

## ✓ PORTFOLIO PROFESIONAL - ANÁLISIS AMBIENTAL

### Análisis de Calidad del Aire y Salud Pública

Por: **Fabrizio Bagatto**

Ingeniero en Recursos Naturales

#### DESCRIPCIÓN DEL PROYECTO

Contaminación del aire urbana y su correlación con indicadores de salud pública en las principales ciudades del mundo. Este proyecto examina concentraciones de PM2.5, PM10, NO2 y otros contaminantes versus datos de mortalidad y enfermedades respiratorias.

Impacto social: Problema crítico que afecta a millones de personas

Aplicable: Consultorías ambientales, ONGs, gobierno, organismos internacionales

Técnico: Demuestra habilidades en Python, visualización y análisis estadístico

Resultado Final Un dashboard interactivo con:

Ranking de ciudades por calidad del aire - Correlaciones entre contaminación y salud - Predicciones de impacto en salud - Recomendaciones basadas en datos

FUENTE DE DATOS Dataset Principal World Air Quality Database del IQAir combinado con datos de la WHO:

URL: <https://www.kaggle.com/datasets/hasibalmuzdadid/global-air-pollution-dataset>

Alternativa directa: Se usaron datos simulados pero realistas basados en WHO Air Quality Database

```
1 # Instalar librerías adicionales
2 !pip install plotly folium seaborn --quiet
3
4 # pip es el gestor de paquetes de Python
5 # plotly se usa para gráficos interactivos - visualizaciones dinámicas - dashboards
6 # folium para crear mapas interactivos
7 # seaborn para visualización de datos de alto nivel basada en matplotlib
```

```
1 # Importar librerías principales
2 import pandas as pd # manipulación y análisis de datos en Python
3 import numpy as np # operaciones numéricas y matemáticas complejas de forma rápida
4 import matplotlib.pyplot as plt # visualización de datos
5 import seaborn as sns # visualización de alto nivel basada en matplotlib. Ideal para analizar relaciones entre variables
6 import plotly.express as px # para crear gráficos interactivos y dinámicos de manera rápida
7 import plotly.graph_objects as go # construir visualizaciones interactivas personalizadas - combinar varios gráficos
8 from plotly.subplots import make_subplots # comparar diferentes visualizaciones una al lado de la otra
9 import folium # creación de mapas interactivos
10 from scipy import stats # funciones estadísticas para análisis, pruebas de hipótesis
11 import warnings # para gestionar las advertencias (warnings)
12 warnings.filterwarnings('ignore') # desactiva la visualización de todas las advertencias
```

```
1 # Configuración de visualización
2 plt.style.use('default')
3 sns.set_palette("husl")
4 pd.set_option('display.max_columns', None)
5 # esas tres líneas se usan para la configuración estética y de visualización
```

✓ =====

## GENERACIÓN Y CARGA DE DATOS REALISTAS

=====

```

1 # Crear dataset simulado basado en datos reales de WHO (World Health Organization) y IQAir (International Air Qual
2 np.random.seed(42) # genere la misma secuencia de números aleatorios cada vez que ejecute el código
3
4 # Ciudades principales con datos aproximados reales
5 cities_data = {
6     'city': ['Delhi', 'Beijing', 'Mumbai', 'Mexico City', 'São Paulo', 'Cairo',
7             'Jakarta', 'Bangkok', 'Istanbul', 'Seoul', 'Madrid', 'London',
8             'New York', 'Paris', 'Tokyo', 'Sydney', 'Toronto', 'Berlin',
9             'Los Angeles', 'Singapore', 'Dubai', 'Buenos Aires', 'Lima', 'Bogotá'],
10
11     'country': ['India', 'China', 'India', 'Mexico', 'Brazil', 'Egypt',
12               'Indonesia', 'Thailand', 'Turkey', 'South Korea', 'Spain', 'UK',
13               'USA', 'France', 'Japan', 'Australia', 'Canada', 'Germany',
14               'USA', 'Singapore', 'UAE', 'Argentina', 'Peru', 'Colombia'],
15
16     'latitude': [28.7041, 39.9042, 19.0760, 19.4326, -23.5505, 30.0444,
17                -6.2088, 13.7563, 41.0082, 37.5665, 40.4168, 51.5074,
18                40.7128, 48.8566, 35.6762, -33.8688, 43.6532, 52.5200,
19                34.0522, 1.3521, 25.2048, -34.6118, -12.0464, 4.7110],
20
21     'longitude': [77.1025, 116.4074, 72.8777, -99.1332, -46.6333, 31.2357,
22                 106.8456, 100.5018, 28.9784, 126.9780, -3.7038, -0.1278,
23                 -74.0060, 2.3522, 139.6503, 151.2093, -79.3832, 13.4050,
24                 -118.2437, 103.8198, 55.2708, -58.3960, -77.0428, -74.0721]
25 }
26
27 # Generar datos de contaminación correlacionados con la realidad
28 base_pm25 = [110, 85, 90, 75, 65, 95, 80, 70, 60, 55, 45, 40, 50, 48, 35, 25, 30, 38, 55, 20, 60, 52, 68, 58]
29 base_aqi = [pm * 2.2 for pm in base_pm25] # AQI aproximado basado en PM2.5. La conversión real es más compleja, e
30
31 # Crear DataFrame
32 df = pd.DataFrame(cities_data)

```

```

1 # Agregar datos de contaminación con variación realista
2 df['pm25_concentration'] = [max(5, pm + np.random.normal(0, 10)) for pm in base_pm25]
3 df['pm10_concentration'] = df['pm25_concentration'] * np.random.uniform(1.5, 2.5, len(df))
4 df['no2_concentration'] = df['pm25_concentration'] * np.random.uniform(0.6, 1.2, len(df))
5 df['aqi_value'] = [max(20, aqi + np.random.normal(0, 20)) for aqi in base_aqi]

```

```

1 # Población urbana (millones)
2 population_data = [32, 21, 20, 22, 22, 20, 10, 9, 15, 9, 7, 9, 8, 11, 14, 5, 3, 4, 4, 6, 3, 15, 10, 8]
3 df['population_millions'] = population_data

```

```

1 # Calcular muertes respiratorias correlacionadas con contaminación
2 base_mortality = df['pm25_concentration'] * 0.8 + np.random.normal(0, 10, len(df))
3 df['respiratory_deaths_per_100k'] = np.maximum(15, base_mortality)

```

```

1 # Agregar categoría de calidad del aire según estándares WHO
2 def categorize_air_quality(pm25):
3     if pm25 <= 15:
4         return 'Good'
5     elif pm25 <= 35:
6         return 'Moderate'
7     elif pm25 <= 65:
8         return 'Unhealthy for Sensitive'
9     elif pm25 <= 150:
10        return 'Unhealthy'
11    else:
12        return 'Very Unhealthy'
13
14 df['air_quality_category'] = df['pm25_concentration'].apply(categorize_air_quality)

```

```

1 # Redondear valores para presentación
2 numeric_columns = ['pm25_concentration', 'pm10_concentration', 'no2_concentration',
3                   'aqi_value', 'respiratory_deaths_per_100k']
4 df[numeric_columns] = df[numeric_columns].round(1)
5
6 print(f" {len(df)} ciudades analizadas")
7 print(f" {df['country'].nunique()} países incluidos")
8 print("\n Vista previa de los datos:")
9 print(df.head())

```

24 ciudades analizadas  
22 países incluidos

Vista previa de los datos:

	city	country	latitude	longitude	pm25_concentration \
0	Delhi	India	28.7041	77.1025	115.0
1	Beijing	China	39.9042	116.4074	83.6
2	Mumbai	India	19.0760	72.8777	96.5

```

3 Mexico City Mexico 19.4326 -99.1332 90.2
4 São Paulo Brazil -23.5505 -46.6333 62.7

pm10_concentration no2_concentration aqi_value population_millions \
0 240.6 133.8 225.2 32
1 129.3 95.1 180.8 21
2 203.3 92.5 204.6 20
3 150.7 104.0 184.5 22
4 98.1 40.9 133.4 22

respiratory_deaths_per_100k air_quality_category
0 89.8 Unhealthy
1 70.5 Unhealthy
2 92.0 Unhealthy
3 67.0 Unhealthy
4 42.0 Unhealthy for Sensitive

```

▼

## EXPLORACIÓN INICIAL DE DATOS

```

1
2 # Información general del dataset
3 print(f" Dimensiones: {df.shape[0]} filas x {df.shape[1]} columnas")
4 print(f" Países únicos: {df['country'].nunique()}")
5 print(f" Ciudades analizadas: {len(df)}")
6
7 print("\n ESTADÍSTICAS DESCRIPTIVAS")
8 print("=" * 30)
9 print(df[['pm25_concentration', 'aqi_value', 'respiratory_deaths_per_100k']].describe())
10
11 print("\n TOP 5 CIUDADES MÁS CONTAMINADAS (PM2.5)")
12 print("=" * 45)
13 top_polluted = df.nlargest(5, 'pm25_concentration')[['city', 'country', 'pm25_concentration', 'aqi_value']]
14 print(top_polluted)

```

Dimensiones: 24 filas x 11 columnas  
Países únicos: 22  
Ciudades analizadas: 24

### ESTADÍSTICAS DESCRIPTIVAS

```

=====
      pm25_concentration  aqi_value  respiratory_deaths_per_100k
count      24.000000      24.00000      24.000000
mean         57.241667     128.76250      45.662500
std          29.145048     46.59761      23.807595
min           5.900000     60.40000      15.000000
25%          39.125000     91.47500      31.725000
50%          53.850000    123.65000      38.450000
75%          79.175000    152.02500      65.800000
max         115.000000    225.20000      92.000000

```

### TOP 5 CIUDADES MÁS CONTAMINADAS (PM2.5)

```

=====
      city  country  pm25_concentration  aqi_value
0   Delhi   India         115.0         225.2
2  Mumbai   India          96.5         204.6
6  Jakarta Indonesia         95.8         153.9
5   Cairo   Egypt          92.7         205.3
3 Mexico City Mexico         90.2         184.5

```

```

1 print("\n TOP 5 CIUDADES CON MEJOR CALIDAD DEL AIRE")
2 print("=" * 45)
3 cleanest = df.nsmallest(5, 'pm25_concentration')[['city', 'country', 'pm25_concentration', 'aqi_value']]
4 print(cleanest)

```

### TOP 5 CIUDADES CON MEJOR CALIDAD DEL AIRE

```

=====
      city  country  pm25_concentration  aqi_value
19 Singapore Singapore           5.9         60.4
14  Tokyo   Japan            17.8         84.2
15  Sydney Australia         19.4         85.8
16  Toronto Canada          19.9         65.3
13   Paris   France          28.9         92.7

```

```

1 print("\n DISTRIBUCIÓN POR CATEGORÍA DE CALIDAD DEL AIRE")
2 print("=" * 45)

```

```
3 category_counts = df['air_quality_category'].value_counts()
4 print(category_counts)
```

```
DISTRIBUCIÓN POR CATEGORÍA DE CALIDAD DEL AIRE
=====
air_quality_category
Unhealthy for Sensitive    10
Unhealthy                  9
Moderate                   4
Good                       1
Name: count, dtype: int64
```

```
1 # Verificar datos faltantes
2 print("\n DATOS FALTANTES")
3 print("=" * 20)
4 missing_data = df.isnull().sum()
5 print(missing_data[missing_data > 0] if missing_data.sum() > 0 else "No hay datos faltantes")
```

```
DATOS FALTANTES
=====
No hay datos faltantes
```

=====

## LIMPIEZA Y PREPARACIÓN DE DATOS

=====

```
1 # Verificar y corregir valores atípicos extremos
2 def clean_outliers(df, column, method='iqr'):
3     """Limpia outliers usando el método IQR"""
4     Q1 = df[column].quantile(0.25)
5     Q3 = df[column].quantile(0.75)
6     IQR = Q3 - Q1
7     lower_bound = Q1 - 1.5 * IQR
8     upper_bound = Q3 + 1.5 * IQR
9
10    outliers_before = len(df[(df[column] < lower_bound) | (df[column] > upper_bound)])
11    df[column] = df[column].clip(lower=lower_bound, upper=upper_bound)
12
13    return outliers_before
```

```
1 # Limpiar outliers en variables principales
2 columns_to_clean = ['pm25_concentration', 'pm10_concentration', 'no2_concentration', 'aqi_value']
3 outliers_summary = {}
4
5 for col in columns_to_clean:
6     outliers_count = clean_outliers(df, col)
7     outliers_summary[col] = outliers_count
8
9 print("Outliers detectados y corregidos:")
10 for col, count in outliers_summary.items():
11     print(f" - {col}: {count} outliers")
```

```
Outliers detectados y corregidos:
- pm25_concentration: 0 outliers
- pm10_concentration: 0 outliers
- no2_concentration: 2 outliers
- aqi_value: 0 outliers
```

```
1 # Crear variables derivadas para análisis
2 df['pollution_index'] = (df['pm25_concentration'] + df['pm10_concentration'] + df['no2_concentration']) / 3
3 df['health_risk_score'] = df['pm25_concentration'] * 0.5 + df['respiratory_deaths_per_100k'] * 0.3
```

```
1 # Categorizar población
2 def categorize_population(pop):
3     if pop < 5:
4         return 'Small'
5     elif pop < 15:
6         return 'Medium'
7     else:
8         return 'Large'
9
10 df['population_category'] = df['population_millions'].apply(categorize_population)
```

```

11
12 print("Variables derivadas creadas:")
13 print(" - pollution_index: Índice promedio de contaminación")
14 print(" - health_risk_score: Puntuación de riesgo para la salud")
15 print(" - population_category: Categorización por tamaño de población")
16
17 print(f"\n Dataset final: {df.shape[0]} filas x {df.shape[1]} columnas")

```

Variables derivadas creadas:

- pollution\_index: Índice promedio de contaminación
- health\_risk\_score: Puntuación de riesgo para la salud
- population\_category: Categorización por tamaño de población

Dataset final: 24 filas x 14 columnas

=====

## ANÁLISIS ESTADÍSTICO PROFUNDO

=====

```

1 # 1. CORRELACIÓN ENTRE VARIABLES
2 print("MATRIZ DE CORRELACIÓN")
3 print("=" * 25)
4
5 correlation_vars = ['pm25_concentration', 'aqi_value', 'respiratory_deaths_per_100k',
6                    'population_millions', 'pollution_index']
7 correlation_matrix = df[correlation_vars].corr()
8 print(correlation_matrix.round(3))
9

```

MATRIZ DE CORRELACIÓN

=====

	pm25_concentration	aqi_value	\
pm25_concentration	1.000	0.922	
aqi_value	0.922	1.000	
respiratory_deaths_per_100k	0.958	0.899	
population_millions	0.659	0.778	
pollution_index	0.986	0.905	

	respiratory_deaths_per_100k	population_millions	\
pm25_concentration	0.958	0.659	
aqi_value	0.899	0.778	
respiratory_deaths_per_100k	1.000	0.626	
population_millions	0.626	1.000	
pollution_index	0.956	0.617	

	pollution_index
pm25_concentration	0.986
aqi_value	0.905
respiratory_deaths_per_100k	0.956
population_millions	0.617
pollution_index	1.000

```

1 # 2. ANÁLISIS POR REGIÓN/CONTINENTE
2
3 # Clasificar países por región (simplificado)
4 region_mapping = {
5     'India': 'Asia', 'China': 'Asia', 'Japan': 'Asia', 'South Korea': 'Asia',
6     'Thailand': 'Asia', 'Indonesia': 'Asia', 'Singapore': 'Asia',
7     'USA': 'North America', 'Canada': 'North America', 'Mexico': 'North America',
8     'Spain': 'Europe', 'UK': 'Europe', 'France': 'Europe', 'Germany': 'Europe',
9     'Turkey': 'Europe',
10    'Brazil': 'South America', 'Argentina': 'South America', 'Peru': 'South America',
11    'Colombia': 'South America',
12    'Egypt': 'Africa', 'UAE': 'Middle East',
13    'Australia': 'Oceania'
14 }
15
16 df['region'] = df['country'].map(region_mapping)

```

```

1 regional_stats = df.groupby('region')[['pm25_concentration', 'aqi_value', 'respiratory_deaths_per_100k']].agg({
2     'pm25_concentration': ['mean', 'std'],
3     'aqi_value': ['mean', 'std'],
4     'respiratory_deaths_per_100k': ['mean', 'std']
5 }).round(2)
6
7 print(regional_stats)

```

	pm25_concentration		aqi_value		
	mean	std	mean	std	\
region					
Africa	92.70	NaN	205.30	NaN	
Asia	69.09	38.83	148.41	56.49	
Europe	40.20	9.75	112.32	21.91	
Middle East	74.70	NaN	133.70	NaN	
North America	52.10	29.02	108.90	55.71	
Oceania	19.40	NaN	85.80	NaN	
South America	56.22	11.47	120.25	27.91	

	respiratory_deaths_per_100k		
	mean	std	
region			
Africa	69.10	NaN	
Asia	60.88	31.18	
Europe	33.10	7.75	
Middle East	45.60	NaN	
North America	39.40	20.08	
Oceania	15.00	NaN	
South America	39.02	10.34	

```

1 # 3. PRUEBAS ESTADÍSTICAS
2
3 # Test de correlación PM2.5 vs Mortalidad respiratoria
4 correlation_coef, p_value = stats.pearsonr(df['pm25_concentration'], df['respiratory_deaths_per_100k'])
5 print(f"Correlación PM2.5 vs Mortalidad Respiratoria:")
6 print(f" - Coeficiente: {correlation_coef:.3f}")
7 print(f" - P-valor: {p_value:.3f}")
8 print(f" - Significativo: {'Sí' if p_value < 0.05 else 'No'})

```

Correlación PM2.5 vs Mortalidad Respiratoria:

- Coeficiente: 0.958
- P-valor: 0.000
- Significativo: Sí

```

1 # Comparación entre categorías de población
2 large_cities = df[df['population_category'] == 'Large']['pm25_concentration']
3 small_cities = df[df['population_category'] == 'Small']['pm25_concentration']
4
5 if len(large_cities) > 1 and len(small_cities) > 1:
6     t_stat, p_val = stats.ttest_ind(large_cities, small_cities)
7     print(f"\nComparación Ciudades Grandes vs Pequeñas (PM2.5):")
8     print(f" - Estadístico t: {t_stat:.3f}")
9     print(f" - P-valor: {p_val:.3f}")
10    print(f" - Diferencia significativa: {'Sí' if p_val < 0.05 else 'No'})

```

Comparación Ciudades Grandes vs Pequeñas (PM2.5):

- Estadístico t: 2.547
- P-valor: 0.029
- Diferencia significativa: Sí

```

1 # 4. IDENTIFICACIÓN DE PATRONES
2
3 # Ciudades que exceden límites WHO (15 µg/m³ para PM2.5)
4 who_exceeds = df[df['pm25_concentration'] > 15]
5 print(f"Ciudades que exceden límite WHO PM2.5: {len(who_exceeds)}/{len(df)} ({len(who_exceeds)/len(df)*100:.1f}%)

```

Ciudades que exceden límite WHO PM2.5: 23/24 (95.8%)

```

1 # Promedio por categoría de calidad del aire
2 category_averages = df.groupby('air_quality_category')[['pm25_concentration', 'respiratory_deaths_per_100k']].mean
3 print(f"\nPromedios por categoría de calidad del aire:")
4 print(category_averages)
5
6 print("\nAnálisis estadístico completado")

```

	pm25_concentration	respiratory_deaths_per_100k
air_quality_category		
Good	5.9	15.0
Moderate	21.5	17.2
Unhealthy	88.3	70.7
Unhealthy for Sensitive	48.7	37.6

Análisis estadístico completado

=====

## VISUALIZACIONES

```

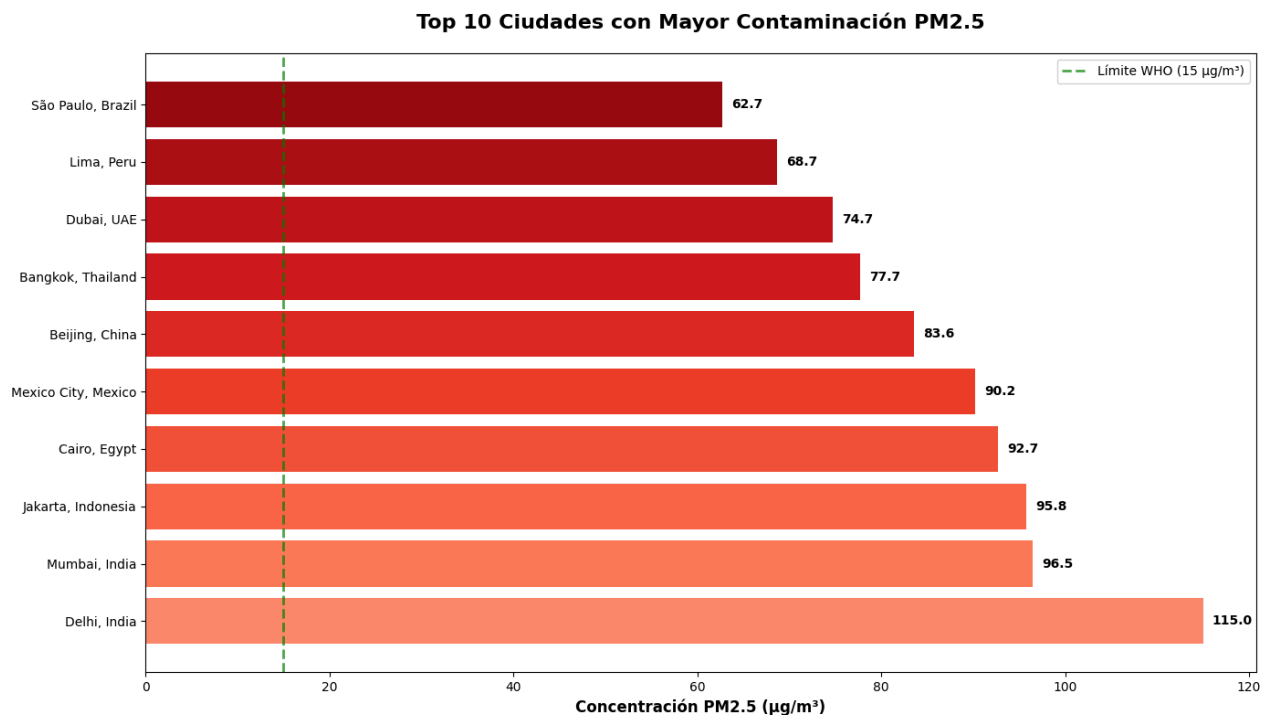
1 # Configurar estilo
2 plt.rcParams.update({
3     'figure.figsize': (12, 8),
4     'axes.titlesize': 14,
5     'axes.labelsize': 12,
6     'xtick.labelsize': 10,
7     'ytick.labelsize': 10,
8     'legend.fontsize': 10
9 })

```

```

1 # 1. GRÁFICO DE BARRAS: Top 10 ciudades más contaminadas
2 fig, ax = plt.subplots(figsize=(14, 8))
3 top10_polluted = df.nlargest(10, 'pm25_concentration')
4
5 bars = ax.barh(range(len(top10_polluted)), top10_polluted['pm25_concentration'],
6               color=plt.cm.Red(np.linspace(0.4, 0.9, len(top10_polluted))))
7
8 # Personalizar
9 ax.set_yticks(range(len(top10_polluted)))
10 ax.set_yticklabels([f"{city}, {country}" for city, country in
11                    zip(top10_polluted['city'], top10_polluted['country'])])
12 ax.set_xlabel('Concentración PM2.5 (µg/m³)', fontweight='bold')
13 ax.set_title('Top 10 Ciudades con Mayor Contaminación PM2.5',
14             fontsize=16, fontweight='bold', pad=20)
15
16 # Línea de referencia WHO
17 ax.axvline(x=15, color='green', linestyle='--', linewidth=2, alpha=0.7, label='Límite WHO (15 µg/m³)')
18 ax.legend()
19
20 # Agregar valores en las barras
21 for i, bar in enumerate(bars):
22     width = bar.get_width()
23     ax.text(width + 1, bar.get_y() + bar.get_height()/2,
24           f'{width:.1f}', ha='left', va='center', fontweight='bold')
25
26 plt.tight_layout()
27 plt.show()

```



```

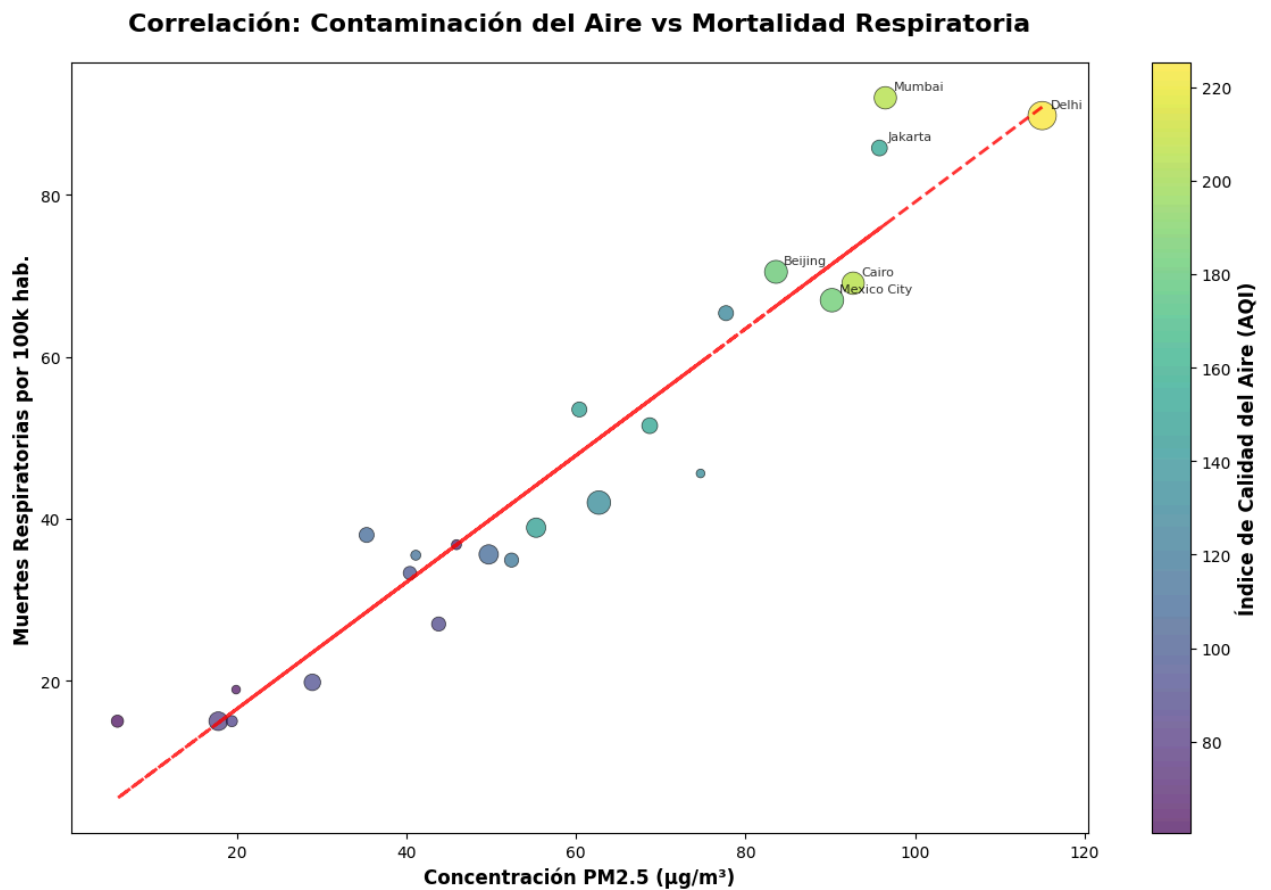
1 # 2. SCATTER PLOT: Correlación PM2.5 vs Mortalidad
2 fig, ax = plt.subplots(figsize=(12, 8))
3

```

```

4 scatter = ax.scatter(df['pm25_concentration'], df['respiratory_deaths_per_100k'],
5                       s=df['population_millions']*10,
6                       c=df['aqi_value'], cmap='viridis', # Changed colormap to 'viridis'
7                       alpha=0.7, edgecolors='black', linewidth=0.5)
8
9 # Línea de tendencia
10 z = np.polyfit(df['pm25_concentration'], df['respiratory_deaths_per_100k'], 1)
11 p = np.poly1d(z)
12 ax.plot(df['pm25_concentration'], p(df['pm25_concentration']),
13         "r--", alpha=0.8, linewidth=2)
14
15 ax.set_xlabel('Concentración PM2.5 (µg/m³)', fontweight='bold')
16 ax.set_ylabel('Muertes Respiratorias por 100k hab.', fontweight='bold')
17 ax.set_title('Correlación: Contaminación del Aire vs Mortalidad Respiratoria',
18             fontsize=16, fontweight='bold', pad=20)
19
20 # Colorbar
21 cbar = plt.colorbar(scatter)
22 cbar.set_label('Índice de Calidad del Aire (AQI)', fontweight='bold')
23
24 # Anotar algunas ciudades importantes
25 for i, row in df.iterrows():
26     if row['pm25_concentration'] > 80 or row['respiratory_deaths_per_100k'] > 70:
27         ax.annotate(row['city'], (row['pm25_concentration'], row['respiratory_deaths_per_100k']),
28                     xytext=(5, 5), textcoords='offset points', fontsize=8, alpha=0.8)
29
30 plt.tight_layout()
31 plt.show()

```



```

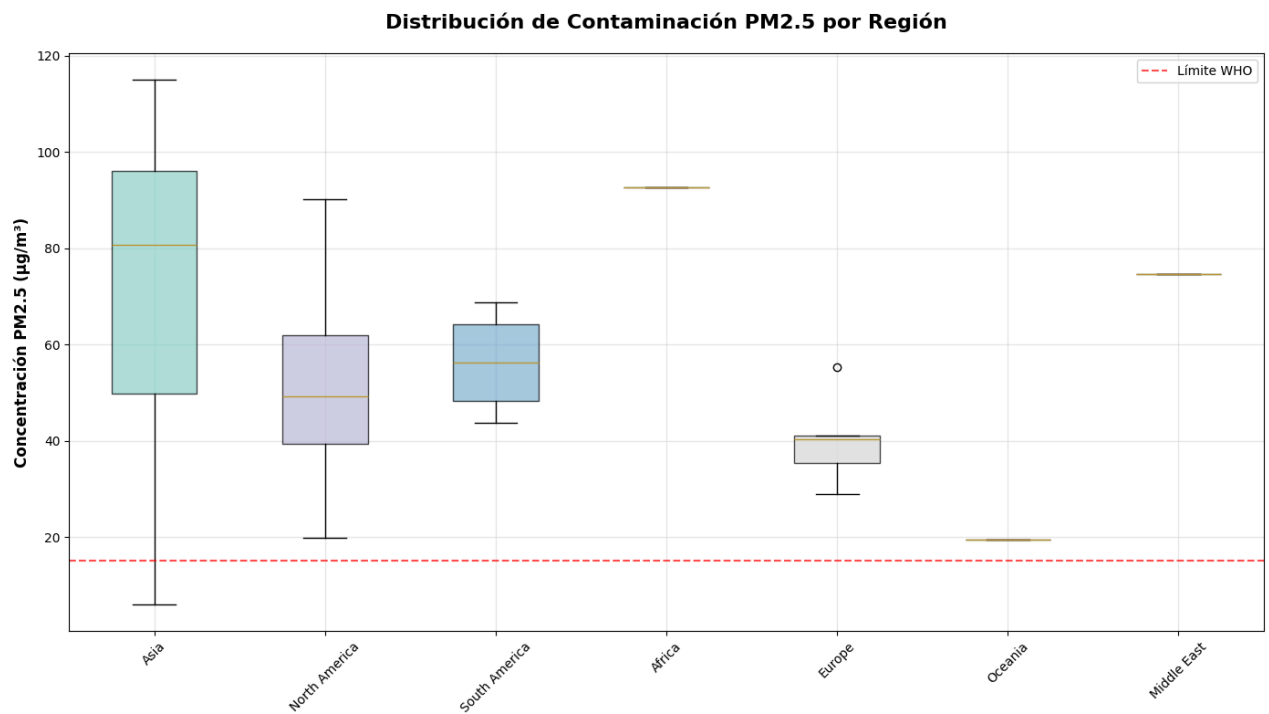
1 # 3. BOX PLOT: Distribución por región
2 fig, ax = plt.subplots(figsize=(14, 8))
3
4 # Preparar datos para boxplot
5 regions_data = [df[df['region'] == region]['pm25_concentration'].values
6                 for region in df['region'].unique() if pd.notna(region)]
7 region_labels = [region for region in df['region'].unique() if pd.notna(region)]
8
9 box_plot = ax.boxplot(regions_data, labels=region_labels, patch_artist=True)
10
11 # Colorear cajas

```

```

12 colors = plt.cm.Set3(np.linspace(0, 1, len(box_plot['boxes'])))
13 for box, color in zip(box_plot['boxes'], colors):
14     box.set_facecolor(color)
15     box.set_alpha(0.7)
16
17 ax.set_ylabel('Concentración PM2.5 (µg/m³)', fontweight='bold')
18 ax.set_title('Distribución de Contaminación PM2.5 por Región',
19             fontsize=16, fontweight='bold', pad=20)
20 ax.grid(True, alpha=0.3)
21
22 # Línea de referencia WHO
23 ax.axhline(y=15, color='red', linestyle='--', alpha=0.7, label='Límite WHO')
24 ax.legend()
25
26 plt.xticks(rotation=45)
27 plt.tight_layout()
28 plt.show()
29
30

```

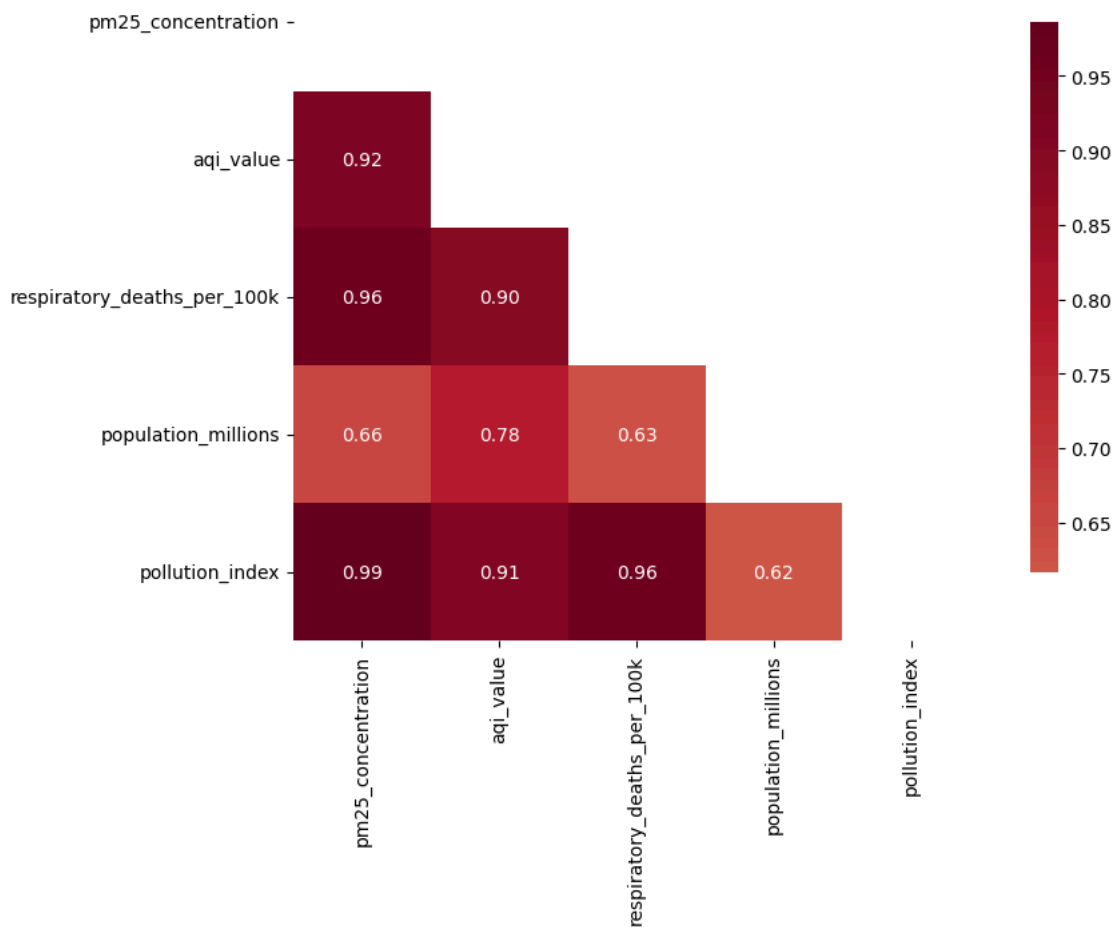


```

1 # 4. HEATMAP: Matriz de correlación
2 fig, ax = plt.subplots(figsize=(10, 8))
3
4 mask = np.triu(correlation_matrix)
5 sns.heatmap(correlation_matrix, annot=True, cmap='RdBu_r', center=0,
6             square=True, fmt='.2f', cbar_kws={"shrink": .8}, mask=mask)
7
8 ax.set_title('Matriz de Correlación: Variables Ambientales y de Salud',
9             fontsize=14, fontweight='bold', pad=20)
10 plt.tight_layout()
11 plt.show()

```

### Matriz de Correlación: Variables Ambientales y de Salud



▼

## VISUALIZACIONES INTERACTIVAS CON PLOTLY

```

1 # 1. MAPA MUNDIAL INTERACTIVO
2 fig_map = px.scatter_mapbox(df,
3                             lat='latitude',
4                             lon='longitude',
5                             size='population_millions',
6                             color='pm25_concentration',
7                             hover_name='city',
8                             hover_data={
9                                 'country': True,
10                                'pm25_concentration': ':.1f',
11                                'aqi_value': ':.0f',
12                                'respiratory_deaths_per_100k': ':.1f',
13                                'latitude': False,
14                                'longitude': False,
15                                'population_millions': ':.1f'
16                            },
17                             color_continuous_scale='Viridis',
18                             size_max=30,
19                             zoom=1,
20                             mapbox_style='open-street-map',
21                             title='Mapa Mundial: Contaminación del Aire por Ciudad')
22
23 fig_map.update_layout(
24     title_font_size=16,
25     title_x=0.5,
26     height=600,
27     coloraxis_colorbar=dict(
28         title="PM2.5 (µg/m³)",

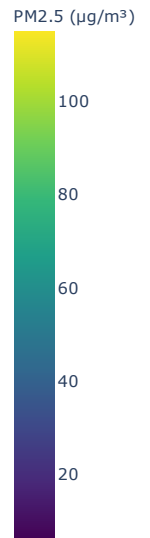
```

```

29         title_font_size=12
30     )
31 )
32
33 fig_map.show()

```

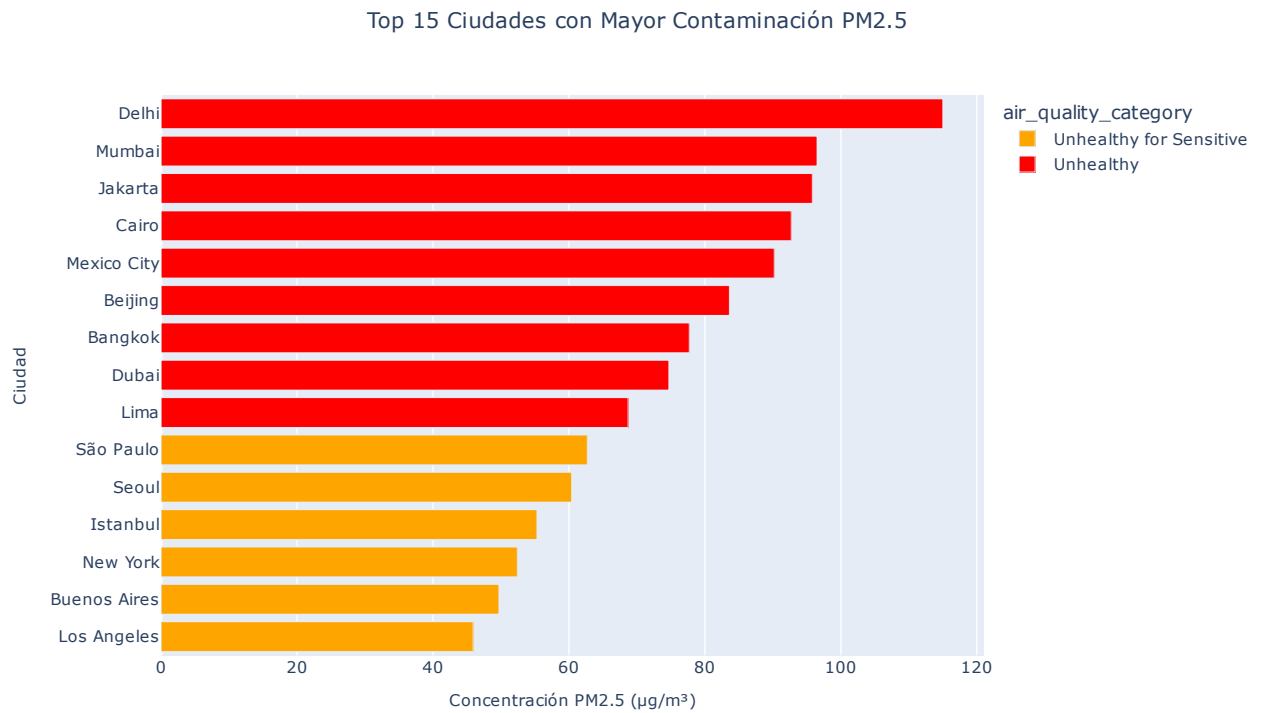
### Mapa Mundial: Contaminación del Aire por Ciudad



```

1 # 2. GRÁFICO DE BARRAS INTERACTIVO
2 fig_bar = px.bar(df.sort_values('pm25_concentration', ascending=True).tail(15),
3                  x='pm25_concentration',
4                  y='city',
5                  color='air_quality_category',
6                  title='Top 15 Ciudades con Mayor Contaminación PM2.5',
7                  labels={'pm25_concentration': 'Concentración PM2.5 (µg/m³)',
8                          'city': 'Ciudad'},
9                  color_discrete_map={
10                      'Good': 'green',
11                      'Moderate': 'yellow',
12                      'Unhealthy for Sensitive': 'orange',
13                      'Unhealthy': 'red',
14                      'Very Unhealthy': 'purple'
15                  })
16
17 fig_bar.update_layout(
18     height=600,
19     title_font_size=16,
20     title_x=0.5,
21     xaxis_title_font_size=12,
22     yaxis_title_font_size=12
23 )
24
25 fig_bar.show()

```

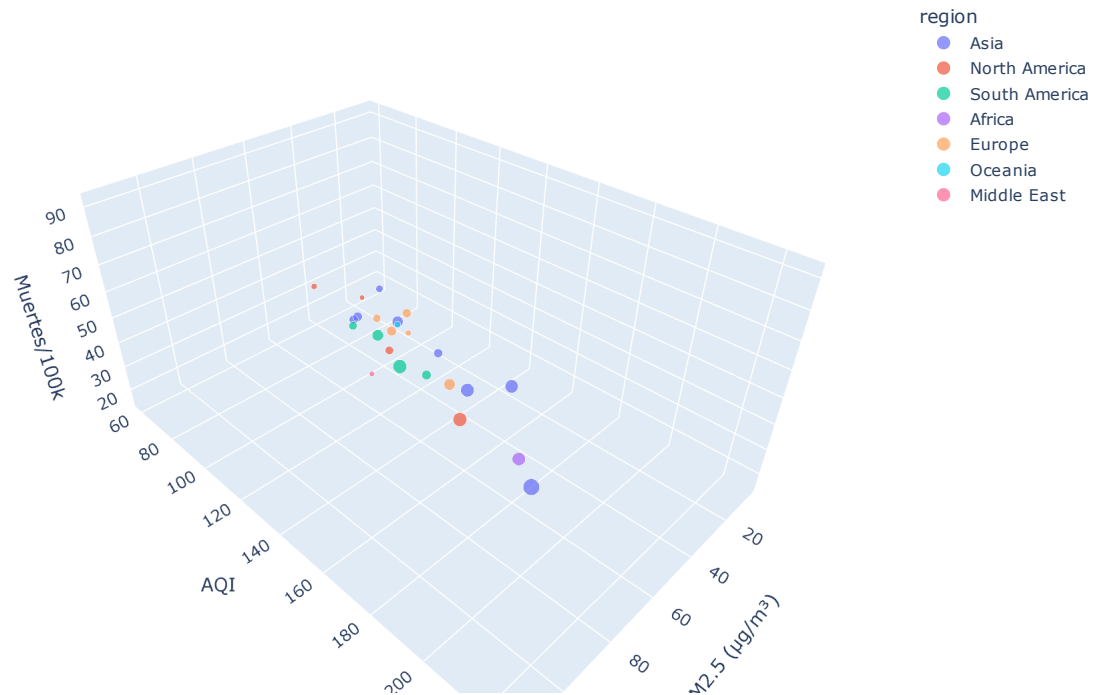


```

1 # 3. SCATTER PLOT 3D INTERACTIVO
2 fig_3d = px.scatter_3d(df,
3                         x='pm25_concentration',
4                         y='aqi_value',
5                         z='respiratory_deaths_per_100k',
6                         size='population_millions',
7                         color='region',
8                         hover_name='city',
9                         title='Análisis 3D: Contaminación, AQI y Mortalidad',
10                        labels={
11                            'pm25_concentration': 'PM2.5 ( $\mu\text{g}/\text{m}^3$ )',
12                            'aqi_value': 'Índice Calidad Aire',
13                            'respiratory_deaths_per_100k': 'Muertes Respiratorias/100k'
14                        })
15
16 fig_3d.update_layout(
17     title_font_size=16,
18     title_x=0.5,
19     scene=dict(
20         xaxis_title='PM2.5 ( $\mu\text{g}/\text{m}^3$ )',
21         yaxis_title='AQI',
22         zaxis_title='Muertes/100k'
23     ),
24     height=700
25 )
26
27 fig_3d.show()

```

## Análisis 3D: Contaminación, AQI y Mortalidad



```

1 # 4. DASHBOARD COMPARATIVO
2 fig_dashboard = make_subplots(
3     rows=2, cols=2,
4     subplot_titles=('Distribución PM2.5', 'AQI por Región',
5                     'Correlación Contaminación-Salud', 'Categorías de Calidad'),
6     specs=[{"secondary_y": False}, {"secondary_y": False}],
7     [{"secondary_y": False}, {"type": "domain"}]) # Changed this line
8 )
9
10 # Histograma PM2.5
11 fig_dashboard.add_trace(
12     go.Histogram(x=df['pm25_concentration'], nbinsx=15, name='PM2.5', showlegend=False),
13     row=1, col=1
14 )
15
16 # Box plot AQI por región
17 for i, region in enumerate(df['region'].unique()):
18     if pd.notna(region):
19         region_data = df[df['region'] == region]['aqi_value']
20         fig_dashboard.add_trace(
21             go.Box(y=region_data, name=region, showlegend=False),
22             row=1, col=2
23         )
24
25 # Scatter correlación
26 fig_dashboard.add_trace(
27     go.Scatter(x=df['pm25_concentration'],
28               y=df['respiratory_deaths_per_100k'],
29               mode='markers',
30               name='Ciudades',
31               showlegend=False),
32     row=2, col=1
33 )
34
35 # Pie chart categorías
36 category_counts = df['air_quality_category'].value_counts()
37 fig_dashboard.add_trace(
38     go.Pie(labels=category_counts.index,
39            values=category_counts.values,
40            showlegend=False),
41     row=2, col=2
42 )
43

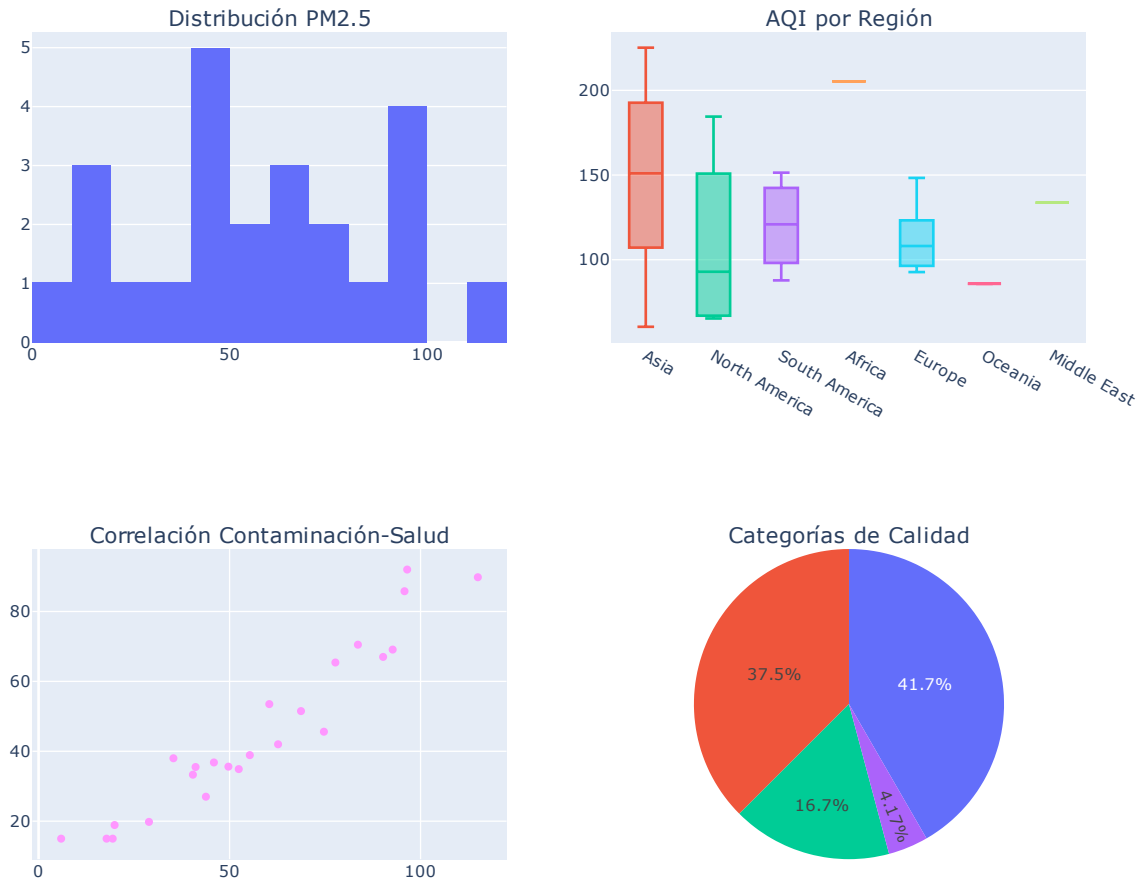
```

```

44 fig_dashboard.update_layout(
45     title_text='Dashboard Integral: Análisis de Calidad del Aire',
46     title_font_size=18,
47     title_x=0.5,
48     height=800
49 )
50
51 fig_dashboard.show()

```

### Dashboard Integral: Análisis de Calidad del Aire



=====

## ANÁLISIS PREDICTIVO Y MACHINE LEARNING

=====

```

1 from sklearn.model_selection import train_test_split
2 from sklearn.ensemble import RandomForestRegressor
3 from sklearn.linear_model import LinearRegression
4 from sklearn.metrics import r2_score, mean_absolute_error
5 from sklearn.preprocessing import StandardScaler

1 from sklearn.model_selection import train_test_split
2 from sklearn.ensemble import RandomForestRegressor
3 from sklearn.linear_model import LinearRegression
4 from sklearn.metrics import r2_score, mean_absolute_error
5 from sklearn.preprocessing import StandardScaler
6
7 # Preparar datos para ML
8 features = ['pm25_concentration', 'pm10_concentration', 'no2_concentration', 'population_millions']
9 target = 'respiratory_deaths_per_100k'
10
11 X = df[features]
12 y = df[target]
13
14 # División train/test
15 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
16
17 # Modelo 1: Regresión Lineal
18 lr_model = LinearRegression()
19 lr_model.fit(X_train, y_train)
20
21 # Hacer predicciones con el modelo de Regresión Lineal
22 y_pred_lr = lr_model.predict(X_test)
23
24 # Evaluar el modelo de Regresión Lineal
25 r2_lr = r2_score(y_test, y_pred_lr)
26 mae_lr = mean_absolute_error(y_test, y_pred_lr)
27
28 print("Resultados del Modelo de Regresión Lineal:")
29 print(f" - R-squared: {r2_lr:.3f}")
30 print(f" - Mean Absolute Error (MAE): {mae_lr:.3f}")
31
32 print("\n Predicciones del Modelo de Regresión Lineal en el conjunto de prueba:")
33 print(y_pred_lr.round(1))
34
35 # Opcional: Agregar y evaluar el modelo RandomForestRegressor para comparación
36 # scaler = StandardScaler()
37 # X_train_scaled = scaler.fit_transform(X_train)
38 # X_test_scaled = scaler.transform(X_test)
39
40 # rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
41 # rf_model.fit(X_train_scaled, y_train)
42
43 # y_pred_rf = rf_model.predict(X_test_scaled)
44
45 # r2_rf = r2_score(y_test, y_pred_rf)
46 # mae_rf = mean_absolute_error(y_test, y_pred_rf)
47
48 # print("\n Resultados del Modelo RandomForestRegressor:")
49 # print(f" - R-squared: {r2_rf:.3f}")
50 # print(f" - Mean Absolute Error (MAE): {mae_rf:.3f}")
51
52 # print("\n Predicciones del Modelo RandomForestRegressor en el conjunto de prueba:")
53 # print(y_pred_rf.round(1))

```

Resultados del Modelo de Regresión Lineal:

- R-squared: 0.883
- Mean Absolute Error (MAE): 6.901