```python
### SCRIPT 13 - PYTHON
# ANÁLISIS CLUSTERING COMPARATIVA K=6 y K=8
#
# ======================================================================
# IMPORTACIÓN DE LIBRERÍAS
# ======================================================================

import pandas as pd
import numpy as np
from datetime import datetime
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.metrics import silhouette_score
import warnings
warnings.filterwarnings('ignore')

print("="*80)
print("PROYECTO: ANÁLISIS CLUSTERING LICITACIONES PÚBLICAS V8")
print("="*80)
print("Dataset combinado Train+Test con clustering K=6 y K=8")

# ======================================================================
# PASO 1: CARGA DE DATASETS BRUTOS
# ======================================================================

print("\n1. CARGA DE DATASETS BRUTOS:")
print("-" * 50)

# Cargar datasets originales
df_train = pd.read_excel('df_train_v5_completo.xlsx')
df_test = pd.read_excel('df_test_v5_completo.xlsx')

print(f"✓ Dataset entrenamiento cargado: {df_train.shape}")
print(f"✓ Dataset test cargado: {df_test.shape}")
print(f"✓ Total registros: {df_train.shape[0] + df_test.shape[0]}")

# Verificar columnas clave
columnas_clave = ['Primera_publicacion_c', 'Presupuesto_licitacion_lote_c',
                  'Precio_final_e_c', 'C_precio_p']
print(f"\nVerificación columnas clave:")
for col in columnas_clave:
    train_exists = col in df_train.columns
    test_exists = col in df_test.columns
    print(f"  {col}: Train={train_exists}, Test={test_exists}")

# ======================================================================
# PASO 2: CREACIÓN DE VARIABLES DERIVADAS
# ======================================================================

print(f"\n2. CREACIÓN DE VARIABLES DERIVADAS:")
print("-" * 50)

# Variable Ahorro_final con fórmula correcta
# Ahorro_final (%) = (Presupuesto_licitacion_lote_c - Precio_final_e_c) / Presupuesto_licitacion_lote_c * 100
df_train['Ahorro_final'] = ((df_train['Presupuesto_licitacion_lote_c'] - df_train['Precio_final_e_c']) /
                            df_train['Presupuesto_licitacion_lote_c'] * 100)

df_test['Ahorro_final'] = ((df_test['Presupuesto_licitacion_lote_c'] - df_test['Precio_final_e_c']) /
                           df_test['Presupuesto_licitacion_lote_c'] * 100)

# Variable Mes_lici desde Primera_publicacion_c
df_train['Mes_lici'] = df_train['Primera_publicacion_c'].dt.month
df_test['Mes_lici'] = df_test['Primera_publicacion_c'].dt.month

print(f"✓ Variables derivadas creadas exitosamente")

# Estadísticas de Ahorro_final
print(f"\nESTADÍSTICAS AHORRO_FINAL:")
```

```python
print(f"Train - Media: {df_train['Ahorro_final'].mean():.2f}%")
print(f"Train - Desv.Std: {df_train['Ahorro_final'].std():.2f}%")
print(f"Train - Sobrecostos: {(df_train['Ahorro_final'] < 0).sum()} ({(df_train['Ahorro_final'] < 0).mean()*100:.1f}%)")

print(f"Test - Media: {df_test['Ahorro_final'].mean():.2f}%")
print(f"Test - Desv.Std: {df_test['Ahorro_final'].std():.2f}%")
print(f"Test - Sobrecostos: {(df_test['Ahorro_final'] < 0).sum()} ({(df_test['Ahorro_final'] < 0).mean()*100:.1f}%)")

# ======================================================================
# PASO 3: UNIÓN DE DATASETS
# ======================================================================

print(f"\n3. UNIÓN DE DATASETS:")
print("-" * 50)

# Crear variable identificadora de origen
df_train['Dataset'] = 'TRAIN'
df_test['Dataset'] = 'TEST'

# Unir datasets
df_combined = pd.concat([df_train, df_test], axis=0, ignore_index=True)

print(f"✓ Dataset combinado creado: {df_combined.shape}")
print(f"  - Registros TRAIN: {(df_combined['Dataset'] == 'TRAIN').sum()}")
print(f"  - Registros TEST: {(df_combined['Dataset'] == 'TEST').sum()}")

# Verificar integridad
print(f"\nVERIFICACIÓN INTEGRIDAD:")
print(f"  - Nulos en Ahorro_final: {df_combined['Ahorro_final'].isnull().sum()}")
print(f"  - Nulos en Mes_lici: {df_combined['Mes_lici'].isnull().sum()}")
print(f"  - Rango Ahorro_final: [{df_combined['Ahorro_final'].min():.1f}%, {df_combined['Ahorro_final'].max():.1f}%]")

# ======================================================================
# PASO 4: PREPARACIÓN PARA CLUSTERING
# ======================================================================

print(f"\n4. PREPARACIÓN PARA CLUSTERING:")
print("-" * 50)

# Variables POST-licitación para clustering (5 variables especificadas)
variables_clustering = [
    'Presupuesto_licitacion_lote_c',  # Valor inicial de licitación
    'N_ofertantes',                   # Número de ofertas recibidas
    'Plazo_m',                        # Plazo de ejecución en meses
    'Baja_p',                         # Porcentaje de baja económica
    'C_precio_p'                      # Peso del criterio precio
]

print(f"Variables para clustering:")
for i, var in enumerate(variables_clustering, 1):
    print(f"  {i}. {var}")

# Crear dataset para clustering
df_clustering = df_combined[variables_clustering + ['Ahorro_final', 'Dataset']].copy()
print(f"\nDataset clustering: {df_clustering.shape}")

# Verificar completitud y limpiar
nulos = df_clustering.isnull().sum()
print(f"Nulos por variable: {nulos.sum()} total")

df_clustering_clean = df_clustering.dropna()
print(f"Dataset limpio: {df_clustering_clean.shape}")

# ======================================================================
# PASO 5: NORMALIZACIÓN Y CLUSTERING
# ======================================================================

print(f"\n5. NORMALIZACIÓN Y CLUSTERING:")
print("-" * 50)
```

```python

# Preparar datos para clustering (solo variables de entrada)
X_clustering = df_clustering_clean[variables_clustering].copy()
print(f"Matriz de clustering: {X_clustering.shape}")

# Normalizar datos
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_clustering)
print(f"✓ Datos normalizados con StandardScaler")

# Realizar clustering K=6 y K=8
kmeans_6 = KMeans(n_clusters=6, random_state=42, n_init=10)
kmeans_8 = KMeans(n_clusters=8, random_state=42, n_init=10)

clusters_6 = kmeans_6.fit_predict(X_scaled)
clusters_8 = kmeans_8.fit_predict(X_scaled)

# Agregar clusters al dataset limpio
df_clustering_clean['Cluster_6'] = clusters_6
df_clustering_clean['Cluster_8'] = clusters_8

print(f"✓ Clustering completado")
print(f"K=6 distribución: {np.bincount(clusters_6)}")
print(f"K=8 distribución: {np.bincount(clusters_8)}")

# Calcular métricas de calidad
silhouette_6 = silhouette_score(X_scaled, clusters_6)
silhouette_8 = silhouette_score(X_scaled, clusters_8)
inertia_6 = kmeans_6.inertia_
inertia_8 = kmeans_8.inertia_

print(f"\nMétricas de calidad:")
print(f"K=6 - Silhouette Score: {silhouette_6:.4f}, Inertia: {inertia_6:.2f}")
print(f"K=8 - Silhouette Score: {silhouette_8:.4f}, Inertia: {inertia_8:.2f}")

# ======================================================================
# PASO 6: INTEGRACIÓN AL DATASET COMPLETO
# ======================================================================

print(f"\n6. INTEGRACIÓN DE CLUSTERS AL DATASET COMPLETO:")
print("-" * 50)

# Crear dataset final con clusters
df_final = df_combined.copy()
df_final['Cluster_6'] = np.nan
df_final['Cluster_8'] = np.nan

# Asignar clusters usando índices
df_final.loc[df_clustering_clean.index, 'Cluster_6'] = df_clustering_clean['Cluster_6'
]
df_final.loc[df_clustering_clean.index, 'Cluster_8'] = df_clustering_clean['Cluster_8'
]

print(f"✓ Dataset final: {df_final.shape}")
print(f"✓ Cluster_6 asignados: {df_final['Cluster_6'].notna().sum()}")
print(f"✓ Cluster_8 asignados: {df_final['Cluster_8'].notna().sum()}")

# Guardar dataset final
df_final.to_excel('df_test_train_v8.xlsx', index=False)
print(f"✓ Dataset final guardado: df_test_train_v8.xlsx")

# ======================================================================
# PASO 7: ANÁLISIS ESTADÍSTICO POR CLUSTER
# ======================================================================

print(f"\n7. ANÁLISIS ESTADÍSTICO POR CLUSTER:")
print("-" * 50)

# Estadísticas K=6
cluster_6_stats = df_clustering_clean.groupby('Cluster_6')['Ahorro_final'].agg([
    'count', 'mean', 'std', 'min', 'max'
]).round(2)
```

```python
cluster_6_stats.columns = ['Count', 'Mean_Ahorro', 'Std_Ahorro', 'Min_Ahorro', 'Max_Ahorro']

# Estadísticas K=8
cluster_8_stats = df_clustering_clean.groupby('Cluster_8')['Ahorro_final'].agg([
    'count', 'mean', 'std', 'min', 'max'
]).round(2)
cluster_8_stats.columns = ['Count', 'Mean_Ahorro', 'Std_Ahorro', 'Min_Ahorro', 'Max_Ahorro']

# Guardar estadísticas
cluster_6_stats.to_csv('estadisticas_k6_v8.csv')
cluster_8_stats.to_csv('estadisticas_k8_v8.csv')

print("✓ Estadísticas calculadas y guardadas")

# ======================================================================
# PASO 8: ANÁLISIS PCA PARA VISUALIZACIÓN
# ======================================================================

print(f"\n8. ANÁLISIS PCA PARA VISUALIZACIÓN:")
print("-" * 50)

# PCA para visualización
pca = PCA(n_components=2, random_state=42)
X_pca = pca.fit_transform(X_scaled)

print(f"✓ PCA completado")
print(f"Varianza explicada PC1: {pca.explained_variance_ratio_[0]:.4f}")
print(f"Varianza explicada PC2: {pca.explained_variance_ratio_[1]:.4f}")
print(f"Varianza total explicada: {pca.explained_variance_ratio_.sum():.4f}")

# Crear dataset para visualización
df_viz = pd.DataFrame({
    'PC1': X_pca[:, 0],
    'PC2': X_pca[:, 1],
    'Cluster_6': clusters_6,
    'Cluster_8': clusters_8,
    'Ahorro_final': df_clustering_clean['Ahorro_final'].values,
    'Dataset': df_clustering_clean['Dataset'].values
})

df_viz.to_csv('datos_pca_v8.csv', index=False)
print(f"✓ Datos PCA guardados")

# ======================================================================
# PASO 9: ANÁLISIS DETALLADO POR CLUSTER
# ======================================================================

print(f"\n9. ANÁLISIS DETALLADO POR CLUSTER:")
print("-" * 50)

def analizar_cluster_detallado(df_data, cluster_col, cluster_id):
    """Analiza un cluster específico con todas las variables"""

    cluster_data = df_data[df_data[cluster_col] == cluster_id]

    # Estadísticas básicas
    count = len(cluster_data)
    mean_ahorro = cluster_data['Ahorro_final'].mean()
    std_ahorro = cluster_data['Ahorro_final'].std()
    min_ahorro = cluster_data['Ahorro_final'].min()
    max_ahorro = cluster_data['Ahorro_final'].max()

    # Variables de clustering
    mean_presupuesto = cluster_data['Presupuesto_licitacion_lote_c'].mean()
    mean_ofertas = cluster_data['N_ofertantes'].mean()
    mean_plazo = cluster_data['Plazo_m'].mean()
    mean_baja = cluster_data['Baja_p'].mean()
    mean_precio = cluster_data['C_precio_p'].mean()

    # Distribución por dataset
```

```
        train_pct = (cluster_data['Dataset'] == 'TRAIN').mean() * 100
        test_pct = (cluster_data['Dataset'] == 'TEST').mean() * 100

        # Categorización
        if mean_ahorro < -7:
            perfil = "Alto Sobrecoste"
            riesgo = "MUY ALTO"
        elif mean_ahorro < -3:
            perfil = "Sobrecoste Moderado"
            riesgo = "ALTO"
        elif mean_ahorro < 2:
            perfil = "Equilibrio"
            riesgo = "MEDIO"
        elif mean_ahorro < 8:
            perfil = "Ahorro Moderado"
            riesgo = "BAJO"
        elif mean_ahorro < 12:
            perfil = "Alto Ahorro"
            riesgo = "MUY BAJO"
        else:
            perfil = "Ahorro Excepcional"
            riesgo = "MUY BAJO"

        return {
            'Cluster': cluster_id,
            'Perfil_Ahorro': perfil,
            'Count': count,
            'Pct_Total': count/len(df_data)*100,
            'Mean_Ahorro': mean_ahorro,
            'Std_Ahorro': std_ahorro,
            'Mean_Presupuesto': mean_presupuesto,
            'Mean_Ofertas': mean_ofertas,
            'Mean_Plazo': mean_plazo,
            'Mean_Baja': mean_baja,
            'Mean_Precio': mean_precio,
            'Train_Pct': train_pct,
            'Test_Pct': test_pct,
            'Riesgo': riesgo
        }

    # Análisis para K=6 y K=8
    analisis_k6 = []
    for i in range(6):
        resultado = analizar_cluster_detallado(df_clustering_clean, 'Cluster_6', i)
        analisis_k6.append(resultado)

    analisis_k8 = []
    for i in range(8):
        resultado = analizar_cluster_detallado(df_clustering_clean, 'Cluster_8', i)
        analisis_k8.append(resultado)

    # Convertir a DataFrames y ordenar por ahorro
    df_analisis_k6 = pd.DataFrame(analisis_k6).sort_values('Mean_Ahorro', ascending=False)
    df_analisis_k8 = pd.DataFrame(analisis_k8).sort_values('Mean_Ahorro', ascending=False)

    # Guardar análisis detallados
    df_analisis_k6.to_csv('analisis_detallado_k6_v8.csv', index=False)
    df_analisis_k8.to_csv('analisis_detallado_k8_v8.csv', index=False)

    print(f"✓ Análisis detallado completado")

    # ============================================================
    # PASO 10: RESUMEN FINAL
    # ============================================================

    print(f"\n{'='*80}")
    print("PROCESO COMPLETADO EXITOSAMENTE - V8")
    print(f"{'='*80}")

    print(f"\n📊 RESUMEN EJECUTIVO:")
    print(f"• Dataset combinado: {df_final.shape[0]} contratos ({df_final.shape[1]}
    variables)")
```

```python
print(f"• Train: {(df_final['Dataset'] == 'TRAIN').sum()} contratos")
print(f"• Test: {(df_final['Dataset'] == 'TEST').sum()} contratos")
print(f"• Variables clustering: {len(variables_clustering)}")
print(f"• Ahorro promedio global: {df_final['Ahorro_final'].mean():.2f}%")
print(f"• Sobrecostos totales: {(df_final['Ahorro_final'] < 0).mean()*100:.1f}%")

print(f"\n🎯 MÉTRICAS CLUSTERING:")
print(f"K=6 - Silhouette: {silhouette_6:.4f}, Clusters exitosos: {(df_analisis_k6['Mean_Ahorro'] > 5).sum()}/6")
print(f"K=8 - Silhouette: {silhouette_8:.4f}, Clusters exitosos: {(df_analisis_k8['Mean_Ahorro'] > 5).sum()}/8")

print(f"\n✅ ARCHIVOS GENERADOS:")
print(f"1. df_test_train_v8.xlsx - Dataset combinado con clusters")
print(f"2. estadisticas_k6_v8.csv - Estadísticas clusters K=6")
print(f"3. estadisticas_k8_v8.csv - Estadísticas clusters K=8")
print(f"4. analisis_detallado_k6_v8.csv - Análisis completo K=6")
print(f"5. analisis_detallado_k8_v8.csv - Análisis completo K=8")
print(f"6. datos_pca_v8.csv - Datos para visualización PCA")

print(f"\n🚀 LISTO PARA SIGUIENTE FASE:")
print(f"Random Forest con clusters como variable objetivo")
print(f"Variables PRE-licitación como predictores")

print(f"\n{'='*80}")
print("FIN DEL SCRIPT")
print(f"{'='*80}")
```