Explicación del Código

Este código realiza un análisis exploratorio de datos (EDA) completo para los datasets proporcionados, con enfoque en las variables TCH (%Sac.Caña) y %Sac.Caña, siguiendo los requisitos del taller práctico:

1. Carga de Datos

Carga los archivos Excel HISTORICO\_SUERTES.xlsx y BD\_IPSA\_1940.xlsx

Si los archivos no están disponibles, genera datos sintéticos basados en la estructura proporcionada en los fragmentos

2. Análisis Exploratorio Básico

Muestra información básica del dataset (forma, tipos de datos)

Identifica columnas relevantes para el análisis de TCH y %Sac.Caña

3. Detección de Valores Faltantes

Calcula el porcentaje de valores faltantes por columna

Genera un mapa de calor para visualizar la distribución de valores faltantes

4. Detección de Outliers

Utiliza el método del rango intercuartílico (IQR) para identificar outliers

Genera visualizaciones (boxplots e histogramas) para TCH y %Sac.Caña

Crea una matriz de correlación para identificar relaciones entre variables

5. Creación de Categorías

Define umbrales para categorizar %Sac.Caña en niveles Alto, Medio y Bajo usando percentiles 33 y 66

Define umbrales para categorizar TCH en niveles Alto, Medio y Bajo usando los mismos percentiles

Genera visualizaciones de la distribución de estas categorías

Analiza la relación entre las categorías de TCH y %Sac.Caña

Si está disponible, realiza un análisis por variedad de caña

6. Análisis Adicional con BD\_IPSA\_1940

Aplica los mismos umbrales de categorización al dataset BD\_IPSA\_1940

Muestra la distribución de categorías en este dataset

El código genera múltiples visualizaciones y guarda los datasets con las nuevas categorías para su uso posterior en modelos de clasificación, tal como se requiere en el taller práctico.

Este análisis proporciona una base sólida para entender la distribución de las variables clave, identificar posibles problemas en los datos y establecer criterios para la categorización que será utilizada en los modelos de clasificación.

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from scipy import stats

# Configuración de visualización

plt.style.use('seaborn-whitegrid')

sns.set\_palette("pastel")

plt.rcParams['figure.figsize'] = (12, 8)

plt.rcParams['font.size'] = 12

# ======================

# CARGA DE LOS DATASETS

# ======================

# Cargar los datasets

try:

    # Intentar cargar HISTORICO\_SUERTES.xlsx que contiene las variables objetivo

    df\_historico = pd.read\_excel('HISTORICO\_SUERTES.xlsx')

    # Intentar cargar BD\_IPSA\_1940.xlsx para el análisis de clasificación

    df\_ipsa = pd.read\_excel('BD\_IPSA\_1940.xlsx')

    print("Archivos cargados correctamente.")

except FileNotFoundError:

    print("Error: No se encontraron los archivos. Cargando datos de ejemplo basados en el fragmento proporcionado...")

    # Crear datos de ejemplo basados en el fragmento proporcionado

    # Para HISTORICO\_SUERTES (simulando con los nombres de columnas identificados)

    columnas\_historico = [

        'Período', 'Hacienda', 'Nombre', 'Zona', 'Tenencia', 'Suerte', 'Suelo', 'Area Neta', 'Dist Km', 'Variedad',

        'Cod.Estado #', 'Cod.Estado', 'F.Siembra', 'D.S.', 'Ult.Riego', 'Edad Ult Cos', 'F.Ult.Corte', 'Destino 1=Semilla',

        'Cod. T.Cultivo', 'Cultivo', 'Fec.Madur.', 'Producto', 'Dosis Madurante', 'Semanas mad.', 'TonUltCorte', 'TCH', 'TCHM',

        'Ton.Azucar', 'Rdto', 'TAH', 'TAHM', 'Sac.Caña Precosecha', 'Edad.Precosecha', '%Sac.Caña', '%Sac.Muestreadora',

        '%ATR', 'KATRHM', '%Fibra Caña', '%AR Jugo', '%ME Min', '%ME Veg', '%ME Tot', 'Brix', 'Pureza', 'Vejez', 'Tipo Quema',

        'T.Corte', 'Cerca de', 'Cosechó', 'Num.Riegos', 'M3 Riego', 'DDUlt.Riego', 'Lluvias (2 Meses Ant.)', 'Lluvias Ciclo',

        'Lluvias 0 -3', 'Lluvias tres a seis', 'Lluvias seis a nueve', 'Luvias 9'

    ]

    # Generar datos sintéticos para TCH y %Sac.Caña

    np.random.seed(42)

    n\_samples = 1000

    # Simular datos realistas para TCH (toneladas por hectárea)

    tch\_values = np.random.normal(loc=80, scale=20, size=n\_samples)

    tch\_values = np.clip(tch\_values, 20, 150)  # Valores realistas entre 20 y 150 TCH

    # Simular datos realistas para %Sac.Caña

    sacarosa\_values = np.random.normal(loc=13, scale=2, size=n\_samples)

    sacarosa\_values = np.clip(sacarosa\_values, 8, 18)  # Valores realistas entre 8% y 18%

    # Crear DataFrame con datos sintéticos

    df\_historico = pd.DataFrame({

        'TCH': tch\_values,

        '%Sac.Caña': sacarosa\_values,

        'Lluvias Ciclo': np.random.normal(loc=1000, scale=300, size=n\_samples),

        'Edad.Precosecha': np.random.normal(loc=12, scale=2, size=n\_samples),

        'Variedad': np.random.choice(['CC01-1940', 'SP79-1011', 'RB867515'], size=n\_samples),

        'Tenencia': np.random.choice(['Propia', 'Arrendada', 'Sociedad'], size=n\_samples),

        'Vejez': np.random.normal(loc=5, scale=2, size=n\_samples)

    })

    # Para BD\_IPSA\_1940

    data\_ipsa = [

        [1617, 'KATANOBU', 80573, '002A', 'Mecanizado Verde', 'CC01-1940', 'SI', 'BONUS 250 EC REGULADOR FISIOLÓGICO', 0.8, 10.3, 13.3, 1, 10.3, 0.4, 12.3, 6, '201906', 164, 0, 3, 2.6],

        [1618, 'KATANOBU', 80573, '002A', 'Mecanizado Verde', 'CC01-1940', 'SI', 'BONUS 250 EC REGULADOR FISIOLÓGICO', 0.8, 6.4, 11.8, 2, 8.3, 2, 12.9, 6, '202006', 130, 112, 3, 3.9],

        [1620, 'LA ALDEA ADIELA', 80270, '1', 'Mecanizado Verde', 'CC01-1940', 'SI', 'BONUS 250 EC REGULADOR FISIOLÓGICO', 1.2, 9.6, 12.4, 4, 7.6, 5.7, 14.6, 11, '201911', 123, 81, 2, 3.5]

    ]

    columnas\_ipsa = ['ID', 'NOME', 'FAZ', 'TAL', 'tipocorte', 'variedad', 'madurada', 'producto', 'dosismad',

                    'semsmad', 'edad', 'cortes', 'me', 'vejez', 'sacarosa', 'mes', 'periodo', 'TCH',

                    'lluvias', 'grupo\_tenencia', 'pct\_diatrea']

    df\_ipsa = pd.DataFrame(data\_ipsa, columns=columnas\_ipsa)

# ======================

# EDA PARA HISTORICO\_SUERTES

# ======================

print("\n" + "="\*50)

print("ANÁLISIS EXPLORATORIO DE DATOS - HISTORICO\_SUERTES")

print("="\*50)

# Información básica del dataset

print("\nInformación básica del dataset:")

print(f"Forma del dataset: {df\_historico.shape}")

print(f"Número de columnas: {df\_historico.shape[1]}")

print(f"Número de filas: {df\_historico.shape[0]}")

# Mostrar las primeras filas

print("\nPrimeras 3 filas del dataset:")

print(df\_historico.head(3))

# Verificar tipos de datos

print("\nTipos de datos de las columnas:")

print(df\_historico.dtypes.value\_counts())

# Identificar columnas relevantes para el análisis

print("\nColumnas relevantes identificadas para el análisis:")

relevant\_columns = [col for col in df\_historico.columns if 'TCH' in col or 'Sac' in col or 'sac' in col or 'lluvia' in col or 'edad' in col or 'vejez' in col or 'variedad' in col]

print(relevant\_columns)

# ======================

# DETECCIÓN DE VALORES FALTANTES

# ======================

print("\n" + "="\*50)

print("DETECCIÓN DE VALORES FALTANTES")

print("="\*50)

# Calcular porcentaje de valores faltantes por columna

missing\_values = df\_historico.isnull().sum()

missing\_percent = (missing\_values / len(df\_historico)) \* 100

# Crear DataFrame con resultados

missing\_data = pd.DataFrame({

    'Valores Faltantes': missing\_values,

    '% Faltante': missing\_percent

})

# Filtrar solo columnas con valores faltantes

missing\_data = missing\_data[missing\_data['Valores Faltantes'] > 0].sort\_values('% Faltante', ascending=False)

if not missing\_data.empty:

    print("\nColumnas con valores faltantes:")

    print(missing\_data.head(10))

    # Visualizar valores faltantes

    plt.figure(figsize=(14, 6))

    sns.heatmap(df\_historico[relevant\_columns].isnull(), cbar=False, yticklabels=False, cmap='viridis')

    plt.title('Mapa de Calor de Valores Faltantes en Columnas Relevantes')

    plt.tight\_layout()

    plt.savefig('valores\_faltantes.png', dpi=300, bbox\_inches='tight')

    plt.close()

    print("✓ Mapa de calor de valores faltantes generado: 'valores\_faltantes.png'")

else:

    print("\nNo se encontraron valores faltantes en el dataset.")

# ======================

# DETECCIÓN DE OUTLIERS

# ======================

print("\n" + "="\*50)

print("DETECCIÓN DE OUTLIERS")

print("="\*50)

# Identificar variables numéricas para análisis de outliers

numeric\_cols = df\_historico.select\_dtypes(include=[np.number]).columns.tolist()

# Función para detectar outliers usando método de IQR

def detect\_outliers\_iqr(df, column):

    Q1 = df[column].quantile(0.25)

    Q3 = df[column].quantile(0.75)

    IQR = Q3 - Q1

    lower\_bound = Q1 - 1.5 \* IQR

    upper\_bound = Q3 + 1.5 \* IQR

    outliers = df[(df[column] < lower\_bound) | (df[column] > upper\_bound)]

    return outliers, lower\_bound, upper\_bound

# Buscar variables relevantes para TCH y %Sac.Caña

tch\_col = next((col for col in df\_historico.columns if 'TCH' in col), None)

sacarosa\_col = next((col for col in df\_historico.columns if 'Sac' in col or 'sac' in col), None)

if tch\_col and sacarosa\_col:

    print(f"\nVariables identificadas para análisis:")

    print(f"- TCH: '{tch\_col}'")

    print(f"- %Sac.Caña: '{sacarosa\_col}'")

    # Análisis de outliers para TCH

    tch\_outliers, tch\_lower, tch\_upper = detect\_outliers\_iqr(df\_historico, tch\_col)

    print(f"\nAnálisis de outliers para {tch\_col}:")

    print(f"- Rango normal: [{tch\_lower:.2f}, {tch\_upper:.2f}]")

    print(f"- Número de outliers: {len(tch\_outliers)} ({len(tch\_outliers)/len(df\_historico)\*100:.2f}%)")

    # Análisis de outliers para %Sac.Caña

    sac\_outliers, sac\_lower, sac\_upper = detect\_outliers\_iqr(df\_historico, sacarosa\_col)

    print(f"\nAnálisis de outliers para {sacarosa\_col}:")

    print(f"- Rango normal: [{sac\_lower:.2f}, {sac\_upper:.2f}]")

    print(f"- Número de outliers: {len(sac\_outliers)} ({len(sac\_outliers)/len(df\_historico)\*100:.2f}%)")

    # Visualización de outliers

    plt.figure(figsize=(14, 10))

    # Boxplot para TCH

    plt.subplot(2, 2, 1)

    sns.boxplot(y=df\_historico[tch\_col])

    plt.title(f'Boxplot de {tch\_col}')

    plt.ylabel(tch\_col)

    # Histograma para TCH

    plt.subplot(2, 2, 2)

    sns.histplot(df\_historico[tch\_col], kde=True)

    plt.axvline(x=tch\_lower, color='r', linestyle='--', label=f'Lower Bound: {tch\_lower:.2f}')

    plt.axvline(x=tch\_upper, color='r', linestyle='--', label=f'Upper Bound: {tch\_upper:.2f}')

    plt.title(f'Distribución de {tch\_col}')

    plt.legend()

    # Boxplot para %Sac.Caña

    plt.subplot(2, 2, 3)

    sns.boxplot(y=df\_historico[sacarosa\_col])

    plt.title(f'Boxplot de {sacarosa\_col}')

    plt.ylabel(sacarosa\_col)

    # Histograma para %Sac.Caña

    plt.subplot(2, 2, 4)

    sns.histplot(df\_historico[sacarosa\_col], kde=True)

    plt.axvline(x=sac\_lower, color='r', linestyle='--', label=f'Lower Bound: {sac\_lower:.2f}')

    plt.axvline(x=sac\_upper, color='r', linestyle='--', label=f'Upper Bound: {sac\_upper:.2f}')

    plt.title(f'Distribución de {sacarosa\_col}')

    plt.legend()

    plt.tight\_layout()

    plt.savefig('distribucion\_tch\_sacarosa.png', dpi=300, bbox\_inches='tight')

    print("✓ Visualización de distribución de TCH y %Sac.Caña generada: 'distribucion\_tch\_sacarosa.png'")

    # Análisis de correlación

    plt.figure(figsize=(12, 10))

    corr\_matrix = df\_historico[[tch\_col, sacarosa\_col] + [col for col in numeric\_cols if col not in [tch\_col, sacarosa\_col]][:8]].corr()

    sns.heatmap(corr\_matrix, annot=True, cmap='coolwarm', fmt=".2f")

    plt.title('Matriz de Correlación entre Variables Relevantes')

    plt.tight\_layout()

    plt.savefig('correlacion\_variables.png', dpi=300, bbox\_inches='tight')

    print("✓ Matriz de correlación generada: 'correlacion\_variables.png'")

else:

    print("\nNo se pudieron identificar las variables TCH y %Sac.Caña en el dataset.")

# ======================

# CREACIÓN DE CATEGORÍAS

# ======================

print("\n" + "="\*50)

print("CREACIÓN DE CATEGORÍAS PARA TCH Y %SAC.CAÑA")

print("="\*50)

if tch\_col and sacarosa\_col:

    # Para %Sac.Caña: Niveles Alto, Medio y Bajo

    # Usaremos percentiles 33 y 66 para dividir en tres categorías

    sacarosa\_low = df\_historico[sacarosa\_col].quantile(0.33)

    sacarosa\_high = df\_historico[sacarosa\_col].quantile(0.66)

    print(f"\nUmbrales para %Sac.Caña:")

    print(f"- Bajo: < {sacarosa\_low:.2f}%")

    print(f"- Medio: {sacarosa\_low:.2f}% - {sacarosa\_high:.2f}%")

    print(f"- Alto: > {sacarosa\_high:.2f}%")

    # Crear columna de categoría para %Sac.Caña

    df\_historico['Categoria\_Sacarosa'] = pd.cut(

        df\_historico[sacarosa\_col],

        bins=[-np.inf, sacarosa\_low, sacarosa\_high, np.inf],

        labels=['Bajo', 'Medio', 'Alto']

    )

    # Para TCH: Niveles Alto, Medio y Bajo

    tch\_low = df\_historico[tch\_col].quantile(0.33)

    tch\_high = df\_historico[tch\_col].quantile(0.66)

    print(f"\nUmbrales para TCH:")

    print(f"- Bajo: < {tch\_low:.2f} TCH")

    print(f"- Medio: {tch\_low:.2f} - {tch\_high:.2f} TCH")

    print(f"- Alto: > {tch\_high:.2f} TCH")

    # Crear columna de categoría para TCH

    df\_historico['Categoria\_TCH'] = pd.cut(

        df\_historico[tch\_col],

        bins=[-np.inf, tch\_low, tch\_high, np.inf],

        labels=['Bajo', 'Medio', 'Alto']

    )

    # Visualizar distribución de categorías

    plt.figure(figsize=(14, 6))

    # Distribución de categorías de Sacarosa

    plt.subplot(1, 2, 1)

    sns.countplot(x='Categoria\_Sacarosa', data=df\_historico, order=['Bajo', 'Medio', 'Alto'])

    plt.title('Distribución de Categorías de %Sac.Caña')

    plt.xlabel('%Sac.Caña')

    plt.ylabel('Frecuencia')

    # Añadir porcentajes en las barras

    for p in plt.gca().patches:

        height = p.get\_height()

        plt.gca().annotate(f'{height/len(df\_historico)\*100:.1f}%',

                           (p.get\_x() + p.get\_width()/2., height),

                           ha='center', va='center',

                           xytext=(0, 5),

                           textcoords='offset points')

    # Distribución de categorías de TCH

    plt.subplot(1, 2, 2)

    sns.countplot(x='Categoria\_TCH', data=df\_historico, order=['Bajo', 'Medio', 'Alto'])

    plt.title('Distribución de Categorías de TCH')

    plt.xlabel('TCH')

    plt.ylabel('Frecuencia')

    # Añadir porcentajes en las barras

    for p in plt.gca().patches:

        height = p.get\_height()

        plt.gca().annotate(f'{height/len(df\_historico)\*100:.1f}%',

                           (p.get\_x() + p.get\_width()/2., height),

                           ha='center', va='center',

                           xytext=(0, 5),

                           textcoords='offset points')

    plt.tight\_layout()

    plt.savefig('distribucion\_categorias.png', dpi=300, bbox\_inches='tight')

    print("✓ Distribución de categorías generada: 'distribucion\_categorias.png'")

    # Análisis cruzado de categorías

    plt.figure(figsize=(10, 8))

    cross\_tab = pd.crosstab(df\_historico['Categoria\_Sacarosa'], df\_historico['Categoria\_TCH'])

    sns.heatmap(cross\_tab, annot=True, fmt='d', cmap='YlGnBu', linewidths=.5)

    plt.title('Relación entre Categorías de %Sac.Caña y TCH')

    plt.ylabel('%Sac.Caña')

    plt.xlabel('TCH')

    plt.tight\_layout()

    plt.savefig('relacion\_categorias.png', dpi=300, bbox\_inches='tight')

    print("✓ Relación entre categorías generada: 'relacion\_categorias.png'")

    # Mostrar proporciones

    print("\nProporción de registros en cada categoría:")

    print(f"- %Sac.Caña:")

    print(df\_historico['Categoria\_Sacarosa'].value\_counts(normalize=True).sort\_index() \* 100)

    print(f"\n- TCH:")

    print(df\_historico['Categoria\_TCH'].value\_counts(normalize=True).sort\_index() \* 100)

    # Análisis por variedad (si está disponible)

    if 'Variedad' in df\_historico.columns:

        plt.figure(figsize=(14, 10))

        # Sacarosa por variedad

        plt.subplot(2, 1, 1)

        sns.boxplot(x='Variedad', y=sacarosa\_col, data=df\_historico)

        plt.title(f'Distribución de %Sac.Caña por Variedad')

        plt.xticks(rotation=45)

        plt.tight\_layout()

        # TCH por variedad

        plt.subplot(2, 1, 2)

        sns.boxplot(x='Variedad', y=tch\_col, data=df\_historico)

        plt.title(f'Distribución de TCH por Variedad')

        plt.xticks(rotation=45)

        plt.tight\_layout()

        plt.savefig('analisis\_por\_variedad.png', dpi=300, bbox\_inches='tight')

        print("✓ Análisis por variedad generado: 'analisis\_por\_variedad.png'")

    # Guardar el DataFrame con las nuevas categorías

    df\_historico.to\_csv('HISTORICO\_SUERTES\_con\_categorias.csv', index=False)

    print("\n✓ Dataset con categorías guardado como 'HISTORICO\_SUERTES\_con\_categorias.csv'")

# ======================

# ANÁLISIS ADICIONAL CON BD\_IPSA\_1940

# ======================

print("\n" + "="\*50)

print("ANÁLISIS ADICIONAL CON BD\_IPSA\_1940")

print("="\*50)

# Verificar si las variables están presentes en BD\_IPSA\_1940

tch\_col\_ipsa = next((col for col in df\_ipsa.columns if 'TCH' in col), None)

sacarosa\_col\_ipsa = next((col for col in df\_ipsa.columns if 'sacarosa' in col or 'Sac' in col), None)

if tch\_col\_ipsa and sacarosa\_col\_ipsa:

    print(f"\nVariables identificadas en BD\_IPSA\_1940:")

    print(f"- TCH: '{tch\_col\_ipsa}'")

    print(f"- Sacarosa: '{sacarosa\_col\_ipsa}'")

    # Crear categorías para BD\_IPSA\_1940 usando los mismos umbrales

    df\_ipsa['Categoria\_Sacarosa'] = pd.cut(

        df\_ipsa[sacarosa\_col\_ipsa],

        bins=[-np.inf, sacarosa\_low, sacarosa\_high, np.inf],

        labels=['Bajo', 'Medio', 'Alto']

    )

    df\_ipsa['Categoria\_TCH'] = pd.cut(

        df\_ipsa[tch\_col\_ipsa],

        bins=[-np.inf, tch\_low, tch\_high, np.inf],

        labels=['Bajo', 'Medio', 'Alto']

    )

    print("\nDistribución de categorías en BD\_IPSA\_1940:")

    print(f"- %Sac.Caña: {df\_ipsa['Categoria\_Sacarosa'].value\_counts().to\_dict()}")

    print(f"- TCH: {df\_ipsa['Categoria\_TCH'].value\_counts().to\_dict()}")

    # Guardar el DataFrame con las nuevas categorías

    df\_ipsa.to\_csv('BD\_IPSA\_1940\_con\_categorias.csv', index=False)

    print("\n✓ Dataset con categorías guardado como 'BD\_IPSA\_1940\_con\_categorias.csv'")

else:

    print("\nNo se pudieron identificar las variables TCH y sacarosa en BD\_IPSA\_1940.")

print("\n" + "="\*50)

print("ANÁLISIS COMPLETADO")

print("="\*50)