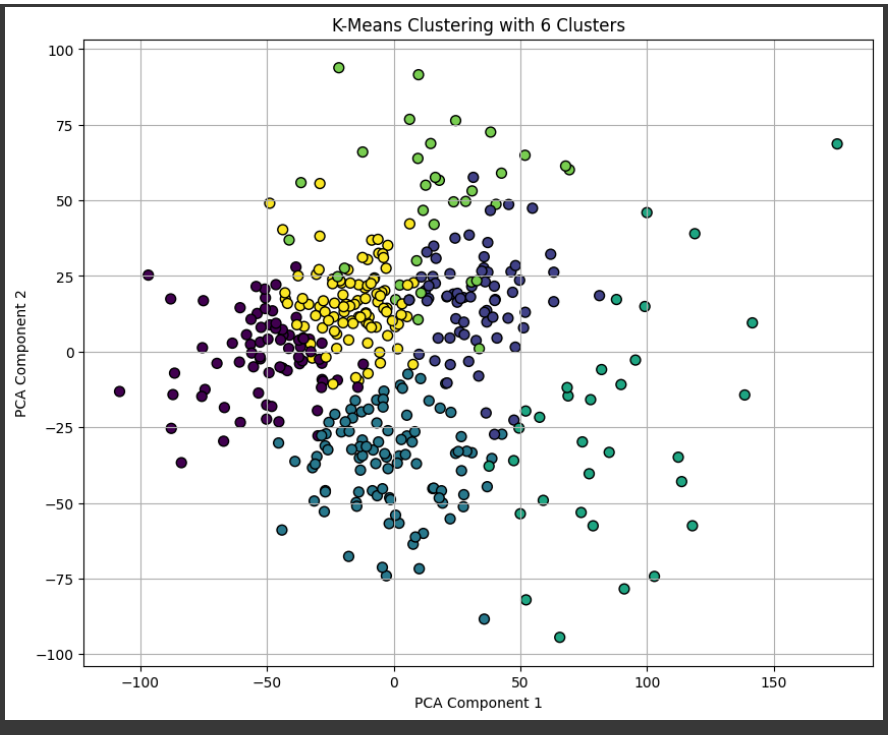
K-Means PCA with 4 clusters: Silhouette Score = 0.2081



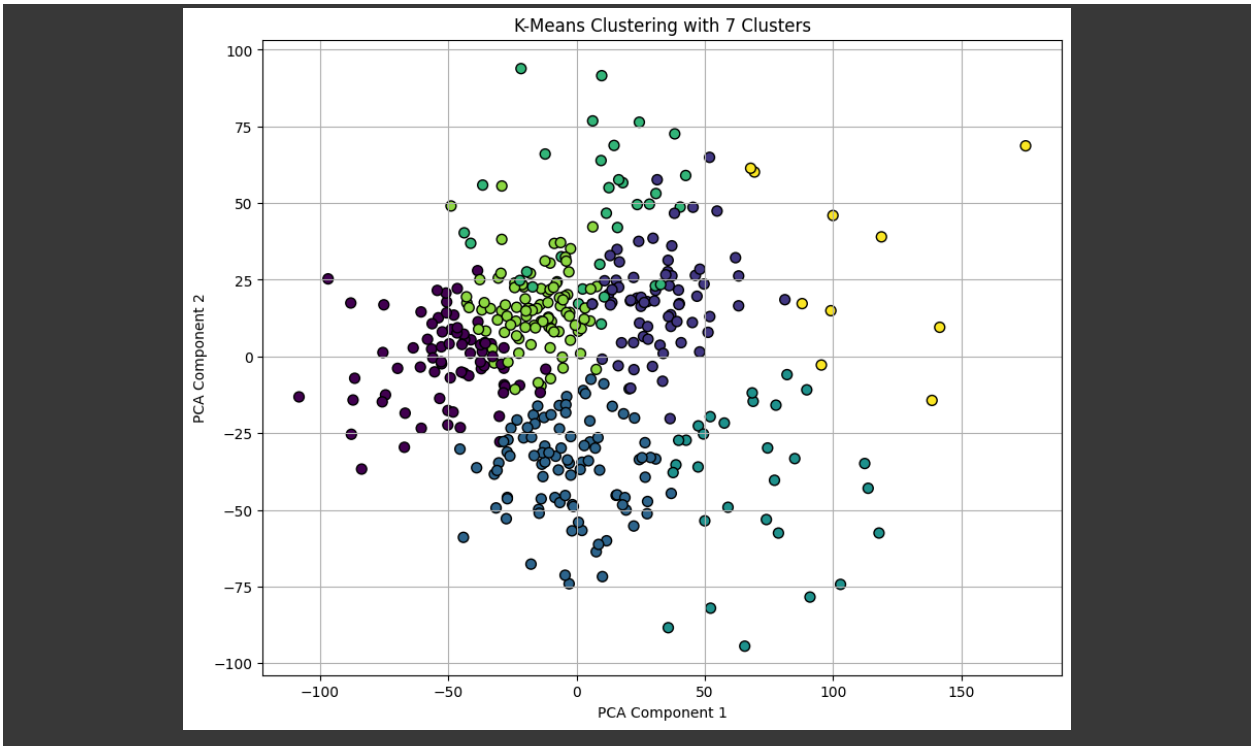
K-Means PCA with 5 clusters: Silhouette Score = 0.1698



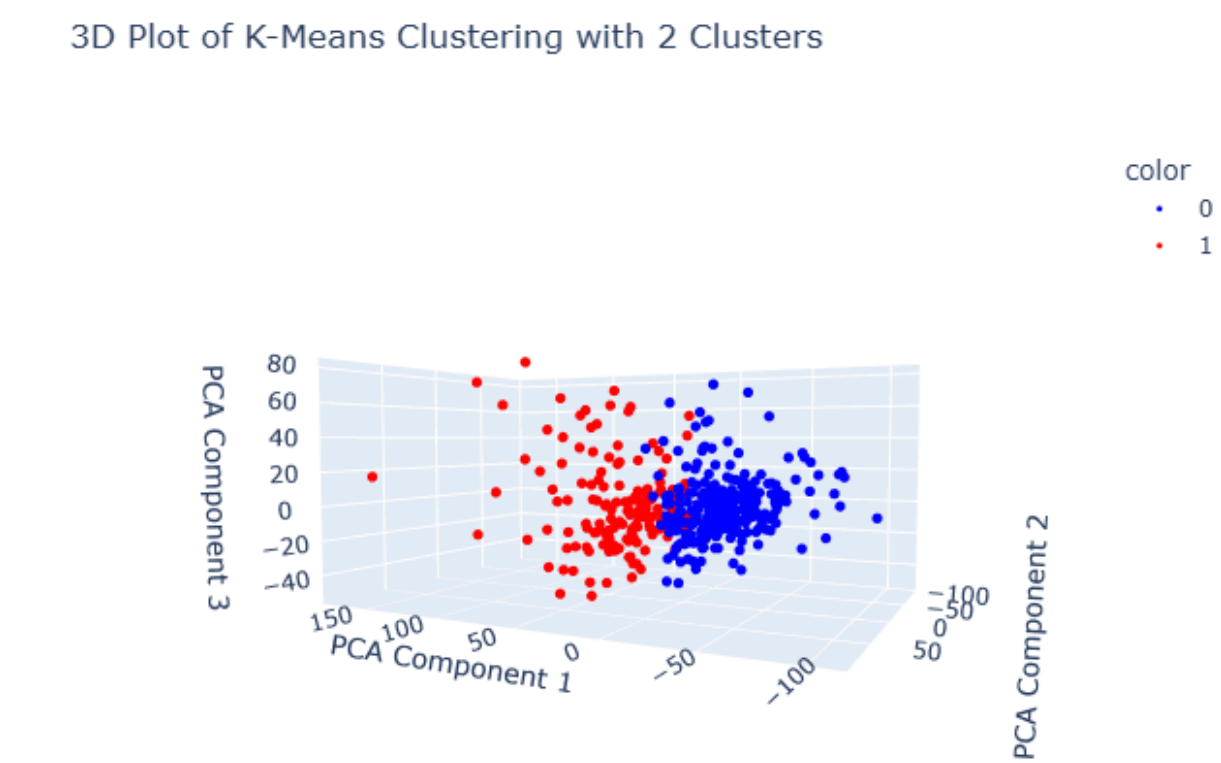
K-Means PCA with 6 clusters: Silhouette Score = 0.1660



K-Means PCA with 7 clusters: Silhouette Score = 0.1693




```
# Example: Choose k=2 (based on your analysis), then visualize
km_2 = KMeans(n_clusters=2, init='k-means++', random_state=42)
km_2.fit(scores_pca)
data_clust['cluster_2_pca'] = km_2.labels_
fig_km = px.scatter_3d(
    x=scores_pca[:, 0],
    y=scores_pca[:, 1],
    z=scores_pca[:, 2],
    color=km_2.labels_.astype(str),
    labels={'x': 'PCA Component 1', 'y': 'PCA Component 2', 'z': 'PCA Component 3'},
    title='3D Plot of K-Means Clustering with 2 Clusters',
    color_discrete_map={'0': 'blue', '1': 'red'}
)
fig_km.update_traces(marker=dict(size=3))
fig_km.show()
```

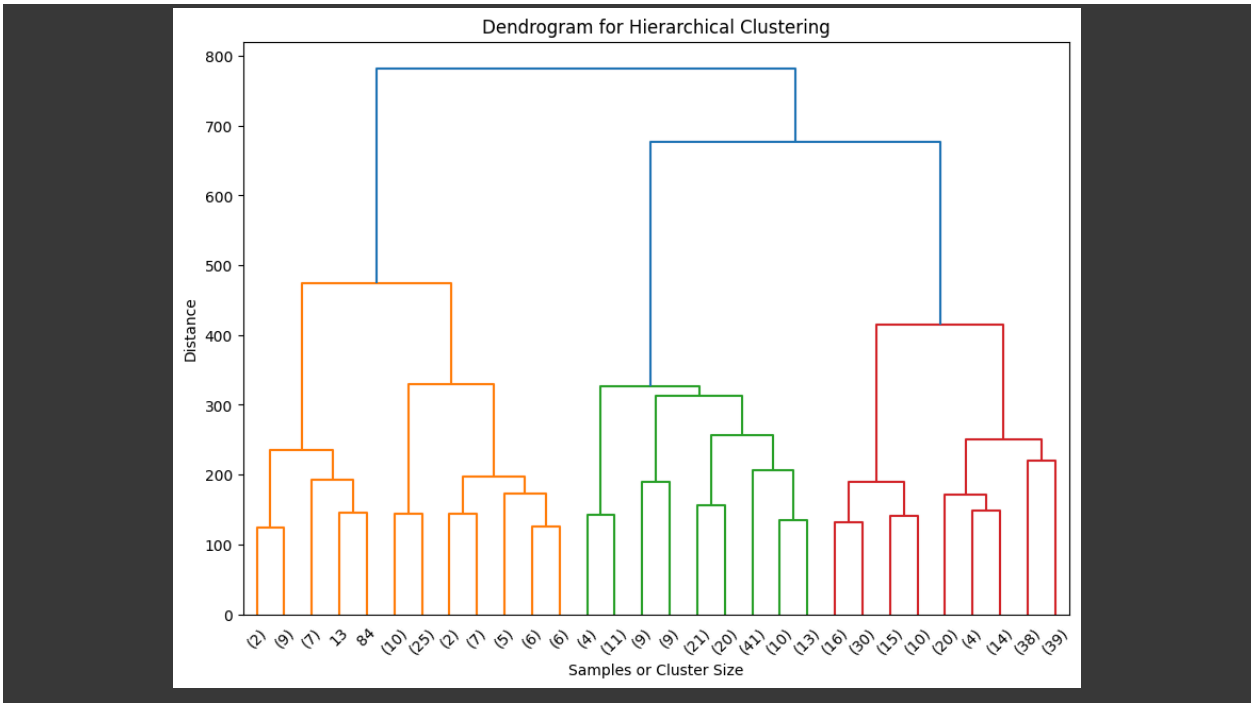


---

9) SAVE CLUSTER RESULTS

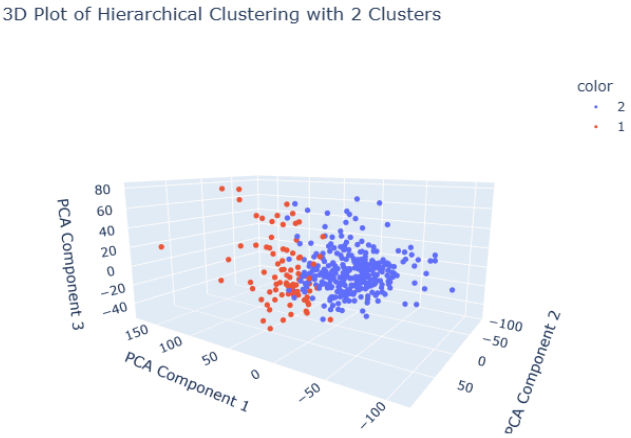
```
output_file = 'final_dataset_with_clusters.xlsx'
data_clust.to_excel(output_file, index=False)
files.download(output_file)
```

```
 ------------------------------------------------------------
 HIERARCHICAL CLUSTERING (WARD LINKAGE + DENDROGRAM) CODE
 ------------------------------------------------------------
from scipy.cluster.hierarchy import linkage, dendrogram, fcluster

link_matrix = linkage(scores_pca, method='ward')

plt.figure(figsize=(10, 7))
plt.title("Dendrogram for Hierarchical Clustering")
dendrogram(link_matrix, truncate_mode='lastp', p=30)
plt.xlabel("Samples or Cluster Size")
plt.ylabel("Distance")
plt.show()

# Test cluster numbers (2 to 10) using silhouette
hier_sil_scores = {}
for n in range(2, 11):
    hier_labels = fcluster(link_matrix, n, criterion='maxclust')
    score = silhouette_score(scores_pca, hier_labels)
    hier_sil_scores[n] = score
    print(f'Hierarchical Clustering with {n} clusters: Silhouette Score = {score:.4f}')

# Find optimal # of clusters
optimal_hier = max(hier_sil_scores, key=hier_sil_scores.get)
print(f"\nOptimal number of clusters (Hierarchical): {optimal_hier} "
      f"with Silhouette Score = {hier_sil_scores[optimal_hier]:.4f}")

# Assign labels & visualize
hier_labels_opt = fcluster(link_matrix, optimal_hier, criterion='maxclust')
data_clust['cluster_hier'] = hier_labels_opt

fig_hier = px.scatter_3d(
    x=scores_pca[:, 0],
    y=scores_pca[:, 1],
    z=scores_pca[:, 2],
    color=data_clust['cluster_hier'].astype(str),
    labels={'x': 'PCA Component 1', 'y': 'PCA Component 2', 'z': 'PCA Component 3'},
    title=f'3D Plot of Hierarchical Clustering with {optimal_hier} Clusters'
)
fig_hier.update_traces(marker=dict(size=3))
fig_hier.show()
```

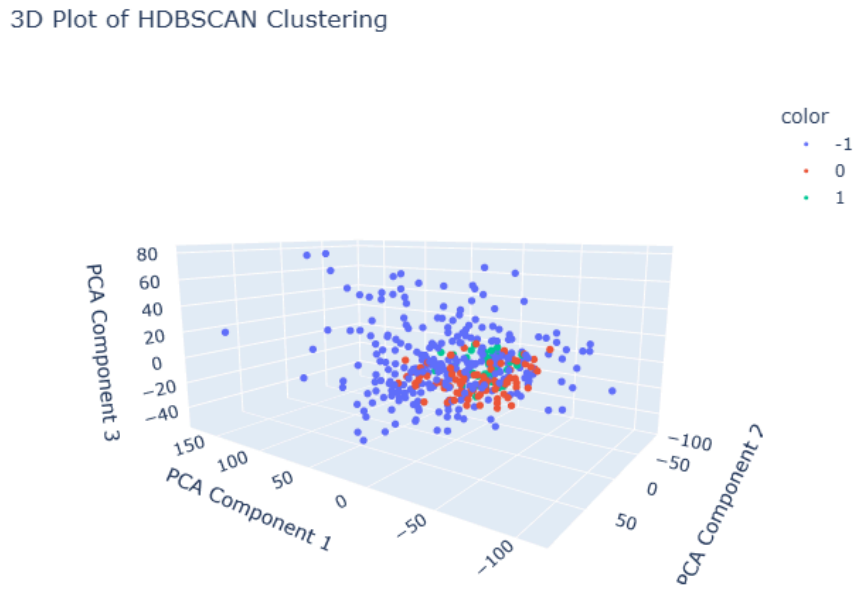


Hierarchical Clustering with 2 clusters: Silhouette Score = 0.2683
Hierarchical Clustering with 3 clusters: Silhouette Score = 0.1835
Hierarchical Clustering with 4 clusters: Silhouette Score = 0.1857
Hierarchical Clustering with 5 clusters: Silhouette Score = 0.1212
Hierarchical Clustering with 6 clusters: Silhouette Score = 0.1160
Hierarchical Clustering with 7 clusters: Silhouette Score = 0.1195
Hierarchical Clustering with 8 clusters: Silhouette Score = 0.1367
Hierarchical Clustering with 9 clusters: Silhouette Score = 0.1192
Hierarchical Clustering with 10 clusters: Silhouette Score = 0.1170

Optimal number of clusters (Hierarchical): 2 with Silhouette Score = 0.2683



```
 ------------------------------------------------------------
 HDBSCAN (DENSITY-BASED) CODE
 ------------------------------------------------------------
hdb = hdbscan.HDBSCAN(min_cluster_size=10, min_samples=1)
hdb_labels = hdb.fit_predict(scores_pca)
data_clust['cluster_hdbscan'] = hdb_labels

# Calculate silhouette ignoring noise (-1) if it exists
if len(set(hdb_labels)) > 1 and -1 in set(hdb_labels):
    mask = (hdb_labels != -1)
    try:
        hdb_score = silhouette_score(scores_pca[mask], hdb_labels[mask])
    except Exception as e:
        hdb_score = -1
else:
    try:
        hdb_score = silhouette_score(scores_pca, hdb_labels)
    except Exception as e:
        hdb_score = -1

print("HDBSCAN Clustering Results:")
print("Unique cluster labels:", set(hdb_labels))
print("Silhouette Score (ignoring noise if present):", hdb_score)

# 3D Plot for HDBSCAN
fig_hdb = px.scatter_3d(
    x=scores_pca[:, 0],
    y=scores_pca[:, 1],
    z=scores_pca[:, 2],
    color=data_clust['cluster_hdbscan'].astype(str),
    labels={'x': 'PCA Component 1', 'y': 'PCA Component 2', 'z': 'PCA Component 3'},
    title='3D Plot of HDBSCAN Clustering'
)
fig_hdb.update_traces(marker=dict(size=3))
fig_hdb.show()
```


3D Plot of HDBSCAN Clustering
color
-1
0
1
PCA Component 3
80
60
40
20
0
−20
−40
PCA Component 1
150
100
50
0
−50
−100
PCA Component 2
−100
−50
0
50

STATISTICAL TESTING CODE

TABLE ONE CODE

```
"SET CORRECT WORKING DIRECTORY: SESSION (TOP TOOLBAR) --> SET WORKING DIRECTORY --> CHOOSE DIRECTORY"
setwd("C:/Users/Ayush_Helli/Desktop/HYRS - Project Data")

"CLEAR MEMORY"
rm(list = ls())

"LOAD THE FOLLOWING REQUIRED PACKAGES (IF MISSING, INSTALL FIRST)"
library(foreign)     # read SPSS files
library(readxl)      # read Excel files
library(gtsummary)   # create tables
library(tidyverse)   # data manipulation
library(jmv)         # normality checks

"READ THE EXCEL FILE (SPECIFY SHEET IF MULTIPLE)"
hcm <- read_excel("hcmDATAfinal2.xlsx", sheet = "dataset")
options(max.print = 1000000)

"ATTACH THE DATAFRAME FOR DIRECT COLUMN REFERENCE"
attach(hcm)
names(hcm)

"OPTIONAL CONVERSION OF VARIABLES TO FACTORS (UNCOMMENT AND ADJUST AS NEEDED)"
# hcm$smoking <- factor(hcm$smoking, labels= c("Never", "Current", "Former"))
# hcm$alcohol_recoded <- factor(hcm$alcohol_recoded, labels= c("None", "Occasional/light", "Moderate/heavy"))
# hcm$ethnicity_recoded <- factor(hcm$ethnicity_recoded, labels= c("White/Caucasian", "Asian", "Others"))
# hcm$marital_recoded <- factor(hcm$marital_recoded, labels= c("Single", "Married/common-law", "Divorced/separated/widowed"))
# hcm$education_recoded <- factor(hcm$education_recoded, labels= c("None", "Educated Up to undergraduate", "Graduate (MSc/PhD)", "Other"))
# hcm$employment_recoded <- factor(hcm$employment_recoded, labels= c("Disability/unemployed", "Employed", "Student", "Retired", "Homemaker"))

"NORMALITY CHECKS"
descriptives(
```

```
  hcm,
  vars = colnames(hcm[, c(103:107)]),
  splitBy = comp_outcome_hcm,
  hist = TRUE,
  qq = TRUE,
  box = TRUE,
  sw = TRUE,
  skew = TRUE,
  kurt = TRUE
)

"TABLE FOR NORMALLY DISTRIBUTED VARIABLES & CATEGORICAL"
hcm %>%
  # Example filter lines if needed:
  # filter(DM == 0) %>%
  # filter(p_sex == 'Female' & (cancer_type == 'Breast Cancer' | (LVEF < 60 & tumor_stage < 4)))
%>%
  select(
    1140, 3, 4, 601, 619, 623, 625, 627, 629, 630, 631,
    632, 645, 647, 648, 652, 653, 149, 151
  ) %>%
  tbl_summary(
    by = cluster_2_pca,
    statistic = list(
      all_continuous() ~ "{mean} ± {sd}",
      all_categorical() ~ "{n} ({p}%)"
    ),
    missing = "no",
    digits = list(
      all_continuous() ~ c(1, 1),
      all_categorical() ~ c(0, 0)
    )
  ) %>%
  add_overall() %>%
  add_p(
    test = list(all_continuous() ~ "t.test"),
    pvalue_fun = function(x) style_pvalue(x, digits = 2)
  ) %>%
  bold_labels() %>%
  italicize_levels() %>%
  bold_p() %>%
  modify_header(
    update = all_stat_cols(FALSE) ~ "**{level}**, N = {n} ({style_percent(p, symbol = TRUE)})"
  )
```

```
"TABLE FOR NON-NORMALLY DISTRIBUTED VARIABLES (& OPTIONAL CATEGORICAL)"
hcm %>%
  select(
    1140, 35, 210, 646, 99, 100, 101, 103, 104, 105,
    106, 107, 556, 656, 657
  ) %>%
  tbl_summary(
    by = cluster_2_pca,
    statistic = list(
      all_continuous() ~ "{median} ({p25}, {p75})",
      all_categorical() ~ "{n} ({p}%)"
    ),
    missing = "no",
    digits = list(
      all_continuous() ~ c(1, 1, 1),
      all_categorical() ~ c(0, 0)
    )
  ) %>%
  add_overall() %>%
  add_p(
    test = list(all_continuous() ~ "wilcox.test"),
    pvalue_fun = function(x) style_pvalue(x, digits = 2)
  ) %>%
  bold_labels() %>%
  italicize_levels() %>%
  bold_p() %>%
  modify_header(
    update = all_stat_cols(FALSE) ~ "**{level}**, N = {n} ({style_percent(p, symbol = TRUE)})"
  )
```

COX MODEL + HAZARD TEST CODE

```
"SET WORKING DIRECTORY (SESSION → SET WORKING DIRECTORY → CHOOSE DIRECTORY)"
setwd("C:/Users/Ayush_Helli/Desktop/HYRS - Project Data")

"CLEAR MEMORY"
rm(list = ls())

"LOAD REQUIRED PACKAGES (INSTALL FIRST IF NEEDED)"
library(tidyverse)  # Data manipulation
library(readxl)     # Reading Excel files
library(gtsummary)  # Creating tables
library(survival)   # Survival analysis
library(survminer)  # Kaplan-Meier curves & related plots

"READ YOUR DATASET"
hcm <- read_excel("hcmDATAfinal.xlsx", sheet = "dataset")
options(max.print = 1000000)
attach(hcm)

names(hcm)  # Inspect variable names in console

"OPTIONAL: DECLARE CATEGORICAL/FACTOR VARIABLES"
# hcm$bcq_smoking_nicotene_collapsed <- factor(hcm$bcq_smoking_nicotene_collapsed, labels = c("Never", "Current", "Former"))
# etc., for other variables as needed

"UNIVARIABLE COX REGRESSION (HAZARD RATIOS)"
hcm %>%
  select(3, 4, 107, 103, 556, 653:654, 1140) %>%
  tbl_uvregression(
    method = coxph,
    y = Surv(time = survival_time_comp_outcome_hcm, event = comp_outcome_hcm),
    exponentiate = TRUE,
    estimate_fun = purrr::partial(style_ratio, digits = 3),
    pvalue_fun   = function(x) style_pvalue(x, digits = 2)
  ) %>%
  bold_p() %>%
  bold_labels() %>%
  modify_table_styling(
    column           = estimate,
    rows             = !is.na(estimate),
    cols_merge_pattern= "{estimate} ({conf.low}-{conf.high})"
```

```
  ) %>%
  modify_header(estimate ~ "**HR (95% CI)**") %>%
  modify_column_hide(ci)

"MULTIVARIABLE COX MODEL"
model_hcm_outcome <- coxph(
  Surv(survival_time_comp_outcome_hcm, comp_outcome_hcm) ~
    cs_age_at_scan +
    cs_sex +
    r_LA_volume_i +
    r_LVMASS_i +
    r_lge_visual_percent_LV_68_subseg +
    factor(cluster_2_pca),
  data = hcm
)

tbl_regression(
  model_hcm_outcome,
  exponentiate = TRUE,
  estimate_fun = function(x) style_number(x, digits = 3),
  pvalue_fun   = function(x) style_pvalue(x, digits = 2)
) %>%
  bold_p() %>%
  bold_labels() %>%
  modify_table_styling(
    column            = estimate,
    rows              = !is.na(estimate),
    cols_merge_pattern= "{estimate} ({conf.low}-{conf.high})"
  ) %>%
  modify_header(estimate ~ "**HR (95% CI)**") %>%
  modify_column_hide(ci)

summary(model_hcm_outcome)

"KAPLAN-MEIER CURVES"
model_smoking <- survfit(
  Surv(survival_time_comp_outcome_hcm, comp_outcome_hcm) ~ cluster_2_pca,
  data = hcm
)

fig <- ggsurvplot(
  model_smoking,
  risk.table        = TRUE,
  pval              = TRUE,
```

```
  ggtheme          = theme_survminer(),
  palette          = c("black", "red", "blue"),
  xlab             = "Survival time (days)",
  ylab             = "Freedom from composite event",
  legend.labs      = c("1", "2"),
  legend.title     = "Cluster Association",
  censor           = FALSE,
  size             = 0.7,
  risk.table.height = 0.3,
  tables.theme     = theme_cleantable()
)

fig

"SAVE FIGURE TO PDF"
pdf("KM_smoking.pdf", width = 7, height = 4.5)
fig
dev.off()

"CHECK NUMBER OF EVENTS"
sum(hcm$comp_outcome_hcm)
```