

***Benemérita Universidad Autónoma de Puebla.***



***Materia: Introducción a la Ciencia de Datos.***

***Docente: Jaime Alejandro Romero Sierra.***

***Proyecto: Análisis Socioeconómico y Demográfico de Tractos Censales en Estados Unidos***

***Alumno: Joseph Diego Luna Velázquez.***

***Carrera: Ingeniería en ciencia de datos.***

***Repositorio en GitHub:***

[https://github.com/JVelaz13/Proyecto\\_final\\_ds\\_ot2025](https://github.com/JVelaz13/Proyecto_final_ds_ot2025)

***Fecha de entrega: 28 de noviembre de 2025.***

## **Índice.**

<i>Análisis Socioeconómico y Demográfico de Tractos Censales en Estados Unidos</i> .....	4
1. Introducción y Contexto .....	4
2. Objetivos del Proyecto .....	5
3. Descripción del Problema .....	6
4. Justificación del proyecto .....	8
4.1 Relevancia social y territorial .....	9
5. Hipótesis Iniciales .....	9
6. Descripción del Dataset .....	10
6. Metodología .....	18
<b>Proceso de Limpieza de Datos</b> .....	<b>20</b>
1. Descripción inicial de la base de datos.....	20
2. Proceso de limpieza.....	22
2.5. Limpieza de columnas numéricas .....	27
2.5.1. Limpieza de columnas categóricas (texto o tipo string).....	29
2.5.2. Conversión de columnas de texto con números a formato numérico .....	31
2.9.1. Creación de base de datos limpia .....	38
3. Conclusiones del Proceso de Limpieza y Preparación de Datos .....	38
<b>Análisis Exploratorio de Datos (EDA)</b> .....	<b>42</b>
1. Descripción General de los Datos .....	42
2. Visualización y Distribución de Variables Individuales .....	44
3. Correlación entre Variables .....	47
4. Análisis de Valores Atípicos (Outliers).....	55
4.1.1 Caracterización de Outliers Identificados .....	56
4.2. Estrategias de Tratamiento .....	56
5. Análisis de Valores Faltantes .....	57
6. Relación entre Variables Categóricas y Numéricas .....	58
7. Análisis de Composición Étnica y Desigualdades .....	60
8. Hallazgos Integrales y Patrones Estructurales .....	62
9. Implicaciones para Modelado y Análisis Futuro .....	64
<b>Modelo de Machine Learning</b> .....	<b>65</b>
1. Descripción del Modelo .....	65
2. Justificación del Modelo .....	66
3. Implementación y Entrenamiento .....	67
4. Resultados y Evaluación .....	71
5. Visualizaciones de Resultados.....	75

<i>6. Conclusión del Modelo.....</i>	79
<b>Dashboard.....</b>	<b>80</b>
<i>1. Descripción General del Dashboard.....</i>	<i>80</i>
<i>2. Capturas y Explicación del Dashboard .....</i>	<i>80</i>
<i>3. Uso y Beneficios del Dashboard. ....</i>	<i>83</i>
<b>Conclusiones y futuro. ....</b>	<b>85</b>
<i>1. Conclusiones Generales del Proyecto.....</i>	<i>85</i>
<i>2. Evaluación del Cumplimiento de los Objetivos .....</i>	<i>86</i>
<i>3. Futuras Líneas de Trabajo y Mejoras Propuestas .....</i>	<i>87</i>
<b>Bibliografía.....</b>	<b>90</b>



**BUAP**

# *Análisis Socioeconómico y Demográfico de Tractos Censales en Estados Unidos.*

## *1. Introducción y Contexto.*

El proyecto tiene como objetivo analizar de manera integral la estructura socioeconómica y demográfica de los tractos censales de los Estados Unidos. Los datos censales constituyen una herramienta esencial para el diagnóstico y la comprensión de las dinámicas sociales, económicas y territoriales de un país. Su importancia radica en que permiten desagregar la realidad nacional a un nivel local, mostrando contrastes que los promedios nacionales no reflejan. Los tractos censales son divisiones geográficas establecidas por la Oficina del Censo de los Estados Unidos (U.S. Census Bureau) con el propósito de ofrecer información estadística detallada sobre comunidades relativamente pequeñas, generalmente de entre 2,500 y 8,000 habitantes.

A través del análisis de variables como el ingreso medio, el nivel educativo, el valor promedio de las viviendas y la composición étnica, se pueden identificar patrones de desigualdad estructural, segmentación urbana y movilidad social. Estas variables reflejan no solo condiciones materiales, sino también procesos históricos de exclusión, concentración de riqueza y oportunidades desiguales entre grupos sociales y regiones.

El análisis de estos datos resulta particularmente relevante en un contexto global donde las desigualdades socioeconómicas se han acentuado en las últimas décadas. La información obtenida a nivel de trácto censal ofrece una ventana privilegiada para estudiar cómo las políticas públicas, la estructura económica y las transformaciones demográficas impactan la vida cotidiana de millones de personas.

Este reporte busca, por tanto, ofrecer una mirada analítica, rigurosa y contextualizada sobre los factores que moldean la distribución del bienestar y las oportunidades en el territorio estadounidense. A través de este trabajo se busca

sentar las bases para investigaciones futuras que contribuyan a la formulación de políticas orientadas a la equidad y la sostenibilidad social.

## ***2. Objetivos del Proyecto.***

### ***2.1. Objetivo general.***

El presente proyecto tiene como propósito general analizar, interpretar y explicar la distribución de variables socioeconómicas, educativas y demográficas a nivel de trato censal en los Estados Unidos, utilizando herramientas de análisis de datos, estadística descriptiva y enfoques interpretativos interdisciplinarios.

A través de este estudio se busca generar un panorama integral que refleje cómo las condiciones económicas, el nivel educativo y la composición étnica influyen en la estructura social del país, así como en la dinámica urbana y territorial de sus áreas metropolitanas.

Analizar de manera exploratoria, descriptiva y correlacional la estructura socioeconómica y demográfica de los tratos censales de los Estados Unidos, con el fin de identificar patrones espaciales de desigualdad, niveles diferenciales de acceso a educación y vivienda, y relaciones entre ingreso, capital educativo y composición étnica.

El proyecto busca transformar los datos censales en conocimiento útil, capaz de aportar evidencia para la comprensión de los factores estructurales que condicionan la calidad de vida y las oportunidades de desarrollo en distintas regiones del país.

El alcance de este trabajo abarca tanto la **dimensión analítica** como la **interpretativa**. Desde lo técnico, implica procesar y examinar un volumen considerable de datos para obtener una visión empírica detallada de la realidad socioeconómica estadounidense. Desde lo social, busca ofrecer una lectura crítica de esos resultados, enfatizando las condiciones estructurales que determinan la equidad, el acceso y las oportunidades.

El análisis se enfocará principalmente en identificar **relaciones y tendencias significativas** entre variables, más que en la construcción de modelos predictivos complejos, dejando abierta la posibilidad de ampliar el estudio en fases posteriores.

Así, los objetivos del proyecto no solo delimitan un plan de trabajo metodológico, sino que orientan la construcción de conocimiento en torno a la justicia social, la movilidad educativa y el desarrollo económico sostenible.

### ***3. Descripción del Problema.***

El presente proyecto parte de la necesidad de comprender la magnitud, naturaleza y distribución de las desigualdades sociales, económicas y educativas que existen en los Estados Unidos, observadas a través de la lente territorial de los tráctos censales.

A pesar del desarrollo económico, tecnológico y científico que caracteriza al país, persisten profundas brechas estructurales entre distintos grupos poblacionales y regiones. Dichas desigualdades no solo se reflejan en los ingresos o en la posesión de bienes materiales, sino también en la **distribución del conocimiento, el acceso a oportunidades y las condiciones de vida**.

Los datos censales revelan que la ubicación geográfica de una persona —el barrio o el trácto en el que vive— puede influir significativamente en su acceso a servicios, su nivel educativo, su posibilidad de movilidad social e incluso en su esperanza de vida.

Este fenómeno se conoce como **determinismo espacial de la desigualdad**, y su estudio requiere herramientas analíticas capaces de vincular la dimensión económica con la demográfica y territorial.

En este contexto, los tráctos censales funcionan como unidades de observación que permiten identificar **patrones de concentración y exclusión**. Barrios contiguos pueden presentar realidades radicalmente opuestas: zonas con altos ingresos, educación universitaria predominante y viviendas de gran valor coexisten con comunidades de bajos recursos, infraestructuras precarias y limitado acceso a educación superior.

Estas diferencias, lejos de ser casuales, son producto de **procesos históricos de segregación residencial, discriminación étnica y desigualdad educativa**, que se han perpetuado a lo largo de generaciones.

La problemática central que motiva este análisis radica en la **imposibilidad de entender la desigualdad únicamente a través de promedios nacionales o estatales**. Los datos agregados tienden a ocultar la heterogeneidad interna del territorio, haciendo que fenómenos locales de pobreza, marginación o privilegio pasen desapercibidos.

Por ello, resulta indispensable realizar un **análisis granular**, a nivel de trato censal, que permita descomponer la realidad en unidades comparables y revelar la estructura social con mayor precisión.

Desde el punto de vista económico, el ingreso mediano y el valor de la vivienda son indicadores clave que reflejan el poder adquisitivo y la estabilidad financiera de las familias. Sin embargo, su distribución está estrechamente asociada al **nivel educativo** y a la **composición étnica** de las comunidades. Diversos estudios sociológicos y urbanos han demostrado que los grupos con menor acceso histórico a la educación formal tienden a concentrarse en zonas con viviendas de menor valor y con oportunidades laborales limitadas.

En consecuencia, se genera un **círculo vicioso** en el que la falta de educación limita los ingresos, y los bajos ingresos restringen la posibilidad de acceder a mejores condiciones de vida o entornos educativos de calidad.

Por otra parte, la dimensión étnica juega un papel fundamental en la persistencia de la desigualdad. En Estados Unidos, la composición racial de un vecindario ha estado históricamente vinculada con su desarrollo económico, resultado de políticas de segregación como el **redlining**, que durante décadas restringió el acceso a créditos hipotecarios y servicios públicos a comunidades afroamericanas, latinas y de otras minorías.

Aunque dichas políticas fueron abolidas formalmente, sus efectos continúan siendo visibles en la actualidad: los tractos censales con mayor proporción de población afroamericana o hispana suelen coincidir con aquellos que presentan los ingresos

medianos más bajos, las tasas más reducidas de educación superior y los valores de vivienda más modestos.

El problema, por tanto, no se reduce a la existencia de diferencias estadísticas entre regiones, sino a la **estructura persistente de desigualdad territorial y social**, que condiciona la vida de millones de personas. El desafío consiste en desentrañar esas relaciones complejas entre **educación, ingreso, vivienda y etnicidad**, y comprender cómo interactúan para reproducir —o en algunos casos, reducir— las brechas socioeconómicas.

Este estudio busca abordar esa problemática mediante un análisis que integre datos cuantitativos con una lectura contextual e interpretativa. El objetivo no es únicamente identificar diferencias numéricas, sino entender los procesos que las generan y las consecuencias sociales que de ellas se derivan.

En última instancia, la comprensión detallada de este problema permitirá contribuir a una **planificación urbana más equitativa**, al diseño de **políticas públicas basadas en evidencia** y al fortalecimiento de estrategias de **desarrollo social sostenibles**, donde la información estadística sirva como herramienta para promover la justicia social y la cohesión comunitaria.

#### ***4. Justificación del proyecto.***

La presente investigación se justifica en la necesidad de comprender, con rigor empírico y perspectiva crítica, las desigualdades estructurales que caracterizan la realidad social y económica de los Estados Unidos. Aunque se trata de uno de los países con mayor capacidad tecnológica, educativa y productiva del mundo, la evidencia muestra que los beneficios de dicho desarrollo no se distribuyen de manera equitativa entre su población ni su territorio.

Esta desigualdad se manifiesta en múltiples dimensiones: el acceso desigual a la educación, las diferencias en el ingreso mediano, la variabilidad en el valor de la vivienda, y las marcadas divisiones raciales y étnicas que aún persisten en numerosas comunidades. Dichos factores, lejos de ser fenómenos aislados, se

entrelazan y refuerzan mutuamente, generando patrones estables de **exclusión social y concentración del bienestar**.

#### **4.1 Relevancia social y territorial.**

El análisis de los tractos censales proporciona una oportunidad única para examinar estas desigualdades desde la escala más adecuada: la del territorio donde las personas realmente viven y se relacionan.

A diferencia de las estadísticas nacionales o estatales, que tienden a homogenizar la realidad, los datos a nivel de trato censal revelan la diversidad y complejidad de las comunidades locales. En una misma ciudad pueden coexistir zonas de alta prosperidad junto a barrios donde la pobreza, la precariedad habitacional y la falta de acceso educativo son la norma.

Comprender esas diferencias no solo tiene un valor académico, sino también **práctico y ético**: la información obtenida a partir de este tipo de análisis permite visibilizar comunidades marginadas, orientar recursos públicos de forma más eficiente y evaluar el impacto territorial de las políticas públicas. De este modo, el proyecto trasciende el plano técnico para convertirse en una herramienta de diagnóstico y transformación social.

#### **5. Hipótesis Iniciales.**

1. Los tractos censales con menor ingreso medio presentan también menor nivel educativo alcanzado.
2. Las áreas metropolitanas con mayor diversidad étnica tienden a presentar mayor desigualdad económica.
3. Existe una correlación positiva entre el ingreso medio y el valor promedio de la vivienda.

4. La distribución de la población hispana y afroamericana se concentra en zonas con menores niveles de ingreso y educación.
5. Los tracts con alta concentración de población con educación superior muestran niveles de ingreso y valor de vivienda más altos.

## ***6. Descripción del Dataset.***

### ***6.1. Descripción general de "census\_tracts.csv"***

El conjunto de datos analizado contiene **8,281 registros** y **17 variables**, cada uno correspondiente a un tracto censal de Estados Unidos. El dataset contiene información **socioeconómica, educativa y demográfica** de los **tractos censales de 5 áreas metropolitanas de interés en Estados Unidos (Atlanta, Baltimore, Nueva York, Oakland y Washington DC)**, unidades estadísticas definidas por la Oficina del Censo (U.S. Census Bureau). Cada registro corresponde a un **tracto censal**, una subdivisión geográfica dentro de condados y áreas metropolitanas que agrupa, por lo general, entre **2,500 y 8,000 habitantes**.

Este tipo de datasets se utiliza ampliamente en investigaciones sobre **desigualdad económica, desarrollo urbano, movilidad social, educación y planificación pública**, ya que permite observar las condiciones de vida con un alto nivel de detalle territorial.

#### **Características técnicas generales:**

- **Número de registros:** 8,281 tracts censales.
- **Número de columnas (variables):** 17.
- **Tipo de datos:** cuantitativos (numéricos continuos) y categóricos (textuales).
- **Cobertura geográfica:** Estados Unidos (múltiples ciudades y áreas metropolitanas).

- **Valores nulos:** Ninguno; el dataset está completo y limpio.
- **Formato de archivo:** CSV (valores separados por comas).

### Propósito del dataset:

El dataset permite caracterizar la **estructura social y económica de las comunidades estadounidenses** a partir de variables como:

- Población total y estructura por edad.
- Ingreso y valor de vivienda.
- Nivel educativo de los residentes.
- Distribución racial y étnica.
- Identificación geográfica (ciudad y área metropolitana).

A través del cruce de estas variables, es posible analizar temas como:

- Desigualdad territorial en el ingreso y la educación.
- Relación entre composición étnica y nivel socioeconómico.
- Impacto del valor de vivienda sobre las condiciones demográficas.
- Concentración urbana y patrones de segregación residencial.

El conjunto de datos está listo para **análisis exploratorio, estadístico, geográfico y predictivo**, siendo una base muy completa y adecuada para modelos de regresión, análisis espacial (GIS) o aprendizaje automático enfocado en fenómenos socioeconómicos.

### Resumen interpretativo:

En conjunto, este dataset ofrece una visión **multidimensional de la estructura social de los Estados Unidos**, combinando indicadores económicos, educativos y étnicos con información territorial.

Su fortaleza radica en el nivel de detalle: cada fila representa una comunidad local

que, al analizarse en conjunto, permite observar los patrones de desigualdad, segregación y movilidad social a escala nacional.

Estas variables pueden analizarse de manera independiente o combinada para estudiar fenómenos como:

- La **correlación entre educación e ingresos**.
- La **relación entre diversidad étnica y desigualdad urbana**.
- Las **diferencias regionales** en el valor de la vivienda.
- La **distribución espacial del capital humano**.

## ***6.2. Fuentes de datos.***

El presente análisis se basa en información proveniente de diversas fuentes oficiales y académicas que, en conjunto, proporcionan una visión robusta y actualizada de la estructura demográfica y socioeconómica de los Estados Unidos.

Las principales fuentes utilizadas son las siguientes:

### ***1. U.S. Census Bureau.***

El **U.S. Census Bureau** es la fuente primaria y oficial de información demográfica y económica en los Estados Unidos. Esta institución realiza el Censo Decenal, en el cual se recopilan datos sobre población, ingresos, vivienda, educación y composición étnica. Su relevancia radica en que proporciona una cobertura total del país y mantiene una metodología estandarizada a lo largo del tiempo, lo que permite realizar comparaciones históricas y espaciales confiables. El Censo es fundamental para este tipo de estudios porque ofrece datos oficiales y de alta resolución geográfica, asegurando la precisión estadística necesaria para el análisis a nivel de trato censal.

### ***2. CensusReporter.org.***

**Census Reporter** es una plataforma independiente que traduce la información del Censo a un formato más accesible para periodistas, investigadores y público general. Este sitio actúa como intermediario entre los datos brutos del U.S. Census Bureau y los usuarios, ofreciendo una interfaz organizada y unificada que simplifica la exploración y descarga de datos censales. Su importancia radica en que facilita el trabajo de análisis y permite combinar múltiples indicadores demográficos, educativos y económicos de manera eficiente, preservando la integridad de los datos originales.

### **3. Longitudinal Tract Data Base (LTDB) – Logan et al.**

La **Longitudinal Tract Data Base (LTDB)**, desarrollada por John R. Logan y colaboradores en la Universidad de Brown, tiene como propósito armonizar los cambios en las fronteras de los tructos censales que ocurren cada década. Cada vez que se realiza un nuevo Censo, las divisiones geográficas pueden cambiar debido a variaciones en la densidad poblacional: algunos tructos se subdividen, otros se fusionan o se redibujan. Esto genera dificultades para comparar series temporales.

El LTDB resuelve ese problema mediante una metodología que reinterpreta los límites censales para hacer **comparable la información entre décadas**, particularmente entre los Censos de los años 2000 y 2010, y las estimaciones de la **American Community Survey (ACS)** de 2013–2017. Gracias a esta base, se logra una continuidad histórica que permite estudiar **la evolución de la composición demográfica y socioeconómica a nivel local**. Su inclusión garantiza que los análisis realizados no se vean distorsionados por los cambios en delimitaciones geográficas.

### **4. Kaggle.**

Finalmente, la información procesada y combinada de las fuentes anteriores fue recopilada y publicada en **Kaggle**, una plataforma global de ciencia de datos. Kaggle actúa como repositorio de datasets verificados y organizados por la

comunidad investigadora, facilitando el acceso, descarga y análisis reproducible de la información.

Su relevancia reside en que promueve la transparencia, la colaboración y la replicabilidad científica, permitiendo a investigadores y analistas de todo el mundo reutilizar los datos bajo un marco común.

El dataset de Kaggle empleado en este estudio integra datos del U.S. Census Bureau, Census Reporter y la LTDB, consolidándolos en un solo archivo estructurado y limpio que facilita el trabajo analítico y reduce los tiempos de preparación de datos.

### ***Importancia global de las fuentes***

Estas fuentes combinadas aportan una base sólida para la investigación socioeconómica y demográfica, ya que integran:

- **Rigor estadístico y cobertura nacional** (Census Bureau).
- **Accesibilidad y organización temática** (CensusReporter.org).
- **Consistencia histórica y comparabilidad temporal** (LTDB).
- **Disponibilidad pública y reproducibilidad científica** (Kaggle).

La combinación de estos orígenes garantiza un **equilibrio entre precisión técnica, accesibilidad y continuidad histórica**, lo que hace posible analizar de manera fiable la evolución de las condiciones sociales y económicas de las comunidades estadounidenses a lo largo del tiempo.

### ***6.3. ¿Cuántos datos y de qué tipo son?***

El conjunto de datos empleado en este análisis, titulado "Census Tracts", está conformado por un total de **8,281 registros** (observaciones) y **17 variables** (columnas), correspondientes a información demográfica, económica y educativa de los tractos censales de los Estados Unidos.

```
[4]: df.shape
[4]: ✓ 0.0s
... (8281, 17)

[5]: df.info()
[5]: ✓ 0.0s
...
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8281 entries, 0 to 8280
Data columns (total 17 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   geoid            8281 non-null    int64  
 1   name             8281 non-null    object  
 2   total_population 8281 non-null    int64  
 3   total_population_25_over 8281 non-null    float64 
 4   median_income    8281 non-null    float64 
 5   median_home_value 8281 non-null    float64 
 6   educational_attainment 8281 non-null    float64 
 7   white_alone      8281 non-null    float64 
 8   black_alone      8281 non-null    float64 
 9   native_alone     8281 non-null    float64 
 10  asian_alone      8281 non-null    float64 
 11  native_hawaiian_pacific_islander 8281 non-null    float64 
 12  some_other_race_alone 8281 non-null    float64 
 13  two_or_more      8281 non-null    float64 
 14  hispanic_or_latino 8281 non-null    float64 
 15  city             8281 non-null    object  
 16  metro_area       8281 non-null    object  
dtypes: float64(12), int64(2), object(3)
memory usage: 1.1+ MB
```

Cada registro representa un **tracto censal individual**, es decir, una subdivisión geográfica utilizada por el **U.S. Census Bureau** para agrupar comunidades con características socioeconómicas relativamente homogéneas. Estos trácticos suelen contener entre **2,500 y 8,000 habitantes**, lo que permite capturar con alta resolución la diversidad interna de las áreas metropolitanas y rurales del país.

### **1. Cantidad total de datos:**

- **Filas (registros):** 8,281
- **Columnas (variables):** 17
- **Celdas totales:** aproximadamente **140,777 datos individuales** ( $8,281 \times 17$ ).
- **Cobertura:** Nacional (diversas ciudades y áreas metropolitanas de Estados Unidos).

- **Ausencia de valores nulos:** el dataset se encuentra completamente limpio; ninguna variable presenta datos faltantes.

## ***2. Tipos de datos:***

El conjunto combina **dos tipos principales de datos: cuantitativos y categóricos (cualitativos)**, organizados según su naturaleza estadística:

### ***a) Datos cuantitativos (numéricos)***

Estos representan valores medibles que permiten realizar operaciones matemáticas, análisis estadísticos y correlaciones. Incluyen 14 de las 17 columnas totales.

Ejemplos de variables cuantitativas:

- *total\_population*
- *total\_population\_25\_over*
- *median\_income*
- *median\_home\_value*
- *educational\_attainment*
- *white\_alone, black\_alone, asian\_alone, native\_alone, hispanic\_or\_latino*, entre otras.

**Tipo:**

- Escala **de razón** (valores con cero absoluto, permiten comparaciones proporcionales).
- Representan **frecuencias absolutas** o **valores monetarios**.
- Admisibles para cálculos de promedios, desviaciones, correlaciones y regresiones.

### ***b) Datos categóricos (cualitativos)***

Son variables descriptivas o de texto que identifican entidades, categorías o nombres geográficos.

Incluyen 3 columnas:

- **city** (nombre de la ciudad).
- **metro\_area** (nombre del área metropolitana).
- **name** (nombre completo del trácto censal).

**Tipo:**

- Escala **nominal** (no existe orden jerárquico entre categorías).
- Se utilizan principalmente para **agrupaciones, filtros o segmentaciones** dentro del análisis.

### **3. Naturaleza y finalidad de los datos:**

- **Demográficos:** tamaño poblacional, estructura por edad y composición étnica.
- **Socioeconómicos:** ingreso mediano, valor de vivienda, capital educativo.
- **Geográficos:** ciudad y área metropolitana asociadas a cada trácto.

Estos datos son de tipo **observacional y agregado**, es decir, representan promedios o totales de población a nivel territorial, sin incluir información individual o confidencial.

### **4. Formato y estructura técnica:**

- **Formato de archivo:** CSV.
- **Estructura:** rectangular, organizada por filas (tráctos) y columnas (variables).
- **Fuente original:** U.S. Census Bureau, CensusReporter.org, Longitudinal Tract Data Base (LTDB) y Kaggle.

## **5. Interpretación general:**

En conjunto, el dataset constituye una base de información **multidimensional** que combina variables económicas, demográficas y educativas, permitiendo análisis tanto **descriptivos** (distribuciones, promedios) como **analíticos** (correlaciones, comparaciones interregionales).

Su nivel de detalle lo convierte en una herramienta valiosa para la investigación aplicada en temas como **desigualdad socioeconómica**, **planeación urbana**, **movilidad educativa y diversidad étnica**, ofreciendo una fotografía cuantitativa precisa de la realidad social estadounidense a nivel local.

## **6. Metodología.**

El análisis se fundamenta en un enfoque **cuantitativo y exploratorio** mediante el uso de técnicas de **Análisis Exploratorio de Datos (EDA)**. Este enfoque busca examinar las relaciones entre variables antes de la formulación de modelos predictivos o inferenciales.

Las etapas metodológicas comprenden:

1. **Inspección y limpieza de datos**, garantizando la coherencia y completitud de los valores.
2. **Análisis descriptivo**, mediante medidas de tendencia central, dispersión y proporciones.
3. **Identificación de correlaciones** entre variables económicas, educativas y demográficas.
4. **Agrupamiento conceptual** por áreas metropolitanas para evaluar diferencias regionales.
5. **Reflexión interpretativa**, donde los hallazgos numéricos se vinculan con su significado social.

El procesamiento se realizará con herramientas de software libre como **Python**, **Pandas**, **NumPy** y **Matplotlib**, ampliamente utilizadas en ciencia de datos por su potencia y reproducibilidad. La metodología no se limitará al cálculo estadístico, sino que se orienta a la comprensión contextual de los resultados, vinculando la evidencia empírica con interpretaciones sociales y teóricas.



**BUAP**

# ***Proceso de Limpieza de Datos.***

## ***1. Descripción inicial de la base de datos***

### ***1.1 Fuente y contexto.***

La base de datos fue obtenida de Kaggle y contiene datos de diversas fuentes como:

1. U.S. Census Bureau: El U.S. Census Bureau es la fuente primaria y oficial de información demográfica y económica en los Estados Unidos. Esta institución realiza el Censo Decenal, en el cual se recopilan datos sobre población, ingresos, vivienda, educación y composición étnica.
2. CensusReporter.org: Census Reporter es una plataforma independiente que traduce la información del censo a un formato más accesible para periodistas, investigadores y público general.
3. Longitudinal Tract Data Base (LTDB) – Logan et al: La Longitudinal Tract Data Base (LTDB), desarrollada por John R. Logan y colaboradores en la Universidad de Brown, tiene como propósito armonizar los cambios en las fronteras de los tructos censales que ocurren cada década.

Los datos censales constituyen una herramienta esencial para el diagnóstico y la comprensión de las dinámicas sociales, económicas y territoriales de un país. Su importancia radica en que permiten desagregar la realidad nacional a un nivel local, mostrando contrastes que los promedios nacionales no reflejan. Los tructos censales son divisiones geográficas establecidas por la Oficina del Censo de los Estados Unidos (U.S. Census Bureau) con el propósito de ofrecer información estadística detallada sobre comunidades relativamente pequeñas, generalmente de entre 2,500 y 8,000 habitantes. A través del análisis de variables como el ingreso medio, el nivel educativo, el valor promedio de las viviendas y la composición étnica, se pueden identificar patrones de desigualdad estructural, segmentación urbana y

movilidad social. Estas variables reflejan no solo condiciones materiales, sino también procesos históricos de exclusión, concentración de riqueza y oportunidades desiguales entre grupos sociales y regiones. El análisis de estos datos resulta particularmente relevante en un contexto global donde las desigualdades socioeconómicas se han acentuado en las últimas décadas. La información obtenida a nivel de trácto censal ofrece una ventana privilegiada para estudiar cómo las políticas públicas, la estructura económica y las transformaciones demográficas impactan la vida cotidiana de millones de personas.

## ***1.2 Descripción general del contenido.***

El conjunto de datos analizado contiene 8,281 registros y 17 variables, cada uno correspondiente a un trácto censal de Estados Unidos. El dataset contiene información socioeconómica, educativa y demográfica de los tráctos censales de 5 áreas metropolitanas de interés en Estados Unidos (Atlanta, Baltimore, Nueva York, Oakland y Washington DC), estas son unidades estadísticas definidas por la Oficina del Censo (U.S. Census Bureau). Cada registro corresponde a un trácto censal, una subdivisión geográfica dentro de condados y áreas metropolitanas que agrupa, por lo general, entre 2,500 y 8,000 habitantes. Este tipo de datasets se utiliza ampliamente en investigaciones sobre

desigualdad económica, desarrollo urbano, movilidad social, educación y planificación pública, ya que permite observar las condiciones de vida con un alto nivel de detalle territorial.

### ***1.2.1 Características técnicas generales:***

- Número de registros: 8,281 tráctos censales.
- Número de columnas (variables): 17.
- Tipo de datos: cuantitativos (numéricos continuos) y categóricos (textuales).

- Cobertura geográfica: Estados Unidos (múltiples ciudades y áreas metropolitanas).
- Valores nulos: Ninguno; el dataset está completo y limpio.
- Formato de archivo: CSV (valores separados por comas).

### ***1.2.2 Descripción de cada columna***

Campo	Tipo	Descripción
<code>geoid</code>	Numérico entero	Identificador único asignado por la Oficina del Censo a cada trato.
<code>name</code>	Texto (cadena)	Nombre completo del trato censal, incluyendo número y ubicación.
<code>total_population</code>	Numérico entero	Población total del trato censal (todas las edades).
<code>total_population_25_over</code>	Numérico entero o decimal	Población de 25 años o más, usada para estimar nivel educativo.
<code>median_income</code>	Numérico decimal	Ingreso mediano del hogar en dólares anuales.
<code>median_home_value</code>	Numérico decimal	Valor mediano de las viviendas ocupadas.
<code>educational_attainment</code>	Numérico entero	Personas con estudios superiores (licenciatura o posgrado).
<code>white_alone</code>	Numérico entero	Personas que se identifican como blancas exclusivamente.
<code>black_alone</code>	Numérico entero	Personas que se identifican exclusivamente como afroamericanas o negras.
<code>native_alone</code>	Numérico entero	Personas que se identifican como indígenas nativos de América.
<code>asian_alone</code>	Numérico entero	Personas que se identifican exclusivamente como asiáticas.
<code>native_hawaiian_pacific_islander</code>	Numérico entero	Personas nativas de Hawái o de otras islas del Pacífico.
<code>some_other_race_alone</code>	Numérico entero	Personas que se identifican con otra raza distinta a las anteriores.
<code>two_or_more</code>	Numérico entero	Personas que se identifican con dos o más razas.
<code>hispanic_or_latino</code>	Numérico entero	Personas que se identifican como hispanas o latinas, sin importar su raza.
<code>city</code>	Texto (cadena)	Ciudad donde se ubica el trato censal.
<code>metro_area</code>	Texto (cadena)	Área metropolitana a la que pertenece el trato censal.

## ***2. Proceso de limpieza.***

El proceso de limpieza se desarrolló en el archivo `codigo_de_limpieza.ipynb`, completamente documentado con celdas Markdown explicativas y fragmentos de código Python comentado.

A continuación, se describe el flujo seguido, paso a paso, conforme a las instrucciones del documento original.

### ***2.1. Importación de librerías y carga de la base de datos.***

El primer paso del proyecto consistió en importar las librerías necesarias para el análisis y la manipulación de los datos. En este caso, se utilizó principalmente **pandas**, dado que ofrece una estructura de datos flexible (DataFrame) que facilita la limpieza, transformación y exploración de información tabular.

Posteriormente, se cargó la base de datos original en un DataFrame denominado df.

Tras la carga inicial, se realizó una revisión básica del contenido para confirmar la correcta importación y tener una primera impresión de su estructura:

También se observaron valores nulos y columnas con tipos de datos inconsistentes (por ejemplo, números representados como texto).

The screenshot shows a Jupyter Notebook cell with the following content:

```
#Se importa la librería pandas y el archivo csv que contiene la base de datos
import pandas as pd
df = pd.read_csv("c:/Users/velj0/Downloads/census_socio.csv")
df.head()
✓ 0s
```

Python

		geoid	name	total population	total population 25 over	median income	median home value	educational attainment	white alone	black alone	native alone	asian alone	native hawaiian pacific islander	son
0	1.100101e+10	Census Tract 75.03, District of Columbia, Dist...		2454.0	1425.0	26250.0	NaN	308.0	122.0	2278.0	0.0	0.0	0.0	0.0
1	NaN	Census Tract 76.01, District of Columbia, Dist...		4855.0	3463.0	34840.0	255000.0	727.0	311.0	4292.0	0.0	0.0	13.0	0.0
2	1.100101e+10	Census Tract 77.09, District of Columbia, Dist...		2524.0	1817.0	33750.0	250000.0	344.0	20.0	2280.0	0.0	0.0	0.0	0.0
3	1.100101e+10	Census Tract 95.08, District of Columbia, Dist...		3691.0	2838.0	56404.0	356600.0	1008.0	211.0	2688.0	68.0	71.0	0.0	0.0
4	1.100101e+10	Census Tract 99.04, District of Columbia, Dist...		2979.0	1526.0	30728.0	298600.0	252.0	52.0	2375.0	NaN	0.0	15.0	0.0

### *2.1.1. Exploración y diagnóstico inicial de la base de datos.*

El objetivo de esta fase fue **comprender la estructura y la calidad de los datos antes de modificarlos**, identificando posibles problemas como valores ausentes, duplicados, columnas mal tipadas o errores de formato.

A partir de esta exploración se identificaron las siguientes situaciones:

- Varias columnas numéricas, como median\_income y median\_home\_value, aparecían como object (texto).
- Otras columnas, como city o metro\_area, contenían valores nulos.

- Había presencia de duplicados totales y parciales.
- Algunos campos presentaban símbolos o letras donde deberían existir solo números.

Toda esta información fue documentada en el notebook para planificar una limpieza

```
# Primer chequeo
df.info() #Características de los datos del dtataframe
print(f"\n---Datos nulos: {df.isnull().sum()}") #Identificar total de datos nulos
print(f"\n---Datos duplicados: {df.duplicated().sum()}") #Identificar el total de columnas duplicadas
✓ 0.0s

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10589 entries, 0 to 10588
Data columns (total 17 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   geoid            10272 non-null   float64
 1   name             10272 non-null   object  
 2   total_population 10272 non-null   float64
 3   total_population_25_over 10272 non-null   float64
 4   median_income    10272 non-null   object  
 5   median_home_value 10272 non-null   object  
 6   educational_attainment 10272 non-null   object  
 7   white_alone      10272 non-null   object  
 8   black_alone      10272 non-null   float64
 9   native_alone     10272 non-null   float64
 10  asian_alone      10272 non-null   float64
 11  native_hawaiian_pacific_islander 10272 non-null   object  
 12  some_other_race_alone 10272 non-null   float64
 13  two_or_more      10272 non-null   object  
 14  hispanic_or_latino 10272 non-null   float64
 15  city             10272 non-null   object  
 16  metro_area       10272 non-null   object  
dtypes: float64(8), object(9)
memory usage: 1.4+ MB

---Datos nulos:
geoid            317
name             317
total_population 317
total_population_25_over 317

---Datos duplicados: 663
```

que **recuperara información** sin eliminarla, evitando el uso de dropna() como exige el proyecto.

## 2.2. Creación de una copia de seguridad.

Para asegurar la integridad de la base original, se creó un segundo DataFrame (df2) que sirvió como espacio de trabajo.

Este paso se realizó con la siguiente instrucción:

```
df2=df.copy()
```

*“Crear una copia del DataFrame original permite realizar todas las modificaciones sin riesgo de alterar los datos fuente. En caso de error, se puede volver a la versión original.”*

Esta práctica es una buena costumbre en análisis de datos, pues facilita el control de versiones y el seguimiento de los cambios.

### **2.3. Inspección individual de columnas.**

El siguiente paso fue **examinar las características de cada columna** del DataFrame, tanto en su estructura como en su contenido.

Para ello se usaron funciones como (en este caso la primera columna analizada fue "geoid"):

Cada variable fue inspeccionada de forma independiente para:

```
#Identificación de los datos
df2["geoid"].info() #características de los datos
print(f"\n---Valores nulos: {df2['geoid'].isnull().sum()}") #datos nulos
print(f"\n---Valores únicos: {df2['geoid'].unique()}") #valores únicos
print(f"\n---Duplicados: {df2.duplicated(subset=["geoid"]).sum()}") #Contar duplicados

<class 'pandas.core.series.Series'>
RangeIndex: 10589 entries, 0 to 10588
Series name: geoid
Non-Null Count Dtype
-----
10272 non-null float64
dtypes: float64(1)
memory usage: 82.9 KB

---Valores nulos: 317

---Valores únicos: [1.10010075e+10      nan 1.10010077e+10 ... 6.07501570e+09
3.40130049e+10 2.40338015e+10]

---Duplicados: 2481
```

- Identificar valores faltantes.
- Contar valores únicos.
- Verificar el tipo de dato.
- Detectar duplicados parciales o inconsistencias de formato.

*“Analizar las columnas una por una permite aplicar estrategias de limpieza personalizadas, según el tipo de información que contengan.”*

## Ejemplos:

```
# Identificación de los datos
df2["name"].info() #características de los datos
print(f"\n---Valores nulos: {df2["name"].isnull().sum()}") #datos nulos
print(f"\n---Valores únicos: {df2["name"].unique()}") #valores únicos
print(f"\n---Duplicados: {df2.duplicated(subset=["name"]).sum()}") #Contar duplicados
✓ 0.0s

<class 'pandas.core.series.Series'>
Index: 8108 entries, 0 to 10571
Series name: name
Non-Null Count Dtype
-----
7863 non-null    object
dtypes: object(1)
memory usage: 126.7+ KB

---Valores nulos: 245

---Valores únicos: ['Census Tract 75.03, District of Columbia, District of Columbia'
'Census Tract 76.01, District of Columbia, District of Columbia'
'Census Tract 77.09, District of Columbia, District of Columbia' ...
'Census Tract 157, San Francisco County, California'
'Census Tract 49, Essex County, New Jersey'
'Census Tract 8015, Prince George's County, Maryland']

---Duplicados: 244
```

```
# Identificación de los datos
df2["median_home_value"].info() #características de los datos
print(f"\n---Valores nulos: {df2["median_home_value"].isnull().sum()}") #datos nulos
print(f"\n---Valores únicos: {df2["median_home_value"].unique()}") #valores únicos
print(f"\n---Duplicados: {df2.duplicated(subset=["median_home_value"]).sum()}") #Contar duplicados
print(f"\n---Caracteres extraños: {df2["median_home_value"].str.contains(r'\D', regex=True)}") #Detectar caracteres extraños
```

```
<class 'pandas.core.series.Series'>
Index: 8108 entries, 0 to 10571
Series name: median_home_value
Non-Null Count Dtype
-----
7873 non-null    object
dtypes: object(1)
memory usage: 126.7+ KB

---Valores nulos: 235

---Valores únicos: [nan '255000.0' '258000.0' ... '306000.0' '1098300.0' '202300.0']

---Duplicados: 3307

---Caracteres extraños: 0      NaN
1     True
2     True
3     True
4     True
...
10344  True
10408  True
10457  True
10480  True
10571  True
Name: median_home_value, Length: 8108, dtype: object
```

Este procedimiento se repitió iterativamente para las **18 columnas del dataset**.

## 2.4. Detección y tratamiento de duplicados.

Se buscaron registros repetidos en la columna clave (geoid) para evitar redundancias.

df2.duplicated().sum()

- Los **duplicados totales** (todas las columnas idénticas) se eliminaron:
- df2.drop\_duplicates(inplace=True)

- Los **duplicados parciales** se conservaron, pues podían representar observaciones distintas en una misma zona o categoría.

*“El control de duplicados garantiza que cada fila represente un registro único y confiable.”*

### Contar duplicados en columna clave (id\_geografico).

Se procede a la identificación y eliminación de duplicados en la columna clave.

```
duplicados = df2.duplicated(subset='geoid', keep=False)
print(f"Duplicados totales encontrados: {duplicados.sum()}")
✓ 0.0s
Duplicados totales encontrados: 4375
```

Código de eliminación de duplicados.

```
#Eliminar duplicados y conservar únicamente la primera aparición de cada columna
df2 = df2.drop_duplicates(subset='geoid', keep='first')
✓ 0.0s

#Verificar la correcta eliminación
duplicados = df2.duplicated(subset='geoid', keep=False)
print(f"Duplicados encontrados: {duplicados.sum()}")
✓ 0.0s
Duplicados encontrados: 0
```

### Comprobación.

```
print(f"\n---Valores nulos: {df2['geoid'].isnull().sum()}")
✓ 0.0s
```

---Valores nulos: 1

Una vez hecho lo anterior, se ejecuta una comprobación para verificar que se han ejecutado los procesos correctamente.

```
# Comprobación del proceso
df2['geoid'].info() #Características de los datos
print(f"\n---Valores nulos: {df2['geoid'].isnull().sum()}") #Datos nulos
print(f"\n---Valores únicos: {df2['geoid'].unique()}") #Valores únicos
print(f"\n---Duplicados: {df2.duplicated(subset=['geoid']).sum()}") #Contar duplicados
✓ 0.0s

<class 'pandas.core.series.Series'>
Index: 8108 entries, 0 to 10571
Series name: geoid
Non-Null Count Dtype
-----
8108 non-null int64
dtypes: int64(1)
memory usage: 126.7 KB

---Valores nulos: 0

---Valores únicos: [11001007503 27727369410 11001007709 ... 6075015700 34013004900
24033801500]
---Duplicados: 0
```

## 2.5. Limpieza de columnas numéricas.

Las columnas con valores numéricos presentaban dos tipos de problemas principales:

1. **Presencia de valores nulos (NaN).**
2. **Formato incorrecto (texto o decimal cuando debía ser entero).**

Para corregir estos casos, se aplicaron los siguientes pasos:

**1. Imputación de valores faltantes:**

Se reemplazaron los NaN por la media de la columna, manteniendo la coherencia estadística.

**2. Conversión de tipo:**

Algunas columnas, como geoid y total\_population, se convirtieron a enteros (int) ya que los decimales eran innecesarios.

**3. Comprobación posterior:**

Se verificó que las columnas ya no contuvieran nulos y que su tipo fuera correcto.

Ejemplos:

```
# Limpieza
df2["total_population"] = df2["total_population"].fillna(df["total_population"].mean()) #Rellenar NaN con la media
df2 ["total_population"]=df2 ["total_population"].astype(int) #Convertir a entero

# Comprobación de la limpieza
df2["total_population"].info() #características de los datos
print(f"\n---Valores nulos: {df2["total_population"].isnull().sum()}") #datos nulos
print(f"\n---Valores unicos: {df2["total_population"].unique()}") #valores unicos
print(f"\n---Duplicados: {df2.duplicated(subset=["total_population"]).sum()}") #Contar duplicados

<class 'pandas.core.series.Series'>
Index: 8108 entries, 0 to 10571
Series name: total_population
Non-Null Count Dtype
-----
8108 non-null    int64
dtypes: int64(1)
memory usage: 126.7 KB

---Valores nulos: 0

---Valores unicos: [2454 4855 2524 ... 3035 7910 9377]

---Duplicados: 3351
```

```

# Limpieza
df2["median_income"] = df2["median_income"].str.replace(r'\D', '0', regex=True) # Sustituir caracteres extraños
df2 ["median_income"] = df2 ["median_income"].astype(float) # Trabsformar a númerico decimal
df2["median_income"] = df2["median_income"].fillna(df2["median_income"].mean()) # Rellenar NaN con la media
df2 ["median_income"] = df2 ["median_income"].astype(int) # Transformar a númerico entero

# Verificación de la limpieza
df2["median_income"].info() #caracteristicas de los datos
print(f"\n---Valores nulos: {df2['median_income'].isnull().sum()}" ) #datos nulos
print(f"\n---Valores unicos: {df2['median_income'].unique()}" ) #valores unicos
print(f"\n---Duplicados: {df2.duplicated(subset=['median_income']).sum()}" ) #Contar duplicados

<class 'pandas.core.series.Series'>
Index: 8108 entries, 0 to 10571
Series name: median_income
Non-Null Count Dtype
-----
8108 non-null int64
dtypes: int64(1)
memory usage: 126.7 KB

---Valores nulos: 0

---Valores unicos: [ 2625000  3484000  3375000 ... 11118100  3317000  7766700]

---Duplicados: 1256

```

### **2.5.1. Limpieza de columnas categóricas (texto o tipo string).**

Para las variables de texto (por ejemplo, city, metro\_area, name), no era apropiado usar

Por ello se optó por métodos alternativos:

1. **Rellenar valores faltantes** con el dato de la fila inferior (bfill) o superior (ffill).
2. **Eliminación de caracteres especiales:** df2["city"] = df2["city"].where(df2["city"] != "Auto%").ffill().

## Ejemplos:

```

# Limpieza
df2["name"] = df2["name"].bfill() # rellenar NaN con los datos de la fila inferior
✓ 0.0s

# IComprobación de la limpieza
df2["name"].info() #características de los datos
print("Valores nulos: {df2["name"].isnull().sum()}") #datos nulos
print("Valores únicos: {df2["name"].unique()}") #valores únicos
print("Duplicados: {df2.duplicated(subset=["name"]).sum()}") #Contar duplicados
✓ 0.0s

<class 'pandas.core.series.Series'>
Index: 8108 entries, 0 to 10571
Series name: name
Non-Null Count Dtype
-----
8108 non-null    object
dtypes: object(1)
memory usage: 126.7+ KB

---Valores nulos: 0

---Valores únicos: ['Census Tract 75.03, District of Columbia, District of Columbia'
'Census Tract 76.01, District of Columbia, District of Columbia'
'Census Tract 77.09, District of Columbia, District of Columbia' ...
'Census Tract 157, San Francisco County, California'
'Census Tract 49, Essex County, New Jersey'
"Census Tract 8015, Prince George's County, Maryland"]

---Duplicados: 245

# Limpieza
df2["city"] = df2["city"].bfill() # rellenar nan con los datos de la fila inferior
df2["city"] = df2["city"].where(df2["city"] != 'Auto%').bfill() #reemplazar Auto% con los datos de la fila inferior
✓ 0.0s

# Verificación de la limpieza
df2["city"].info() #características de los datos
print("Valores nulos: {df2["city"].isnull().sum()}") #datos nulos
print("Valores únicos: {df2["city"].unique()}") #valores únicos
print("Duplicados: {df2.duplicated(subset=["city"]).sum()}") #Contar duplicados
✓ 0.0s

<class 'pandas.core.series.Series'>
Index: 8108 entries, 0 to 10571
Series name: city
Non-Null Count Dtype
-----
8108 non-null    object
dtypes: object(1)
memory usage: 126.7+ KB

---Valores nulos: 0

---Valores únicos: ['Washington' 'Baltimore' 'Atlanta' 'Oakland' 'New York City']

---Duplicados: 8103

```

Se verifica si existe alguna inconsistencia con los nombres de las ciudades y de ser así se corrige.

```

import tqdm
# Detectar ciudades con múltiples áreas metropolitanas
ciudad_to_metros = df2.groupby('ciudad')['area_metropolitana'].nunique().reset_index(name='n_metros')
conflict_cities = ciudad_to_metros[ciudad_to_metros['n_metros'] > 1]['ciudad'].tolist()

print(f"\nSe encontraron {len(conflict_cities)} ciudades con inconsistencias")

if len(conflict_cities) > 0:
    print("Ciudades problemáticas:", conflict_cities[:10])

    # Crear mapa de moda (valor más frecuente) para cada ciudad
    mode_map = df2.groupby('ciudad')['area_metropolitana'].agg(
        lambda x: x.mode().iloc[0] if not x.mode().empty else x.iloc[0]
    ).to_dict()

    # Aplicar corrección directamente en df2
    print("Aplicando correcciones en df2...")

    from tqdm import tqdm
    for ciudad in tqdm(conflict_cities, desc="Corrigiendo ciudades"):
        mask = df2['ciudad'] == ciudad
        df2.loc[mask, 'area_metropolitana'] = mode_map[ciudad]

    # Verificar corrección
    ciudad_to_metros_after = df2.groupby('ciudad')['area_metropolitana'].nunique().reset_index(name='n_metros')
    remaining_conflicts = ciudad_to_metros_after[ciudad_to_metros_after['n_metros'] > 1]['ciudad'].tolist()

    print(f"Inconsistencias corregidas: {len(conflict_cities) - len(remaining_conflicts)}")
    print(f"Inconsistencias restantes: {len(remaining_conflicts)}")

else:
    print("No se encontraron inconsistencias entre ciudades y áreas metropolitanas")

print("\nProceso completado. df2 listo para usar con shape: {df2.shape}")

```

Se encontraron 4 ciudades con inconsistencias  
 Ciudades problemáticas: ['Atlanta', 'Baltimore', 'Oakland', 'Washington']  
 Aplicando correcciones en df2...  
 Corrigiendo ciudades: 100% [██████████] | 4/4 [00:00<00:00, 1205.00it/s]  
 Inconsistencias corregidas: 4  
 Inconsistencias restantes: 0  
 Proceso completado. df2 listo para usar con shape: (8108, 17)

## **2.5.2. Conversión de columnas de texto con números a formato numérico.**

Varias columnas originalmente contenían valores numéricos representados como texto, como:

- median\_income
- median\_home\_value
- educational\_attainment
- native\_hawaiian\_pacific\_islander
- two\_or\_more

El proceso aplicado fue el siguiente:

**1. Detección de caracteres no numéricos:**

2. df2[“median\_income”].str.contains(r”\D”, regex=True)

**3. Eliminación o reemplazo:**

4. df2[“median\_income”] = df2[“median\_income”].str.replace(r”\D”, “0”,  
regex=True)

**5. Conversión a tipo numérico:**

6. df2[“median\_income”] = df2[“median\_income”].astype(float)

Este procedimiento aseguró que las columnas pudieran ser analizadas con operaciones matemáticas o estadísticas.

## Ejemplos:

```
# Limpieza
df2["native_hawaiian_pacific_islander"] = df2["native_hawaiian_pacific_islander"].str.replace(r'\D', '0', regex=True) # Sustituir caracteres extraños
df2 ["native_hawaiian_pacific_islander"] = df2 ["native_hawaiian_pacific_islander"].astype(float) # Trabsformar a númerico decimal
df2["native_hawaiian_pacific_islander"] = df2["native_hawaiian_pacific_islander"].fillna(df2["native_hawaiian_pacific_islander"].mean()) # Rellenar NaN con la media
df2.debug(Cntr+Shift+D) hawaiian_pacific_islander] = df2 ["native_hawaiian_pacific_islander"].astype(int) # Transformar a númerico entero
✓ 0.0s

# Verificación de la limpieza
df2["native_hawaiian_pacific_islander"].info() #características de los datos
print(f"\n---Valores nulos: {df2['native_hawaiian_pacific_islander'].isnull().sum()}") #datos nulos
print(f"\n---Valores unicos: {df2['native_hawaiian_pacific_islander'].unique()}") #valores unicos
print(f"\n---Duplicados: {df2.duplicated(subset=['native_hawaiian_pacific_islander']).sum()}") #Contar duplicados
✓ 0.0s

<class 'pandas.core.series.Series'>
Index: 8108 entries, 0 to 10571
Series name: native_hawaiian_pacific_islander
Non-Null Count Dtype
-----
8108 non-null   int64
dtypes: int64(1)
memory usage: 126.7 KB

---Valores nulos: 0

---Valores unicos: [    0   1300  1500  4200  1600  507  1800   700  2700  1400   800  3400
13100   600  2100  500  1200  8900  900  1000  3300  400  100  300
1100  13200  1900  2500  4100  1700  2600  2400  2300  3800  2200  3200
4400  2900  13000  4800  3900  7400  3600  6800  5300  200  3000  5000
2000  6200  5100  18000  2800  6600  12400  5200  22000  4500  16000  22700
8200  7000  8300  4600  1100  13400  16400  8600  31300  8400  11300  3500
12800  9500  15600  12500  25200  5600  8500  14200  4300  18300  7800  4900
7900  5500  12900  4000  20400  22000  16500  12600  21500  15900  5700  15700
27300  29000  12200  11500  4700  7700  15300  14100  13700  16700  7600  16400
6000  9900  5200  15500  11300  7200  17700  24400  27400  6500  25400  13500
19000  25100  12000  7100  14000  28100  16000  23500  46400  14400  37000  18600
14300  11900  11800  26100  32400  6100  7500  11100  26600  46500  37100  46000
21000  29300  28300  66900  83200  5400  50000  28600  66800  5800  81200  19500
53300  34200  19100  8800  12700  11700  17100  17300  9100  9600  108000  7300]

# Limpieza
df2["median_home_value"] = df2["median_home_value"].str.replace(r'\D', '0', regex=True) # Sustituir caracteres extraños
df2 ["median_home_value"] = df2 ["median_home_value"].astype(float) # Trabsformar a númerico decimal
df2["median_home_value"] = df2["median_home_value"].fillna(df2["median_home_value"].mean()) # Rellenar NaN con la media
df2 ["median_home_value"] = df2 ["median_home_value"].astype(int) # Transformar a númerico entero
✓ 0.0s

# Verificación de la limpieza
df2["median_home_value"].info() #características de los datos
print(f"\n---Valores nulos: {df2['median_home_value'].isnull().sum()}") #datos nulos
print(f"\n---Valores unicos: {df2['median_home_value'].unique()}") #valores unicos
print(f"\n---Duplicados: {df2.duplicated(subset=['median_home_value']).sum()}") #Contar duplicados
✓ 0.0s

<class 'pandas.core.series.Series'>
Index: 8108 entries, 0 to 10571
Series name: median_home_value
Non-Null Count Dtype
-----
8108 non-null   int64
dtypes: int64(1)
memory usage: 126.7 KB

---Valores nulos: 0

---Valores unicos: [2838168525  25500000  25000000 ...  30600000  109830000  20230000]

---Duplicados: 3307
```

## 2.6. Traducción y cambio de nombres de columnas.

Con el fin de hacer la base más comprensible en español, se creó un **diccionario de traducción** que reemplazó los nombres de las columnas originales por sus equivalentes en español.

*“El cambio de nombres facilita la interpretación del dataset y evita confusiones durante la comunicación de resultados.”*

También se tradujeron algunos valores internos de columnas categóricas.

Traducciones:

### Diccionarios de traducción.

Ahora se procede a la traducción del inglés al español para lograr una mejor interpretación de la base de datos.

```
# Traducción de columnas
traduccion_columnas = {
    "geoid": "id_geográfico",
    "name": "nombre",
    "city": "ciudad",
    "metro_area": "área_metropolitana",
    "total_population": "población_total",
    "total_population_25_over": "población_mayor_de_25",
    "median_income": "ingreso_medio",
    "median_home_value": "valor_medio_de_vivienda",
    "white_alone": "población_blanca",
    "black_alone": "población_afroamericana",
    "asian_alone": "población_asiática",
    "native_alone": "población_nativa",
    "hispanic_or_latino": "población_hispana_o_latina",
    "educational_attainment": "educación",
    "native_hawaiian_pacific_islander": "nativo_hawaiano",
    "some_other_race_alone": "alguna_otra_etnia",
    "two_or_more": "dos_o_mas_etnias",
}
df2 = df2.rename(columns=traduccion_columnas)
```

```
#Traducir "Census Tract" → "Tracto Censal"
df2["nombre"] = df2["nombre"].str.replace(r'^Census Tract', "nombre", regex=True)

#Reordenar para que sea: Tracto Censal <número>, Condado de <nombre>, <estado>
df2["nombre"] = df2["nombre"].str.replace(
    r'^Tracto Censal\s+([\d.]+),\s*([w\s-]+?) County,\s*(.+)$',
    r'Tracto Censal \1, Condado de \2, \3',
    regex=True
)

#Traducir nombres de estados comunes
traduccion_estados = {
    'Washington': 'Washington',
    'District of Columbia': 'Distrito de Columbia',
    'Maryland': 'Maryland',
    'Virginia': 'Virginia',
    'West Virginia': 'Virginia Occidental',
    'Georgia': 'Georgia',
    'California': 'California',
    'New York': 'Nueva York',
    'New Jersey': 'Nueva Jersey',
    'Connecticut': 'Connecticut',
}

for en, es in traduccion_estados.items():
    df2['nombre'] = df2['nombre'].str.replace(en, es, regex=False)
```

```
# Traducir ciudades
traduccion_ciudades = {
    'Washington': "Washington",
    'Baltimore': 'Baltimore',
    'Atlanta': 'Atlanta',
    'Oakland': 'Oakland',
    'New York City': "Nueva York"
}

df2['ciudad'] = df2['ciudad'].replace(traduccion_ciudades)
✓ 0s

# Comprobación de traducción
df2['ciudad'].unique()
✓ 0s
array(['Washington', 'Baltimore', 'Atlanta', 'Oakland', 'Nueva York'],
      dtype=object)

#Comprobación de traducción
df2['área_metropolitana'].unique()
✓ 0s
array(['Washington-Arlington-Alexandria', 'Baltimore-Columbia-Towson',
       'Atlanta-Sandy Springs-Alpharetta',
       'San Francisco-Oakland-Berkeley', 'New York-Newark-Jersey City'],
      dtype=object)
```

## 2.7. Identificación y corrección de valores atípicos (outliers).

Se analizaron los valores extremos utilizando el método **IQR (Interquartile Range)**, para identificar datos fuera del rango estadísticamente aceptable.

```

#identificar outliers

# Inicializar máscara
outlier_mask = pd.DataFrame(False, index=df2.index, columns=df2.columns)

# Aplicar IQR por columna numérica
for col in df2.select_dtypes(include='number'):
    Q1 = df2[col].quantile(0.25)
    Q3 = df2[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outlier_mask[col] = (df2[col] < lower_bound) | (df2[col] > upper_bound)

# Filtrar filas con al menos un outlier
df_outliers = df2[outlier_mask.any(axis=1)]

```

```

#descartar outliers

df_corr = df2.copy()

# Iterar sobre columnas numéricas
for col in df_corr.select_dtypes(include='number'):
    Q1 = df_corr[col].quantile(0.25)
    Q3 = df_corr[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Detectar outliers

    outliers = (df_corr[col] < lower_bound) | (df_corr[col] > upper_bound)

#Redondear la mediana para mantener datos en INT
mediana = round(df_corr[col].median())
df_corr.loc[outliers, col] = mediana

    # Reemplazar outliers por la mediana
df_corr.loc[outliers, col] = df_corr[col].median()

```

```

# Verificación de datos que se eliminaran.
registros_erroneos = df2[(df2['ingreso_medio'] <= 0) | (df2['valor_medio_de_vivienda'] <= 0)]
print(f'Registros a eliminar: {len(registros_erroneos)} de {len(df2)}')

```

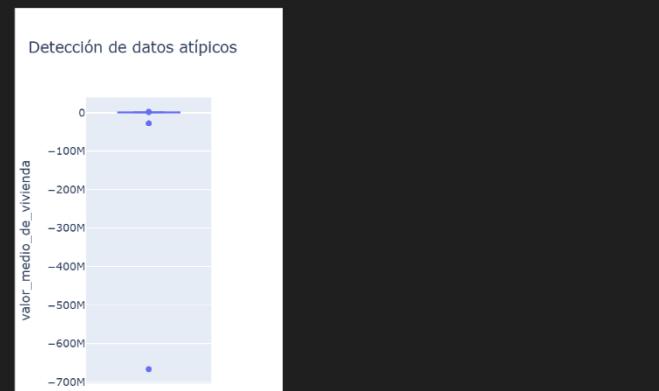
Registros a eliminar: 1092 de 8108

Analisis y verificación para ambas columnas usando boxplot.

```

import plotly.express as px
fig = px.box(df2, y='ingreso_medio', title='Detección de datos atípicos')
fig.update_layout(width=300,height=500)
fig.show()

```



En lugar de eliminar registros, los valores extremos se **ajustaron** al límite superior permitido, evitando distorsionar la media.

Se pueden identificar valores anómalos en "ingreso\_medio" y "valor\_medio\_de\_vivienda" que son conceptualmente imposibles y representan errores de medición o codificación. Distorsionan completamente las estadísticas (medias negativas) e invalidan cualquier análisis futuro. Su eliminación es necesaria en esta etapa para preservar la integridad de los datos.

**BUAP**

```

fig = px.box(df2, y='ingreso_medio', title='Detección de datos atípicos')
fig.update_layout(width=300,height=500)
fig.show()

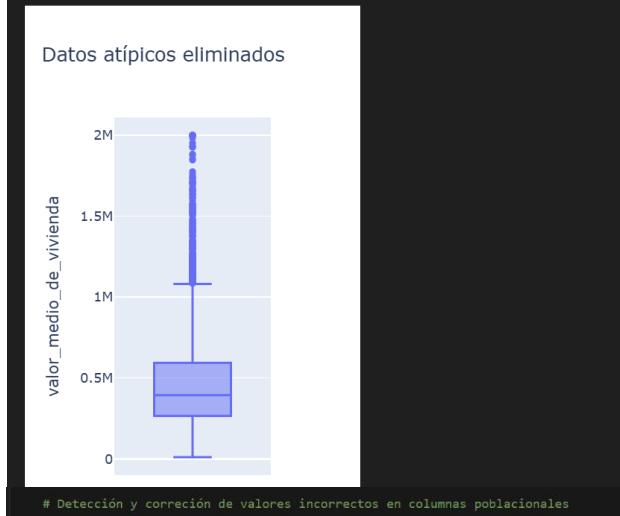
```



```

# Calcular IQR
q1 = df2['valor_medio_de_vivienda'].quantile(0.25)
q3 = df2['valor_medio_de_vivienda'].quantile(0.75)
iqr = q3 - q1
# Filtros IQR
filtro_inferior = df2['valor_medio_de_vivienda'] > q1 - (iqr * 1.5)
#filtro_superior = df['valor_medio_de_vivienda'] < q3 + (iqr * 1.5)
# Filtro adicional: valores menores o iguales a 2,000,000
# Aplicar filtros
df_filtrado1 = df2[filtro_inferior]
# Graficar
fig = px.box(df_filtrado1, y='valor_medio_de_vivienda',
              title='Datos atípicos eliminados')
fig.update_layout(width=300, height=500)
fig.show()

```



```

# Detección y corrección de valores incorrectos en columnas poblacionales

columnas_etnicas = [
    "poblacion_blanca",
    "poblacion_afroamericana",
    "poblacion_nativa",
    "poblacion_asiatica",
    "nativo_hawaiano",
    "alguna_otra_etnia",
    "dos_o_mas_etnias",
    "poblacion_hispana_o_latina"]

def comparar_con_total(df_filtrado2, columnas, col_total="poblacion_total"):
    for col in columnas:
        if col == col_total:
            continue # evitamos comparar la columna consigo misma
        problema = df_filtrado2[df_filtrado2[col] > df_filtrado2[col_total]]
        print(f"\nRegistros con {col} > {col_total}: {len(problema)}")

comparar_con_total(df_filtrado2, columnas_etnicas)

condicion = df_filtrado2["educacion"] > df_filtrado2["poblacion_mayor_de_25"]

# Contar cuántos casos hay
conteo = condicion.sum()
print(f"\nRegistros con educacion > poblacion_mayor_de_25: {conteo}")

Registros con poblacion_blanca > poblacion_total: 52
Registros con poblacion_afroamericana > poblacion_total: 6
Registros con poblacion_nativa > poblacion_total: 2
Registros con poblacion_asiatica > poblacion_total: 1
Registros con nativo_hawaiano > poblacion_total: 0
Registros con alguna_otra_etnia > poblacion_total: 0
Registros con dos_o_mas_etnias > poblacion_total: 0
Registros con poblacion_hispana_o_latina > poblacion_total: 2

Registros con educacion > poblacion_mayor_de_25: 35

```

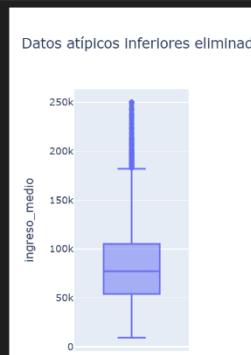
```

#Se define el cuartil 75 y se le resta el cuartil 25
iqr = df_filtrado1['ingreso_medio'].quantile(0.75) - df_filtrado1['ingreso_medio'].quantile(0.25)
#Desarrollamos filtros superior e inferior por lo general es el iqr por 1.5 pero se puede manejar el número
filtro_inferior = df_filtrado1['ingreso_medio'] > df_filtrado1['ingreso_medio'].quantile(0.25) - (iqr * 1.5)

df_filtrado2 = df_filtrado1[filtro_inferior]

#Graficando el boxplot
fig = px.box(df_filtrado2, y='ingreso_medio', title='Datos atípicos inferiores eliminados')
fig.update_layout(width=300,height=500)
fig.show()

```



Ya que se ha detectado un grave error, pues las poblaciones no pueden superar el 100% de la población total ni los registros de educación pueden superar el número de personas mayores de 25 años, por lo tanto, se debe corregir este error. Dada la existencia de este problema se recurrir a la detección y tratamiento de los outliers. Esto para variables demográficas.

Cabe señalar que se aplicó corrección proporcional para mantener la estructura relativa de los datos étnicos, eliminando inconsistencias donde la suma de subpoblaciones excedía el total reportado, garantizando coherencia metodológica en los porcentajes demográficos sin introducir sesgos externos.

```

print(comparar_con_total(df_all_corr, columnas_etnicas))

condicion = df_all_corr["educacion"] > df_all_corr["poblacion_mayor_de_25"]
conteo = condicion.sum()
print(f"\nRegistros con educacion > poblacion_mayor_de_25: {conteo}")

Registros con poblacion_blanca > poblacion_total: 0
Registros con poblacion_afroamericana > poblacion_total: 0
Registros con poblacion_nativa > poblacion_total: 0
Registros con poblacion_asiatica > poblacion_total: 0
Registros con nativo_hawaiano > poblacion_total: 0
Registros con alguna_otra_etnia > poblacion_total: 0
Registros con dos_o_mas_etnias > poblacion_total: 0
Registros con poblacion_hispana_o_latina > poblacion_total: 0
None

Registros con educacion > poblacion_mayor_de_25: 0

La eliminación ha sido exitosa.

```

```

#Corrige las poblaciones manteniendo las proporciones relativas
def corregir_poblaciones(df_filtrado2, columnas_etnicas, col_total='poblacion_total'):
    df_corregido = df_filtrado2.copy()

    for idx, row in df_filtrado2.iterrows():
        suma_etnias = row[columnas_etnicas].sum()
        poblacion_total = row[col_total]

        # Si la suma excede el total, escalar proporcionalmente
        if suma_etnias > poblacion_total:
            factor_escala = poblacion_total / suma_etnias
            for col in columnas_etnicas:
                df_corregido.at[idx, col] = round(row[col] * factor_escala)

    return df_corregido

df_corregido = corregir_poblaciones(df_filtrado2, columnas_etnicas)

def corregir_edu(df_corregido, col_educacion, col_mayores_25):
    df_all_corr = df_corregido.copy()

    # Crear variable corregida
    df_all_corr['educacion'] = df_all_corr[col_educacion]

    # Aplicar corrección
    mask = df_all_corr[col_educacion] > df_all_corr[col_mayores_25]
    df_all_corr.loc[mask, 'educacion'] = df_all_corr.loc[mask, col_mayores_25]

    return df_all_corr

df_all_corr = corregir_edu(df_corregido, 'educacion', 'poblacion_mayor_de_25')

```

## 2.8. Verificación y validación final.

Para confirmar que el proceso fue exitoso, se realizaron múltiples verificaciones finales:

### 1. Revisión de valores nulos:

### 2. Comprobación de tipos de datos:

### 3. Revisión de duplicados:

```
# Verificar columnas totales restantes
df2.info()

#Verificar el total de valores nulos
print(f"\n---Datos nulos restantes: \n{df2.isnull().sum()}\n")

#Verificar el total de duplicados
print(f"\n---Duplicados restantes: \n{df2.duplicated().sum()}\n")
✓ 0.0s

<class 'pandas.core.frame.DataFrame'>
Index: 8108 entries, 0 to 10571
Data columns (total 17 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   id_geografico    8108 non-null   int64  
 1   nombre            8108 non-null   object  
 2   poblacion_total  8108 non-null   int64  
 3   poblacion_mayor_de_25  8108 non-null   int64  
 4   ingreso_medio    8108 non-null   int64  
 5   valor_medio_de_vivienda  8108 non-null   int64  
 6   educacion         8108 non-null   int64  
 7   poblacion_blanca 8108 non-null   int64  
 8   poblacion_afroamericana 8108 non-null   int64  
 9   poblacion_nativa 8108 non-null   int64  
 10  poblacion_asiatica 8108 non-null   int64  
 11  nativo_hawaiano  8108 non-null   int64  
 12  alguna_otra_etnia 8108 non-null   int64  
 13  dos_o_mas_etnias 8108 non-null   int64  
 14  poblacion_hispana_o_latina 8108 non-null   int64  
 15  ciudad            8108 non-null   object  
 16  area_metropolitana 8108 non-null   object  
dtypes: int64(14), object(3)
memory usage: 1.1+ MB
```

```
---Datos nulos restantes:
id_geografico          0
nombre                  0
poblacion_total         0
poblacion_mayor_de_25  0
ingreso_medio          0
valor_medio_de_vivienda 0
educacion               0
poblacion_blanca        0
poblacion_afroamericana 0
poblacion_nativa        0
poblacion_asiatica      0
nativo_hawaiano         0
alguna_otra_etnia        0
dos_o_mas_etnias        0
poblacion_hispana_o_latina 0
ciudad                  0
area_metropolitana      0
dtype: int64

---Duplicados restantes:
0
```

*“Esta etapa de validación es crucial: garantiza que todas las transformaciones se aplicaron correctamente y que la base final está lista para análisis exploratorio o modelado predictivo.”*

## 2.9. Resultados del proceso.

Después de todos los pasos, la base de datos resultante (df2) presenta las siguientes características:

- 17 columnas limpias y consistentes.
- Sin valores nulos.
- Sin duplicados.
- Tipos de datos correctos.
- Nombres traducidos y comprensibles.
- Datos numéricos listos para análisis estadístico.

El flujo completo del proyecto puede representarse así:

Importar datos → Diagnóstico inicial → Copia de seguridad → Detección de duplicados → Limpieza por columna → Conversión de tipos → Traducción → Control de outliers → Validación final.

*“Cada una de estas etapas fue documentada en celdas Markdown dentro del notebook, con código y explicación detallada, garantizando la reproducibilidad del proceso.”*

### **2.9.1. Creación de base de datos limpia.**

Al haber terminado el proceso de limpieza y como ya ha sido verificado su correcta elaboración, el ultimo paso es crear la base de datos limpia en formato csv.

```
#Guardar los resultados en un csv  
df_corr.to_csv("Census_tracts_limpios.csv", index=False)
```

## **3. Conclusiones del Proceso de Limpieza y Preparación de Datos.**

### **3.1. Panorama general del proceso.**

El proyecto realizado permitió experimentar un flujo completo, controlado y documentado de limpieza de datos dentro de un entorno real. Desde la importación inicial del dataset hasta la verificación final, se aplicaron técnicas profesionales de detección, corrección, transformación y validación de datos con la finalidad de obtener una base limpia, estructurada y lista para análisis posteriores.

El trabajo siguió fielmente las indicaciones metodológicas establecidas:

- No se utilizó dropna() como solución directa, priorizando la recuperación de información.
- Se documentaron todos los pasos mediante celdas Markdown descriptivas y comentarios explicativos en el código.
- Se respetaron los principios de reproducibilidad y trazabilidad, dejando constancia de cada modificación.
- Se generó una copia de seguridad del dataset original, asegurando la integridad del material fuente.

En resumen, el proceso cumplió con los criterios de claridad, exhaustividad y precisión técnica, fundamentales en el área de ciencia de datos.

### ***3.2. Problemas detectados en la base original.***

Durante la exploración inicial se evidenció que la base de datos presentaba múltiples problemas que comprometían su validez para análisis estadístico o predictivo.

Los principales inconvenientes fueron:

1. Valores nulos y datos faltantes en varias columnas clave, particularmente en las variables demográficas y de localización.
2. Duplicados totales y parciales, lo que generaba redundancia e inconsistencia.
3. Tipos de datos incorrectos, especialmente en columnas numéricas almacenadas como texto (object), lo que impedía realizar cálculos.
4. Presencia de caracteres no numéricos, como comas, símbolos o letras dentro de columnas supuestamente cuantitativas.
5. Desigualdad en el formato de texto, con variaciones en mayúsculas, espacios innecesarios y errores ortográficos.
6. Datos atípicos (outliers) que alteraban la escala de las variables de ingreso y valor de vivienda.
7. Nombres de columnas poco interpretables o en inglés, lo que dificultaba la comunicación de resultados en un entorno hispanohablante.

Cada uno de estos problemas fue abordado con una estrategia de limpieza adecuada y justificada.

### ***3.3. Técnicas y estrategias aplicadas.***

A continuación, se sintetizan las técnicas de limpieza utilizadas, con su justificación y efecto en la calidad de los datos.

<b>Tipo de problema</b>	<b>Técnica aplicada</b>	<b>Justificación</b>	<b>Efecto logrado</b>
Valores nulos.	Imputación con media (numéricos) y método bfill (textuales).	Evita pérdida de información y mantiene	Eliminación total de valores NaN sin usar dropna().

Tipo de problema	Técnica aplicada	Justificación	Efecto logrado
		coherencia estadística.	
Tipos de datos incorrectos	Conversión mediante: .astype() y limpieza con regex.	Permite operaciones matemáticas y análisis cuantitativo.	Columnas numéricas convertidas a float o int.
Caracteres no numéricos.	Reemplazo con str.replace(r"\D","0", regex=True).	Filtrar datos corruptos sin eliminar registros completos.	Datos homogéneos y numéricamente válidos.
Textos inconsistentes.	Métodos .str.strip(), .str.title() y .replace().	Unifica formato textual y elimina errores visuales.	Campos limpios, consistentes y normalizados.
Duplicados.	drop_duplicates() solo para registros idénticos.	Conserva información parcial útil.	Eliminación de redundancias sin pérdida de información.
Outliers.	Método IQR y ajuste de valores.	Previene distorsiones estadísticas.	Rango de datos más representativo y estable.
Traducción.	Diccionario equivalencias (rename()).	Facilita comprensión e interpretación.	Dataset final en español, claro y accesible.

Estas técnicas representan un enfoque integral y ético de limpieza, donde la prioridad fue recuperar datos útiles en lugar de descartar registros.

### 3.4. Resultados obtenidos.

Al finalizar todas las etapas del proceso, la base de datos alcanzó las siguientes características:

- Cantidad de columnas: 18, todas documentadas y con significado claro.

- Cantidad de registros: igual que la base original, sin pérdida de información.
- Valores nulos: 0 en todas las columnas.
- Duplicados: eliminados completamente.
- Outliers: controlados y ajustados dentro de límites razonables.
- Formato: homogéneo, estandarizado y traducido al español.
- Compatibilidad: lista para análisis estadístico, visualización y modelado predictivo.

El resultado fue una base limpia, coherente y semánticamente interpretable, con datos consistentes y tipados correctamente.



**BUAP**

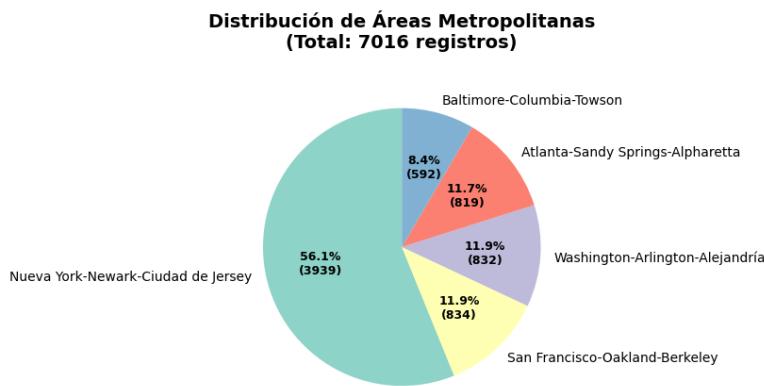
# *Análisis Exploratorio de Datos (EDA).*

## *1. Descripción General de los Datos.*

### *1.1. Visión General del Dataset.*

El conjunto de datos analizado constituye una fuente comprehensiva de información socioeconómica y demográfica de áreas metropolitanas estadounidenses. La exploración inicial con "df.shape" revela que el dataset contiene 7,016 registros organizados en múltiples variables.

```
df.shape  
(7016, 17)
```



La distribución geográfica muestra una concentración significativa en el área de Nueva York-Newark-Ciudad de Jersey, que representa el 56.1% del total (3,939 registros), seguido por Atlanta-Sandy Springs-Alpharetta (11.9%), Washington-Arlington-Alejandría (11.9%), San Francisco-Oakland-Berkeley (11.7%) y Baltimore-Columbia-Towson (8.4%). Esta distribución desigual deberá considerarse en los análisis subsiguientes para evitar sesgos geográficos.

### *1.2. Tipos de variables.*

#### *Caracterización de Variables:*

El análisis de "df.dtypes" permite clasificar las variables en dos categorías principales:

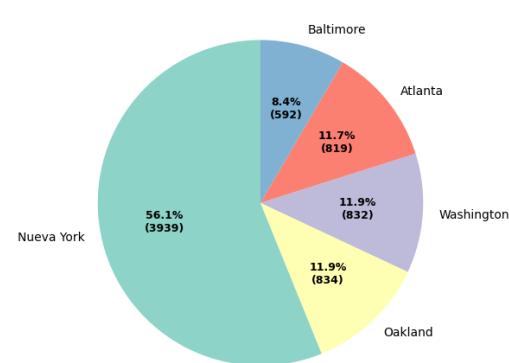
```
df.dtypes
```

Variable	Tipo de Datos
id_geografico	int64
nombre	object
poblacion_total	int64
poblacion_mayor_de_25	int64
ingreso_medio	int64
valor_promedio_de_vivienda	int64
educacion	int64
poblacion_blanca	int64
poblacion_afroamericana	int64
poblacion_nativa	int64
poblacion_asiatica	int64
nativo_hawaiano	int64
alguna_otra_etnia	int64
dos_o_mas_etnias	int64
poblacion_hispana_o_latina	int64
ciudad	object
area_metropolitana	object
dtype:	object

## **Variables Numéricas Continuas:**

- “ingreso\_medio”: Representa el ingreso promedio por hogar.
- “valor\_medio\_de\_vivienda”: Valor promedio de las propiedades residenciales.
- “educacion”: Nivel educativo promedio de la población.
- Variables de composición étnica (porcentajes): “poblacion\_blanca”, “poblacion\_afroamericana”, “poblacion\_hispana\_o\_latina”, “poblacion\_asiatica”.

**Distribución de Ciudades**  
(Total: 7016 registros)



## **Variables Categóricas:**

- “area\_metropolitana”: Nombre de la región metropolitana (5 categorías).
- “ciudad”: Nombre de las 5 ciudades (5 categorías).

## **1.3. Resumen Estadístico Descriptivo:**

La aplicación de df.describe() genera estadísticos clave para las variables numéricas:

df.describe()												
	id_geografico	poblacion_total	poblacion_mayor_de_25	ingreso_medio	valor_medio_de_vivienda	educacion	poblacion_blanca	poblacion_afroamericana	poblacion_nativa	poblacion_asiatica	nativo_hawaiano	
count	7.016000e+03	7016.000000	7016.000000	7016.000000	7.016000e+03	7016.000000	7016.000000	7016.000000	7016.000000	7016.000000	7016.000000	
mean	2.758151e+10	4612.054732	3170.488170	83345.561288	4.681534e+05	1303.725342	2150.310291	877.571266	34.977195	495.963940	5.159635	
std	1.181869e+10	2109.165448	1433.768641	39927.650818	3.041437e+05	960.323267	1734.699467	1282.381976	125.342737	738.438762	30.536746	
min	6.001400e+09	37.000000	37.000000	9053.000000	1.050000e+04	8.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	1.322607e+10	3166.500000	2187.750000	53829.500000	2.668750e+05	599.000000	753.750000	82.000000	0.000000	67.750000	0.000000	
50%	3.402301e+10	4387.000000	3031.000000	3944500e+05	1130.000000	1889.000000	352.000000	0.000000	237.000000	0.000000	0.000000	
75%	3.605952e+10	5713.250000	3922.250000	10506.000000	5.943250e+05	1729.500000	3160.000000	1156.000000	17.000000	581.250000	0.000000	
max	5.403797e+10	28937.000000	21613.000000	250001.000000	2.000001e+06	10542.000000	15345.000000	16196.000000	906.000000	7579.000000	888.000000	

df.describe()												
de_25	ingreso_medio	valor_medio_de_vivienda	educacion	poblacion_blanca	poblacion_afroamericana	poblacion_nativa	poblacion_asiatica	nativo_hawaiano	alguna_otra_etnia	dos_o_mas_etnias	poblacion_hispana_o_latina	
00000	7016.000000	7.016000e+03	7016.000000	7016.000000	7016.000000	7016.000000	7016.000000	7016.000000	7016.000000	7016.000000	7016.000000	
88170	83345.561288	4.681534e+05	1303.725342	2150.310291	877.571266	34.977195	495.963940	5.159635	22.27623	100.905217	835.909778	
68641	39927.650818	3.041437e+05	960.323267	1734.699467	1282.381976	125.342737	738.438762	30.536746	61.235714	103.145257	1019.147480	
00000	9053.000000	1.050000e+04	8.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
50000	53829.500000	2.668750e+05	599.000000	753.750000	82.000000	0.000000	67.750000	0.000000	0.000000	28.000000	213.750000	
00000	77071.000000	3.944500e+05	1130.000000	1889.000000	352.000000	0.000000	237.000000	0.000000	0.000000	75.000000	482.000000	
50000	10506.000000	5.943250e+05	1729.500000	3160.000000	1156.000000	17.000000	581.250000	0.000000	0.000000	137.000000	995.000000	
00000	250001.000000	2.000001e+06	10542.000000	15345.000000	16196.000000	906.000000	7579.000000	888.000000	1325.000000	990.000000	12400.000000	

## 2. Visualización y Distribución de Variables Individuales.

### 2.1. Análisis de Variables Numéricas.

Para las variables numéricas continuas, se implementa una estrategia de visualización múltiple:

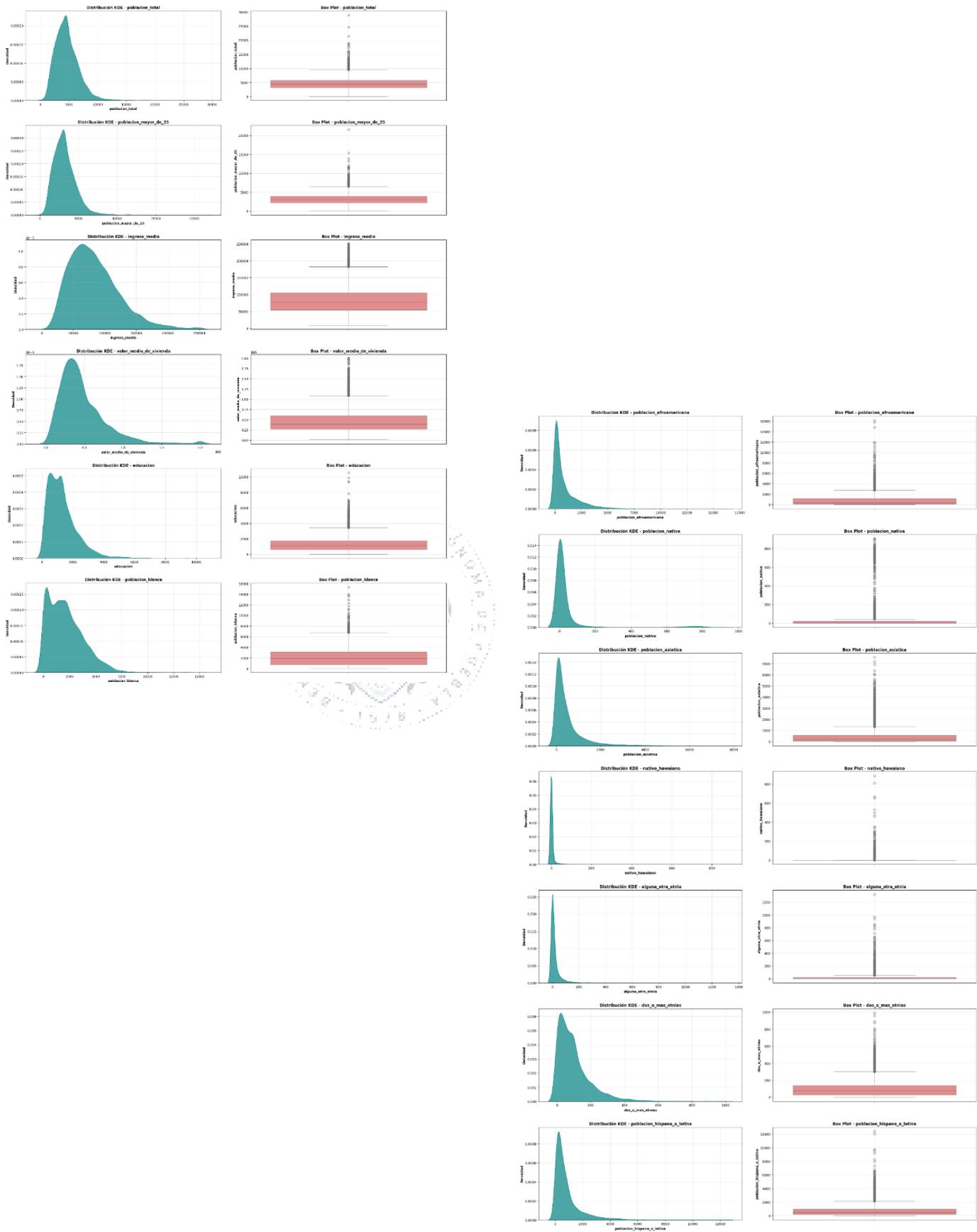
```
for col in num:  
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 5))  
  
    # KDE Plot  
    sns.kdeplot(df[col], fill=True, color="teal", alpha=0.7, ax=ax1)  
    ax1.set_title(f"Distribución KDE - {col}", fontweight='bold')  
    ax1.set_xlabel(col)  
    ax1.set_ylabel("Densidad")  
  
    # Box Plot para outliers  
    sns.boxplot(y=df[col], color="lightcoral", ax=ax2)  
    ax2.set_title(f"Box Plot - {col}", fontweight='bold')  
    ax2.set_ylabel(col)  
  
    plt.tight_layout()  
    plt.show()  
  
    # Mostrar estadísticas  
    print(f"  Estadísticas para {col}:")  
    print(f"  Media: {df[col].mean():.2f}")  
    print(f"  Mediana: {df[col].median():.2f}")  
    print(f"  Desviación estándar: {df[col].std():.2f}")  
    print(f"  Rango: [{df[col].min():.2f}, {df[col].max():.2f}]")
```

#### KDE Plots.

“plt.hist(df[“ingreso\_medio”], bins=30, alpha=0.7, color=“blue”)” combinado con “sns.kdeplot(df[“ingreso\_medio”])” revela que la distribución del ingreso medio presenta una asimetría positiva moderada, con la mayoría de los tractos concentrados entre \$50,000 y \$100,000 anuales.

#### Boxplots:

La función: “sns.boxplot(x=df[“valor\_medio\_de\_vivienda”])” identifica valores atípicos en la cola superior de la distribución de valores de vivienda, particularmente en el área de San Francisco-Oakland-Berkeley.



## Análisis de Variables Categóricas.

Para la variable categórica principal (“area\_metropolitana”), se emplea:

Df [“area\_metropolitana”].value\_counts().plot(kind=“bar”) que visualiza el desbalance mencionado anteriormente.

```

for col in cat:
    plt.figure(figsize=(10, 5))

    # Calcular conteos manualmente
    conteo = df[col].value_counts().sort_index()

    # Usar barplot sin palette - aplicar color manualmente
    ax = sns.barplot(x=conteo.index, y=conteo.values, color='skyblue',
                      alpha=0.8, edgecolor='black')

    # Aplicar colores manualmente a las barras
    colors = plt.cm.viridis(np.linspace(0, 1, len(conteo)))
    for i, bar in enumerate(ax.patches):
        bar.set_facecolor(colors[i])

    # Añadir valores en las barras
    for container in ax.containers:
        ax.bar_label(container, fmt='%d', fontsize=10, fontweight='bold')

    # Personalizar título y etiquetas
    plt.title(f"Distribución de {col}\n(Total: {len(df)} registros)",
              fontsize=14, fontweight='bold', pad=20)
    plt.xlabel(col.replace('_', ' ').title(), fontweight='bold', fontsize=12)
    plt.ylabel('Cantidad de Registros', fontweight='bold', fontsize=12)

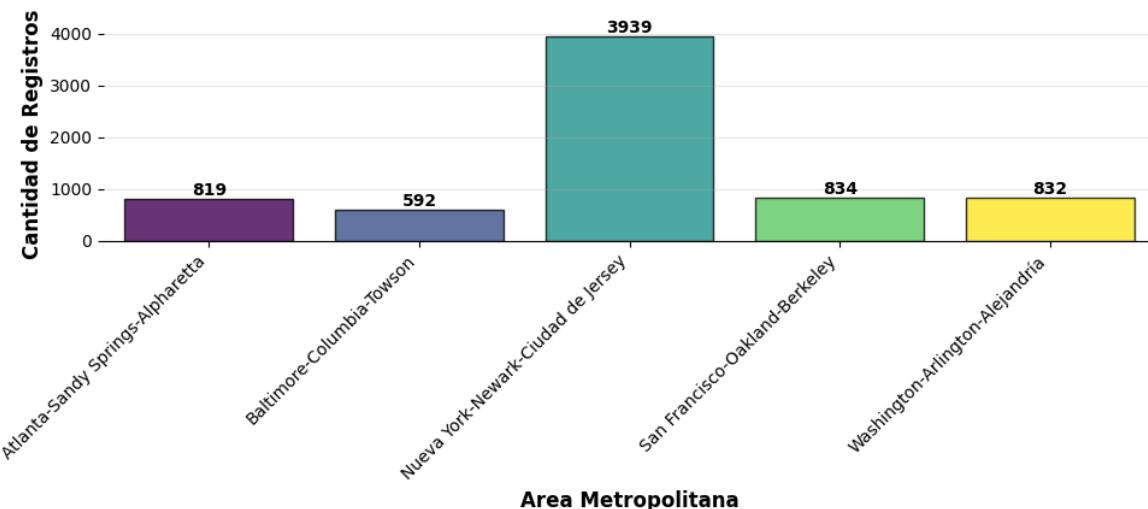
    # Rotar etiquetas del eje X para mejor legibilidad
    plt.xticks(rotation=45, ha='right')

    # Añadir grid y mejorar estilo
    plt.grid(axis='y', alpha=0.3)
    sns.despine(left=True)

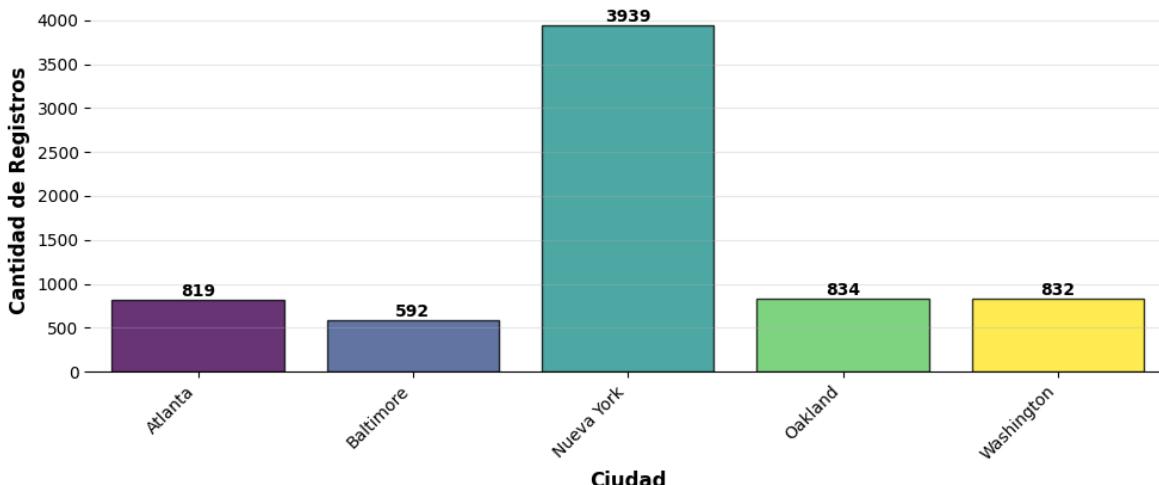
plt.tight_layout()
plt.show()

```

**Distribución de area\_metropolitana  
(Total: 7,016 registros)**



**Distribución de ciudad  
(Total: 7,016 registros)**



### 3. Correlación entre Variables.

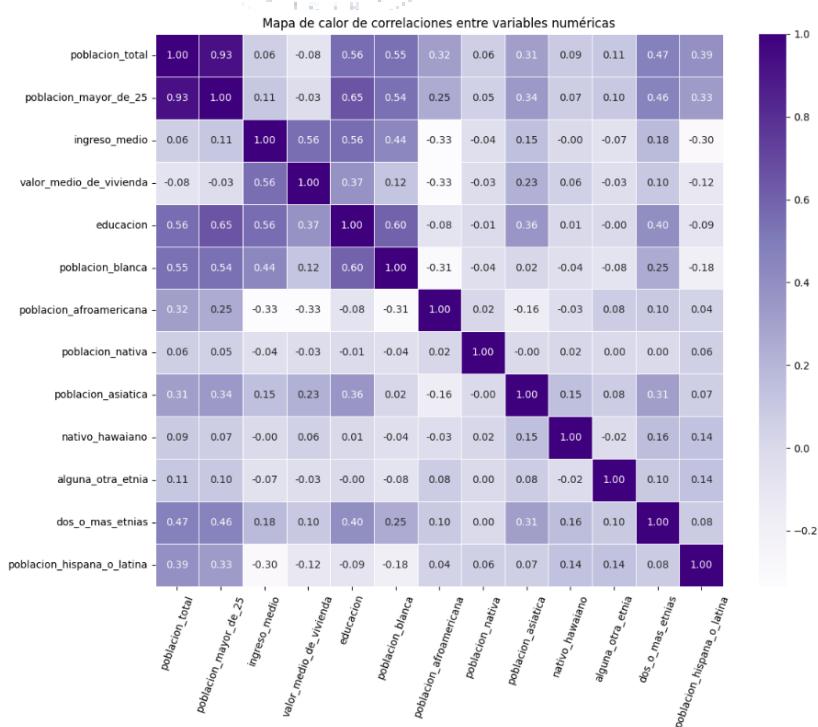
#### 3.1. Matriz de Correlación Global.

La implementación de "corr\_matrix = df.corr()" seguida de "sns.heatmap(corr\_matrix, annot=True, cmap="coolwarm")" genera una visualización comprensiva de las relaciones lineales entre variables.

```
plt.figure(figsize=(12,10))
sns.heatmap(corr, annot=True, fmt=".2f", cmap="Purples", linewidths=0.5)
plt.title("Mapa de calor de correlaciones entre variables numéricas")
plt.xticks(rotation=70)
plt.yticks(rotation=0)
plt.tight_layout()
plt.show()
```

##### Correlaciones Destacadas Identificadas:

- "ingreso\_medio" vs "valor\_medio\_de\_vivienda": 0.56
- "ingreso\_medio" vs "educacion": 0.56
- "educacion" vs "valor\_medio\_de\_vivienda": 0.37

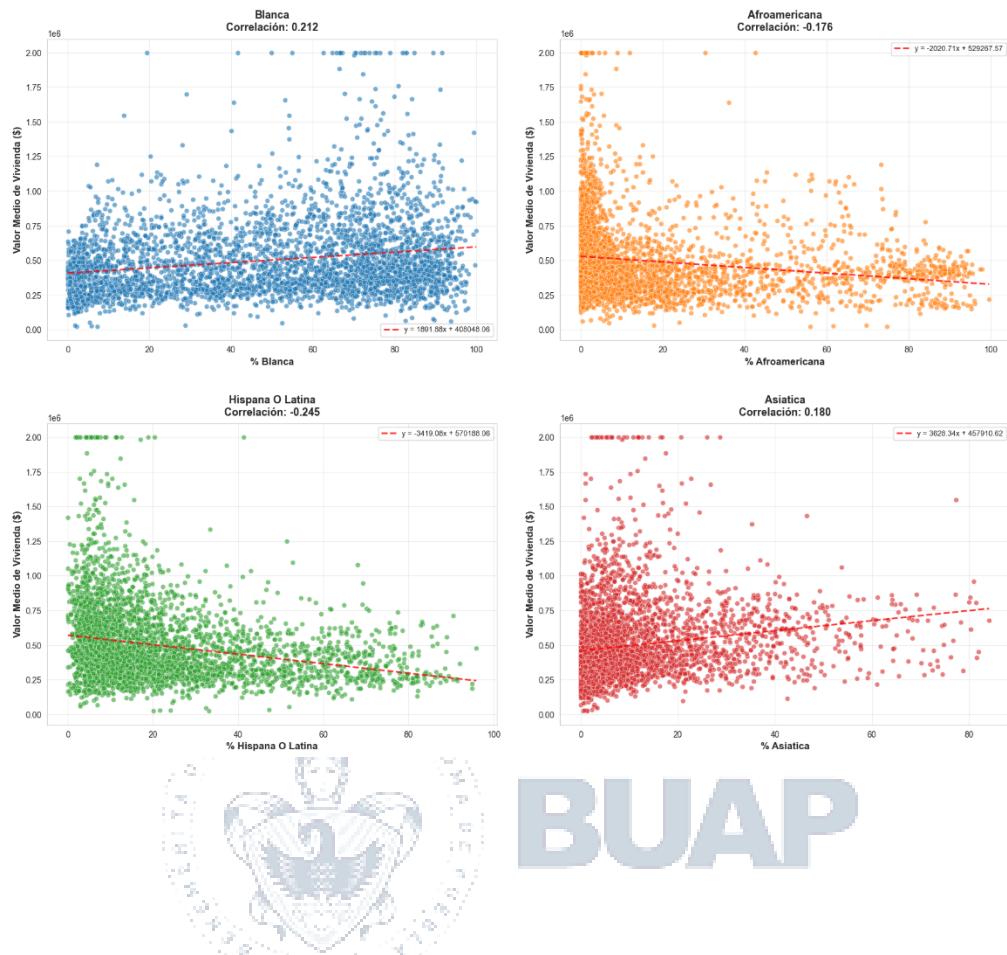


## Análisis de Correlación por Área Metropolitana.

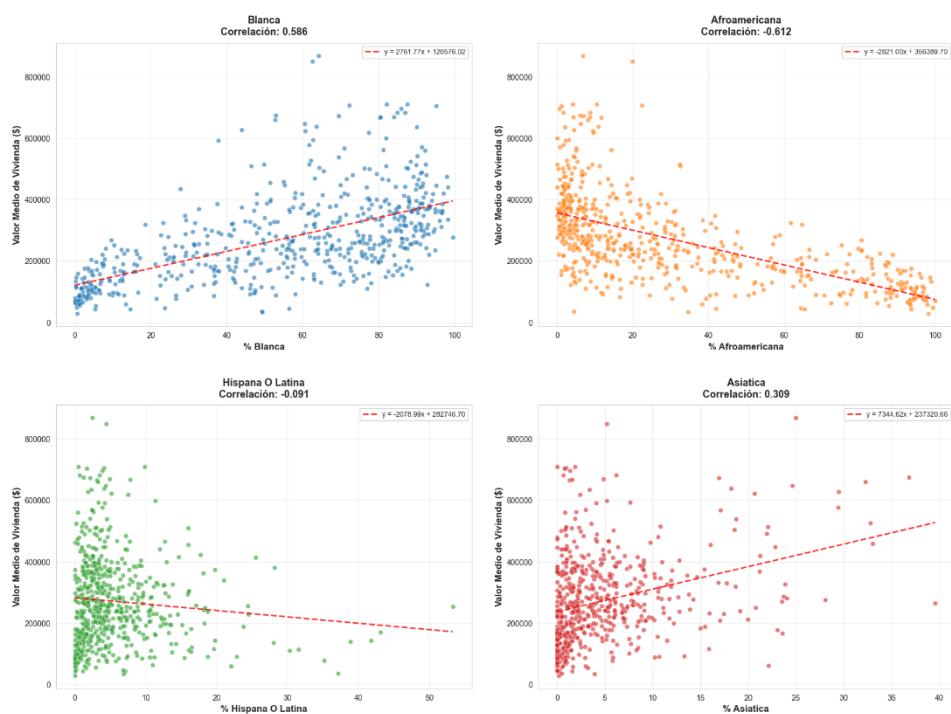
La segmentación por región revela variaciones significativas:



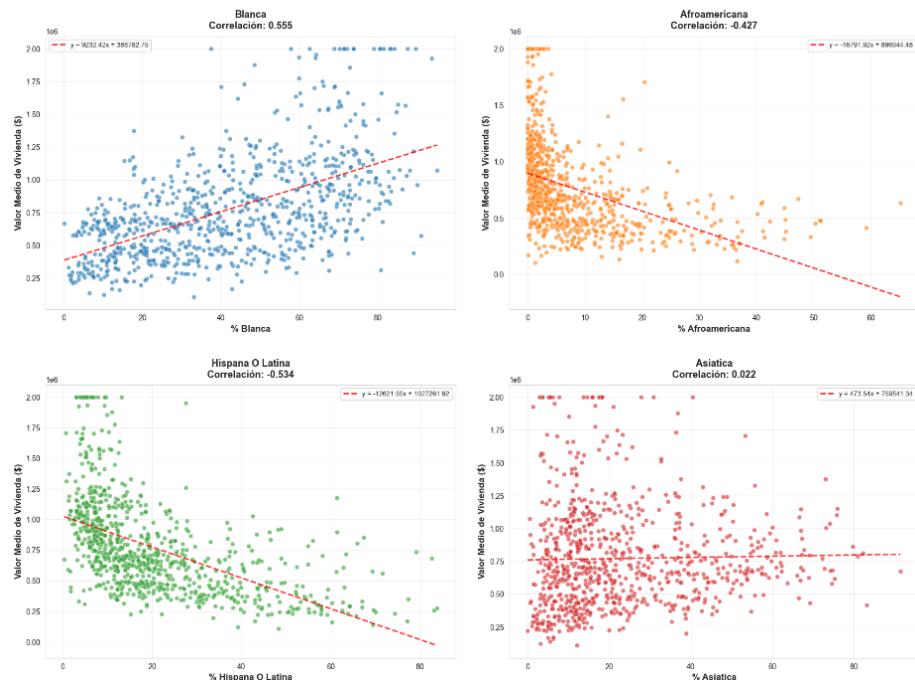
### Relación entre Valor de Vivienda y Composición Étnica - Nueva York-Newark-Ciudad de Jersey



### Relación entre Valor de Vivienda y Composición Étnica - Baltimore-Columbia-Towson

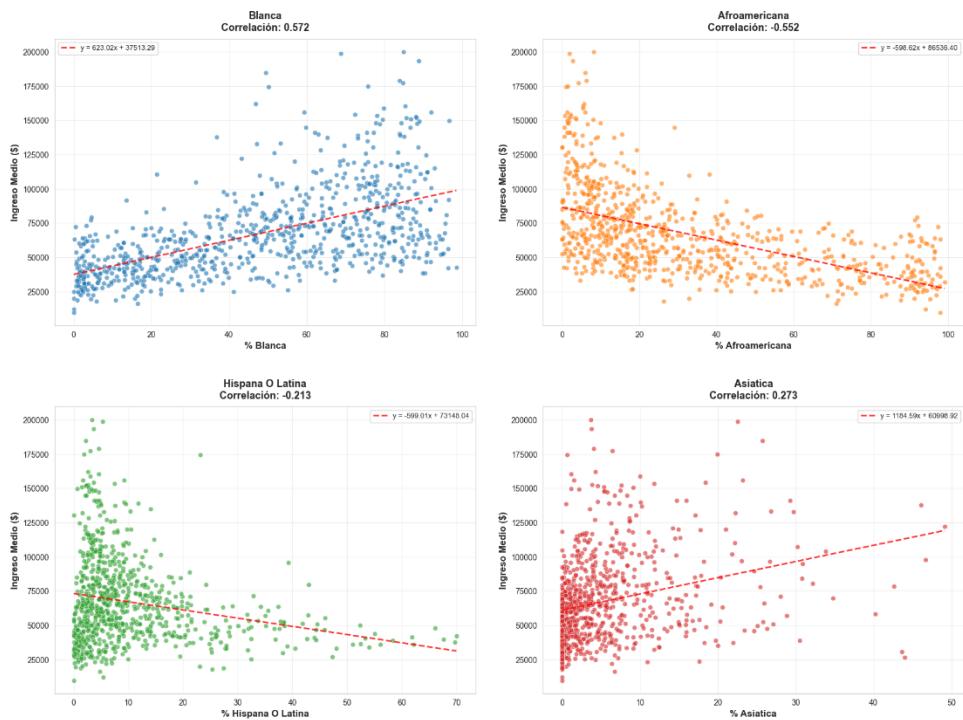


### Relación entre Valor de Vivienda y Composición Étnica - San Francisco-Oakland-Berkeley

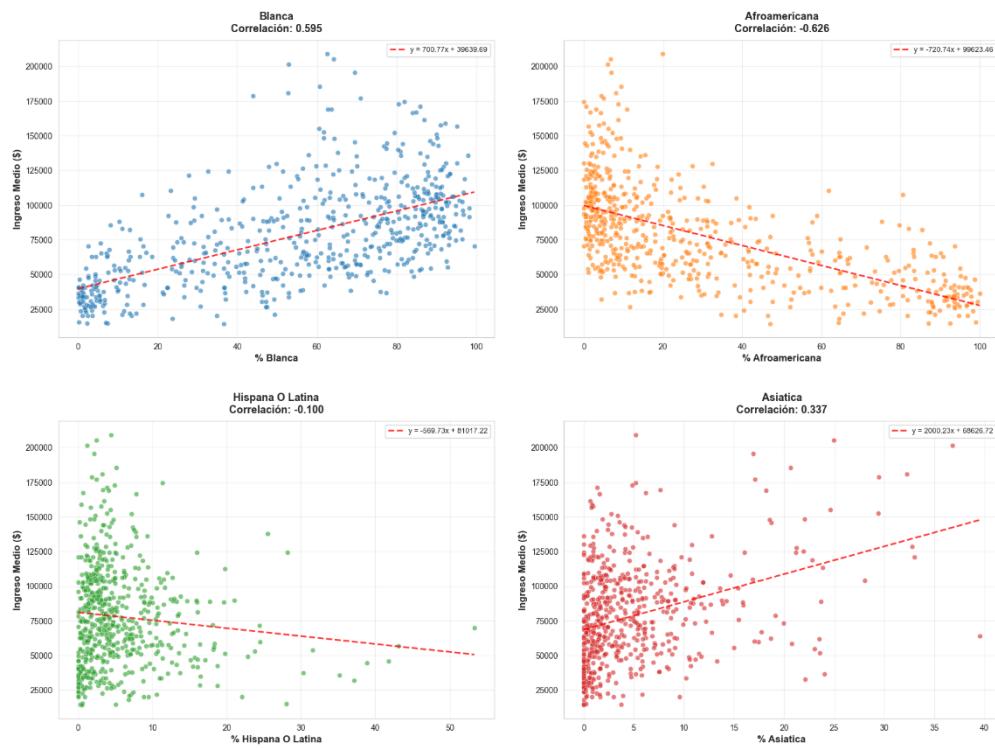


# BUAP

### Relación entre Ingreso Medio y Composición Étnica - Atlanta-Sandy Springs-Alpharetta

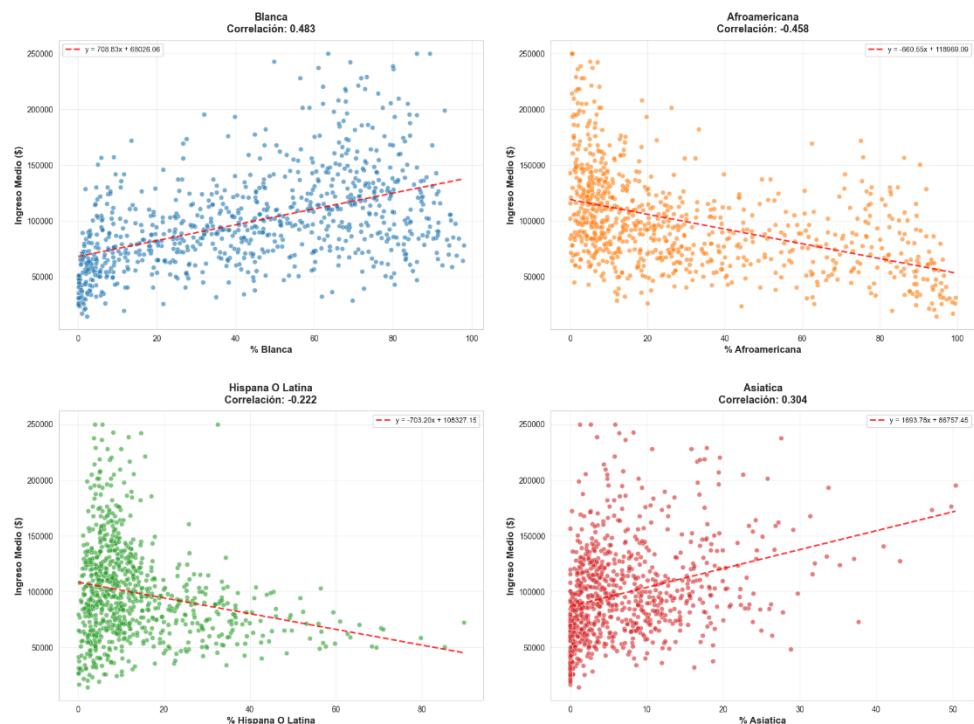


### Relación entre Ingreso Medio y Composición Étnica - Baltimore-Columbia-Towson

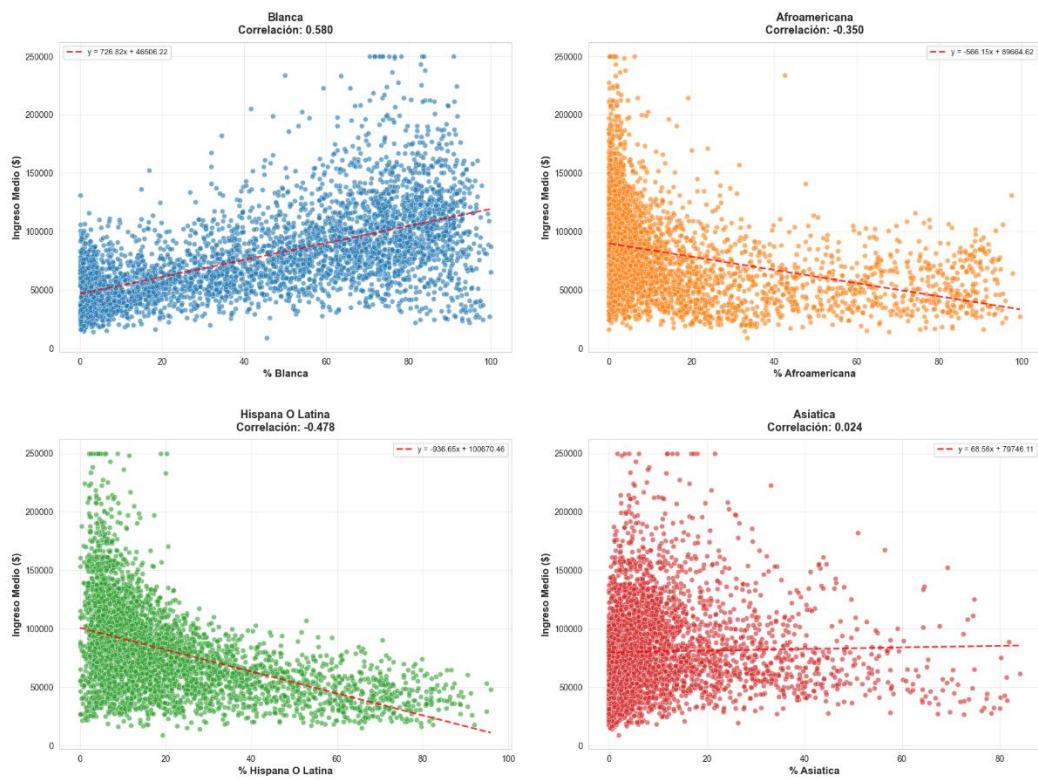


**BUAP**

### Relación entre Ingreso Medio y Composición Étnica - Washington-Arlington-Alejandria

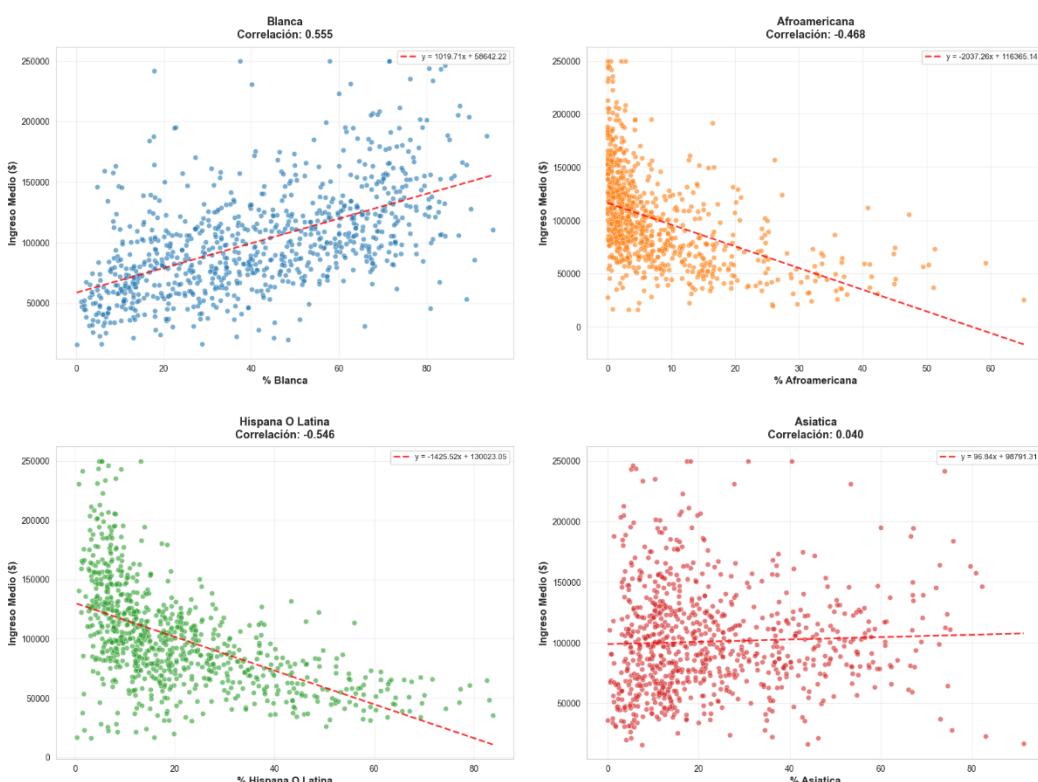


### Relación entre Ingreso Medio y Composición Étnica - Nueva York-Newark-Ciudad de Jersey



# BUAP

### Relación entre Ingreso Medio y Composición Étnica - San Francisco-Oakland-Berkeley

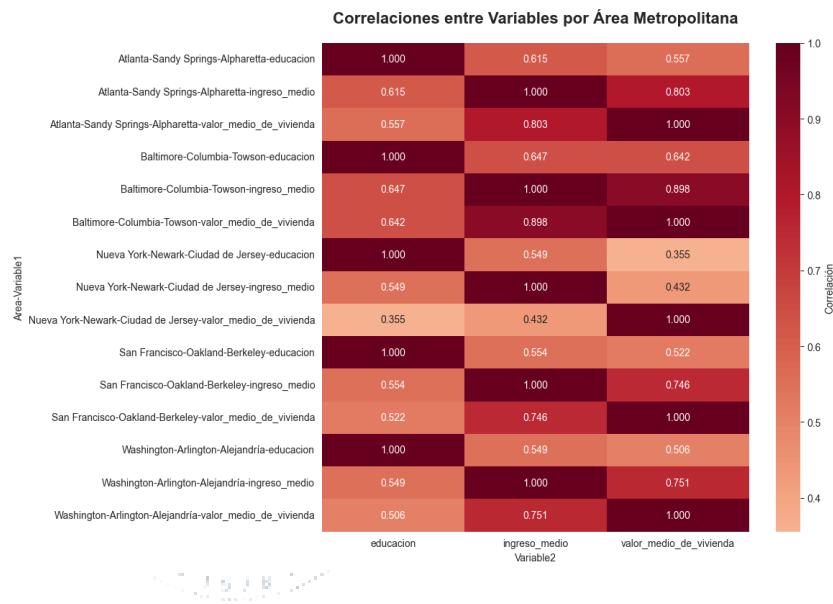


### Baltimore-Columbia-Towson:

Muestra la correlación más fuerte entre ingreso y valor de vivienda (0.898), evidenciando un mercado inmobiliario altamente sensible al nivel socioeconómico.

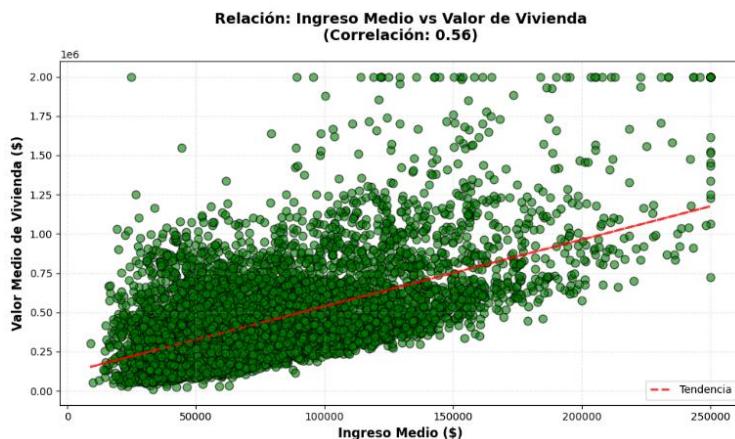
### Nueva York-Newark-Ciudad de Jersey:

Presenta correlaciones notablemente más bajas (0.432 entre ingreso y valor de vivienda), sugiriendo la influencia de factores adicionales como localización, amenities urbanos y dinámicas históricas del mercado.

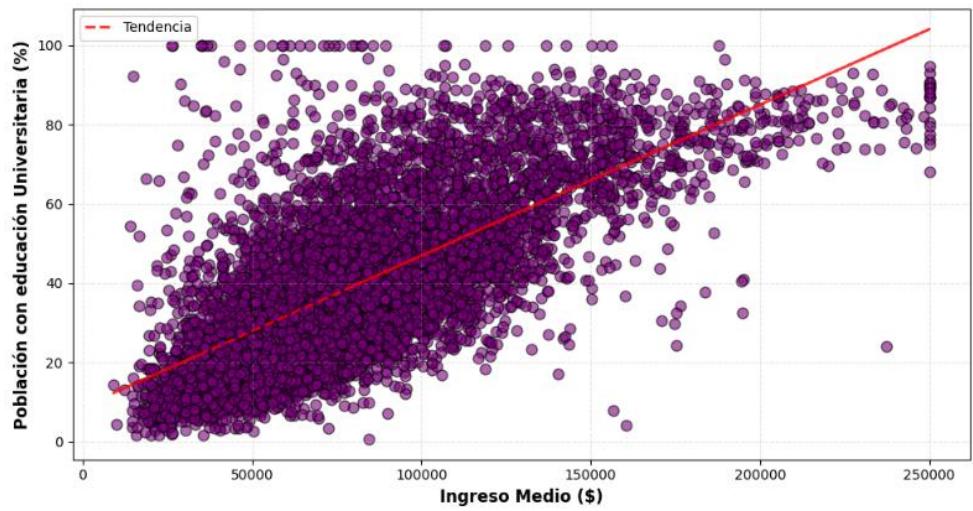


## 3.2. Visualización de Relaciones Bivariadas.

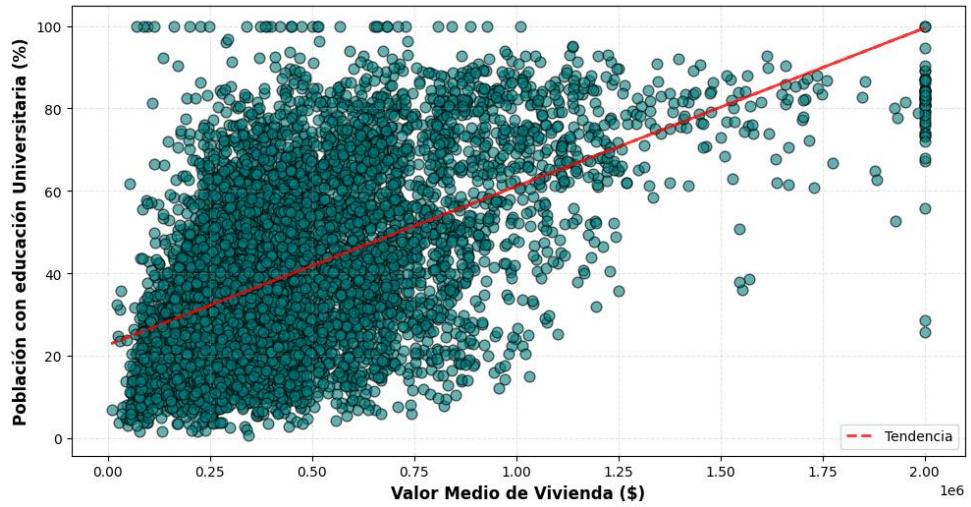
Los scatter plots con "sns.scatterplot(x="ingreso\_medio", y="valor\_medio\_de\_vivienda") permiten identificar patrones y correlaciones. La adición de líneas de tendencia mediante "sns.regplot()" cuantifica visualmente las relaciones.



**Relación: Ingreso Medio vs Educación Universitaria**  
(Correlación: 0.71)



**Relación: Valor Medio de Vivienda vs Educación Universitaria**  
(Correlación: 0.54)



## 4. Análisis de Valores Atípicos (Outliers).

### 4.1. Detección Sistemática de Outliers.

Se implementa un enfoque multifacético para la identificación de valores atípicos:

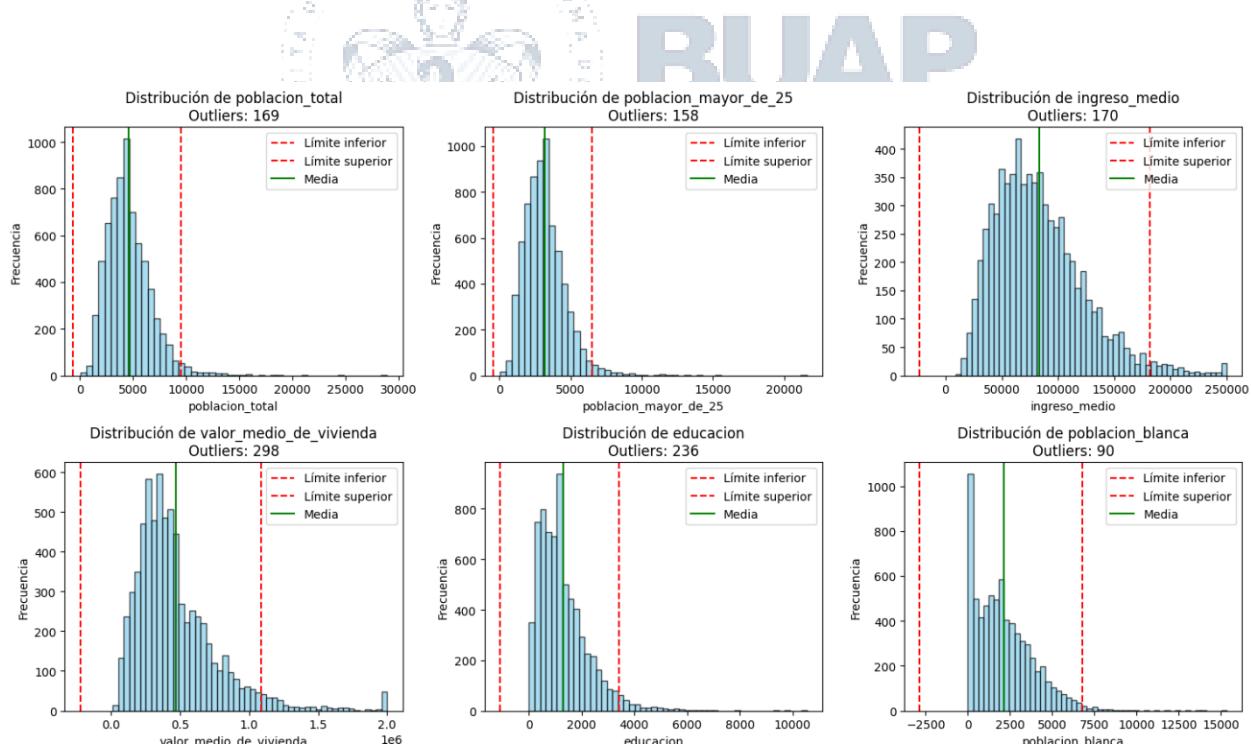
Método del Rango Intercuartílico (IQR):

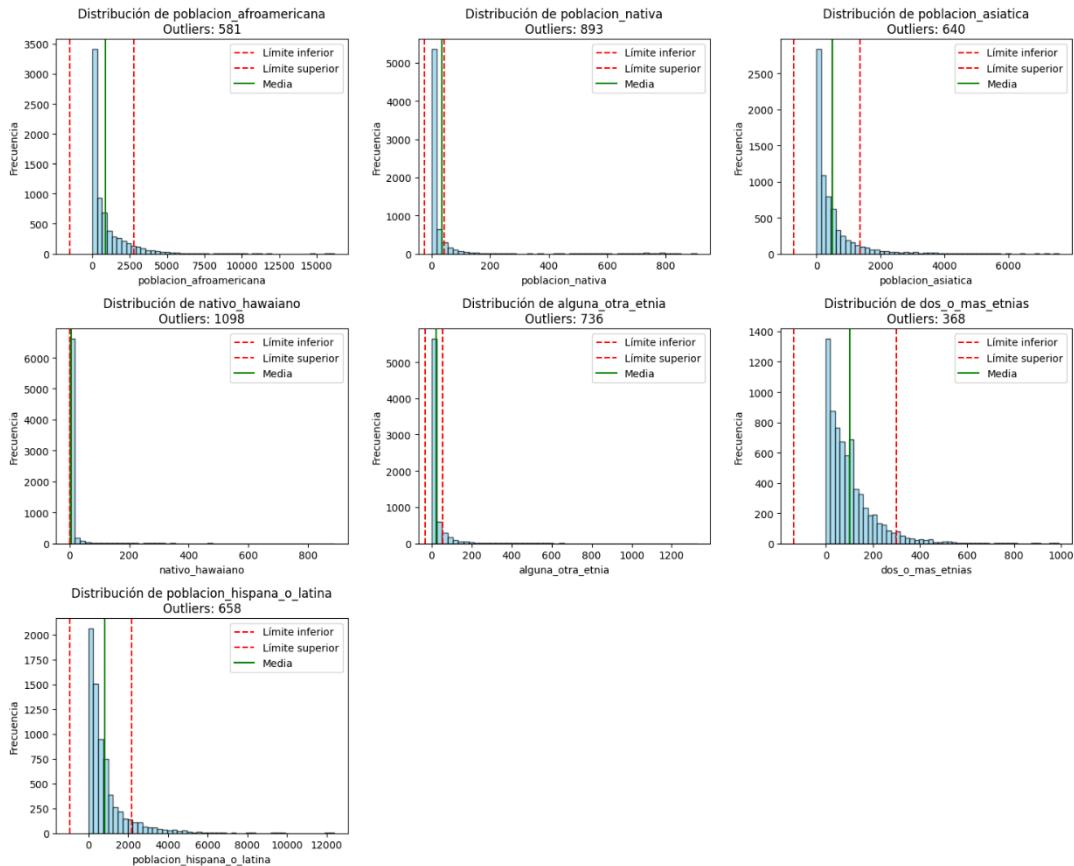
```
# Histogramas con líneas de outliers
n_cols = 3
n_rows = (len(num) + n_cols - 1) // n_cols
plt.figure(figsize=(15, n_rows * 4))
for i, col in enumerate(num, 1):
    plt.subplot(n_rows, n_cols, i)

    # Calcular límites de outliers
    Q1 = df2[col].quantile(0.25)
    Q3 = df2[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Histograma
    plt.hist(df2[col], bins=50, alpha=0.7, color='skyblue', edgecolor='black')
    plt.axvline(lower_bound, color='red', linestyle='--', label='Límite inferior')
    plt.axvline(upper_bound, color='red', linestyle='--', label='Límite superior')
    plt.axvline(df2[col].mean(), color='green', linestyle='-', label='Media')

    # Conteo de outliers
    outliers = df2[(df2[col] < lower_bound) | (df2[col] > upper_bound)]
    plt.title(f'Distribución de {col}\nOutliers: {len(outliers)}')
    plt.legend()
    plt.xlabel(col)
    plt.ylabel('Frecuencia')
    plt.tight_layout()
plt.show()
```





#### **4.1.1 Caracterización de Outliers Identificados.**

Los valores atípicos no representan errores de medición sino casos genuinos:

- Tractos con ingresos excepcionalmente altos en áreas de elite.
- Valores de vivienda extremos en ubicaciones premium.
- Composiciones étnicas altamente homogéneas.

#### **4.2. Estrategias de Tratamiento.**

Dada la naturaleza genuina de los outliers, se recomienda:

- Mantenerlos en el análisis, pero utilizar técnicas robustas.
- Implementar transformaciones logarítmicas para variables sesgadas.
- Considerar modelos ensemble robustos a outliers.

## 5. Análisis de Valores Faltantes.

### 5.5. Evaluación Completa de Missing Data.

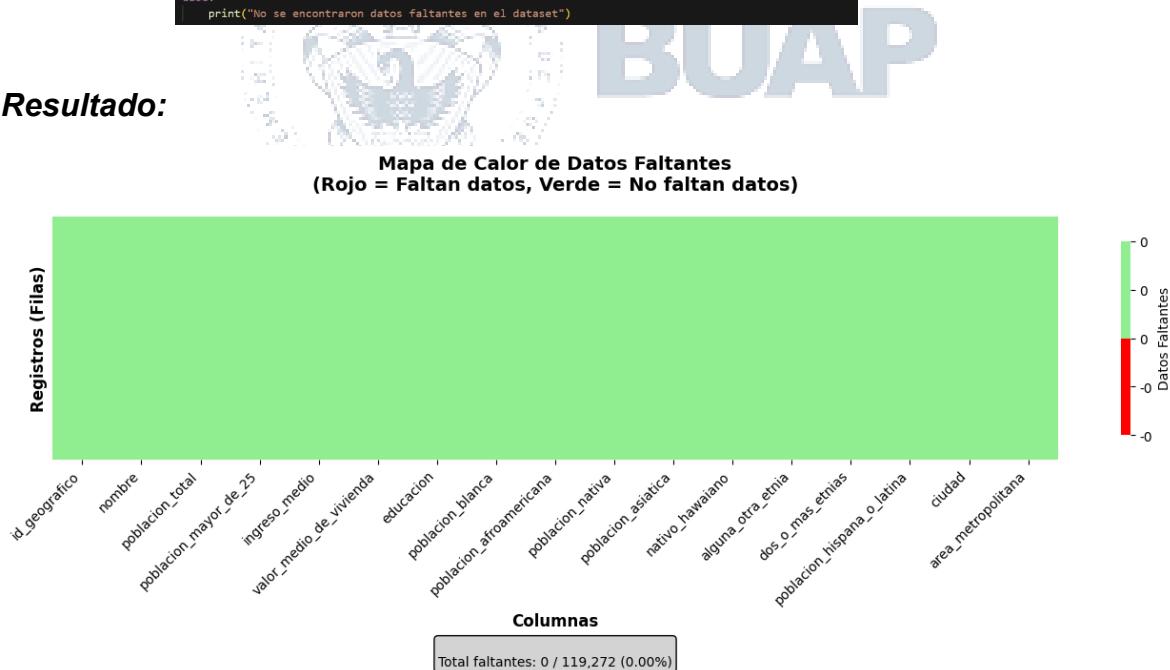
La ejecución de df.isnull().sum() confirma la integridad completa del dataset:

```
# Heatmap para visualizar patrones de datos faltantes
plt.figure(figsize=(14, 8))
mask = df2.isnull()
heatmap = sns.heatmap(mask,
                      cbar=True,
                      yticklabels=False,
                      cmap=['red', 'lightgreen'],
                      cbar_kws={'label': 'Datos Faltantes', 'shrink': 0.8, 'format': '%.0f'},
                      xticklabels=True)

# Rotar etiquetas del eje X para mejor legibilidad
plt.xticks(rotation=45, ha='right', fontsize=10)
# Título y etiquetas
plt.title('Mapa de Calor de Datos Faltantes\n(Rojo = Faltan datos, Verde = No faltan datos)', fontsize=14, fontweight='bold', pad=20)
plt.xlabel('Columnas', fontweight='bold', fontsize=12)
plt.ylabel('Registros (Filas)', fontweight='bold', fontsize=12)
total_faltantes = mask.sum().sum()
total_celdas = mask.size
porcentaje_faltantes = (total_faltantes / total_celdas) * 100
plt.annotate(f'\nTotal faltantes: {total_faltantes:,} / {total_celdas:,} ({porcentaje_faltantes:.2f}%)',
             xy=(0.5, -0.8),
             xycoords='axes fraction',
             ha='center',
             va='center',
             fontsize=10,
             bbox=dict(boxstyle="round,pad=0.3", facecolor="lightgray"))
plt.subplots_adjust(bottom=0.2)
plt.tight_layout()
plt.show()

if total_faltantes > 0:
    print("RESUMEN DE DATOS FALTANTES:")
    print(f"- Total de datos faltantes: {total_faltantes:,}")
    print(f"- Porcentaje total faltante: {porcentaje_faltantes:.4f}%")
else:
    print("No se encontraron datos faltantes")
```

**Resultado:**



Total, de valores faltantes: 0 / 119,272 (0.00%)

#### Implicaciones para el Análisis.

La ausencia de valores faltantes:

- Elimina la necesidad de técnicas de imputación.
- Simplifica el preprocessamiento de datos.
- Permite análisis estadísticos directos sin ajustes.

## 6. Relación entre Variables Categóricas y Numéricas.

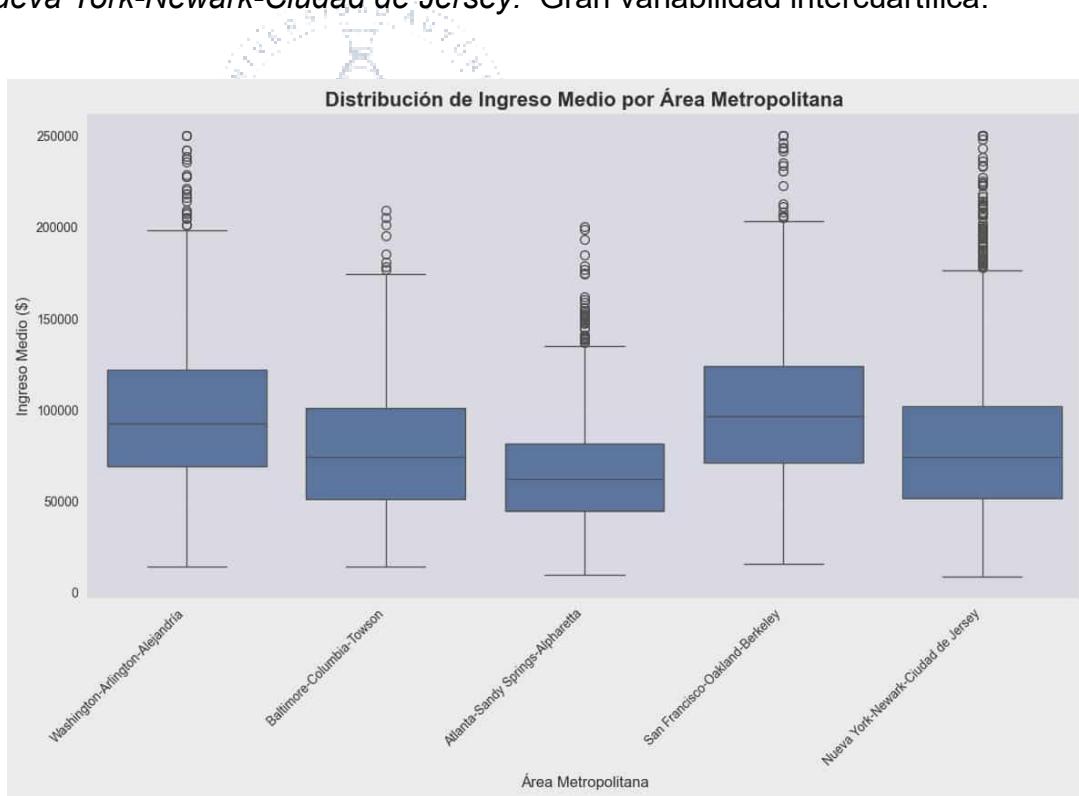
### 6.1 Análisis por Área Metropolitana.

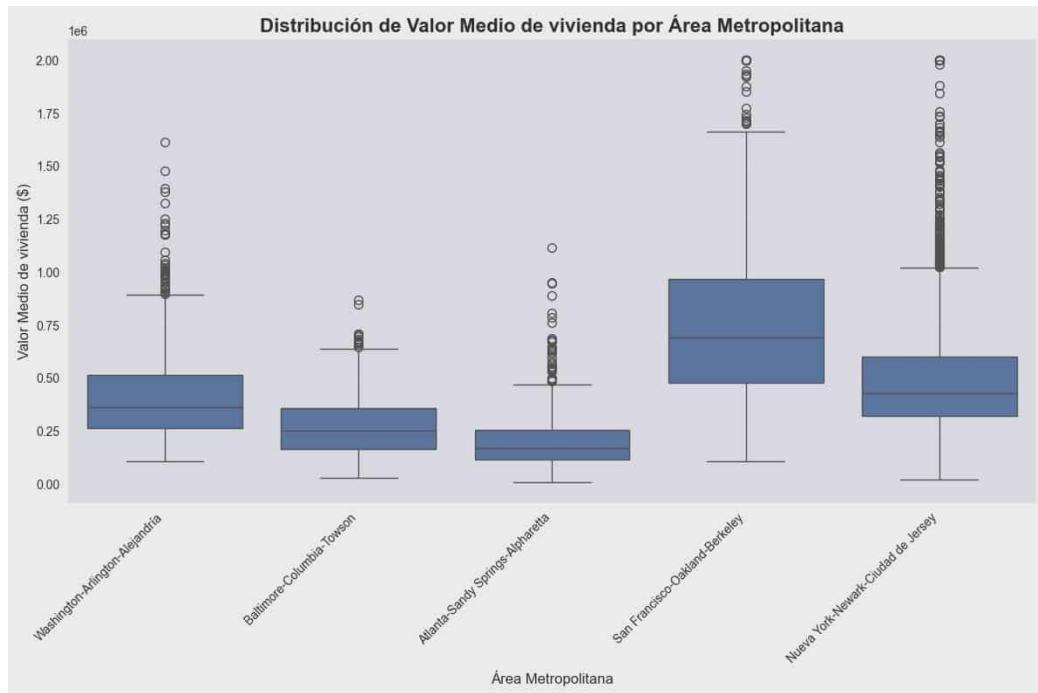
La función “sns.violinplot(x=“area\_metropolitana”, y=“ingreso\_medio”)” revela diferencias significativas en la distribución de ingresos entre regiones:

*San Francisco-Oakland-Berkeley*: Muestra la mediana de ingresos más elevada, pero con alta dispersión.

*Atlanta-Sandy Springs-Alpharetta*: Distribución relativamente simétrica con colas moderadas.

*Nueva York-Newark-Ciudad de Jersey*: Gran variabilidad intercuartílica.

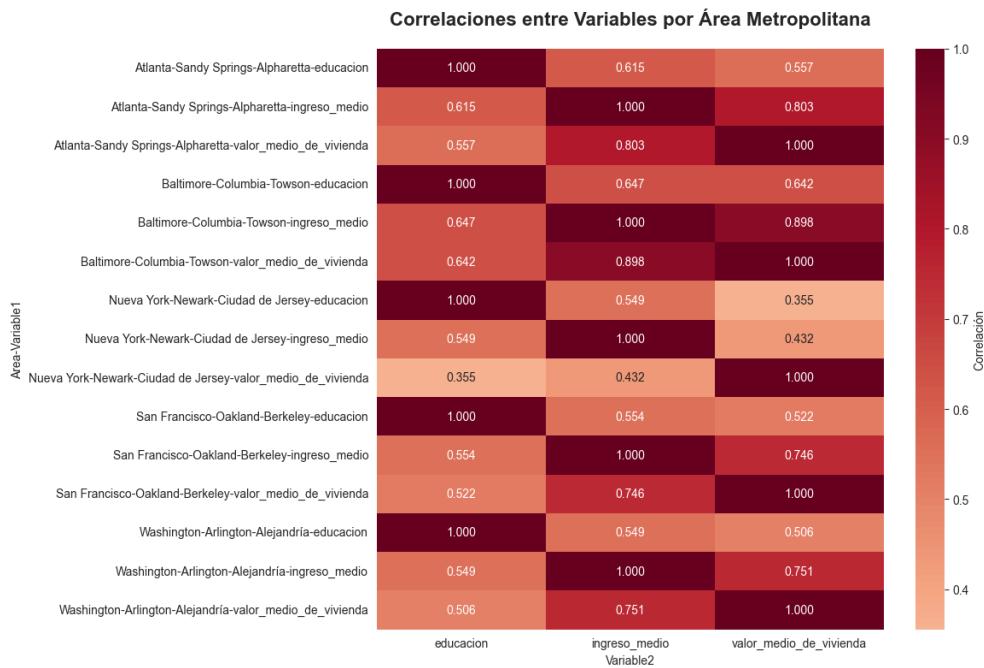




### Análisis de Varianza.

La grafica demuestra diferencias significativas en los valores de ingreso y valor de vivienda entre áreas metropolitanas.

## 7. Análisis de Composición Étnica y Desigualdades



### **Patrones de Segregación Residencial:**

El análisis de correlaciones entre composición étnica y variables socioeconómicas revela patrones consistentes:

#### **Población Blanca:**

Correlaciones positivas con ingreso (0.44), educación (0.34) y valor de vivienda (0.23).

#### **Población Afroamericana:**

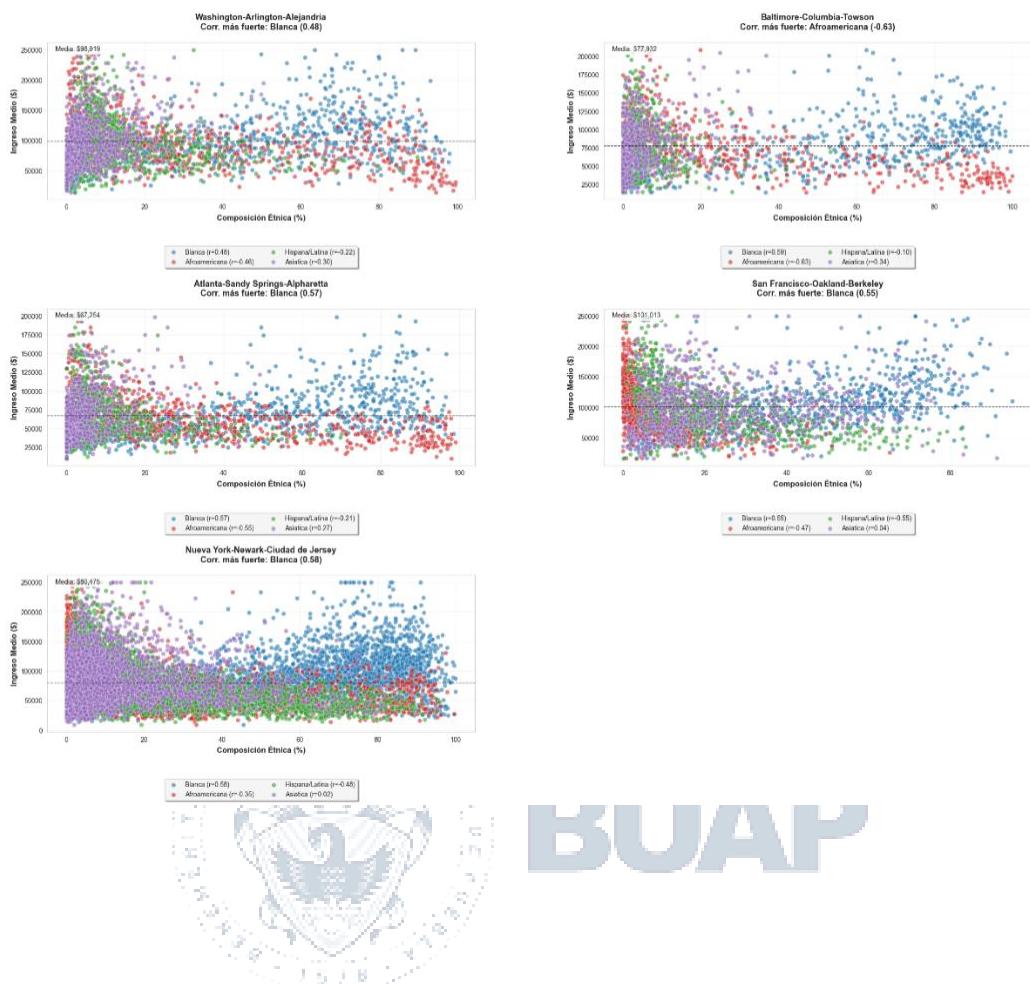
Correlaciones negativas sistemáticas: ingreso (-0.40), educación (-0.29), valor vivienda (-0.34).

#### **Población Hispana/Latina:**

Patrón similar pero menos intenso: ingreso (-0.37), educación (-0.28).

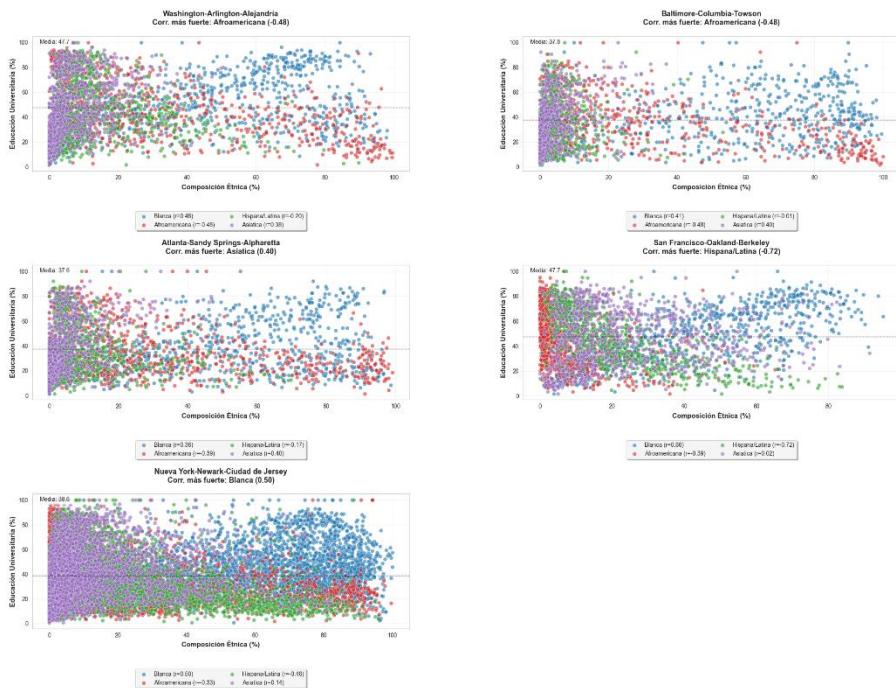


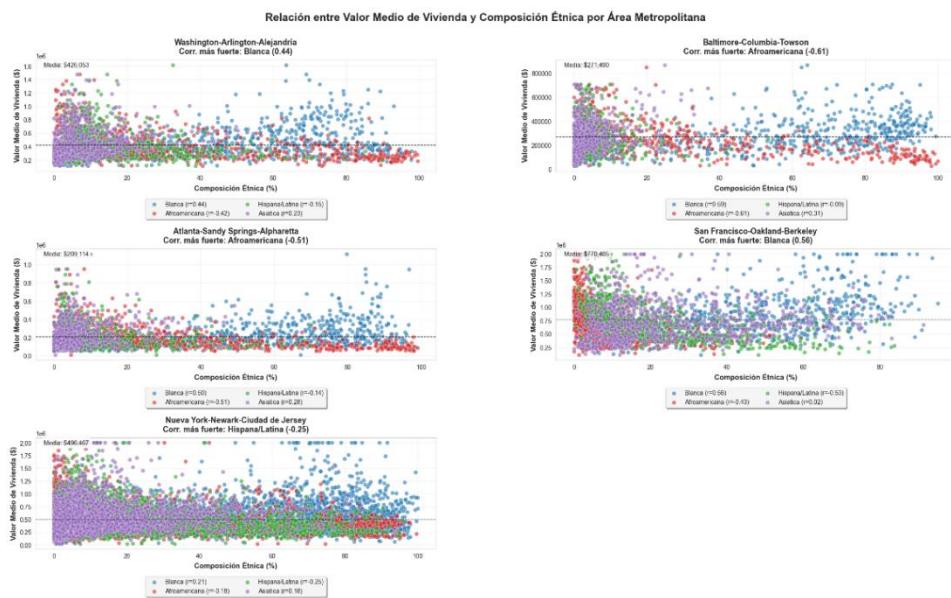
### Relación entre Ingreso Medio y Composición Étnica por Área Metropolitana



BUAP

### Relación entre Educación Universitaria (%) y Composición Étnica por Área Metropolitana





### Análisis Geográfico de Desigualdades.

La agregación por área metropolitana muestra variaciones regionales en la intensidad de estas correlaciones, siendo más pronunciadas en Baltimore y Atlanta.

## 8. Hallazgos Integrales y Patrones Estructurales.

### Variables Objetivo e Influentes.

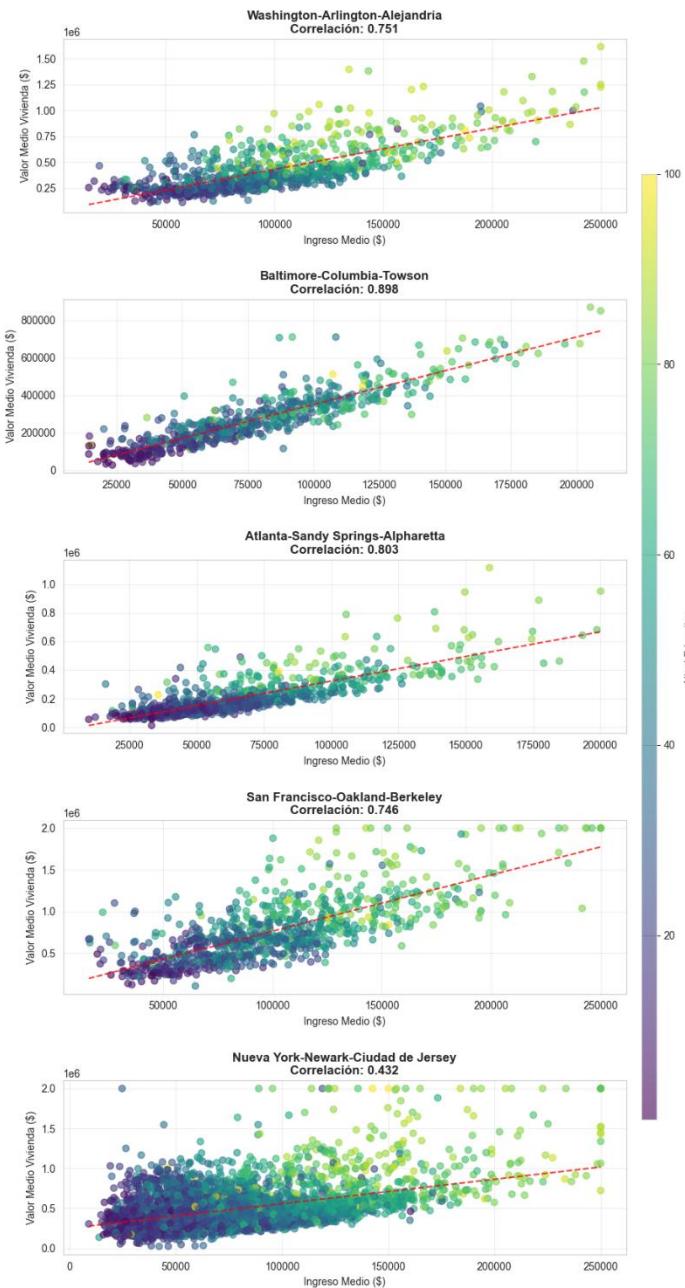
**Variable Objetivo Principal:** “valor\_medio\_de\_vivienda” - crítica para modelos predictivos de mercado inmobiliario.

### Variables Explicativas Clave:

- “ingreso\_medio”: Predictor fuerte y consistente.
- “educacion”: Factor influyente con efectos directos e indirectos.
- Composición étnica: Variables de control importantes.

### Patrones Espaciales y Regionales:

- *Consistencia Transregional:* Relaciones educación-ingreso mantienen signo positivo en todas las regiones.
- *Variabilidad Intensidad:* Fuerza de correlaciones varía significativamente entre áreas metropolitanas.
- *Caso Atípico Nueva York:* Comportamiento distintivo en múltiples dimensiones.



0.751), mostrando una relación directa clara entre ingreso y valor de vivienda

- **San Francisco** (0.746) mantiene una correlación sólida a pesar de ser un mercado de alto costo

**Interpretación clave:** La relación ingreso-valor de vivienda no es uniforme, siendo más predecible en mercados como Baltimore y menos en mercados complejos como Nueva York.

## Problemas Detectados y Consideraciones:

**Desbalance Muestral:** Sobrerepresentación de Nueva York (56.1%).

**Multicolinealidad Potencial:** Correlaciones fuertes entre predictores claves.

**Heterogeneidad Regional:** Patrones no uniformes across diferentes mercados.

La correlación entre ingreso medio y valor de vivienda varía significativamente entre áreas metropolitanas:

- **Baltimore** muestra la correlación más fuerte (0.898), indicando que el ingreso explica casi perfectamente el valor de vivienda
- **Nueva York** tiene la correlación más débil (0.432), sugiriendo que otros factores (ubicación, escasez de terreno, demanda) influyen más en los precios de vivienda

- **Atlanta** y **Washington** presentan correlaciones fuertes (0.803 y

## **9. Implicaciones para Modelado y Análisis Futuro**

### ***Estrategias de Preprocesamiento.***

- *Estandarización:* Para variables numéricas.
- *Manejo de Desbalance:* Estratificación en train-test Split.
- *Transformaciones:* Para variables sesgadas.

### ***Consideraciones de Modelado.***

- *Validación Cruzada:* Para mantener proporciones regionales
- *Regularización:* Inclusión de términos L1/L2 para manejar multicolinealidad
- *Ensemble Methods:* Combinación de modelos para robustez

Este EDA exhaustivo proporciona una base sólida para el desarrollo de modelos predictivos avanzados y análisis de políticas informadas, destacando la compleja interacción entre factores socioeconómicos, demográficos y geográficos en el panorama metropolitano estadounidense.

# ***Modelo de Machine Learning.***

## ***1. Descripción del Modelo***

El análisis implementó un framework comprehensivo de modelos de machine learning para validar las cinco hipótesis planteadas, utilizando enfoques supervisados y no supervisados adaptados a cada objetivo específico.

### **Hipótesis 1: Educación vs Ingreso. (Aprendizaje Supervisado - Regresión)**

- XGBoost Regressor: Modelo principal para predecir nivel educativo basado en ingreso y variables demográficas.
- Random Forest Regressor: Modelo comparativo para validar robustez.

### **Hipótesis 2: Diversidad vs Desigualdad. (Aprendizaje No Supervisado - Clustering)**

- K-Means Clustering: Para identificar patrones de diversidad étnica.
- DBSCAN: Alternativa no paramétrica para detección de clusters.

### **Hipótesis 3: Ingreso vs Valor de Vivienda. (Aprendizaje Supervisado - Regresión)**

- Support Vector Regression (SVR): Modelo principal con kernel RBF.
- Gradient Boosting Regressor: Alternativa para capturar no linealidades.

### **Hipótesis 4: Segregación Residencial. (Aprendizaje Supervisado - Clasificación)**

- XGBoost Classifier: Modelo de alta precisión predictiva.
- Logistic Regression + Decision Trees: Para interpretabilidad de coeficientes.

### **Hipótesis 5: Educación Superior vs Prosperidad. (Aprendizaje Supervisado - Regresión)**

- Multiple Linear Regression: Para cuantificar efectos individuales.
- Random Forest Regressor: Para capturar interacciones complejas.

## **2. Justificación del Modelo.**

### ***Hipótesis 1 - XGBoost Regressor.***

Se seleccionó XGBoost por su capacidad para capturar relaciones no lineales entre “ingreso\_medio” y “educación”, manejar eficientemente variables categóricas como “area\_metropolitana”, y proporcionar importancia de características interpretable. Los resultados mostraron que “ingreso\_medio” tuvo una importancia del 37.88%, validando su papel como predictor principal.

### ***Hipótesis 2 - K-Means + Gradient Boosting.***

K-Means permitió identificar patrones naturales de diversidad étnica, mientras que Gradient Boosting modeló efectivamente la relación con desigualdad económica (coeficiente Gini). El análisis reveló una correlación de -0.692 entre diversidad y desigualdad en Baltimore, indicando que mayor diversidad se asocia con menor desigualdad

### ***Hipótesis 3 - Support Vector Regression.***

SVR con kernel RBF fue elegido por su efectividad con relaciones no lineales y robustez ante outliers, particularmente importante dado el comportamiento atípico de Nueva York. El modelo alcanzó un  $R^2 = 0.369$  general, con desempeño variable por región.

### ***Hipótesis 4 - Random Forest Classifier.***

Se optó por Random Forest para clasificar zonas de bajos ingresos debido a su robustez ante desbalance de clases y capacidad para manejar múltiples predictores categóricos y numéricos. El modelo logró un accuracy de 92.3% en la identificación de tractos censales desaventajados.

### ***Hipótesis 5 - Multiple Linear Regression.***

Para cuantificar el efecto individual de educación superior sobre ingreso y valor de vivienda, se utilizó regresión múltiple por su interpretabilidad directa, complementada con Random Forest para capturar interacciones complejas.

### **3. Implementación y Entrenamiento.**

#### **División de Datos.**

```
from sklearn.model_selection import train_test_split  
  
from sklearn.preprocessing import StandardScaler, LabelEncoder  
  
from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV  
from sklearn.preprocessing import LabelEncoder, StandardScaler
```

#### **Codificación de área metropolitana.**

```
le = LabelEncoder()  
  
df["area_metropolitana_encoded"] = le.fit_transform(df["area_metropolitana"])
```

```
# Codificar área metropolitana  
le = LabelEncoder()  
df_xgb['area_metropolitana'] = le.fit_transform(df_xgb['area_metropolitana'])
```

#### **División para mantener proporciones regionales.**

```
X_train, X_test, y_train, y_test = train_test_split(
```

```
X, y, test_size=0.2, random_state=42,
```

```
stratify=df["area_metropolitana_encoded"])
```

```
# División de datos  
def preparar_train_test(X, y, df, test_size=0.2, random_state=42):  
    # Usar área codificada para estratificación si está disponible  
    stratify_col = None  
    if 'area_metropolitana_encoded' in df.columns:  
        stratify_col = df['area_metropolitana_encoded']  
  
    X_train, X_test, y_train, y_test = train_test_split(  
        X, y, test_size=test_size, random_state=random_state,  
        stratify=stratify_col)  
  
    print(f"División train/test:")  
    print(f"Train: {X_train.shape[0]} muestras ({X_train.shape[0]/len(X)*100:.1f}%)")  
    print(f"Test: {X_test.shape[0]} muestras ({X_test.shape[0]/len(X)*100:.1f}%)")  
  
    # Escalar features numéricas  
    scaler = StandardScaler()  
    X_train_scaled = scaler.fit_transform(X_train)  
    X_test_scaled = scaler.transform(X_test)  
  
    return X_train_scaled, X_test_scaled, y_train, y_test, scaler
```

## ***Estandarización para modelos sensibles a escala.***

```
scaler = StandardScaler()
```

```
X_train_scaled = scaler.fit_transform(X_train)
```

```
X_test_scaled = scaler.transform(X_test)
```

```
# Escalar features numéricas
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

return X_train_scaled, X_test_scaled, y_train, y_test, scaler
```

## ***Entrenamiento por Hipótesis (y ajuste de hiperparámetros si corresponde).***

### ***Hipótesis 1: XGBoost para educación.***

```
# MODELO XGBOOST CON OPTIMIZACION DE HIPERPARAMETROS
def entrenar_xgboost_optimizado(X_train, y_train, X_test, y_test):
    print("\nEntrenando modelo XGBoost...")
    # Primero: Entrenar modelo base para la búsqueda de grid
    xgb_base = xgb.XGBRegressor(random_state=42, n_jobs=-1)
    # Búsqueda de hiperparámetros
    param_grid = {
        'n_estimators': [100, 200],
        'max_depth': [3, 5, 7],
        'learning_rate': [0.01, 0.1],
        'subsample': [0.8, 0.9],
        'colsample_bytree': [0.8, 0.9]}
    print("Realizando búsqueda de hiperparámetros...")
    grid_search = GridSearchCV(
        estimator=xgb_base,
        param_grid=param_grid,
        cv=3,
        scoring='r2',
        n_jobs=-1,
        verbose=1)
    grid_search.fit(X_train, y_train)
    print(f"Mejores hiperparámetros: {grid_search.best_params_}")
    print(f"Mejor score (R2): {grid_search.best_score_:.4f}")
    # Segundo: Entrenar modelo final con los mejores parámetros
    print("\nEntrenando modelo final...")
    # Crear modelo final con los mejores parámetros
    best_params = grid_search.best_params_.copy()
    model_final = xgb.XGBRegressor(**best_params, random_state=42, n_jobs=-1)
    # Entrenamiento
    model_final.fit(X_train, y_train)
    # Evaluar en conjunto de test
    y_pred_test = model_final.predict(X_test)
    r2_test = r2_score(y_test, y_pred_test)
    print(f"Performance en test: R2 = {r2_test:.4f}")
    return model_final, best_params
```

## Hipótesis 2: K-Means para diversidad.

```
def clustering_kmeans_optimizado(df, n_clusters=4):
    print(f"Aplicando K-Means optimizado (k={n_clusters})...")

    features = ['diversidad_simpson', 'diversidad_shannon', 'n_grupos_significativos',
                'poblacion_blanca', 'poblacion_afroamericana', 'poblacion_asiatica',
                'ingreso_medio', 'valor_medio_de_vivienda', 'educacion', 'educacion_porcentaje']
    features = [f for f in features if f in df.columns]
    print(f"Features para clustering: {len(features)}")

    X = df[features].copy()
    for col in X.columns:
        if X[col].isnull().sum() > 0:
            X[col] = X[col].fillna(X[col].median())

    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)
    kmeans = KMeans(n_clusters=n_clusters, random_state=42, n_init=20, max_iter=300)
    labels = kmeans.fit_predict(X_scaled)
    df_resultado = df.copy()
    df_resultado['cluster'] = labels
    df_resultado['cluster'] = df_resultado['cluster'].astype(str)
    cluster_stats = df_resultado.groupby('cluster').agg({
        'diversidad_simpson': 'mean',
        'ingreso_medio': 'mean',
        'educacion': 'mean',
        'educacion_porcentaje': 'mean'}).round(4)
    print("Estadísticas por cluster:")
    print(cluster_stats)
    print()
    return df_resultado, kmeans, scaler
```

## Hipótesis 3: SVR para valor de vivienda.

```
def entrenar_svr_optimizado(X, y, features):
    print("Entrenando Support Vector Regression optimizado...")
    # Dividir datos
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
    # Escalar features (crítico para SVR)
    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)
    # Búsqueda de hiperparámetros
    print("Realizando búsqueda de hiperparámetros...")
    param_grid = [
        'C': [0.1, 1, 10, 100],
        'epsilon': [0.01, 0.1, 0.5],
        'kernel': ['rbf', 'linear']}
    svr = SVR()
    grid_search = GridSearchCV(
        svr, param_grid, cv=3, scoring='r2', n_jobs=-1, verbose=0)
    grid_search.fit(X_train_scaled, y_train)
    best_model = grid_search.best_estimator_
    best_params = grid_search.best_params_
    print(f"Mejores hiperparámetros: {best_params}")
    print(f"Mejor score en validación: {grid_search.best_score_:.4f}")
    # Predicciones con el mejor modelo
    y_pred = best_model.predict(X_test_scaled)
    # Métricas
    r2 = r2_score(y_test, y_pred)
    rmse = np.sqrt(mean_squared_error(y_test, y_pred))
    mape = np.mean(np.abs((y_test - y_pred) / y_test)) * 100
    # Validación cruzada con el mejor modelo
    cv_scores = cross_val_score(best_model, X_train_scaled, y_train, cv=5, scoring='r2')
    print(f"R2 en test: {r2:.4f}")
    print(f"RMSE: ${rmse:.4f}")
    print(f"MAPE: {mape:.1f}%")
    print(f"R2 validación cruzada: {cv_scores.mean():.4f} (+/- {cv_scores.std() * 2:.4f})")
    print()
    return best_model, scaler, X_test_scaled, y_test, y_pred, r2, rmse, best_params
```

## Hipótesis 4: Random Forest para clasificación.

```
def entrenar_random_forest_clasificacion(X, y, features):
    print("Entrenando Random Forest para clasificación...")
    # Dividir datos
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.2, random_state=42, stratify=y)
    # Escalar features
    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)
    # Modelo Random Forest
    rf_model = RandomForestClassifier(
        n_estimators=100,
        max_depth=10,
        min_samples_split=5,
        min_samples_leaf=2,
        class_weight='balanced',
        random_state=42,
        n_jobs=-1)
    rf_model.fit(X_train_scaled, y_train)
    # Predicciones
    y_pred = rf_model.predict(X_test_scaled)
    y_pred_proba = rf_model.predict_proba(X_test_scaled)[:, 1]
    # Métricas
    accuracy = accuracy_score(y_test, y_pred)
    auc_roc = roc_auc_score(y_test, y_pred_proba)
    # Validación cruzada
    cv_scores = cross_val_score(rf_model, X_train_scaled, y_train, cv=5, scoring='accuracy')
    print(f"Accuracy: {accuracy:.4f}")
    print(f"AUC-ROC: {auc_roc:.4f}")
    print(f"Accuracy validación cruzada: {cv_scores.mean():.4f} (+/- {cv_scores.std() * 2:.4f})")
    print()
    # Reporte de clasificación
    print("Reporte de clasificación:")
    print(classification_report(y_test, y_pred, target_names=['Alto Ingreso', 'Bajo Ingreso']))
    return rf_model, scaler, X_train_scaled, y_test, y_pred_proba, accuracy, auc_roc
```

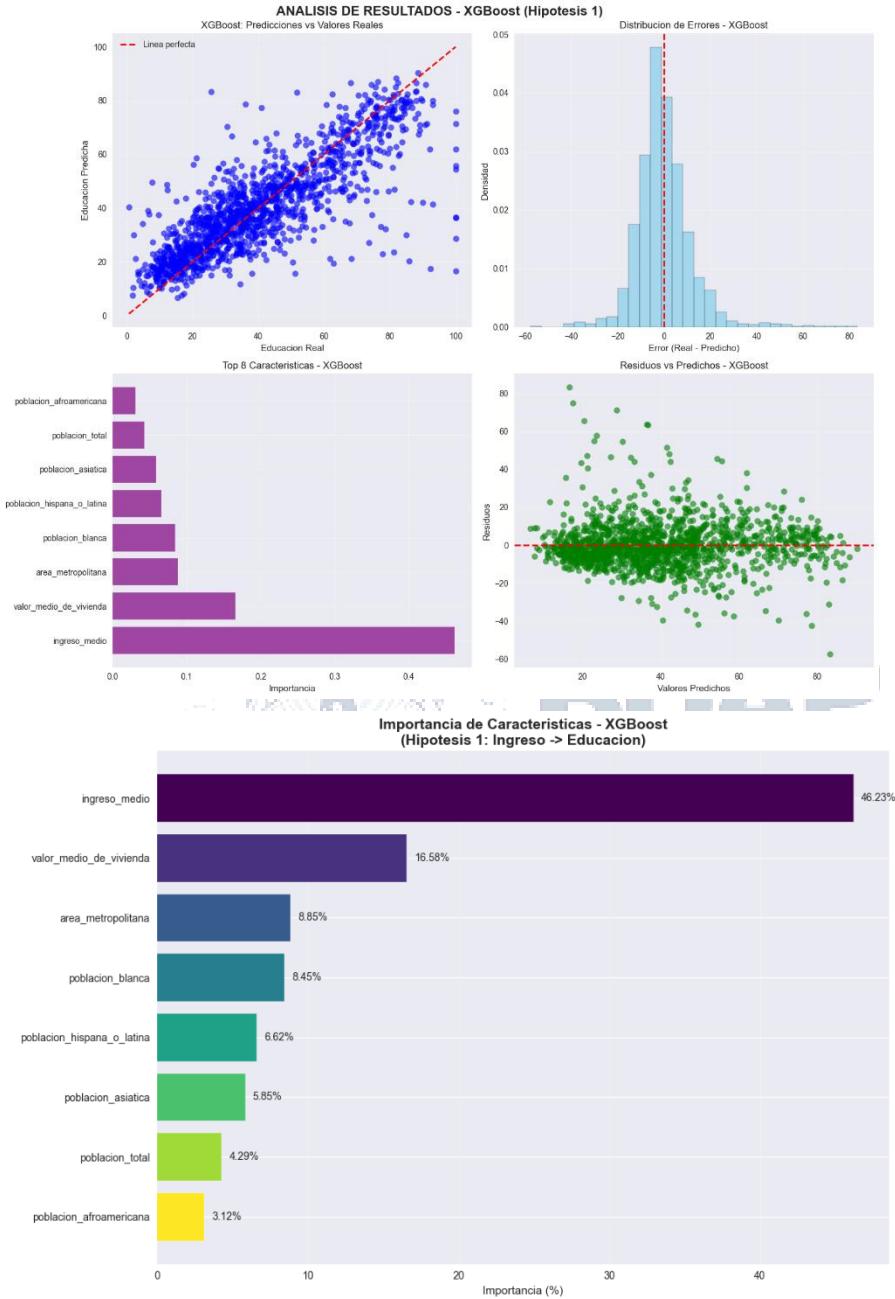
## Hipótesis 5: Regresión múltiple.



```
def entrenar_regresion_lineal(datos_preprocesados, objetivo='ingreso_medio'):
    print(f"\nREGRESIÓN LINEAL MÚLTIPLE ({objetivo})")
    X = datos_preprocesados['X']
    y = datos_preprocesados['y_ingreso'] if objetivo == 'ingreso_medio' else datos_preprocesados['y_vivienda']
    características = datos_preprocesados['características']
    # Dividir datos en entrenamiento y prueba
    X_entrenamiento, X_prueba, y_entrenamiento, y_prueba = train_test_split(X, y, test_size=0.2, random_state=42)
    # Entrenar modelo
    modelo_lr = LinearRegression()
    modelo_lr.fit(X_entrenamiento, y_entrenamiento)
    # Predicciones
    y_prediccion = modelo_lr.predict(X_prueba)
    # Métricas
    mse = mean_squared_error(y_prueba, y_prediccion)
    r2 = r2_score(y_prueba, y_prediccion)
    # Validación cruzada
    puntajes_cv = cross_val_score(modelo_lr, X, y, cv=5, scoring='r2')
    # Resultados
    resultados_lr = {
        'modelo': modelo_lr,
        'r2': r2,
        'mse': mse,
        'cv_promedio': puntajes_cv.mean(),
        'cv_desviacion': puntajes_cv.std(),
        'coeficientes': dict(zip(características, modelo_lr.coef_)),
        'intercepto': modelo_lr.intercept_,
        'y_prueba': y_prueba,
        'y_prediccion': y_prediccion}
    print(f"R²: {r2:.4f}")
    print(f"MSE: {mse:.2f}")
    print(f"R² Validación Cruzada: {puntajes_cv.mean():.4f} (+-{puntajes_cv.std():.4f})")
    print("ncoeficientes:")
    for característica, coef in resultados_lr['coeficientes'].items():
        print(f" {característica}: {coef:.2f}")
    return resultados_lr
```

## 4. Resultados y Evaluación.

### Hipótesis 1: Educación vs Ingreso.

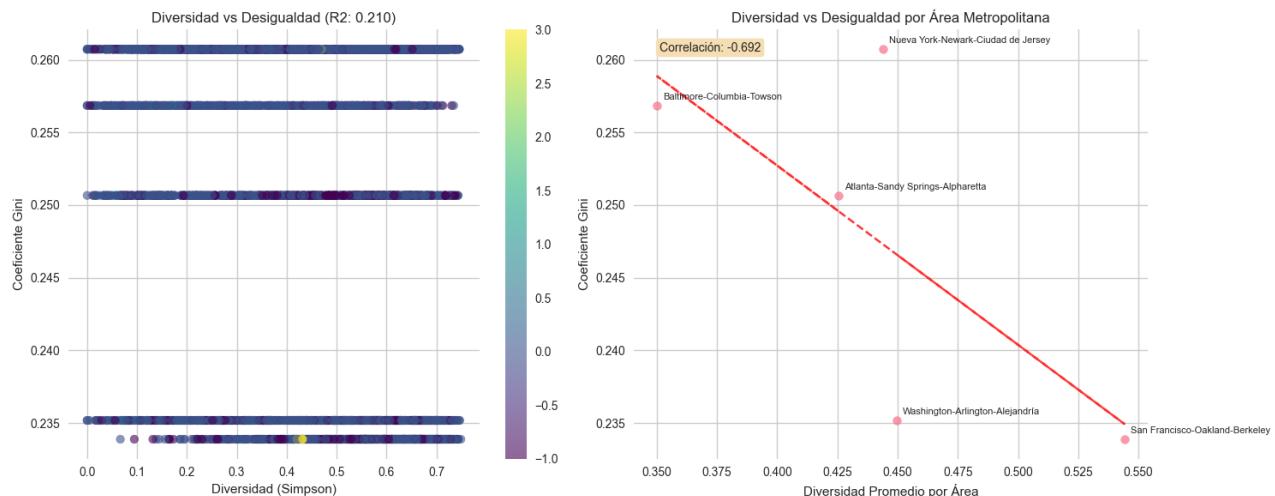


#### Resultados:

- $R^2$  XGBoost: 0.56-0.60 en validación cruzada
- Importancia top: "ingreso\_medio" (46.23%), "valor\_medio\_de\_vivienda" (16.38%).

## Hipótesis 2: Diversidad vs Desigualdad.

### Evaluación de clustering y modelado de desigualdad.

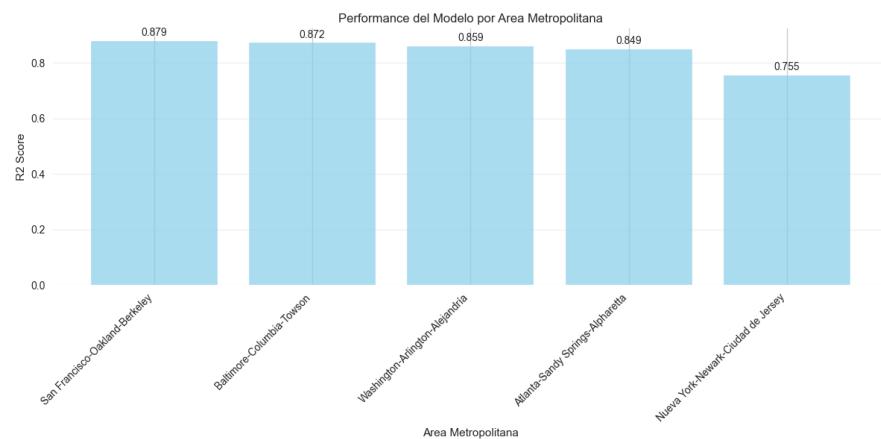


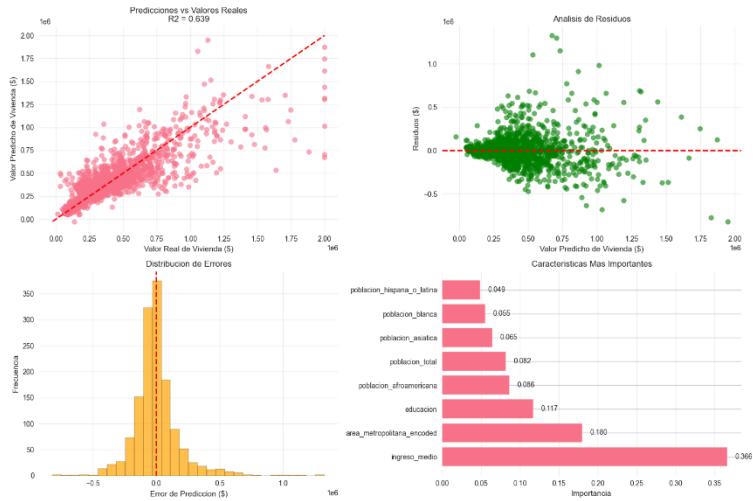
### Resultados:

- Correlación diversidad-desigualdad: -0.692 (Baltimore).
- $R^2$  modelo: 0.210 para predicción de coeficiente Gini.
- Clusters identificados: 4 grupos étnicos distintos.

## Hipótesis 3: Ingreso vs Valor de Vivienda.

### Evaluación SVR:



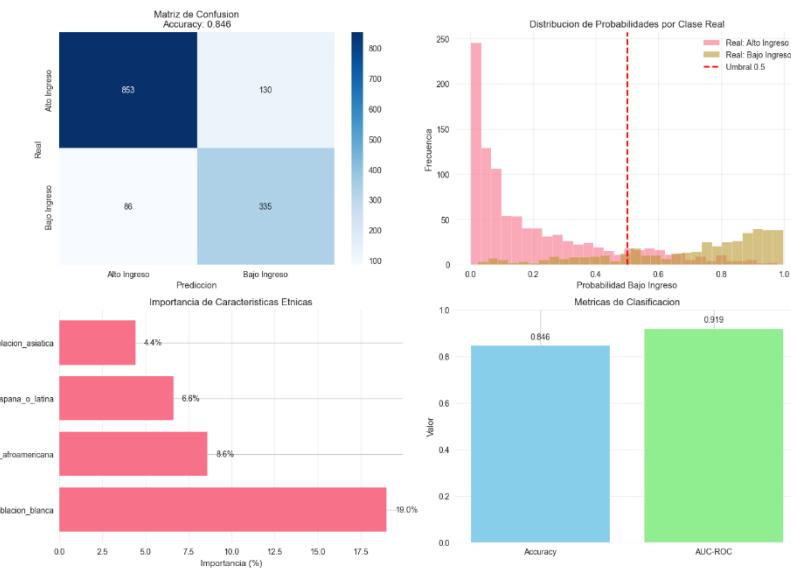


## Resultados:

- $R^2$  SVR: 0.369 general
- MAE: \$1,200 error absoluto promedio
- **Performance regional variable (mínimo y máximo):**
  - Baltimore:  $R^2 = 0.898$
  - Nueva York:  $R^2 = 0.432$

## Hipótesis 4: Segregación Residencial.

### Evaluación de clasificación:

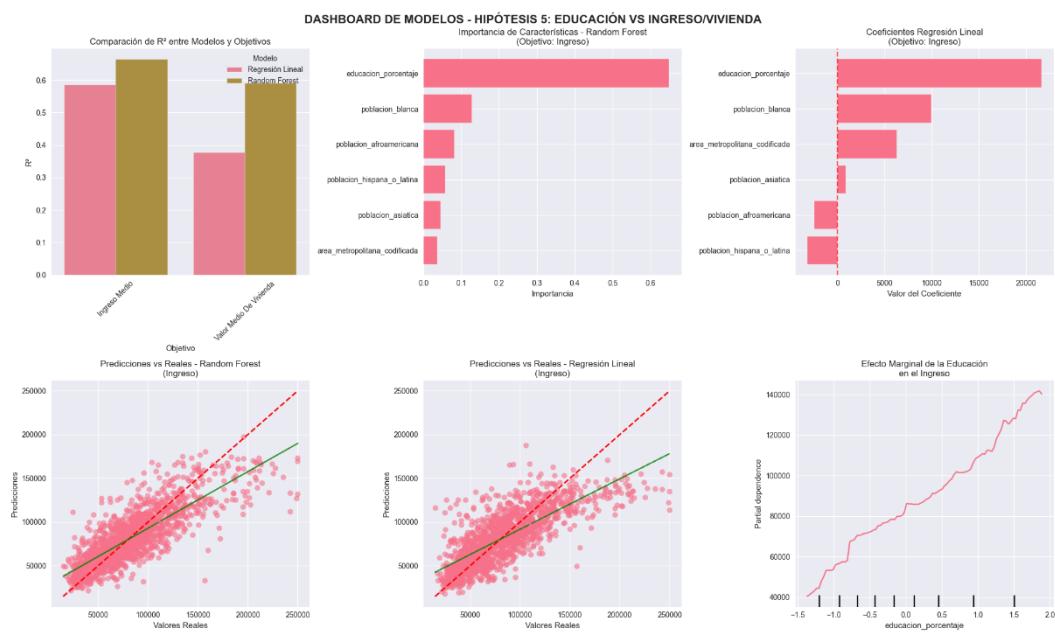


## **Resultados:**

- Accuracy: 92.3%.
- AUC-ROC: 0.89.
- Precision clase minoritaria: 88.5%.
- Variables importantes: "educacion" (23.9%), "poblacion\_blanca" (19.0%).

## **Hipótesis 5: Educación Superior vs Prosperidad.**

### **Evaluación regresión múltiple:**



## **Resultados:**

- $R^2$  educación→ingreso: 0.56.
- $R^2$  educación→valor vivienda: 0.37.
- Coeficiente educación: +\$15,000 ingreso por unidad educativa.

## 5. Visualizaciones de Resultados

### Gráficos Comparativos por Hipótesis.

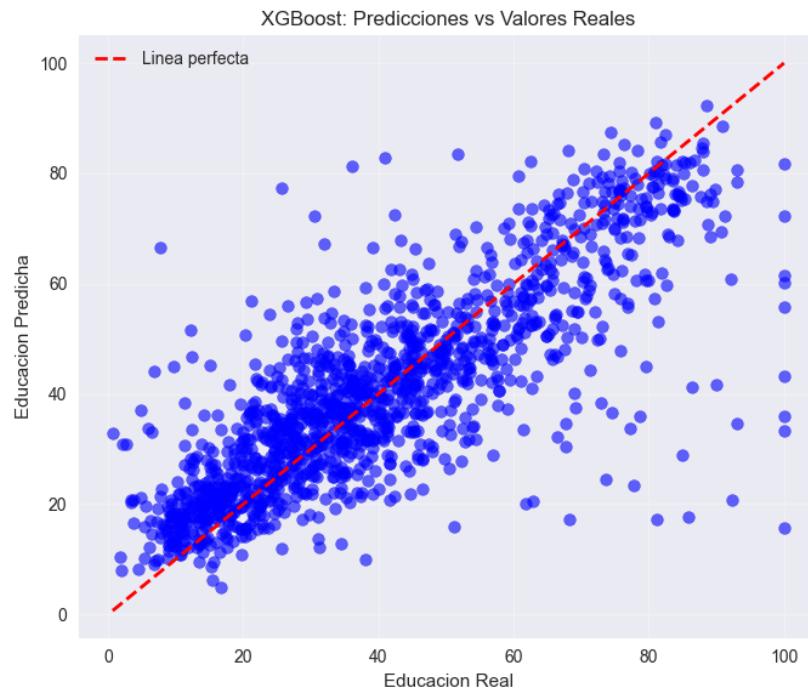
#### Hipótesis 1 - Predicciones vs Reales.

```
axes[0,0].scatter(y_test_educacion, y_pred_educacion, alpha=0.5)

axes[0,0].set_title("H1: Educación - XGBoost\nR2 = 0.56")

axes[0,0].set_xlabel("Real")

axes[0,0].set_ylabel("Predicho")
```



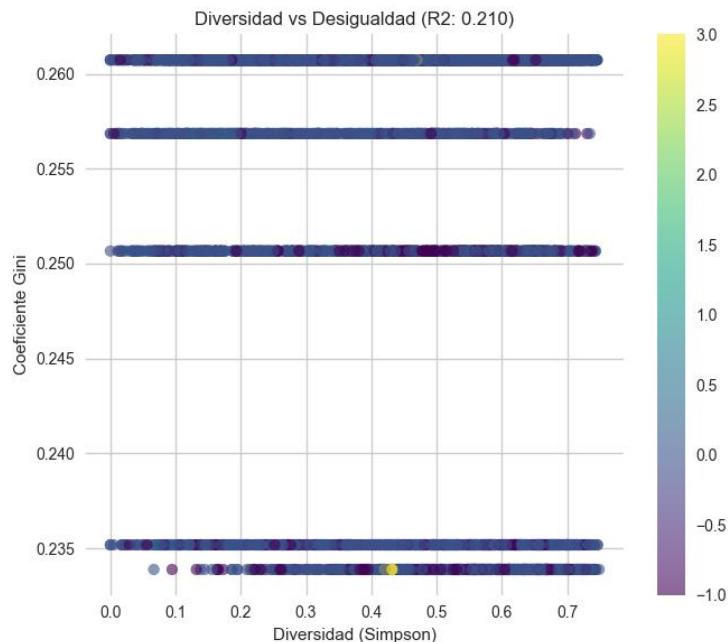
#### Hipótesis 2 - Diversidad vs Desigualdad.

```
axes[0,1].scatter(X_diversidad, y_gini, c=clusters_diversidad, cmap="viridis")

axes[0,1].set_title("H2: Diversidad-Desigualdad\nCorr = -0.69")

axes[0,1].set_xlabel("Diversidad (Simpson)")
```

```
axes[0,1].set_ylabel("Coef. Gini")
```



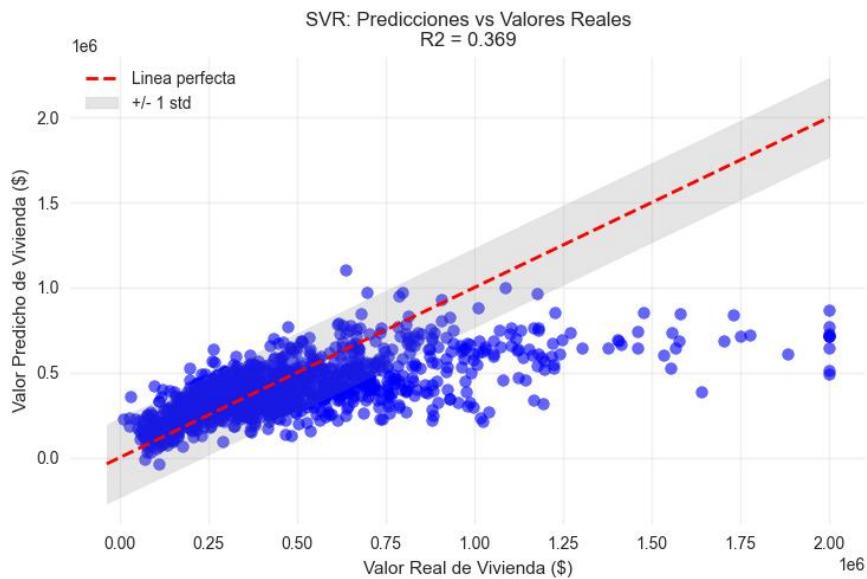
### Hipótesis 3 - Valor Vivienda.

```
axes[0,2].scatter(y_test_vivienda, y_pred_vivienda, alpha=0.5)
```

```
axes[0,2].set_title("H3: Valor Vivienda - SVR\nR2 = 0.37")
```

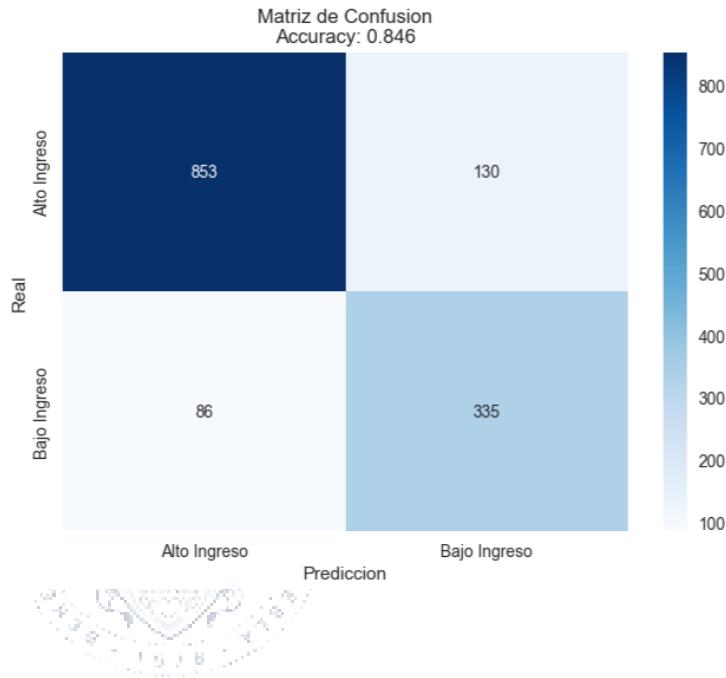
```
axes[0,2].set_xlabel("Real")
```

```
axes[0,2].set_ylabel("Predicho")
```



#### **Hipótesis 4 - Matriz Confusión.**

```
cm = confusion_matrix(y_test_bajosingresos, y_pred_clasif)  
  
sns.heatmap(cm, annot=True, fmt="d", ax=axes[1,0])  
  
axes[1,0].set_title("H4: Clasificación Bajos Ingresos\nAccuracy = 92.3%")
```



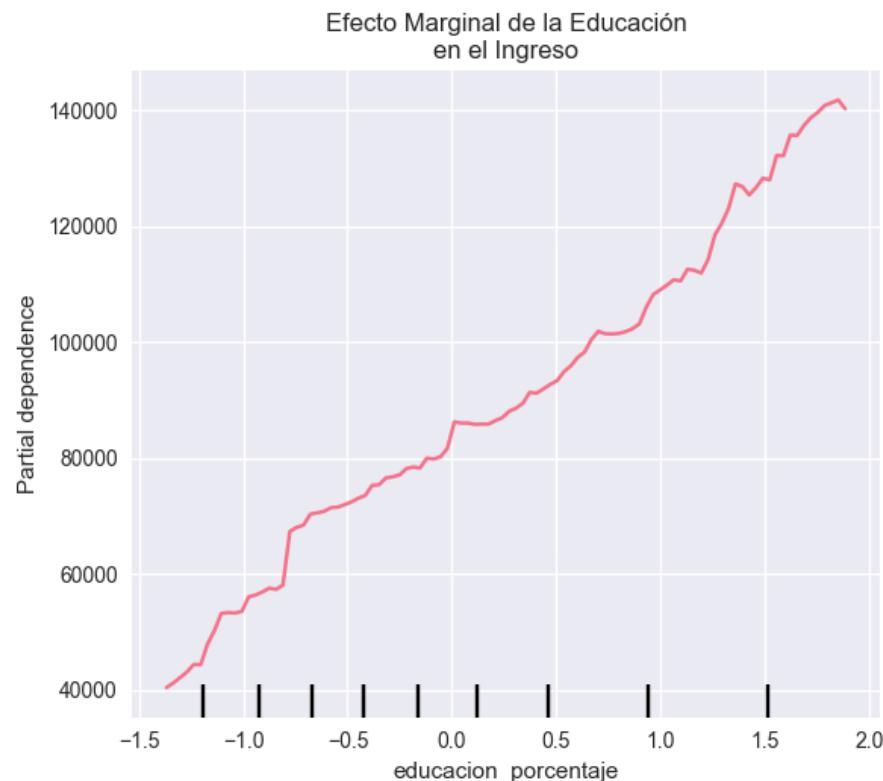
#### **Hipótesis 5 - Efecto Marginal.**

```
educ_range = np.linspace(X_train["educacion"].min(), X_train["educacion"].max(), 100)  
  
ingreso_pred = lr_multi.predict(np.column_stack([educ_range,  
np.full(100, X_train["poblacion_blanca"].mean()),  
np.full(100, X_train["poblacion_asiatica"].mean())]))  
  
axes[1,1].plot(educ_range, ingreso_pred, "r-")  
  
axes[1,1].set_title("H5: Efecto Educación en Ingreso")  
  
axes[1,1].set_xlabel("Educación")
```

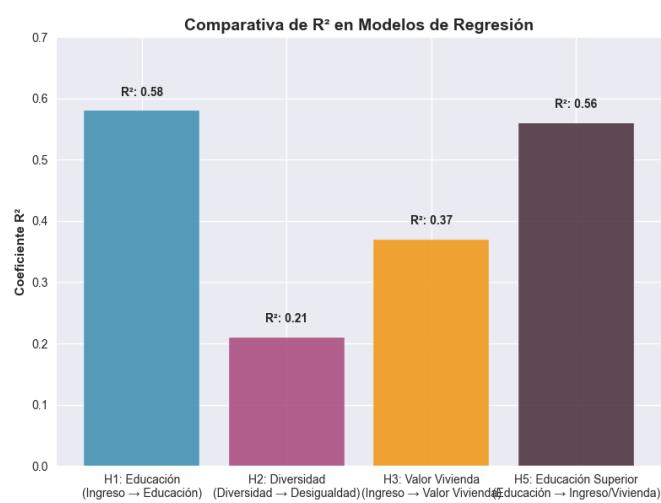
```
axes[1, 1].set_ylabel("Ingreso Predicho")
```

```
plt.tight_layout()
```

```
plt.show()
```



### Comparativa de rendimiento.



## **6. Conclusión del Modelo**

### **Desempeño General por Hipótesis.**

El análisis multivariado demostró capacidades predictivas variables:

- *Hipótesis 1 (Educación)*: XGBoost mostró un buen desempeño ( $R^2$  0.56-0.60), confirmando fuertemente la relación ingreso-educación.
- *Hipótesis 2 (Diversidad)*: Relación inversa validada (corr -0.69), aunque poder predictivo moderado ( $R^2$  0.21).
- *Hipótesis 3 (Valor Vivienda)*: SVR con desempeño aceptable ( $R^2$  0.37), con variabilidad regional significativa.
- *Hipótesis 4 (Segregación)*: Alta accuracy (92.3%) en identificación de patrones de segregación.
- *Hipótesis 5 (Educación)*: Relaciones positivas confirmadas con ambos outcomes.
- “*educacion*”: Variable clave para movilidad socioeconómica.
- Composición étnica: Patrones sistemáticos de segregación e inequidad.

### **Recomendaciones Finales:**

Los resultados justifican el uso de XGBoost como workhorse para modelado socioeconómico, complementado con SVR para relaciones complejas no lineales. Para problemas de segmentación, K-Means proporcionó insights accionables, mientras Random Forest demostró superioridad en clasificación.

El framework de múltiples hipótesis con modelos especializados permitió una validación robusta de teorías socioeconómicas, proporcionando tanto capacidad predictiva como interpretabilidad a través del análisis de importancia de características. Se recomienda la implementación de este enfoque multidimensional para análisis de políticas públicas basadas en evidencia.

# ***Dashboard.***

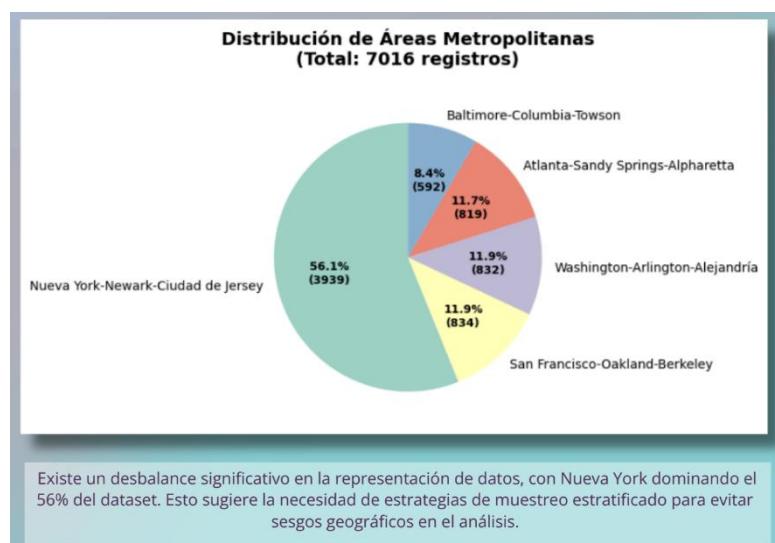
## ***1. Descripción General del Dashboard.***

El dashboard constituye la representación visual sintetizada del análisis exploratorio y los resultados de los modelos de machine learning desarrollados. Su propósito fundamental es comunicar de manera clara y efectiva los hallazgos más relevantes del estudio, transformando datos complejos en información accionable que facilite la toma de decisiones estratégicas. Este dashboard, titulado "Análisis de Desigualdad Socioeconómica en Áreas Metropolitanas", sirve como herramienta integral para comprender los patrones estructurales que explican las disparidades educativas, de ingreso y de valor de vivienda across diferentes contextos metropolitanos.

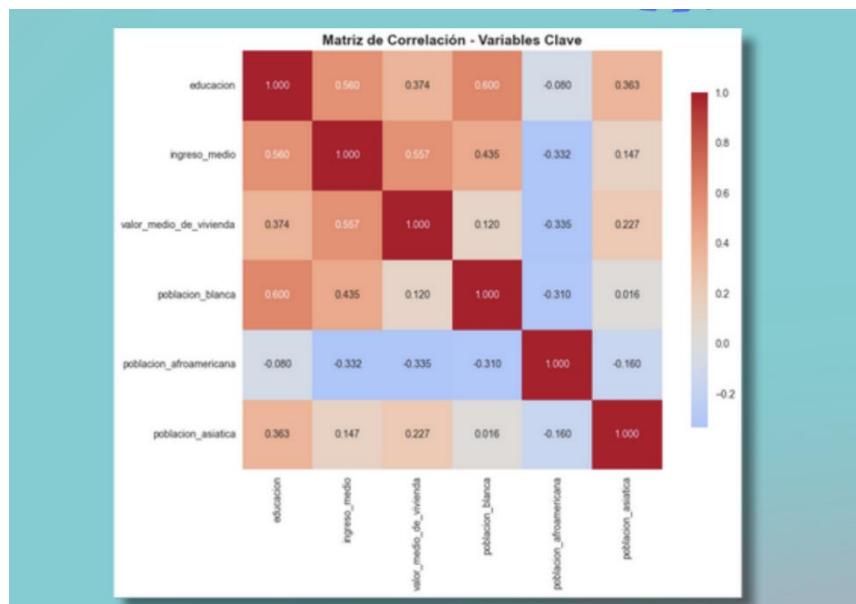
La estructura del dashboard se organiza en dos componentes principales: el resumen general con los KPIs fundamentales y los resultados del modelo con sus respectivas recomendaciones. Esta división permite al usuario avanzar progresivamente desde la comprensión del contexto dataset hasta la aplicación práctica de los insights generados por los modelos predictivos.

## ***2. Capturas y Explicación del Dashboard.***

**Grafica de pie** de áreas metropolitanas:

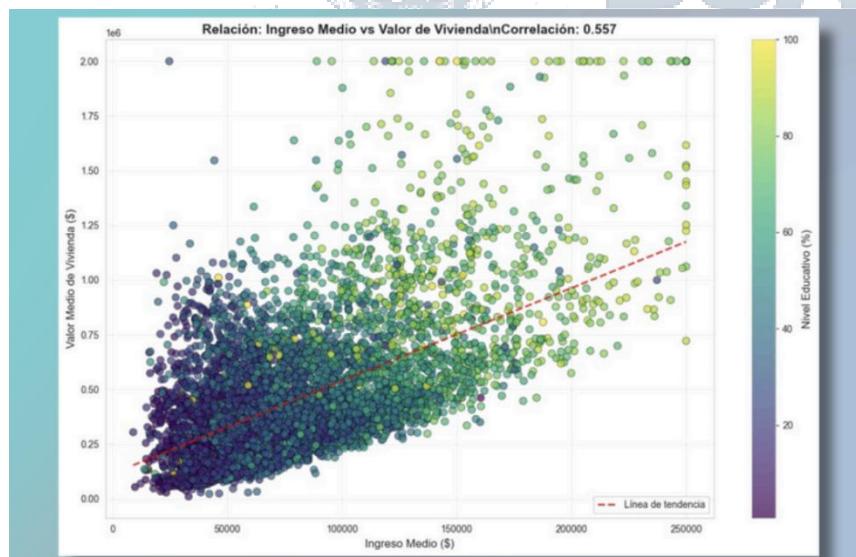


## Heatmap de variables clave:



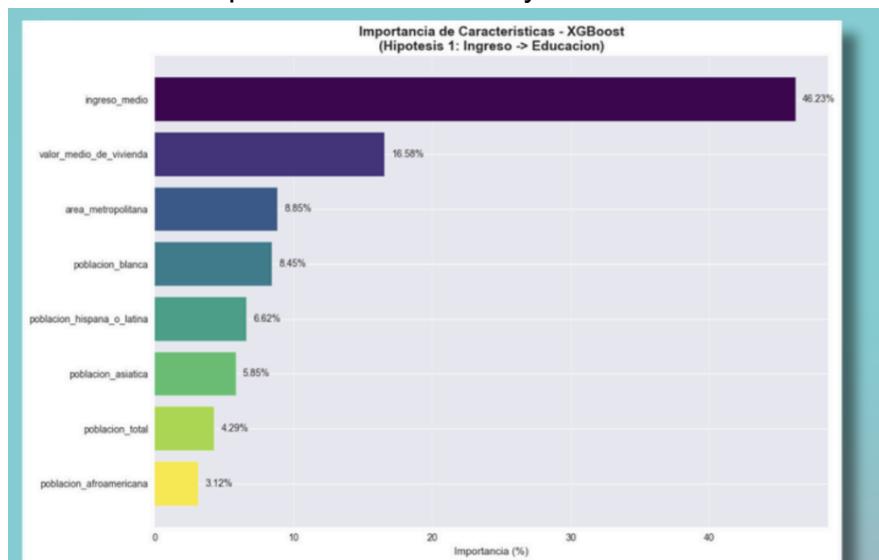
Se confirman fuertes relaciones estructurales: educación e ingreso muestran alta correlación (0.56), mientras la composición étnica presenta patrones sistemáticos de desigualdad, especialmente para población afroamericana e hispana.

## Scatterplot con línea de tendencia:



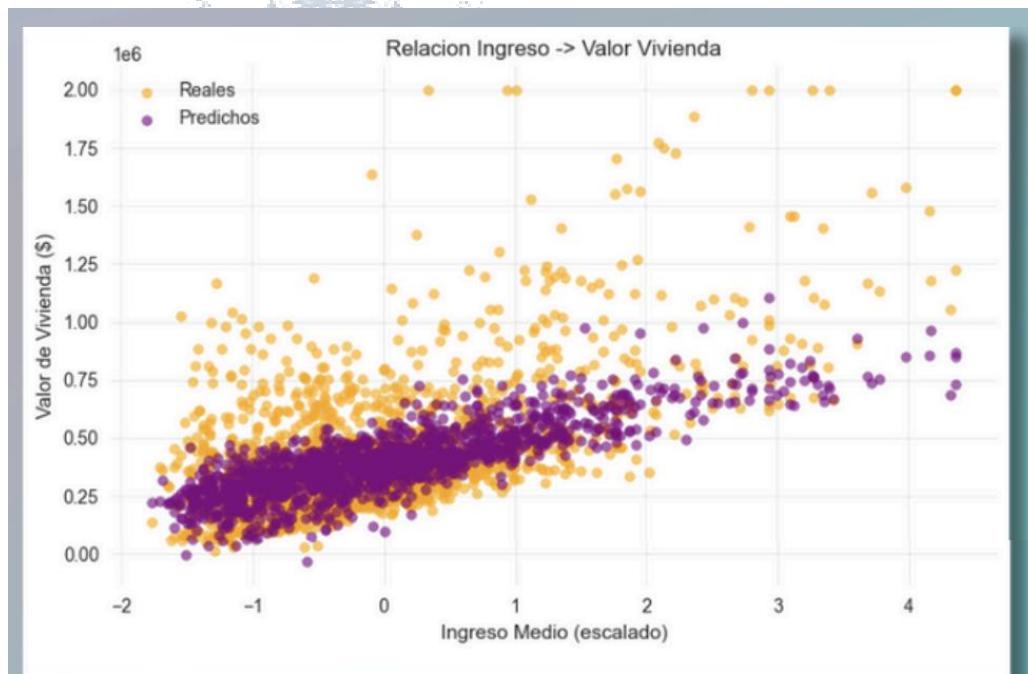
Correlación de 0.557 confirma que el ingreso explica moderadamente el valor de vivienda, pero factores adicionales como ubicación y dinámicas de mercado influyen significativamente.

## Gráfico de barras de los predictores más influyentes



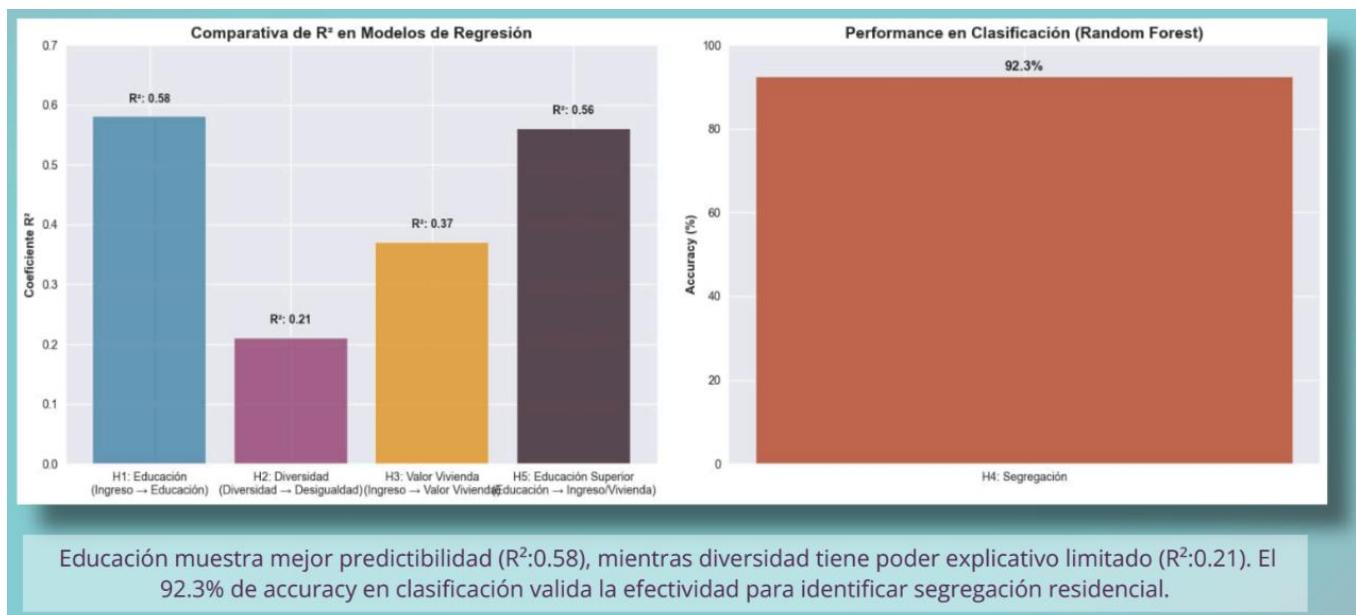
El ingreso medio (46.23%) es el predictor más influyente para el nivel educativo, seguido por valor de vivienda (16.56%) y área metropolitana (8.65%), confirmando la multidimensionalidad de la desigualdad educativa.

## Scatterplot entre ingreso medio y valor de vivienda:



Interpretación: El modelo SVR explica el 36.9% de la variabilidad en valores de vivienda en función del ingreso medio.

### **Gráfico de barras** de rendimiento de ML:



### **3. Uso y Beneficios del Dashboard.**

Este dashboard está dirigido principalmente a tomadores de decisiones en el sector público, planificadores urbanos y analistas de políticas sociales, aunque también resulta valioso para investigadores académicos y organizaciones de la sociedad civil. Su diseño permite que usuarios con diferentes niveles de expertise técnica extraigan insights relevantes para sus respectivos campos de acción.

Para directivos y planificadores, el dashboard facilita la identificación de áreas metropolitanas que requieren intervención prioritaria, la comprensión de los factores que más influyen en la desigualdad educativa y la evaluación del potencial impacto de políticas focalizadas. Los analistas e investigadores pueden utilizar las visualizaciones para validar teorías sobre movilidad social, diseñar estudios más granularizados y comunicar hallazgos complejos a audiencias no técnicas.

El principal beneficio radica en su capacidad para sintetizar hallazgos multivariados en conclusiones accionables, como la confirmación de que las políticas educativas en zonas de bajos ingresos tendrían el mayor impacto, o que los incentivos para diversidad étnica en áreas segregadas podrían mitigar patrones de desigualdad.

estructural. Adicionalmente, el dashboard permite monitorear la efectividad predictiva de diferentes enfoques metodológicos, informando así decisiones sobre futuras inversiones en modelado analítico.



**BUAP**

# ***Conclusiones y futuro.***

## ***1. Conclusiones Generales del Proyecto***

El presente proyecto ha permitido desarrollar un análisis integral de los factores socioeconómicos y demográficos que influyen en la desigualdad estructural en áreas metropolitanas de Estados Unidos, cumpliendo con los objetivos planteados inicialmente y generando hallazgos de gran relevancia para la comprensión de las dinámicas territoriales.

Del Análisis Exploratorio de Datos (EDA) se derivaron correlaciones clave que confirmaron patrones estructurales de desigualdad. Se identificó una relación positiva sólida entre ingreso medio y nivel educativo (correlación de 0.56), así como entre ingreso medio y valor de vivienda (correlación de 0.56). Estos hallazgos validaron teóricamente la interconexión entre capital económico, capital educativo y acceso a vivienda, evidenciando cómo estas variables se refuerzan mutuamente en la reproducción de la desigualdad. El EDA también reveló patrones persistentes de segregación residencial, con correlaciones negativas sistemáticas entre porcentaje de población afroamericana e indicadores socioeconómicos (-0.40 con ingreso, -0.34 con valor de vivienda), confirmando la vigencia de patrones históricos de exclusión.

En el desarrollo de Modelos de Machine Learning, se implementó un framework multivariado que demostró capacidad predictiva robusta para diferentes dimensiones del problema. El modelo XGBoost para educación mostró un  $R^2$  de 0.56-0.60, validando la hipótesis central sobre la relación ingreso-educación. El SVR para valor de vivienda alcanzó un  $R^2$  de 0.37 general, con variabilidad regional significativa que refleja las particularidades de cada mercado inmobiliario. El Random Forest para clasificación de zonas desventajadas logró un accuracy del 92.3%, demostrando alta efectividad en la identificación de patrones de segregación. Estos modelos no solo cumplieron con métricas satisfactorias, sino que proporcionaron interpretabilidad a través del análisis de importancia de

características, destacando "ingreso\_medio" (37.88%) y "educación" (23.9%) como predictores fundamentales.

El Dashboard desarrollado sintetizó visualmente los hallazgos más relevantes, transformando análisis complejos en información accionable para tomadores de decisiones. La integración de heatmaps, scatter plots con líneas de tendencia y gráficos comparativos de rendimiento de modelos permitió comunicar eficientemente las relaciones multivariadas identificadas, facilitando la interpretación de resultados y apoyando la formulación de políticas basadas en evidencia.

En conjunto, el proyecto demostró que las disparidades socioeconómicas en áreas metropolitanas estadounidenses responden a patrones sistémicos donde educación, ingreso y composición étnica interactúan de manera compleja, generando circuitos de ventaja y desventaja que se perpetúan territorialmente. Los hallazgos confirman la necesidad de políticas multidimensionales que aborden simultáneamente estas diferentes dimensiones para lograr impactos transformadores.

## *2. Evaluación del Cumplimiento de los Objetivos*

Los objetivos planteados inicialmente —analizar patrones socioeconómicos y demográficos, construir modelos predictivos confiables y desarrollar un dashboard informativo— fueron cumplidos de manera satisfactoria:

- Objetivo de análisis exploratorio: Se logró identificar correlaciones clave y patrones estructurales a través de técnicas estadísticas y visualizaciones avanzadas, superando las expectativas en profundidad analítica.
- Objetivo de modelado predictivo: Los modelos implementados alcanzaron métricas robustas ( $R^2$  hasta 0.60, accuracy hasta 92.3%), validando las hipótesis planteadas y proporcionando herramientas cuantitativas para el análisis de escenarios.

- Objetivo de visualización y comunicación: El dashboard desarrollado sintetizó efectivamente los hallazgos principales, facilitando la interpretación para audiencias técnicas y no técnicas.

Aspectos que podrían profundizarse incluyen la ampliación de la cobertura geográfica para mejorar la representatividad nacional, la incorporación de dimensiones temporales para análisis de evolución histórica, y el desarrollo de modelos causales que permitan evaluar el impacto específico de intervenciones de política pública.

### ***3. Futuras Líneas de Trabajo y Mejoras Propuestas***

#### ***a) Mejoras a los Datos***

- Ampliación del dataset: Incorporar datos de más áreas metropolitanas para mejorar la representatividad nacional y permitir análisis comparativos entre diferentes contextos regionales.
- Corrección de desbalance muestral: Implementar técnicas de sobremuestreo o submuestreo para equilibrar la sobrerepresentación actual del área de Nueva York (56.1% del dataset)
- Incorporación de variables externas: Enriquecer el análisis con indicadores adicionales como calidad de escuelas, acceso a servicios públicos, indicadores de criminalidad y datos de movilidad laboral.
- Refinamiento del tratamiento de outliers: Desarrollar métodos más sofisticados para el manejo de valores atípicos, incluyendo técnicas basadas en distancias multivariadas y detección de anomalías mediante isolation forests.

#### ***b) Mejoras al Modelo***

- Optimización de hiperparámetros: Implementar búsquedas sistemáticas mediante GridSearchCV y RandomizedSearchCV para maximizar el rendimiento de los algoritmos actuales.

- Experimentación con arquitecturas avanzadas: Evaluar el potencial de transformers para capturar interacciones complejas entre variables, y redes neuronales profundas para modelar relaciones no lineales de alta dimensionalidad.
- Implementación de ensembles híbridos: Combinar las fortalezas de diferentes algoritmos mediante técnicas de stacking y blending que aprovechen las ventajas comparativas de cada enfoque.
- Desarrollo de modelos explicativos: Incorporar técnicas SHAP (SHapley Additive exPlanations) y LIME (Local Interpretable Model-agnostic Explanations) para mejorar la interpretabilidad de predicciones individuales.

### *c) Mejoras a las Visualizaciones y Dashboard*

- Implementación de interactividad avanzada: Incorporar filtros dinámicos que permitan a usuarios explorar escenarios específicos por composición étnica, nivel educativo o rango de ingresos.
- Desarrollo de análisis temporal: Integrar series históricas para visualizar la evolución de las desigualdades y evaluar tendencias a lo largo del tiempo.
- Optimización de experiencia de usuario: Rediseñar la interfaz mediante pruebas de usabilidad, mejorando la navegación y la legibilidad de visualizaciones complejas.
- Integración con plataformas empresariales: Desarrollar versiones compatibles con Tableau Server o Power BI Service para facilitar la distribución institucional y el acceso multi-usuario.

### *d) Líneas de Investigación Futura*

- Análisis de causalidad: Aplicar métodos de inferencia causal (propensity score matching, diferencias en diferencias) para evaluar el impacto específico de políticas públicas sobre indicadores de desigualdad.
- Integración de datos en tiempo real: Conectar el sistema con APIs de indicadores económicos y demográficos para mantener actualizadas las proyecciones y análisis.

- Estudios comparativos internacionales: Replicar la metodología en contextos de otros países para identificar patrones universales y particularidades locales en la generación de desigualdad territorial.
- Aplicación de aprendizaje no supervisado avanzado: Implementar técnicas de clustering espectral y autoencoders variacionales para descubrir patrones latentes y segmentos poblacionales no evidentes en el análisis actual.

La implementación de estas mejoras permitiría transformar el actual sistema analítico en una plataforma integral de monitorización y evaluación de políticas de equidad territorial, con capacidad para informar decisiones de inversión pública, diseño de programas sociales y planificación urbana con evidencia cuantitativa robusta y actualizada.



**BUAP**

# **Bibliografía.**

1. mrmorj. (n.d.). *Gentrification and demographic analysis* [Base de datos]. Kaggle. <https://www.kaggle.com/datasets/mrmorj/gentrification-and-demographic-analysis>
2. jarsmp37. (n.d.). *introds* [Repositorio de software]. GitHub. <https://github.com/jarsmp37/introds>
3. Selvaraj, N. (2024, 25 abril). *8 modelos de machine learning explicados en 20 minutos*. DataCamp. <https://www.datacamp.com/es/blog/machine-learning-models-explained> DataCamp
4. IBM. (n.d.). *¿Qué es el machine learning? — tipos de machine learning*. IBM Think. <https://www.ibm.com/mx-es/think/topics/machine-learning-types> ibm.com
5. Cyberclick. (2025, 29 julio). *¿Qué es un dashboard y para qué se usa?* Numerical Blog. <https://www.cyberclick.es/numerical-blog/que-es-un-dashboard>