# Model 3

use\_cols = [

"GAD1","GAD2", "GAD3", "GAD4", "GAD5", "GAD6", "GAD7", "PHQ1", "PHQ2", "PHQ3", "PHQ4", "PHQ5", "PHQ6", "PHQ7", "PHQ8", "PHQ9"]

df = pd.read\_csv("0917-athlete-anxiety-cluster2.csv", usecols=use\_cols)

df["ROLE"].astype(str)

df["GAD1\_TEXT"] = df["GAD1"].astype(str)

df["GAD2\_TEXT"] = df["GAD2"].astype(str)

df["GAD3\_TEXT"] = df["GAD3"].astype(str)

df["GAD4\_TEXT"] = df["GAD4"].astype(str)

df["GAD5\_TEXT"] = df["GAD5"].astype(str)

df["GAD6\_TEXT"] = df["GAD6"].astype(str)

df["GAD7\_TEXT"] = df["GAD7"].astype(str)

df["PHQ1\_TEXT"] = df["PHQ1"].astype(str)

df["PHQ2\_TEXT"] = df["PHQ2"].astype(str)

df["PHQ3\_TEXT"] = df["PHQ3"].astype(str)

df["PHQ4\_TEXT"] = df["PHQ4"].astype(str)

df["PHQ5\_TEXT"] = df["PHQ5"].astype(str)

df["PHQ6\_TEXT"] = df["PHQ6"].astype(str)

df["PHQ7\_TEXT"] = df["PHQ7"].astype(str)

df["PHQ8\_TEXT"] = df["PHQ8"].astype(str)

df["PHQ9\_TEXT"] = df["PHQ9"].astype(str)

def row\_to\_text(row):

return (

f"This student athlete felt nervous, anxious, or on edge {row['GAD1\_TEXT']} during the last two weeks."

f"This student athlete was not able to stop or control worrying {row['GAD2\_TEXT']} during the last two weeks."

f"This student athlete worried too much about different things {row['GAD3\_TEXT']} during the last two weeks."

f"This student athlete had trouble relaxing {row['GAD4\_TEXT']} during the last two weeks."

f"This student athlete felt so restless that it was hard to sit still {row['GAD5\_TEXT']} during the last two weeks."

f"This student athlete became easily annoyed or irritable {row['GAD6\_TEXT']} during the last two weeks."

f"This student athlete felt afraid as if something awful might happen {row['GAD7\_TEXT']} during the last two weeks."

f"This student athlete had little interest or pleasure in doing things {row['PHQ1\_TEXT']} during the last two weeks."

f"This student athlete felt down, depressed, or hopeless {row['PHQ2\_TEXT']} during the last two weeks."

f"This student athlete had trouble sleeping problems {row['PHQ3\_TEXT']} during the last two weeks."

f"This student athlete felt tired or had little energy {row['PHQ4\_TEXT']} during the last two weeks."

f"This student athlete had a poor appetite or overate {row['PHQ5\_TEXT']} during the last two weeks."

f"This student athlete felt like a failure {row['PHQ6\_TEXT']} during the last two weeks."

f"This student athlete had trouble concentrating problems {row['PHQ7\_TEXT']} during the last two weeks."

f"This student athlete was so fidgety or restless that they were moving around a lot more than usual {row['PHQ8\_TEXT']} during the last two weeks."

f"This student athlete had thoughts that they would be better off dead or of hurting themselves {row['PHQ9\_TEXT']} during the last two weeks."

)

df['text'] = df.apply(row\_to\_text, axis=1)

model = SentenceTransformer('sentence-transformers/all-roberta-large-v1')

embeddings = model.encode(

df['text'].tolist(),

convert\_to\_numpy=True,

normalize\_embeddings=True,

show\_progress\_bar=True

)

bert\_df = pd.DataFrame(embeddings)

bert\_df.to\_csv("0917athlete——anxiety——cluster2\_embeddings\_bert\_1024d.csv", index=False)

pca = PCA()

pca.fit(embeddings)

explained = np.cumsum(pca.explained\_variance\_ratio\_)

optimal\_dims = np.argmax(explained >= 0.85) + 1

print(f"✅ Optimal PCA dimensions: {optimal\_dims}")

pca = PCA(n\_components=optimal\_dims)

reduced = pca.fit\_transform(embeddings)

pca\_df = pd.DataFrame(reduced)

pca\_df.to\_csv(f"embeddings\_pca\_{reduced.shape[1]}d.csv", index=False)

sil\_scores = []

K\_range = range(2, min(10, len(df)))

for k in K\_range:

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

labels = km.fit\_predict(reduced)

sil\_scores.append(silhouette\_score(reduced, labels))

best\_k = K\_range[np.argmax(sil\_scores)]

print(f"✅ Optimal number of clusters (K): {best\_k} (Silhouette score = {max(sil\_scores):.3f})")

plt.plot(K\_range, sil\_scores, marker='o')

plt.title("Silhouette Scores vs Cluster Number")

plt.xlabel("Number of Clusters (K)")

plt.ylabel("Silhouette Score")

plt.grid(True)

plt.show()

final\_kmeans = KMeans(n\_clusters=best\_k, n\_init='auto', random\_state=42).fit(reduced)

labels = final\_kmeans.labels\_

sil\_low\_eu = silhouette\_score(reduced, labels) sil\_high\_cos = silhouette\_score(embeddings, labels, metric="cosine") print(f"[Trans] Silhouette (PCA-space, euclid) = {sil\_low\_eu:.3f}")

print(f"[Trans] Silhouette (ORIGINAL embeddings, cosine) = {sil\_high\_cos:.3f}")

pca\_2d = PCA(n\_components=2, random\_state=42)

reduced\_2d = pca\_2d.fit\_transform(embeddings)

plt.figure()

plt.scatter(reduced\_2d[:, 0], reduced\_2d[:, 1], c=labels, cmap='viridis', s=50)

plt.title(f"2D PCA Projection of Clusters (K={best\_k})")

plt.grid(True); plt.show()

pca\_3d = PCA(n\_components=3, random\_state=42)

reduced\_3d = pca\_3d.fit\_transform(embeddings)

fig = plt.figure()

ax = fig.add\_subplot(111, projection='3d')

ax.scatter(reduced\_3d[:, 0], reduced\_3d[:, 1], reduced\_3d[:, 2], c=labels, cmap='viridis', s=50)

ax.set\_title(f"3D PCA Projection of Clusters (K={best\_k})")

plt.show()

final\_kmeans = KMeans(n\_clusters=best\_k, random\_state=42)

df['Cluster'] = final\_kmeans.fit\_predict(reduced)

print("\n📊 Clustered Results:")

# print(df[['text', 'Cluster']])

# Model 1/2 part

from sklearn.preprocessing import StandardScaler

raw\_features = df.drop(columns=['text', 'Cluster', 'ROLE', 'ROLE\_TEXT'], errors='ignore') \

.select\_dtypes(include=np.number)

scaler = StandardScaler()

raw\_Xs = scaler.fit\_transform(raw\_features)

# raw\_Xs = raw\_features.values

sil\_scores\_raw = []

for k in K\_range:

km = KMeans(n\_clusters=k, n\_init='auto', random\_state=42) # ✅ n\_init

labels = km.fit\_predict(raw\_Xs) # ✅ 用 raw\_Xs

sil\_scores\_raw.append(silhouette\_score(raw\_Xs, labels))

best\_k\_raw = K\_range[np.argmax(sil\_scores\_raw)]

print(f"🔵 Raw Data + KMeans: Best K = {best\_k\_raw} (Silhouette = {max(sil\_scores\_raw):.3f})")

plt.plot(K\_range, sil\_scores\_raw, marker='o')

plt.title("Silhouette Scores (Raw Data + KMeans)")

plt.xlabel("Number of Clusters (K)")

plt.ylabel("Silhouette Score")

plt.grid(True)

plt.show()

pca\_r\_full = PCA().fit(raw\_Xs)

expl\_r = np.cumsum(pca\_r\_full.explained\_variance\_ratio\_)

dims\_r = np.argmax(expl\_r >= 0.85) + 1

pca\_raw = PCA(n\_components=dims\_r, random\_state=42)

raw\_reduced = pca\_raw.fit\_transform(raw\_Xs)

sil\_scores\_raw\_pca = []

for k in K\_range:

km = KMeans(n\_clusters=k, n\_init='auto', random\_state=42) # ✅ n\_init

labels = km.fit\_predict(raw\_reduced)

sil\_scores\_raw\_pca.append(silhouette\_score(raw\_reduced, labels))

best\_k\_raw\_pca = K\_range[np.argmax(sil\_scores\_raw\_pca)]

print(f"🟢 Raw Data + PCA + KMeans: Best K = {best\_k\_raw\_pca} (Silhouette = {max(sil\_scores\_raw\_pca):.3f})")

raw\_pca\_kmeans\_final = KMeans(n\_clusters=best\_k\_raw\_pca, n\_init='auto', random\_state=42).fit(raw\_reduced)

labels\_raw\_pca = raw\_pca\_kmeans\_final.labels\_

sil\_low\_raw = silhouette\_score(raw\_reduced, labels)

sil\_high\_raw = silhouette\_score(raw\_Xs, labels)

print(f"[Raw] Silhouette (PCA-space) = {sil\_low\_raw:.3f}")

print(f"[Raw] Silhouette (ORIGINAL raw\_Xs) = {sil\_high\_raw:.3f}")

plt.plot(K\_range, sil\_scores\_raw\_pca, marker='o')

plt.title("Silhouette Scores (Raw Data + PCA + KMeans)")

plt.xlabel("Number of Clusters (K)")

plt.ylabel("Silhouette Score")

plt.grid(True)

plt.show()

model\_names = ['BERT + PCA + KMeans', 'Raw + KMeans', 'Raw + PCA + KMeans']

best\_ks = [best\_k, best\_k\_raw, best\_k\_raw\_pca]

sil\_scores\_max = [max(sil\_scores), max(sil\_scores\_raw), max(sil\_scores\_raw\_pca)]

comparison\_df = pd.DataFrame({

'Model': model\_names,

'Best K': best\_ks,

'Max Silhouette Score': sil\_scores\_max

})

comparison\_df.to\_csv("0917athlete——anxiety——cluster\_model\_comparison.csv", index=False)

print("\n📊 Model Comparison Summary:")

print(comparison\_df)