TAREA SLIDES 10

Utilizando el siguiente código de python:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from statsmodels.tsa.seasonal import seasonal_decompose
import os
import warnings
warnings.filterwarnings('ignore')

print("=== HANDS-ON: ANÁLISIS DE EROSIÓN COSTERA EN CASTELLDEFELS ===")
print("=" * 70)

#
==========================================================================
======
# 1. CARGAR Y PREPARAR DATOS
#
==========================================================================
======
print("\n📁 1. CARGANDO DATOS...")

# Cargar datos - USA EL NOMBRE CORRECTO DE TU ARCHIVO
df = pd.read_csv('datos2.csv')  # ⬅️ AJUSTA SI ES NECESARIO

print("✅ Datos cargados correctamente")
print("\n🔍 INFORMACIÓN DEL DATASET:")
print("Columnas:", df.columns.tolist())
print(f"Tamaño del dataset: {df.shape}")
print(f"Transectos únicos: {df['transect_id'].nunique()}")
print(f"Rango de fechas: {df['date'].min()} a {df['date'].max()}")
print(f"Valores distance_m: {df['distance_m'].isna().sum()} NaNs de
{len(df)} total")

#
==========================================================================
======
```

```
# 2. CREAR TABLA PIVOTE
#
================================================================================
======
print("\n" + "="*50)
print("2. CREANDO TABLA PIVOTE...")

# Convertir fecha
df['date'] = pd.to_datetime(df['date'])

# Crear tabla pivote
pivot_df = df.pivot(index='date', columns='transect_id',
values='distance_m')
print(f"Tabla pivote: {pivot_df.shape[0]} fechas × {pivot_df.shape[1]}
transectos")

#
================================================================================
======
# 3. EXPLORAR DATASET COMPLETO - GRÁFICA DE NANS POR TRANSECTO
#
================================================================================
======
print("\n" + "="*50)
print("3. EXPLORACIÓN DEL DATASET...")

# Porcentaje de NaNs por transecto
print("\n📈 PORCENTAJE DE NaNs POR TRANSECTO:")
nan_percent = pivot_df.isna().sum() / len(pivot_df) * 100
print(nan_percent.sort_values())

# Gráfica de NaNs por transecto
plt.figure(figsize=(12, 6))
nan_percent.sort_values(ascending=False).plot(kind='bar',
color='skyblue')
plt.axhline(y=20, color='red', linestyle='--', label='Límite 20%')
plt.title('Porcentaje de Valores Faltantes por Transecto')
plt.ylabel('Porcentaje de NaNs (%)')
plt.xlabel('Transecto')
plt.legend()
plt.xticks(rotation=45)
plt.tight_layout()
plt.savefig('nans_por_transecto.png', dpi=300, bbox_inches='tight')
```

```
print("✅ Gráfica de NaNs por transecto guardada")

#
================================================================================
======
# 4. ESTRATEGIA ROBUSTA DE FILTRADO Y CALIDAD
#
================================================================================
======
print("\n" + "="*50)
print("4. ESTRATEGIA ROBUSTA DE FILTRADO Y CALIDAD...")

# Análisis exhaustivo de completitud
print("\n📊 ANÁLISIS DE COMPLETITUD TEMPORAL:")
completitud_transectos = (1 - pivot_df.isna().sum() / len(pivot_df)) *
100
completitud_fechas = (1 - pivot_df.isna().sum(axis=1) /
pivot_df.shape[1]) * 100

print(f"• Completitud promedio transectos:
{completitud_transectos.mean():.1f}%")
print(f"• Completitud promedio fechas:
{completitud_fechas.mean():.1f}%")

# Aplicar criterios de calidad estándar
print("\n🎯 APLICANDO CRITERIOS DE CALIDAD ESTÁNDAR:")

# Criterio 1: Transectos con al menos 70% de completitud
transectos_calidad = completitud_transectos[completitud_transectos >=
70].index
print(f"• Transectos con ≥70% completitud:
{len(transectos_calidad)}/{len(pivot_df.columns)}")

# Criterio 2: Fechas con al menos 50% de transectos presentes
fechas_calidad = completitud_fechas[completitud_fechas >= 50].index
print(f"• Fechas con ≥50% completitud:
{len(fechas_calidad)}/{len(pivot_df)}")

# Aplicar filtros
df_calidad = pivot_df.loc[fechas_calidad, transectos_calidad]
print(f"• Dataset de calidad: {df_calidad.shape[0]} fechas ×
{df_calidad.shape[1]} transectos")
```

```
# Si no hay suficientes transectos con 70%, usar los mejores
disponibles
if len(transectos_calidad) < 3:
    print("⚠️  Pocos transectos con 70% completitud, usando los 5
mejores...")
    transectos_calidad = completitud_transectos.nlargest(5).index
    df_calidad = pivot_df.loc[fechas_calidad, transectos_calidad]
    print(f"• Dataset ajustado: {df_calidad.shape[0]} fechas ×
{df_calidad.shape[1]} transectos")

#
==========================================================================
======
# 5. IMPUTACIÓN CON MÉTODOS ROBUSTOS
#
==========================================================================
======
print("\n" + "="*50)
print("5. IMPUTACIÓN CON MÉTODOS ROBUSTOS...")

# Método 1: Interpolación temporal por transecto
print("🔄 Aplicando interpolación temporal...")
df_interpolado = df_calidad.interpolate(
    method='linear',     # Interpolación lineal entre puntos temporales
    limit=3,            # Máximo 3 puntos consecutivos faltantes
    limit_direction='both'
)

# Método 2: Rellenar con media móvil para gaps pequeños
for col in df_interpolado.columns:
    if df_interpolado[col].isna().sum() > 0:
        df_interpolado[col] = df_interpolado[col].fillna(
            df_interpolado[col].rolling(window=5, min_periods=1).mean()
        )

# Método 3: Verificar resultados
nan_final = df_interpolado.isna().sum().sum()
print(f"✅ NaNs después de imputación: {nan_final}
({(nan_final/df_interpolado.size)*100:.2f}%)")

#
==========================================================================
======
```

```
# 6. SERIE TEMPORAL AGREGADA (SIN ELIMINAR OUTLIERS)
#
================================================================================
======
print("\n" + "="*50)
print("6. CONSTRUYENDO SERIE TEMPORAL AGREGADA...")

# Crear serie temporal agregada SIN eliminar outliers
serie_agregada = df_interpolado.mean(axis=1)

# Rellenar posibles NaNs restantes con interpolación
serie_agregada = serie_agregada.interpolate(method='linear')

print(f"• Puntos temporales finales: {len(serie_agregada)}")
print(f"• Completitud serie:
{serie_agregada.notna().sum()/len(serie_agregada)*100:.1f}%")
print(f"• Rango temporal: {serie_agregada.index.min()} a
{serie_agregada.index.max()}")

#
================================================================================
======
# 7. VALIDACIÓN DE CALIDAD
#
================================================================================
======
print("\n" + "="*50)
print("7. VALIDACIÓN DE CALIDAD...")

def validar_calidad_serie(serie):
    """Validación completa de la serie temporal"""
    resultados = {}

    # Completitud
    resultados['completitud'] = serie.notna().sum() / len(serie) * 100

    # Consistencia temporal (no demasiados saltos bruscos)
    diferencias = serie.diff().abs()
    saltos_bruscos = (diferencias > diferencias.quantile(0.95)).sum()
    resultados['saltos_bruscos'] = saltos_bruscos

    return resultados
```

```
# Aplicar validación
calidad = validar_calidad_serie(serie_agregada)
print("📋 MÉTRICAS DE CALIDAD:")
print(f"  • Completitud: {calidad['completitud']:.1f}%")
print(f"  • Saltos bruscos: {calidad['saltos_bruscos']} (> percentil
95)")

# Umbrales de aceptación
if calidad['completitud'] >= 80 and calidad['saltos_bruscos'] <
len(serie_agregada) * 0.05:
    print("✅ CALIDAD ACEPTABLE - Continuando con análisis")
    mean_series = serie_agregada
else:
    print("⚠️  CALIDAD SUBÓPTIMA - Continuando con análisis pero con
precaución")
    mean_series = serie_agregada

#
================================================================================
======
# 8. DETECCIÓN DE OUTLIERS USANDO EL MÉTODO DEL BOXPLOT
#
================================================================================
======
print("\n" + "="*50)
print("8. DETECCIÓN DE OUTLIERS (Método Boxplot)...")

# Preparar datos para boxplot por año
mean_series_df = pd.DataFrame({'distance': mean_series.values},
index=mean_series.index)
mean_series_df['year'] = mean_series_df.index.year

# Detectar outliers usando el método del boxplot por año
outliers_mask = pd.Series(False, index=mean_series.index)

for year in mean_series_df['year'].unique():
    year_data = mean_series_df[mean_series_df['year'] ==
year]['distance']

    # Calcular quartiles para cada año
    Q1 = year_data.quantile(0.25)
    Q3 = year_data.quantile(0.75)
    IQR = Q3 - Q1
```

```
    # Definir límites para outliers
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Identificar outliers para este año
    year_outliers = (year_data < lower_bound) | (year_data >
upper_bound)

    # Actualizar la máscara general
    for idx in year_data[year_outliers].index:
        outliers_mask.loc[idx] = True

outliers_count = outliers_mask.sum()

print(f"🔍 OUTLIERS DETECTADOS POR AÑO:")
for year in mean_series_df['year'].unique():
    year_outliers_count = outliers_mask[mean_series_df['year'] ==
year].sum()
    year_total = (mean_series_df['year'] == year).sum()
    print(f"   • {year}: {year_outliers_count} outliers de {year_total}
puntos ({year_outliers_count/year_total*100:.1f}%)")

print(f"📊 TOTAL: {outliers_count} outliers de {len(mean_series)}
puntos ({outliers_count/len(mean_series)*100:.1f}%)")

# 8.2 Resampling mensual
print("\n📅 REMUESTREO MENSUAL...")
monthly_series = mean_series.resample('M').mean()
monthly_series = monthly_series.interpolate(method='linear')
print(f"Serie mensual: {len(monthly_series)} puntos")

#
================================================================================
======
# 9. VISUALIZACIÓN EXPLORATORIA CON MISMOS OUTLIERS
#
================================================================================
======
print("\n" + "="*50)
print("9. VISUALIZACIÓN EXPLORATORIA...")

# Crear figura con múltiples subgráficas
```

```
fig, axes = plt.subplots(2, 2, figsize=(15, 10))

# Gráfica 1: Serie temporal completa
axes[0,0].plot(mean_series.index, mean_series.values, linewidth=1,
alpha=0.7, label='Diaria')
axes[0,0].plot(monthly_series.index, monthly_series.values,
linewidth=2, color='red', label='Mensual')
# Marcar outliers en la serie temporal
outliers_dates = mean_series[outliers_mask]
axes[0,0].scatter(outliers_dates.index, outliers_dates.values,
color='red', s=30, zorder=5, label=f'Outliers ({outliers_count})')
axes[0,0].set_title('Evolución Temporal de la Distancia a la Costa')
axes[0,0].set_ylabel('Distancia (m)')
axes[0,0].legend()
axes[0,0].grid(True, alpha=0.3)

# Gráfica 2: Distribución
axes[0,1].hist(mean_series.values, bins=50, edgecolor='black',
alpha=0.7, color='lightblue')
axes[0,1].axvline(mean_series.mean(), color='red', linestyle='--',
label=f'Media: {mean_series.mean():.1f}m')
# Marcar región de outliers
Q1_global = mean_series.quantile(0.25)
Q3_global = mean_series.quantile(0.75)
IQR_global = Q3_global - Q1_global
lower_bound_global = Q1_global - 1.5 * IQR_global
upper_bound_global = Q3_global + 1.5 * IQR_global
axes[0,1].axvline(lower_bound_global, color='orange', linestyle=':',
label='Límite outliers')
axes[0,1].axvline(upper_bound_global, color='orange', linestyle=':')
axes[0,1].set_title('Distribución de Distancias a la Costa')
axes[0,1].set_xlabel('Distancia (m)')
axes[0,1].set_ylabel('Frecuencia')
axes[0,1].legend()

# Gráfica 3: Boxplot por año (OUTLIERS AUTOMÁTICOS)
sns.boxplot(data=mean_series_df, x='year', y='distance', ax=axes[1,0])
axes[1,0].set_title('Distribución por Año (Boxplot)')
axes[1,0].set_xlabel('Año')
axes[1,0].set_ylabel('Distancia (m)')
axes[1,0].tick_params(axis='x', rotation=45)

# Gráfica 4: Outliers temporales (MISMOS QUE BOXPLOT)
```

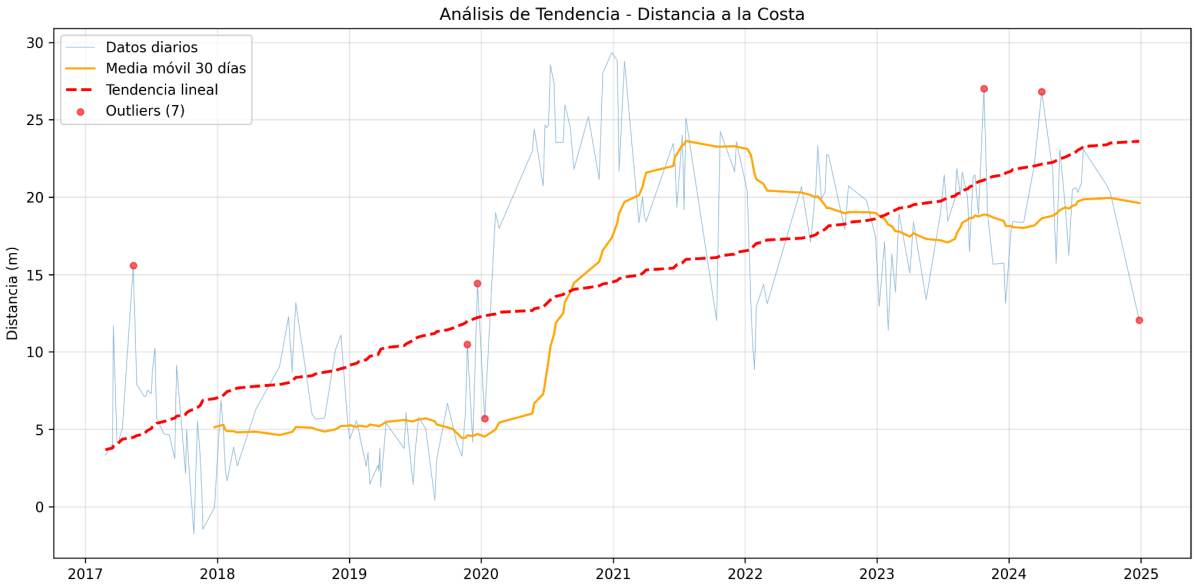
```
colors = ['red' if outlier else 'blue' for outlier in outliers_mask]
axes[1,1].scatter(mean_series.index, mean_series.values, c=colors,
alpha=0.6, s=20)
axes[1,1].set_title(f'Outliers Detectados ({outliers_count} puntos)')
axes[1,1].set_ylabel('Distancia (m)')
axes[1,1].grid(True, alpha=0.3)

plt.tight_layout()
plt.savefig('exploracion_completa.png', dpi=300, bbox_inches='tight')
print("✅ Gráfica exploratoria guardada (OUTLIERS CONSISTENTES)")

#
================================================================================
======
# 10. ANÁLISIS ESTADÍSTICO
#
================================================================================
======
print("\n" + "="*50)
print("10. ANÁLISIS ESTADÍSTICO...")

print("\n📊 ESTADÍSTICAS DESCRIPTIVAS:")
print(mean_series.describe())

# Estadísticas por año
print("\n📅 ESTADÍSTICAS POR AÑO:")
yearly_stats = mean_series.groupby(mean_series.index.year).agg(['mean',
'std', 'min', 'max'])
print(yearly_stats)

#
================================================================================
======
# 11. ANÁLISIS DE TENDENCIAS
#
================================================================================
======
print("\n" + "="*50)
print("11. ANÁLISIS DE TENDENCIAS...")

# 11.1 Moving averages
ma_30 = mean_series.rolling(window=30).mean()
```

```python
# 11.2 Regresión lineal (tendencia)
X = np.arange(len(mean_series)).reshape(-1, 1)
y = mean_series.values

model = LinearRegression()
model.fit(X, y)
y_pred = model.predict(X)

trend_slope = model.coef_[0]  # Pendiente por día
r2 = model.score(X, y)

print(f"📈 TENDENCIA LINEAL:")
print(f"   - Pendiente: {trend_slope:.6f} m/día")
print(f"   - R²: {r2:.4f}")

# Gráfica de tendencias
plt.figure(figsize=(12, 6))
plt.plot(mean_series.index, mean_series.values, alpha=0.5, label='Datos
diarios', linewidth=0.5)
plt.plot(ma_30.index, ma_30.values, label='Media móvil 30 días',
linewidth=1.5, color='orange')
plt.plot(mean_series.index, y_pred, label='Tendencia lineal',
linewidth=2, color='red', linestyle='--')
# Marcar outliers en la tendencia
plt.scatter(outliers_dates.index, outliers_dates.values, color='red',
s=20, alpha=0.6, label=f'Outliers ({outliers_count})')
plt.title('Análisis de Tendencia - Distancia a la Costa')
plt.ylabel('Distancia (m)')
plt.legend()
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.savefig('analisis_tendencia.png', dpi=300, bbox_inches='tight')
print("✅ Gráfica de tendencia guardada")

#
==========================================================================
======
# 12. DESCOMPOSICIÓN DE SERIES TEMPORALES
#
==========================================================================
======
print("\n" + "="*50)
print("12. DESCOMPOSICIÓN DE SERIES TEMPORALES...")
```

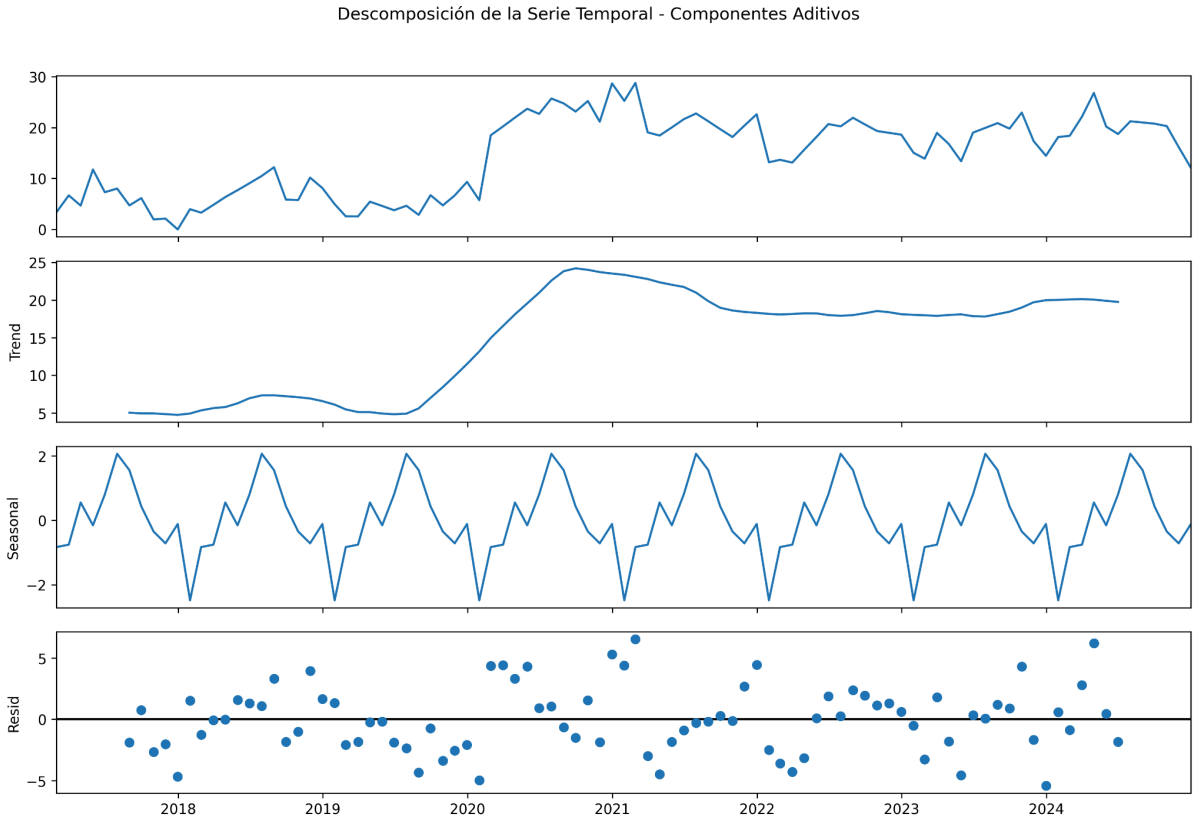
```
try:
    # Usar serie mensual para mejor descomposición
    decomposition = seasonal_decompose(monthly_series,
model='additive', period=12)

    fig = decomposition.plot()
    fig.set_size_inches(12, 8)
    plt.suptitle('Descomposición de la Serie Temporal - Componentes
Aditivos', y=1.02)
    plt.tight_layout()
    plt.savefig('descomposicion_series.png', dpi=300,
bbox_inches='tight')
    print("✅ Gráfica de descomposición guardada")

except Exception as e:
    print(f"⚠️  No se pudo realizar la descomposición: {e}")

#
================================================================================
======
# 13. RESUMEN Y CONCLUSIONES
#
================================================================================
======
print("\n" + "="*70)
print("📋 RESUMEN FINAL Y CONCLUSIONES")
print("="*70)

print(f"\n📊 DATOS GENERALES:")
print(f"   • Período analizado:
{mean_series.index.min().strftime('%Y-%m-%d')} a
{mean_series.index.max().strftime('%Y-%m-%d')}")
print(f"   • Transectos analizados: {len(transectos_calidad)} (de
{len(pivot_df.columns)} totales)")
print(f"   • Puntos temporales: {len(mean_series)}")
print(f"   • Distancia promedio: {mean_series.mean():.1f} m")
print(f"   • Calidad de la serie: {calidad['completitud']:.1f}%
completitud")

print(f"\n📈 TENDENCIAS:")
print(f"   • Pendiente: {trend_slope:.6f} m/día ({trend_slope*365:.3f}
m/año)")
```

```
print(f"   • R² de la tendencia: {r2:.4f}")

print(f"\n🔍 PATRONES DETECTADOS:")
print(f"   • Outliers: {outliers_count} puntos
({outliers_count/len(mean_series)*100:.1f}%)")
print(f"   • Variabilidad interanual: {mean_series.std():.1f} m")
print(f"   • Saltos bruscos: {calidad['saltos_bruscos']} puntos")

if outliers_count > 0:
    print(f"\n📅 OUTLIERS POR AÑO:")
    for year in mean_series_df['year'].unique():
        year_outliers_count = outliers_mask[mean_series_df['year'] ==
year].sum()
        if year_outliers_count > 0:
            print(f"   • {year}: {year_outliers_count} outliers")

print(f"\n💡 PREGUNTAS PARA REFLEXIÓN:")
print(f"   • ¿La tendencia es consistente en todos los transectos?")
print(f"   • ¿Hay patrones estacionales visibles?")
print(f"   • ¿Los outliers coinciden con eventos climáticos
conocidos?")
print(f"   • ¿La calidad de datos es suficiente para conclusiones
robustas?")

print(f"\n" + "="*70)
print("✅ ANÁLISIS COMPLETADO")
print("📁 Gráficas guardadas:")
print("   - nans_por_transecto.png")
print("   - exploracion_completa.png")
print("   - analisis_tendencia.png")
print("   - descomposicion_series.png")
print("="*70)
```

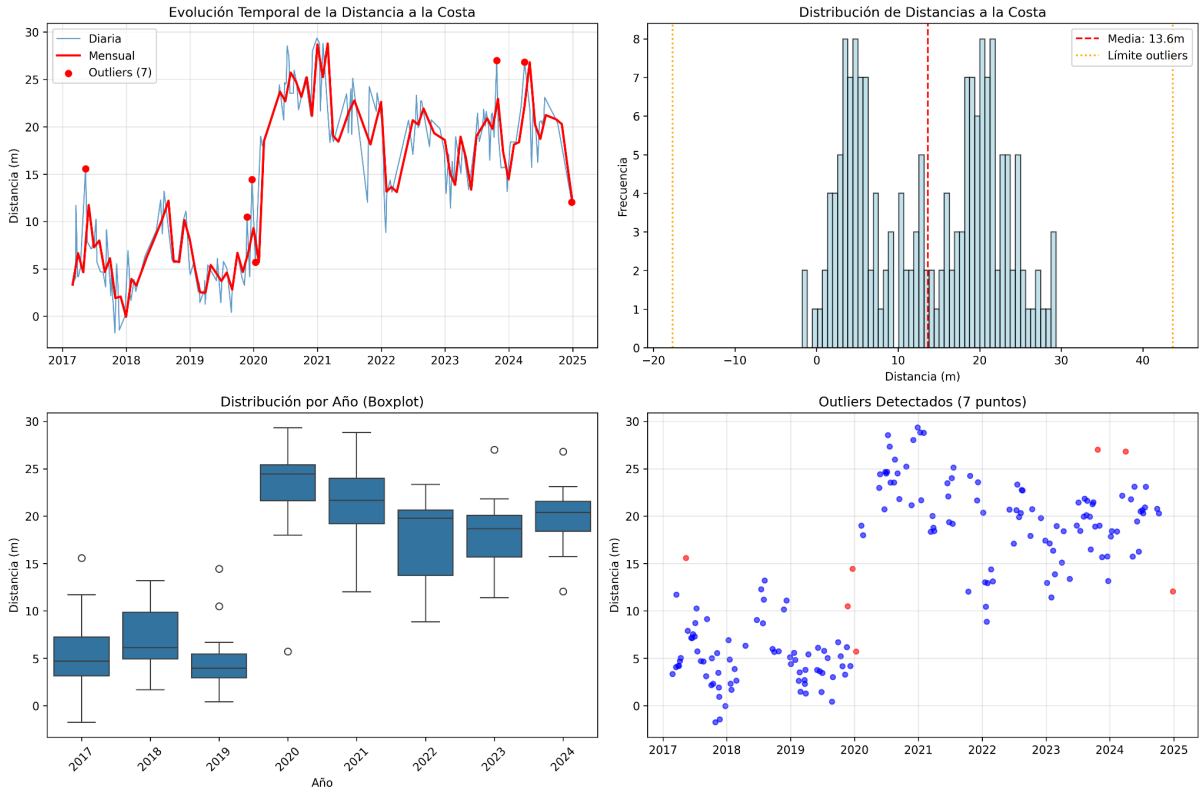
Hemos llegado a las siguientes gráficas:



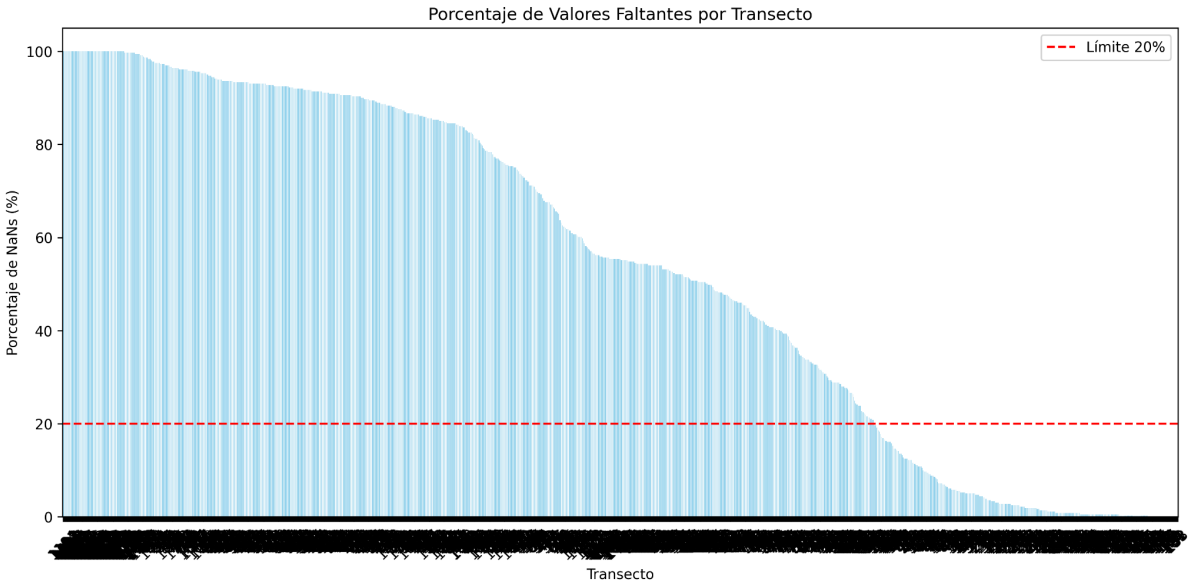
En esta gráfica podemos sacar la conclusión que si se puede observar que si existe una tendencia a largo plazo. Tiene una tendencia a crecer más o menos de forma constante. Podemos observar también que usando la media móvil cada mes que entre 2020 y 2021 se produjo un aumento bastante grande de la distancia de la costa (anchura) probablemente debido al COVID-19, puesto que supuso una mucho menor influencia del ser humano en éstas, por ello el ecosistema y la naturaleza tuvo un periodo de tiempo en el que regenerarse. Por esta razón, también se puede explicar la presencia de tantos outliers en ese momento.



Con esta gráfica podemos asegurar que si hay un patrón de estación, que sería la seasonal, es una especie de montaña o pico que ocurre durante el verano, debido a la energía de las olas. En verano, las olas son suaves, por ello las playas están suministradas por la arena que traen estas olas. Por otro lado, en invierno, las olas son más pesadas, por lo que arrastra arena al mar haciendo las playas más pequeñas.



Los outliers de 2024 se podrían explicar por diferentes razones actividad humana extrema (obras), cambios bruscos en las condiciones oceanográficas, pero yo creo que la que mejor lo explica es que hubo diferentes eventos meteorológicos extremos, es decir, principalmente las lluvias y tormentas torrenciales en la Dana.



En esta gráfica, se puede observar un patrón claro de la disminución de NANs con el tiempo, esto posiblemente se explique por la mejora de las diferentes formas de medición que filtran mejor las zonas de interés eliminando las manchas, entre otras formas de NANs.