Análisis de Datos con Python y Streamlit

Herramientas para Estadísticas Básicas, Limpieza de Datos,Análisis Visual Interactivo de Datos (ENAHO) y Análisis de Gastos en Servicios Básicos con MANOVA

Edilberto Wilson Mamani Emanuel

4 de julio de 2025

1. Analizador de CSV - Estadísticas Básicas

Propósito: Esta herramienta permite cargar archivos CSV y realizar análisis estadísticos básicos de forma intuitiva. Está diseñada para usuarios que no tienen experiencia avanzada en estadística, proporcionando explicaciones claras de cada concepto.

Funcionalidades principales:

- Carga automática de archivos CSV con detección de codificación
- Cálculo de estadísticas descriptivas (media, mediana, moda, desviación estándar)
- Visualizaciones interactivas (histogramas, gráficos de caja)
- Interpretación automática de resultados
- Interfaz web amigable con Streamlit

Conceptos estadísticos incluidos:

- Medidas de tendencia central (promedio, mediana, moda)
- Medidas de dispersión (desviación estándar, rango, cuartiles)
- Análisis de distribución de datos
- Interpretación de coeficiente de variación

1.1. Código Fuente

```
import streamlit as st
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sn
from io import StringIO

# Configurar la p gina
st.set_page_config(
```

```
    page_title="       Analizador de CSV - Estad sticas  B sicas",
    page_icon="      ",
    layout="wide"
)

# T tulo principal
st.title("       Analizador de CSV - Estad sticas  B sicas")
st.markdown("###   Sube   tu archivo CSV y descubre qu   dicen tus datos!"
    )

# Sidebar con informaci n
st.sidebar.markdown("##        Gu a  R pida")
st.sidebar.markdown("""
**  Qu    puedes hacer aqu ?**
- Subir un archivo CSV
- Ver estad sticas  b sicas
- Entender qu   significan los n meros
- Ver gr ficos  f ciles de entender
""")

# Funci n para explicar conceptos estad sticos
def explicar_concepto(concepto):
    explicaciones = {
        "promedio": "       **Promedio (Media)**: Es como dividir una
    pizza entre todos. Si tienes 10 pizzas y 5 personas, cada uno come 2
    pizzas en promedio.",
        "mediana": "       **Mediana**: Es el valor del medio cuando
    ordenas todos los n meros de menor a mayor. Como el estudiante que
    queda en el medio en una fila ordenada por altura.",
        "moda": "       **Moda**: Es el valor que m s se repite. Como
    el color favorito que eligi  la mayor a de personas en una encuesta
    .",
        "desviacion": "       **Desviaci n Est ndar**: Nos dice qu
    tan dispersos est n los datos. Si es peque a, los datos est n muy
    juntos. Si es grande, est n muy separados.",
        "minimo": "       **M nimo**: El valor m s  peque o en tus
    datos.",
        "maximo": "       **M ximo**: El valor m s grande en tus datos
    .",
        "rango": "       **Rango**: La diferencia entre el valor m s
    grande y el m  s  peque o.",
        "cuartiles": "       **Cuartiles**: Dividen tus datos en 4
    partes iguales. Como dividir a todos los estudiantes en 4 grupos del
    mismo tama o  seg n sus calificaciones."
    }
    return explicaciones.get(concepto, "")

# Subir archivo
uploaded_file = st.file_uploader(
    "       Sube tu archivo CSV aqu ",
    type=['csv'],
    help="Arrastra y suelta tu archivo CSV o haz clic para seleccionarlo
    "
)

if uploaded_file is not None:
    try:
        # Funci n para detectar la codificaci n del archivo
```

```
53        def leer_csv_con_codificacion(archivo):
54            # Lista de codificaciones comunes
55            codificaciones = ['utf-8', 'latin1', 'iso-8859-1', 'cp1252',
    'utf-16']
56
57            for codificacion in codificaciones:
58                try:
59                    # Resetear el puntero del archivo
60                    archivo.seek(0)
61                    # Intentar leer con la codificaci n actual
62                    df = pd.read_csv(archivo, encoding=codificacion)
63                    st.success(f"    Archivo le do correctamente con
    codificaci n: {codificacion}")
64                    return df
65                except UnicodeDecodeError:
66                    continue
67                except Exception as e:
68                    # Si hay otro error, intentar con la siguiente
    codificaci n
69                    continue
70
71            # Si ninguna codificaci n funcion , intentar con 'errors=
    ignore'
72            try:
73                archivo.seek(0)
74                df = pd.read_csv(archivo, encoding='utf-8', errors='
    ignore')
75                st.warning("       Se ley  el archivo ignorando algunos
     caracteres especiales")
76                return df
77            except:
78                raise Exception("No se pudo leer el archivo con ninguna
    codificaci n  com n")
79
80        # Leer el archivo CSV con detecci n autom tica de
    codificaci n
81        df = leer_csv_con_codificacion(uploaded_file)
82
83        # Mostrar informaci n b sica del archivo
84        st.success(f"      Archivo  cargado exitosamente!")
85
86        col1, col2, col3 = st.columns(3)
87        with col1:
88            st.metric("       Total de filas", len(df))
89        with col2:
90            st.metric("       Total de columnas", len(df.columns))
91        with col3:
92            st.metric("       Columnas num ricas", len(df.select_dtypes
    (include=[np.number]).columns))
93
94        # Mostrar vista previa de los datos
95        st.subheader("       Vista previa de tus datos")
96        st.dataframe(df.head(10), use_container_width=True)
97
98        # Obtener solo columnas num ricas
99        columnas_numericas = df.select_dtypes(include=[np.number]).
    columns.tolist()
100
```

```
        if len(columnas_numericas) > 0:
            st.subheader("       Estad sticas  B sicas")

            # Selector de columna
            columna_seleccionada = st.selectbox(
                "       Selecciona una columna para analizar:",
                columnas_numericas,
                help="Elige la columna de la que quieres ver las estad sticas"
            )

            if columna_seleccionada:
                datos_columna = df[columna_seleccionada].dropna()

                # Calcular estad sticas
                promedio = datos_columna.mean()
                mediana = datos_columna.median()
                moda = datos_columna.mode().iloc[0] if not datos_columna.mode().empty else "No hay moda"
                desviacion = datos_columna.std()
                minimo = datos_columna.min()
                maximo = datos_columna.max()
                rango = maximo - minimo
                q1 = datos_columna.quantile(0.25)
                q3 = datos_columna.quantile(0.75)

                # Mostrar estad sticas en tarjetas
                st.markdown(f"###        Estad sticas de la columna: **{columna_seleccionada}**")

                # Fila 1 de m tricas
                col1, col2, col3, col4 = st.columns(4)
                with col1:
                    st.metric("       Promedio", f"{promedio:.2f}")
                    st.info(explicar_concepto("promedio"))

                with col2:
                    st.metric("       Mediana", f"{mediana:.2f}")
                    st.info(explicar_concepto("mediana"))

                with col3:
                    st.metric("       M nimo", f"{minimo:.2f}")
                    st.info(explicar_concepto("minimo"))

                with col4:
                    st.metric("       M ximo", f"{maximo:.2f}")
                    st.info(explicar_concepto("maximo"))

                # Fila 2 de m tricas
                col1, col2, col3, col4 = st.columns(4)
                with col1:
                    st.metric("       Desviaci n  Est ndar", f"{desviacion:.2f}")
                    st.info(explicar_concepto("desviacion"))

                with col2:
                    st.metric("       Moda", f"{moda}")
                    st.info(explicar_concepto("moda"))
```

```
155
156                 with col3:
157                     st.metric("       Rango", f"{rango:.2f}")
158                     st.info(explicar_concepto("rango"))
159
160                 with col4:
161                     st.metric("       Datos  v lidos", len(datos_columna
    ))
162                     st.info("Cantidad de datos que no est n  vac os")
163
164                 # Interpretaci n  autom tica
165                 st.subheader("          Qu   significa esto?")
166
167                 interpretacion = []
168
169                 if abs(promedio - mediana) / promedio < 0.1:
170                     interpretacion.append("    **Datos balanceados**: Tu
     promedio y mediana son muy similares, esto significa que tus datos
    est n  bien distribuidos.")
171                 else:
172                     interpretacion.append("       **Datos desbalanceados
    **: Tu promedio y mediana son diferentes, esto puede indicar que
    tienes valores muy altos o muy bajos que afectan el promedio.")
173
174                 cv = (desviacion / promedio) * 100 if promedio != 0 else
     0
175                 if cv < 15:
176                     interpretacion.append(f"       **Datos consistentes
    **: Tus datos  var an poco (variaci n del {cv:.1f}%), est n
    bastante agrupados.")
177                 elif cv < 30:
178                     interpretacion.append(f"       **Datos moderadamente
     variables**: Tus datos tienen variaci n media ({cv:.1f}%).")
179                 else:
180                     interpretacion.append(f"       **Datos muy variables
    **: Tus datos  est n  muy dispersos (variaci n del {cv:.1f}%).")
181
182                 for interp in interpretacion:
183                     st.markdown(interp)
184
185                 # Gr ficos
186                 st.subheader("       Visualizaciones")
187
188                 tab1, tab2, tab3 = st.tabs(["       Histograma", "
     Gr fico de Caja", "       Estad sticas Resumidas"])
189
190                 with tab1:
191                     st.markdown("**  Qu   es un histograma?**       ")
192                     st.markdown("Es como un gr fico de barras que
    muestra cu ntos  datos tienes en cada rango de valores. Te ayuda a
    ver la forma de tus datos.")
193
194                     fig, ax = plt.subplots(figsize=(10, 6))
195                     ax.hist(datos_columna, bins=20, color='skyblue',
    alpha=0.7, edgecolor='black')
196                     ax.set_xlabel(columna_seleccionada)
197                     ax.set_ylabel('Frecuencia (  Cuntas   veces aparece
    ?)')
```

```
198                     ax.set_title(f'Distribuci n de {
    columna_seleccionada}')
199                     ax.grid(True, alpha=0.3)
200                     st.pyplot(fig)
201 
202                 with tab2:
203                     st.markdown("**  Qu   es un gr fico de caja?**
        ")
204                     st.markdown("Es como un resumen visual de tus datos.
     La caja muestra d nde est  la mayor a de tus datos, y las l neas
     muestran los valores extremos.")
205 
206                     fig, ax = plt.subplots(figsize=(10, 6))
207                     ax.boxplot(datos_columna, vert=False)
208                     ax.set_xlabel(columna_seleccionada)
209                     ax.set_title(f'Resumen visual de {
    columna_seleccionada}')
210                     ax.grid(True, alpha=0.3)
211                     st.pyplot(fig)
212 
213                 with tab3:
214                     st.markdown("**       Resumen completo de
    estad sticas**")
215 
216                     # Crear tabla de resumen
217                     resumen = pd.DataFrame({
218                         'Estad stica': ['Promedio', 'Mediana', 'Moda',
    'Desviaci n Est ndar', 'M nimo', 'M ximo', 'Rango', 'Cuartil 1
    (25%)', 'Cuartil 3 (75%)'],
219                         'Valor': [f"{promedio:.2f}", f"{mediana:.2f}",
    str(moda), f"{desviacion:.2f}", f"{minimo:.2f}", f"{maximo:.2f}", f"{
    rango:.2f}", f"{q1:.2f}", f"{q3:.2f}"],
220                         'Significado': [
221                             'Valor t pico de tus datos',
222                             'Valor del medio cuando ordenas los datos',
223                             'Valor que m s se repite',
224                             'Qu  tan dispersos est n los datos',
225                             'Valor m s peque o',
226                             'Valor m s grande',
227                             'Diferencia entre m ximo y m nimo',
228                             '25% de datos est n por debajo de este
    valor',
229                             '75% de datos est n por debajo de este
    valor'
230                         ]
231                     })
232 
233                     st.dataframe(resumen, use_container_width=True)
234 
235                 # An lisis de todas las columnas num ricas
236                 if len(columnas_numericas) > 1:
237                     st.subheader("       Resumen de todas las columnas
    num ricas")
238 
239                     # Descripci n estad stica de todas las columnas
240                     descripcion = df[columnas_numericas].describe()
241                     st.dataframe(descripcion, use_container_width=True)
242 
```

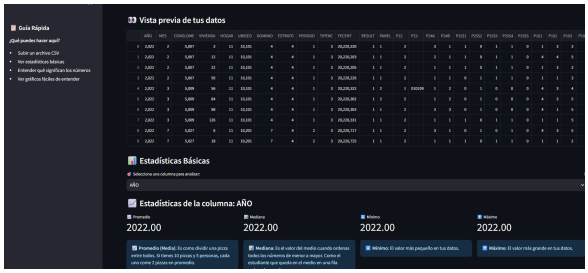
```
243                     st.markdown("**  Qu   significa cada fila?**")
244                     st.markdown("""
245                     - **count**: Cu ntos datos v lidos tienes en cada
    columna
246                     - **mean**: El promedio de cada columna
247                     - **std**: La desviaci n est ndar (dispersi n) de
     cada columna
248                     - **min**: El valor m nimo de cada columna
249                     - **25%**: El 25% de los datos est n por debajo de
    este valor
250                     - **50%**: La mediana (valor del medio)
251                     - **75%**: El 75% de los datos est n por debajo de
    este valor
252                     - **max**: El valor m ximo de cada columna
253                     """)
254         else:
255             st.warning("       No se encontraron columnas num ricas en
    tu archivo CSV. Las estad sticas solo se pueden calcular para datos
    num ricos.")
256             st.markdown("**  Qu   puedes hacer?**")
257             st.markdown("- Verifica que tu archivo tenga columnas con
    n meros")
258             st.markdown("- Aseg rate de que los n meros no est n
    escritos como texto")
259
260     except Exception as e:
261         st.error(f"       Hubo un error al procesar tu archivo: {str(e)}
    ")
262         st.markdown("**Posibles soluciones:**")
263         st.markdown("- **Problema de codificaci n**: Abre tu archivo
    CSV en un editor de texto y gu rdalo como UTF-8")
264         st.markdown("- **Archivo da ado**: Verifica que tu archivo CSV
    no est  corrupto")
265         st.markdown("- **Formato incorrecto**: Aseg rate de que sea un
    archivo CSV v lido")
266         st.markdown("- **Caracteres especiales**: Evita usar caracteres
    como   ,   ,   ,   ,   ,    en los nombres de columnas")
267
268         # Mostrar informaci n adicional para debug
269         st.markdown("###        Informaci n t cnica:")
270         st.code(f"Error espec fico: {str(e)}", language='text')
271
272         # Sugerencias espec ficas para codificaci n
273         st.markdown("###        Si tienes problemas de codificaci n:")
274         st.markdown("""
275         **Opci n 1 - En Excel:**
276         1. Abre tu archivo en Excel
277         2. Ve a 'Archivo'     'Guardar como'
278         3. Selecciona 'CSV UTF-8 (delimitado por comas)'
279         4. Guarda el archivo
280
281         **Opci n 2 - En Google Sheets:**
282         1. Sube tu archivo a Google Sheets
283         2. Ve a 'Archivo'     'Descargar'     'Valores separados por
    comas (.csv)'
284         3. Usa el archivo descargado
285         """)
286
```

```
        st.markdown("###          Sigues  teniendo problemas?")
        st.markdown("Env ame las primeras l neas de tu archivo CSV y te ayudo a solucionarlo.")

else:
    # Mostrar informaci n de ayuda cuando no hay archivo
    st.info("       Sube un archivo CSV para comenzar el an lisis")

    st.markdown("###          Qu   son las estad sticas b sicas?")
    st.markdown("""
    Las estad sticas b sicas son como un resumen de tus datos que te ayuda a entender:

    - **          Cul   es el valor t pico?** (Promedio)
    - **          Cul   es el valor del medio?** (Mediana)
    - **          Cul   es el valor m s com n?** (Moda)
    - **          Qu   tan dispersos est n los datos?** (Desviaci n est ndar)
    - **               Cules   son los valores extremos?** (M nimo y m ximo)
    """)

    st.markdown("###        Ejemplo de archivo CSV")
    st.markdown("Tu archivo CSV debe verse algo as :")

    ejemplo_csv = """Nombre,Edad,Salario,Experiencia
Juan,25,45000,2
Mar a,30,55000,5
Pedro,35,65000,8
Ana,28,50000,3"""

    st.code(ejemplo_csv, language='csv')

    st.markdown("**       Consejos para tu archivo CSV:**")
    st.markdown("- La primera fila debe contener los nombres de las columnas")
    st.markdown("- Los datos deben estar separados por comas")
    st.markdown("- Los n meros no deben contener s mbolos como $ o %")
    st.markdown("- Guarda el archivo con extensi n .csv")

# Footer
st.markdown("---")
st.markdown("       **Consejo**: Las estad sticas te ayudan a tomar mejores decisiones basadas en tus datos.  No  necesitas ser un experto para entender lo que significan!")
```

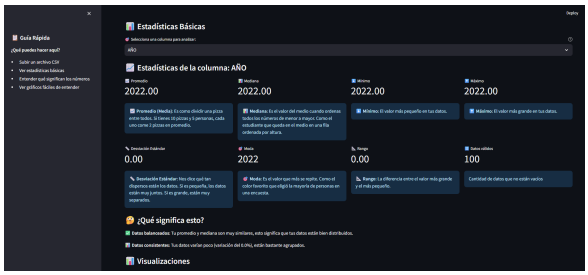
Listing 1: Analizador de CSV - Estadísticas Básicas

1.2. Capturas de Pantalla

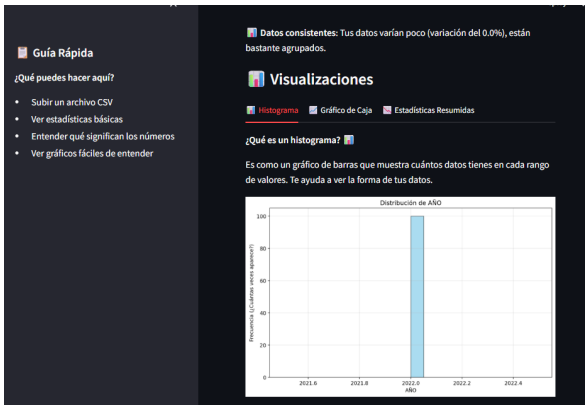
(a) Interfaz principal de carga de archivos



(b) Vista previa de datos cargados



(a) Métricas estadísticas calculadas



(b) Histograma de distribución

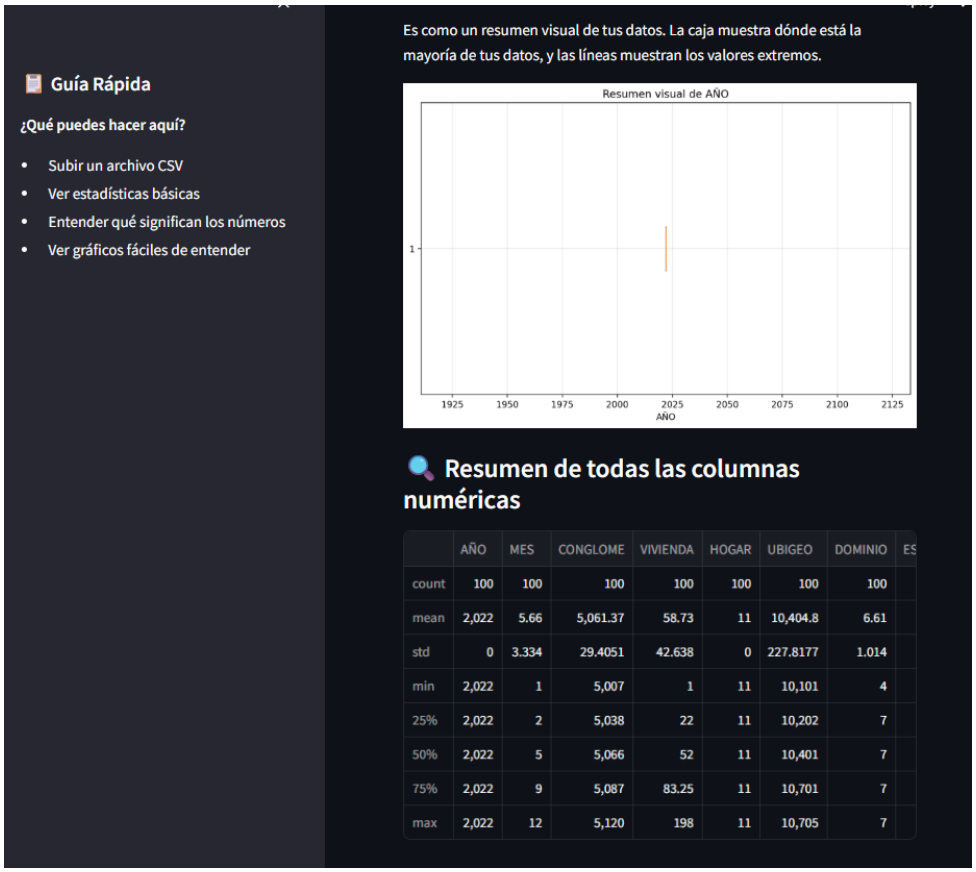


Figura 3: Gráfico de caja (boxplot) y resumen estadístico

1.3. Análisis Técnico

Aspectos técnicos destacados:

- **Detección automática de codificación:** El sistema prueba múltiples codificaciones (UTF-8, Latin-1, ISO-8859-1) para garantizar la correcta lectura de archivos CSV con caracteres especiales.
- **Interfaz adaptativa:** La aplicación se adapta automáticamente al tipo de datos encontrados, mostrando solo las opciones relevantes para cada conjunto de datos.
- **Interpretación automatizada:** El sistema compara automáticamente media y mediana para detectar asimetrías, y calcula el coeficiente de variación para evaluar la dispersión de los datos.
- **Visualizaciones interactivas:** Utiliza tanto Matplotlib como Plotly para crear gráficos informativos que ayudan a entender la distribución de los datos.

2. Limpieza de Datos y Detección de Outliers

Propósito: Esta herramienta se enfoca en la identificación y corrección de problemas comunes en conjuntos de datos, incluyendo valores faltantes, duplicados y valores atípicos (outliers). Es fundamental para preparar los datos antes del análisis.

Funcionalidades principales:

- Detección automática de datos faltantes con visualización de patrones
- Identificación de outliers usando métodos IQR y Z-Score
- Eliminación automática de duplicados
- Imputación de valores faltantes con múltiples estrategias
- Generación de reportes detallados de calidad de datos
- Descarga de datos limpios en formato CSV

Métodos de detección de outliers:

- **IQR (Rango Intercuartílico):** Método robusto que usa Q1 y Q3 para definir límites
- **Z-Score:** Método basado en desviaciones estándar desde la media
- Visualizaciones con gráficos de caja y dispersión

2.1. Código Fuente

```
import streamlit as st
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import warnings
warnings.filterwarnings('ignore')

# Configurar la p gina
st.set_page_config(
    page_title="       Limpieza de Datos y Detecci n de Outliers",
    page_icon="      ",
    layout="wide"
)

# T tulo principal
st.title("       Limpieza de Datos y Detecci n de Outliers")
st.markdown("###   Encuentra   y limpia los datos problem ticos en tu
    archivo CSV!")

# Sidebar con informaci n
st.sidebar.markdown("##        Gu a de Limpieza")
st.sidebar.markdown("""
**  Qu   hace esta herramienta?**
-        Detecta datos faltantes
-        Encuentra outliers (datos raros)
-        Sugiere c mo limpiar tus datos
-        Muestra gr ficos f ciles de entender
-        Genera reporte completo
""")
```

```
34
35 # Funci n para explicar conceptos
36 def explicar_concepto(concepto):
37     explicaciones = {
38         "outliers": "       **Outliers (Datos At picos)**: Son valores
    que est n muy lejos del resto. Como una persona de 2.5 metros en un
    grupo de personas normales.",
39         "datos_faltantes": "    **Datos Faltantes**: Son espacios
    vac os en tu tabla. Como cuando no respondiste una pregunta en una
    encuesta.",
40         "iqr": "       **M todo IQR**: Usa cuartiles para encontrar
    datos raros. Si un dato est  muy lejos del 'grupo principal', lo
    marca como extra o.",
41         "zscore": "       **M todo Z-Score**: Mide qu  tan 'normal' es
     un dato. Si est  a m s de 3 desviaciones del promedio, es raro.",
42         "duplicados": "       **Datos Duplicados**: Filas que se repiten
     exactamente igual. Como tener la misma persona dos veces en una
    lista.",
43         "limpieza": "       **Limpieza de Datos**: Es como ordenar tu
    cuarto. Quitas lo que no sirve y organizas lo que s  ."
44     }
45     return explicaciones.get(concepto, "")
46
47 # Funci n para leer CSV con m ltiples codificaciones
48 def leer_csv_con_codificacion(archivo):
49     codificaciones = ['utf-8', 'latin1', 'iso-8859-1', 'cp1252', 'utf-16
    ']
50
51     for codificacion in codificaciones:
52         try:
53             archivo.seek(0)
54             df = pd.read_csv(archivo, encoding=codificacion)
55             st.success(f"    Archivo le do correctamente con
    codificaci n: {codificacion}")
56             return df
57         except UnicodeDecodeError:
58             continue
59         except Exception as e:
60             continue
61
62     try:
63         archivo.seek(0)
64         df = pd.read_csv(archivo, encoding='utf-8', errors='ignore')
65         st.warning("       Se ley  el archivo ignorando algunos
    caracteres especiales")
66         return df
67     except:
68         raise Exception("No se pudo leer el archivo con ninguna
    codificaci n  com n")
69
70 # Funci n para detectar outliers con IQR
71 def detectar_outliers_iqr(df, columna):
72     Q1 = df[columna].quantile(0.25)
73     Q3 = df[columna].quantile(0.75)
74     IQR = Q3 - Q1
75     limite_inferior = Q1 - 1.5 * IQR
76     limite_superior = Q3 + 1.5 * IQR
77
```

```
78      outliers = df[(df[columna] < limite_inferior) | (df[columna] >
    limite_superior)]
79      return outliers, limite_inferior, limite_superior
80
81  # Funci n para detectar outliers con Z-Score
82  def detectar_outliers_zscore(df, columna, umbral=3):
83      z_scores = np.abs(stats.zscore(df[columna].dropna()))
84      outliers = df[z_scores > umbral]
85      return outliers, umbral
86
87  # Funci n para generar reporte de limpieza
88  def generar_reporte_limpieza(df):
89      reporte = {}
90
91      # Informaci n general
92      reporte['filas_totales'] = len(df)
93      reporte['columnas_totales'] = len(df.columns)
94      reporte['celdas_totales'] = len(df) * len(df.columns)
95
96      # Datos faltantes
97      reporte['datos_faltantes'] = df.isnull().sum().sum()
98      reporte['porcentaje_faltantes'] = (reporte['datos_faltantes'] /
    reporte['celdas_totales']) * 100
99
100     # Duplicados
101     reporte['filas_duplicadas'] = df.duplicated().sum()
102     reporte['porcentaje_duplicados'] = (reporte['filas_duplicadas'] /
    reporte['filas_totales']) * 100
103
104     # Columnas con problemas
105     reporte['columnas_con_faltantes'] = df.isnull().sum()[df.isnull().
    sum() > 0].to_dict()
106
107     return reporte
108
109 # Subir archivo
110 uploaded_file = st.file_uploader(
111     "       Sube tu archivo CSV aqu ",
112     type=['csv'],
113     help="Arrastra tu archivo CSV o haz clic para seleccionarlo"
114 )
115
116 if uploaded_file is not None:
117     try:
118         # Leer el archivo
119         df_original = leer_csv_con_codificacion(uploaded_file)
120         df = df_original.copy()
121
122         # Informaci n b sica
123         st.success("      Archivo   cargado exitosamente!")
124
125         # M tricas principales
126         col1, col2, col3, col4 = st.columns(4)
127         with col1:
128             st.metric("       Total de filas", len(df))
129         with col2:
130             st.metric("       Total de columnas", len(df.columns))
131         with col3:
```

```
            st.metric("       Columnas num ricas", len(df.select_dtypes
    (include=[np.number]).columns))
        with col4:
            st.metric("       Columnas de texto", len(df.select_dtypes(
    include=['object']).columns))

        # Generar reporte de limpieza
        reporte = generar_reporte_limpieza(df)

        # Mostrar vista previa
        st.subheader("       Vista previa de tus datos")
        st.dataframe(df.head(10), use_container_width=True)

        # Tabs principales
        tab1, tab2, tab3, tab4, tab5 = st.tabs([
            "       Reporte General",
            "    Datos Faltantes",
            "       Outliers (Datos At picos)",
            "       Limpieza Autom tica",
            "       Reporte Final"
        ])

        with tab1:
            st.subheader("       Reporte General de Calidad")

            # M tricas de calidad
            col1, col2 = st.columns(2)

            with col1:
                st.metric(
                    "    Datos Faltantes",
                    f"{reporte['datos_faltantes']:,}",
                    f"{reporte['porcentaje_faltantes']:.1f}% del total"
                )

                st.metric(
                    "       Filas Duplicadas",
                    f"{reporte['filas_duplicadas']:,}",
                    f"{reporte['porcentaje_duplicados']:.1f}% del total"
                )

            with col2:
                # Gr fico de calidad general
                fig = go.Figure(data=[
                    go.Bar(
                        name='Datos Completos',
                        x=['Calidad de Datos'],
                        y=[100 - reporte['porcentaje_faltantes']],
                        marker_color='green'
                    ),
                    go.Bar(
                        name='Datos Faltantes',
                        x=['Calidad de Datos'],
                        y=[reporte['porcentaje_faltantes']],
                        marker_color='red'
                    )
                ])

```

```
188                 fig.update_layout(
189                     title='Calidad General de los Datos',
190                     yaxis_title='Porcentaje',
191                     barmode='stack',
192                     height=300
193                 )
194                 st.plotly_chart(fig, use_container_width=True)
195 
196             # Calidad por columna
197             st.subheader("       Calidad por Columna")
198 
199             calidad_columnas = []
200             for columna in df.columns:
201                 faltantes = df[columna].isnull().sum()
202                 porcentaje_faltantes = (faltantes / len(df)) * 100
203                 porcentaje_completos = 100 - porcentaje_faltantes
204 
205                 calidad_columnas.append({
206                     'Columna': columna,
207                     'Datos Completos': int(len(df) - faltantes),
208                     'Datos Faltantes': int(faltantes),
209                     '% Completos': round(porcentaje_completos, 1),
210                     '% Faltantes': round(porcentaje_faltantes, 1),
211                     'Tipo de Dato': str(df[columna].dtype)
212                 })
213 
214             df_calidad = pd.DataFrame(calidad_columnas)
215             st.dataframe(df_calidad, use_container_width=True)
216 
217             # Interpretaci n
218             st.info(explicar_concepto("datos_faltantes"))
219             st.info(explicar_concepto("duplicados"))
220 
221         with tab2:
222             st.subheader("    An lisis de Datos Faltantes")
223 
224             if reporte['datos_faltantes'] > 0:
225                 # Mapa de calor de datos faltantes
226                 st.markdown("###        Mapa de Calor de Datos Faltantes
    ")
227                 st.markdown("**  Qu   muestra?** Las  reas  rojas son
    donde faltan datos. Las azules donde est n completos.")
228 
229                 # Crear matriz de valores faltantes
230                 missing_matrix = df.isnull().astype(int)
231 
232                 fig, ax = plt.subplots(figsize=(12, 8))
233                 sns.heatmap(missing_matrix, cbar=True, cmap='RdYlBu_r',
    ax=ax)
234                 ax.set_title('Mapa de Datos Faltantes (Rojo = Falta,
    Azul = Completo)')
235                 st.pyplot(fig)
236 
237                 # Estad sticas por columna
238                 st.markdown("###        Estad sticas Detalladas")
239 
240                 columnas_problematicas = []
241                 for columna, faltantes in reporte['
```

```
    columnas_con_faltantes'].items():
242                     porcentaje = (faltantes / len(df)) * 100
243
244                     # Determinar nivel de problema
245                     if porcentaje < 5:
246                         nivel = "       M nimo"
247                     elif porcentaje < 20:
248                         nivel = "       Moderado"
249                     elif porcentaje < 50:
250                         nivel = "       Alto"
251                     else:
252                         nivel = "       Cr tico"
253
254                     columnas_problematicas.append({
255                         'Columna': columna,
256                         'Datos Faltantes': faltantes,
257                         'Porcentaje': f"{porcentaje:.1f}%",
258                         'Nivel de Problema': nivel,
259                         'Recomendaci n': 'Eliminar columna' if
    porcentaje > 70 else 'Rellenar valores' if porcentaje > 30 else '
    Eliminar filas'
260                     })
261
262                 df_problematicas = pd.DataFrame(columnas_problematicas)
263                 st.dataframe(df_problematicas, use_container_width=True)
264
265                 # Recomendaciones
266                 st.markdown("###        Recomendaciones para Datos
    Faltantes")
267                 st.markdown("""
268                 - **       M nimo (< 5%)**: Eliminar las filas con
    datos faltantes
269                 - **       Moderado (5-20%)**: Rellenar con promedio,
    mediana o moda
270                 - **       Alto (20-50%)**: Considerar t cnicas
    avanzadas de imputaci n
271                 - **       Cr tico (> 50%)**: Evaluar si eliminar la
    columna completa
272                 """)
273
274             else:
275                 st.success("        Excelente ! No tienes datos
    faltantes en tu archivo.")
276
277         with tab3:
278             st.subheader("       Detecci n de Outliers (Datos At picos
    )")
279
280             st.info(explicar_concepto("outliers"))
281
282             # Obtener columnas num ricas
283             columnas_numericas = df.select_dtypes(include=[np.number]).
    columns.tolist()
284
285             if len(columnas_numericas) > 0:
286                 # Selector de columna
287                 columna_analizar = st.selectbox(
288                     "       Selecciona una columna para analizar
```

```
    outliers:",
289                     columnas_numericas
290                 )
291
292                 if columna_analizar:
293                     # Selector de m todo
294                     metodo = st.radio(
295                         "       Selecciona el m todo de detecci n:",
296                         ["IQR (Recomendado)", "Z-Score"],
297                         help="IQR es m s robusto, Z-Score es m s
    sensible"
298                     )
299
300                     col1, col2 = st.columns(2)
301
302                     with col1:
303                         if metodo == "IQR (Recomendado)":
304                             st.info(explicar_concepto("iqr"))
305
306                             # Detectar outliers con IQR
307                             outliers, limite_inf, limite_sup =
    detectar_outliers_iqr(df, columna_analizar)
308
309                             st.metric("       Outliers detectados", len(
    outliers))
310                             st.metric("       L mite inferior", f"{
    limite_inf:.2f}")
311                             st.metric("       L mite superior", f"{
    limite_sup:.2f}")
312
313                         else:
314                             st.info(explicar_concepto("zscore"))
315
316                             # Detectar outliers con Z-Score
317                             outliers, umbral = detectar_outliers_zscore(
    df, columna_analizar)
318
319                             st.metric("       Outliers detectados", len(
    outliers))
320                             st.metric("       Umbral Z-Score", f"{umbral
    }")
321
322                     with col2:
323                         # Gr fico de caja
324                         fig = px.box(df, y=columna_analizar, title=f'
    Detecci n de Outliers: {columna_analizar}')
325                         fig.update_layout(height=400)
326                         st.plotly_chart(fig, use_container_width=True)
327
328                     # Mostrar outliers detectados
329                     if len(outliers) > 0:
330                         st.subheader("       Outliers Detectados")
331
332                         # Mostrar estad sticas de outliers
333                         col1, col2, col3 = st.columns(3)
334                         with col1:
335                             st.metric("       Total de outliers", len(
    outliers))
```

```
                        with col2:
                            st.metric("       Valor m ximo outlier", f"
{outliers[columna_analizar].max():.2f}")
                        with col3:
                            st.metric("       Valor m nimo outlier", f"
{outliers[columna_analizar].min():.2f}")

                        # Tabla de outliers
                        st.dataframe(outliers, use_container_width=True)

                        # Gr fico de dispersi n
                        st.subheader("       Visualizaci n de Outliers"
)

                        # Crear gr fico de dispersi n
                        fig = px.scatter(
                            df,
                            x=df.index,
                            y=columna_analizar,
                            title=f'Outliers en {columna_analizar}',
                            labels={'x': ' ndice  de Fila', 'y':
columna_analizar}
                        )

                        # Destacar outliers
                        fig.add_scatter(
                            x=outliers.index,
                            y=outliers[columna_analizar],
                            mode='markers',
                            marker=dict(color='red', size=10),
                            name='Outliers'
                        )

                        st.plotly_chart(fig, use_container_width=True)

                        # Recomendaciones
                        st.subheader("         Qu   hacer con los
outliers?")
                        porcentaje_outliers = (len(outliers) / len(df))
* 100

                        if porcentaje_outliers < 1:
                            st.success(f"    Solo {porcentaje_outliers
:.1f}% son outliers. Puedes eliminarlos de forma segura.")
                        elif porcentaje_outliers < 5:
                            st.warning(f"       {porcentaje_outliers:.1f
}% son outliers. Considera si son errores o datos v lidos.")
                        else:
                            st.error(f"       {porcentaje_outliers:.1f}%
 son outliers. Revisa si hay problemas en la recolecci n de datos.")

                    else:
                        st.success("        Excelente  ! No se detectaron
 outliers en esta columna.")

                # An lisis de todas las columnas
                st.subheader("       Resumen de Outliers en Todas las
Columnas")
```

```
383
384                 resumen_outliers = []
385                 for columna in columnas_numericas:
386                     outliers_iqr, _, _ = detectar_outliers_iqr(df,
    columna)
387                     outliers_zscore, _ = detectar_outliers_zscore(df,
    columna)
388
389                     resumen_outliers.append({
390                         'Columna': columna,
391                         'Outliers IQR': len(outliers_iqr),
392                         'Outliers Z-Score': len(outliers_zscore),
393                         '% Outliers IQR': f"{(len(outliers_iqr)/len(df)
    *100):.1f}%",
394                         '% Outliers Z-Score': f"{(len(outliers_zscore)/
    len(df)*100):.1f}%"
395                     })
396
397                 df_resumen = pd.DataFrame(resumen_outliers)
398                 st.dataframe(df_resumen, use_container_width=True)
399
400             else:
401                 st.warning("       No se encontraron columnas num ricas
     para analizar outliers.")
402
403         with tab4:
404             st.subheader("       Limpieza Autom tica de Datos")
405
406             st.info(explicar_concepto("limpieza"))
407
408             # Opciones de limpieza
409             st.markdown("###        Opciones de Limpieza")
410
411             col1, col2 = st.columns(2)
412
413             with col1:
414                 st.markdown("**          Eliminar Datos**")
415                 eliminar_duplicados = st.checkbox("Eliminar filas
    duplicadas", value=True)
416                 eliminar_filas_vacias = st.checkbox("Eliminar filas
    completamente vac as", value=True)
417
418                 # Umbral para eliminar filas con muchos faltantes
419                 umbral_faltantes = st.slider(
420                     "Eliminar filas con m s de X% datos faltantes:",
421                     min_value=0, max_value=100, value=50, step=5
422                 )
423
424             with col2:
425                 st.markdown("**       Rellenar Datos Faltantes**")
426                 rellenar_numericos = st.selectbox(
427                     "Rellenar columnas num ricas con:",
428                     ["No rellenar", "Promedio", "Mediana", "Moda", "Cero
    "]
429                 )
430
431                 rellenar_texto = st.selectbox(
432                     "Rellenar columnas de texto con:",
```

```
                    ["No rellenar", "Moda", "Texto personalizado"]
                )

                if rellenar_texto == "Texto personalizado":
                    texto_personalizado = st.text_input("Texto para
rellenar:", value="Sin datos")

            # Bot n para limpiar
            if st.button("       Limpiar Datos", type="primary"):
                df_limpio = df.copy()
                acciones_realizadas = []

                # Eliminar duplicados
                if eliminar_duplicados:
                    duplicados_antes = df_limpio.duplicated().sum()
                    df_limpio = df_limpio.drop_duplicates()
                    if duplicados_antes > 0:
                        acciones_realizadas.append(f"    Eliminadas {
duplicados_antes} filas duplicadas")

                # Eliminar filas completamente vac as
                if eliminar_filas_vacias:
                    filas_vacias = df_limpio.isnull().all(axis=1).sum()
                    df_limpio = df_limpio.dropna(how='all')
                    if filas_vacias > 0:
                        acciones_realizadas.append(f"    Eliminadas {
filas_vacias} filas completamente vac as")

                # Eliminar filas con muchos faltantes
                if umbral_faltantes < 100:
                    umbral_decimal = umbral_faltantes / 100
                    filas_antes = len(df_limpio)
                    df_limpio = df_limpio.dropna(thresh=len(df_limpio.
columns) * (1 - umbral_decimal))
                    filas_eliminadas = filas_antes - len(df_limpio)
                    if filas_eliminadas > 0:
                        acciones_realizadas.append(f"    Eliminadas {
filas_eliminadas} filas con m s del {umbral_faltantes}% de datos
faltantes")

                # Rellenar datos num ricos
                if rellenar_numericos != "No rellenar":
                    columnas_numericas = df_limpio.select_dtypes(include
=[np.number]).columns
                    for columna in columnas_numericas:
                        faltantes_antes = df_limpio[columna].isnull().
sum()
                        if faltantes_antes > 0:
                            if rellenar_numericos == "Promedio":
                                df_limpio[columna] = df_limpio[columna].
fillna(df_limpio[columna].mean())
                            elif rellenar_numericos == "Mediana":
                                df_limpio[columna] = df_limpio[columna].
fillna(df_limpio[columna].median())
                            elif rellenar_numericos == "Moda":
                                df_limpio[columna] = df_limpio[columna].
fillna(df_limpio[columna].mode().iloc[0] if not df_limpio[columna].
mode().empty else 0)
```

```
                            elif rellenar_numericos == "Cero":
                                df_limpio[columna] = df_limpio[columna].
fillna(0)

                            acciones_realizadas.append(f"    Rellenados
{faltantes_antes} valores faltantes en '{columna}' con {
rellenar_numericos.lower()}")

            # Rellenar datos de texto
            if rellenar_texto != "No rellenar":
                columnas_texto = df_limpio.select_dtypes(include=['
object']).columns
                for columna in columnas_texto:
                    faltantes_antes = df_limpio[columna].isnull().
sum()
                    if faltantes_antes > 0:
                        if rellenar_texto == "Moda":
                            df_limpio[columna] = df_limpio[columna].
fillna(df_limpio[columna].mode().iloc[0] if not df_limpio[columna].
mode().empty else "Sin datos")
                        elif rellenar_texto == "Texto personalizado"
:
                            df_limpio[columna] = df_limpio[columna].
fillna(texto_personalizado)

                        acciones_realizadas.append(f"    Rellenados
{faltantes_antes} valores faltantes en '{columna}'")

            # Mostrar resultados
            st.subheader("       Resultados de la Limpieza")

            if acciones_realizadas:
                for accion in acciones_realizadas:
                    st.write(accion)

                # Comparaci n antes/despu s
                col1, col2 = st.columns(2)

                with col1:
                    st.markdown("**       Antes de la limpieza:**")
                    st.metric("Filas", len(df))
                    st.metric("Datos faltantes", df.isnull().sum().
sum())
                    st.metric("Duplicados", df.duplicated().sum())

                with col2:
                    st.markdown("**    Despu s de la limpieza:**")
                    st.metric("Filas", len(df_limpio))
                    st.metric("Datos faltantes", df_limpio.isnull().
sum().sum())
                    st.metric("Duplicados", df_limpio.duplicated().
sum())

                # Mostrar datos limpios
                st.subheader("       Datos Limpios")
                st.dataframe(df_limpio.head(10), use_container_width
=True)

```

```
                    # Guardar datos limpios en session state
                    st.session_state['datos_limpios'] = df_limpio

                    # Bot n para descargar
                    csv_limpio = df_limpio.to_csv(index=False)
                    st.download_button(
                        label="       Descargar Datos Limpios",
                        data=csv_limpio,
                        file_name="datos_limpios.csv",
                        mime="text/csv"
                    )

                else:
                    st.info("       No se realizaron cambios. Tus datos ya est n limpios.")

        with tab5:
            st.subheader("       Reporte Final de Limpieza")

            # Generar reporte final
            st.markdown("###        Resumen Ejecutivo")

            # Crear m tricas de resumen
            col1, col2, col3, col4 = st.columns(4)

            with col1:
                calidad_general = 100 - reporte['porcentaje_faltantes']
                st.metric("       Calidad General", f"{calidad_general:.1f}%")

            with col2:
                problemas_totales = reporte['datos_faltantes'] + reporte['filas_duplicadas']
                st.metric("       Problemas Detectados", f"{problemas_totales:,}")

            with col3:
                columnas_problematicas = len(reporte['columnas_con_faltantes'])
                st.metric("       Columnas con Problemas", columnas_problematicas)

            with col4:
                if 'datos_limpios' in st.session_state:
                    mejora = len(df) - len(st.session_state['datos_limpios'])
                    st.metric("       Filas Eliminadas", mejora)
                else:
                    st.metric("       Filas Eliminadas", "No aplicado")

            # Reporte detallado
            st.markdown("###        Reporte Detallado")

            reporte_detallado = f"""
            **       REPORTE DE CALIDAD DE DATOS**

            **       Informaci n General:**
            - Total de filas: {reporte['filas_totales']:,}
```

```
574             - Total de columnas: {reporte['columnas_totales']:,}
575             - Total de celdas: {reporte['celdas_totales']:,}
576 
577             **    Problemas Detectados:**
578             - Datos faltantes: {reporte['datos_faltantes']:,} ({reporte
    ['porcentaje_faltantes']:.1f}%)
579             - Filas duplicadas: {reporte['filas_duplicadas']:,} ({
    reporte['porcentaje_duplicados']:.1f}%)
580 
581             **       An lisis por Columna:**
582             """
583 
584             for columna, faltantes in reporte['columnas_con_faltantes'].
    items():
585                 porcentaje = (faltantes / len(df)) * 100
586                 reporte_detallado += f"\n- {columna}: {faltantes}
    faltantes ({porcentaje:.1f}%)"
587 
588             if len(reporte['columnas_con_faltantes']) == 0:
589                 reporte_detallado += "\n-     No hay columnas con datos
    faltantes"
590 
591             # Recomendaciones
592             reporte_detallado += f"""
593 
594             **       Recomendaciones:**
595             """
596 
597             if reporte['porcentaje_faltantes'] < 5:
598                 reporte_detallado += "\n-     Calidad excelente.
    M nimos problemas detectados."
599             elif reporte['porcentaje_faltantes'] < 20:
600                 reporte_detallado += "\n-        Calidad buena.
    Considerar limpiar datos faltantes."
601             elif reporte['porcentaje_faltantes'] < 50:
602                 reporte_detallado += "\n-        Calidad regular.
    Limpieza necesaria antes del an lisis."
603             else:
604                 reporte_detallado += "\n-        Calidad cr tica.
    Revisar proceso de recolecci n de datos."
605 
606             if reporte['filas_duplicadas'] > 0:
607                 reporte_detallado += f"\n-           Eliminar {reporte['
    filas_duplicadas']} filas duplicadas"
608 
609             if len(columnas_numericas) > 0:
610                 reporte_detallado += "\n-        Revisar outliers en
    columnas num ricas"
611 
612             st.markdown(reporte_detallado)
613 
614             # Gr fico de resumen
615             st.markdown("###        Visualizaci n del Reporte")
616 
617             # Crear gr fico de pastel para mostrar distribuci n de
    problemas
618             fig = go.Figure(data=[go.Pie(
619                 labels=['Datos Completos', 'Datos Faltantes', '
```

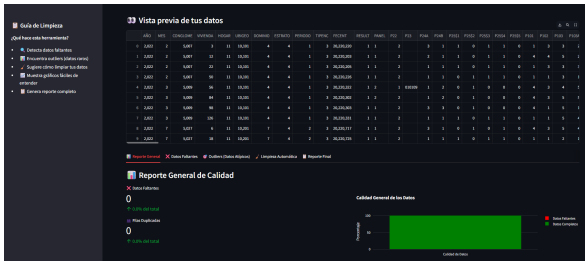
```
    Duplicados'],
620                 values=[
621                     reporte['celdas_totales'] - reporte['datos_faltantes
    '] - reporte['filas_duplicadas'],
622                     reporte['datos_faltantes'],
623                     reporte['filas_duplicadas']
624                 ],
625                 hole=0.3,
626                 marker_colors=['green', 'red', 'orange']
627             )])
628 
629             fig.update_layout(
630                 title="Distribuci n de Calidad de Datos",
631                 annotations=[dict(text='Calidad<br>General', x=0.5, y
    =0.5, font_size=16, showarrow=False)]
632             )
633 
634             st.plotly_chart(fig, use_container_width=True)
635 
636             # Descargar reporte
637             st.download_button(
638                 label="       Descargar Reporte Completo",
639                 data=reporte_detallado,
640                 file_name="reporte_limpieza_datos.txt",
641                 mime="text/plain"
642             )
643 
644             # Conclusiones finales
645             st.markdown("###        Conclusiones y Pr ximos Pasos")
646 
647             conclusiones = []
648 
649             if reporte['porcentaje_faltantes'] < 5 and reporte['
    filas_duplicadas'] == 0:
650                 conclusiones.append("       **  Excelente  !** Tus datos
    est n en muy buena condici n.")
651             elif reporte['porcentaje_faltantes'] < 20:
652                 conclusiones.append("    **Buena calidad** general, solo
     necesitas limpieza menor.")
653             else:
654                 conclusiones.append("       **Necesitas limpieza** antes
     de hacer an lisis importantes.")
655 
656             if len(columnas_numericas) > 0:
657                 conclusiones.append("       **Revisa los outliers** en
    la pesta a correspondiente.")
658 
659             if reporte['filas_duplicadas'] > 0:
660                 conclusiones.append("          **Elimina los duplicados
    ** para evitar sesgos en tu an lisis.")
661 
662             conclusiones.append("       **Usa la limpieza autom tica**
    para mejorar la calidad de tus datos.")
663 
664             for conclusion in conclusiones:
665                 st.markdown(conclusion)
666 
667             # Recordatorio importante
```

```
668             st.warning("       **Recordatorio importante**: Siempre revisa manualmente los cambios propuestos antes de aplicarlos a tus datos originales.")
669
670     except Exception as e:
671         st.error(f"       Hubo un error al procesar tu archivo: {str(e)}")
672         st.markdown("**Posibles soluciones:**")
673         st.markdown("- **Problema de codificaci n**: Guarda tu archivo CSV como UTF-8")
674         st.markdown("- **Archivo da ado**: Verifica que tu archivo CSV no est  corrupto")
675         st.markdown("- **Formato incorrecto**: Aseg rate de que sea un archivo CSV v lido")
676         st.markdown("- **Caracteres especiales**: Evita caracteres especiales en nombres de columnas")
677
678         # Informaci n t cnica
679         st.markdown("###        Informaci n t cnica:")
680         st.code(f"Error espec fico: {str(e)}", language='text')
681
682 else:
683     # P gina de inicio cuando no hay archivo
684     st.info("       Sube un archivo CSV para comenzar el an lisis de limpieza")
685
686     st.markdown("###          Qu   son los datos sucios?")
687     st.markdown("""
688     Los datos "sucios" son datos que tienen problemas como:
689
690     - **    Datos faltantes**: Celdas vac as o con valores nulos
691     - **       Duplicados**: Filas que se repiten exactamente
692     - **       Outliers**: Valores muy diferentes al resto (datos at picos)
693     - **       Inconsistencias**: Diferentes formatos para el mismo tipo de dato
694     - **       Errores de tipeo**: Nombres mal escritos o caracteres raros
695     """)
696
697     st.markdown("###         Por   qu   limpiar los datos?")
698     st.markdown("""
699     - **       An lisis  m  s precisos**: Resultados  m  s confiables
700     - **       Mejores decisiones**: Conclusiones basadas en datos correctos
701     - **    Procesamiento  m  s  r pido**: Menos datos problem ticos
702     - **       Patrones  m  s claros**: Tendencias  m  s  f ciles de identificar
703     """)
704
705     st.markdown("###        Ejemplo de datos que necesitan limpieza:")
706
707     ejemplo_sucio = """Nombre,Edad,Salario,Ciudad
708 Juan,25,45000,Madrid
709 Mar a,,55000,Barcelona
710 Pedro,35,65000,Madrid
711 Ana,28,50000,
712 Juan,25,45000,Madrid
```

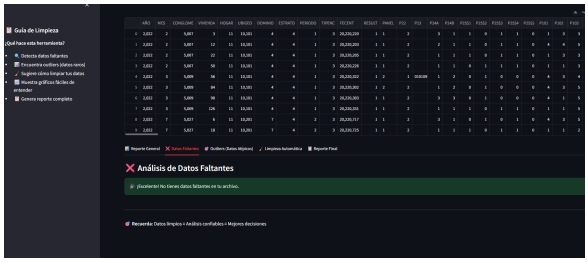
```
Carlos,200,70000,Valencia"""

    st.code(ejemplo_sucio, language='csv')

    st.markdown("**       Problemas en este ejemplo:**")
    st.markdown("- Edad faltante para Mar a")
    st.markdown("- Ciudad faltante para Ana")
    st.markdown("- Juan est  duplicado")
    st.markdown("- Carlos tiene 200 a os (posible outlier)")

    st.markdown("###        Lo que esta herramienta hace por ti:")
    st.markdown("""
    - **        Detecta autom ticamente** todos los problemas
    - **        Muestra gr ficos** f ciles de entender
    - **        Limpia autom ticamente** con un clic
    - **        Genera reportes** detallados
    - **        Da recomendaciones** espec ficas
    """)

# Footer
st.markdown("---")
st.markdown("       **Recuerda**: Datos limpios = An lisis confiables =
    Mejores decisiones")
```

Listing 2: Limpieza de Datos y Detección de Outliers

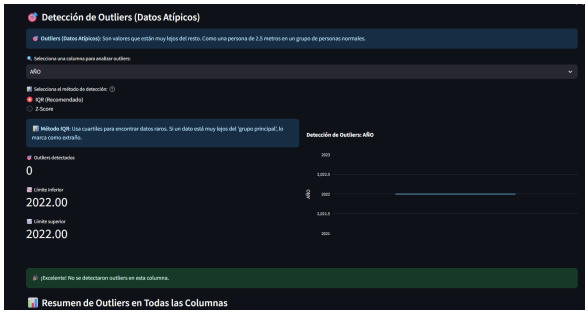
2.2. Capturas de Pantalla



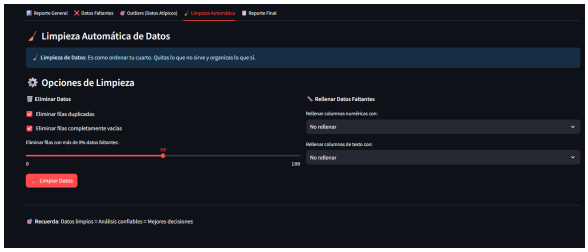
(a) Reporte general de calidad de datos



(b) Mapa de calor de datos faltantes



(a) Detección de outliers con método IQR



(b) Opciones de limpieza automática

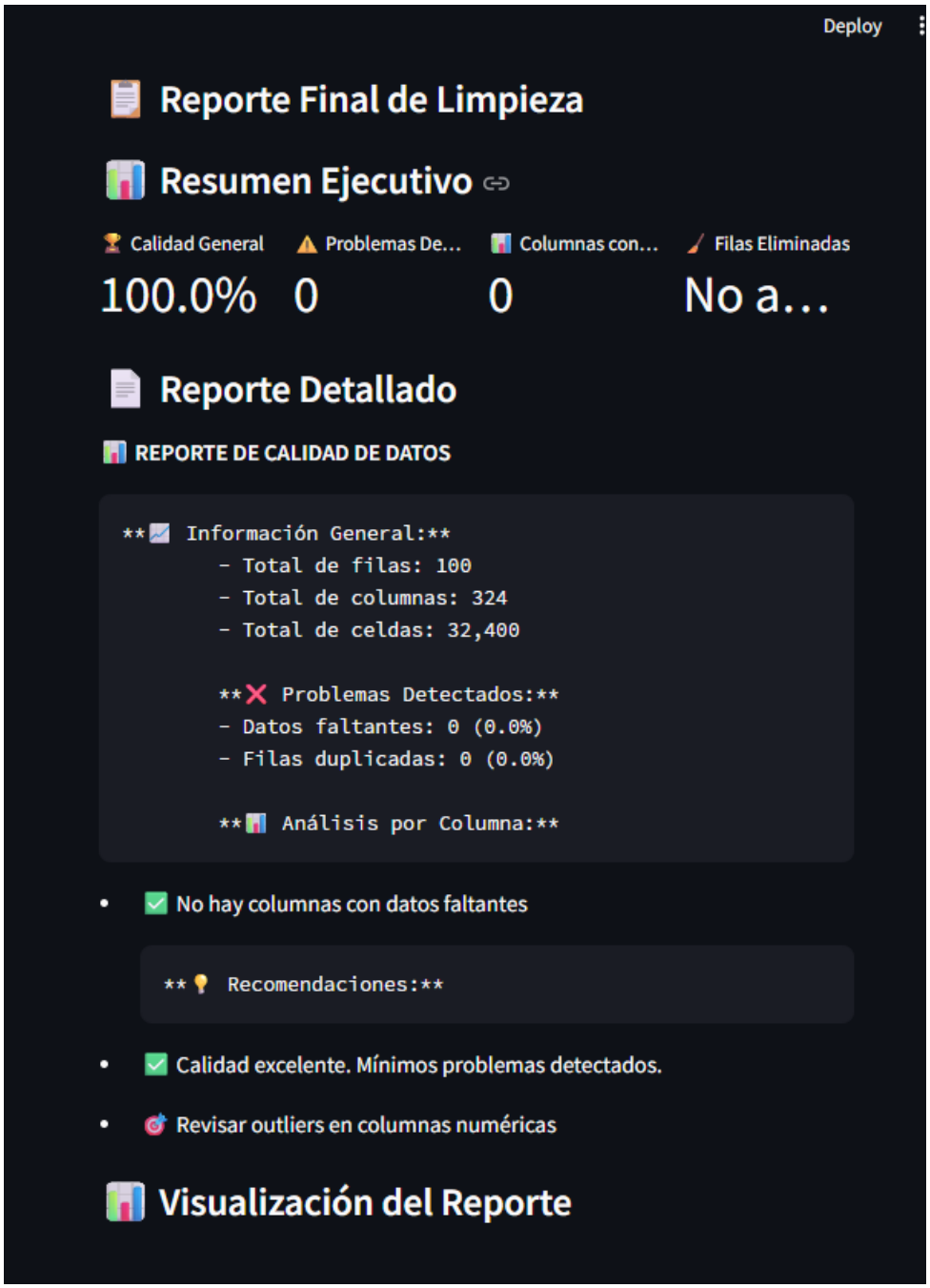


Figura 6: Reporte final con métricas de mejora

2.3. Análisis Técnico

Aspectos técnicos destacados:

- **Detección de outliers múltiple:** Implementa tanto el método IQR como Z-Score, permitiendo al usuario elegir el más apropiado según sus necesidades.
- **Visualización avanzada:** Utiliza Plotly para crear gráficos interactivos que permiten explorar los datos de forma dinámica, incluyendo mapas de calor para patrones de datos faltantes.
- **Estrategias de imputación:** Ofrece múltiples opciones para el tratamiento de

valores faltantes: eliminación, imputación por media/mediana/moda, o valores personalizados.

- **Análisis de calidad:** Calcula métricas de calidad de datos y genera reportes automáticos con recomendaciones específicas basadas en los problemas encontrados.
- **Preservación de datos:** Mantiene los datos originales intactos y trabaja con copias, permitiendo comparaciones antes/después de la limpieza.

3. Análisis Visual Interactivo de Datos (ENAHO)

Propósito: Facilitar la exploración y comprensión de conjuntos de datos CSV (especialmente los de encuestas como ENAHO) mediante visualizaciones interactivas, permitiendo a usuarios de cualquier nivel técnico realizar análisis exploratorios sin necesidad de programar.

Funcionalidades principales:

- **Carga robusta de archivos CSV:** Sube y lee archivos CSV automáticamente, gestionando diversas codificaciones de caracteres para evitar errores.
- **Normalización de nombres de columnas:** Limpia y estandariza los nombres de las columnas para un manejo más sencillo y consistente.
- **Matriz de correlación dinámica:** Genera un mapa de calor interactivo que muestra las relaciones lineales entre las variables numéricas seleccionadas.
- **Análisis de tendencias y distribuciones por categoría:** Visualiza cómo una variable numérica se comporta a través de diferentes grupos (ej. estrato social o región) usando gráficos de línea y de caja.
- **Mapa de calor de promedios combinados:** Explora el promedio de una variable numérica para cada combinación de dos categorías distintas, identificando patrones clave.

Características técnicas:

- **Framework Streamlit:** Desarrollado sobre Streamlit para una interfaz de usuario web intuitiva y fácil de usar.
- **Manipulación de datos con Pandas:** Utiliza la potente librería Pandas para el procesamiento, limpieza y análisis de los DataFrames.
- **Visualizaciones interactivas con Plotly Express:** Emplea Plotly para crear gráficos dinámicos y de alta calidad que facilitan la exploración de los datos.
- **Manejo integral de errores:** Incorpora validaciones y mensajes claros para guiar al usuario ante problemas de carga o selección de datos.

3.1. Código Fuente

```
import streamlit as st
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
import numpy as np
import io

st.set_page_config(page_title="An lisis ENAHO 2022", layout="wide")
st.title("       An lisis Visual de Datos ENAHO")

# Subida de archivo
archivo = st.file_uploader("       Sube tu archivo CSV", type=["csv"])

df = None # Inicializar df fuera del bloque if

if archivo is not None:
    try:
        # Intentar cargar con codificaciones comunes
        codificaciones = ['utf-8', 'latin1', 'iso-8859-1', 'cp1252']
        codificacion_exitosa = None
        for encoding in codificaciones:
            try:
                archivo.seek(0)
                df = pd.read_csv(archivo, encoding=encoding, low_memory=
    False)
                codificacion_exitosa = encoding
                st.success(f"    Datos cargados correctamente con
    codificaci n: {encoding}")
                break
            except UnicodeDecodeError:
                continue
            except Exception as e:
                st.error(f"Error al leer con {encoding}: {e}")
                df = None

        if df is None:
            st.error("    No se pudo cargar el archivo CSV con ninguna
    de las codificaciones intentadas. Por favor, verifica el formato y la
     codificaci n de tu archivo.")
            st.stop()

        st.dataframe(df.head())

        # --- Limpiar los nombres de las columnas para mayor robustez
    ---
        original_columns = df.columns.tolist()
        df.columns = df.columns.str.strip().str.replace(' ', '_').str.
    replace('.', '_').str.replace('-', '_')

        if isinstance(df.columns, pd.MultiIndex):
            df.columns = df.columns.map('_'.join)
            st.info("Detectado y simplificado un  ndice  de columnas
    multi-nivel.")

        if original_columns != df.columns.tolist():
            st.info("Se han limpiado los nombres de las columnas. Los
```

```
    nuevos nombres son: " + ", ".join(df.columns.tolist()))
50
51          # Selecci n de todas las columnas num ricas disponibles
52          todas_columnas_numericas = df.select_dtypes(include=np.number).
    columns.tolist()
53
54          # --- 1. Matriz de correlaci n ---
55          st.subheader("1       Matriz de correlaci n de Variables
    Num ricas Relevantes")
56          st.markdown("Selecciona las variables num ricas que consideres
    m s relevantes para analizar su correlaci n.")
57
58          # Nuevo: Selector m ltiple para elegir las columnas a
    correlacionar
59          selected_numeric_columns = st.multiselect(
60              "Elige las columnas num ricas para la matriz de
    correlaci n:",
61              todas_columnas_numericas,
62              default=todas_columnas_numericas[:5] if len(
    todas_columnas_numericas) > 0 else [] # Selecciona las primeras 5 por
     defecto si existen
63          )
64
65          if len(selected_numeric_columns) >= 2:
66              corr = df[selected_numeric_columns].corr()
67
68              fig_corr = px.imshow(
69                  corr,
70                  text_auto=".2f", # Formato a 2 decimales
71                  aspect="auto",
72                  color_continuous_scale='RdBu_r', # Escala de color Rojo-
    Azul invertida
73                  title="Mapa de Correlaci n entre Variables Num ricas
    Seleccionadas"
74              )
75
76              # Ajustar el tama o del texto dentro de las celdas y las
    etiquetas de los ejes
77              # El tama o de la fuente se har  m s grande si hay menos
    columnas
78              font_size_labels = 12 if len(selected_numeric_columns) < 15
    else 8
79              font_size_text = 12 if len(selected_numeric_columns) < 10
    else 8
80
81              fig_corr.update_traces(textfont_size=font_size_text)
82              fig_corr.update_xaxes(tickangle=-45, tickfont_size=
    font_size_labels)
83              fig_corr.update_yaxes(tickfont_size=font_size_labels)
84
85              # Ajustar layout para mejor visualizaci n
86              fig_corr.update_layout(
87                  height=max(500, len(selected_numeric_columns) * 40), #
    Altura din mica
88                  width=max(500, len(selected_numeric_columns) * 40), #
    Ancho din mico
89                  title_x=0.5,
90                  title_font_size=20
```

```
            )

            st.plotly_chart(fig_corr, use_container_width=True)

            st.markdown("""
            **Interpretaci n de la Matriz de Correlaci n:**
            - **Valores cercanos a 1:** Indican una fuerte correlaci n positiva (cuando una variable aumenta, la otra tambi n tiende a aumentar).
            - **Valores cercanos a -1:** Indican una fuerte correlaci n negativa (cuando una variable aumenta, la otra tiende a disminuir).
            - **Valores cercanos a 0:** Indican poca o ninguna correlaci n lineal entre las variables.
            """)

        else:
            st.warning("Por favor, selecciona al menos 2 columnas num ricas para generar la matriz de correlaci n.")

        # --- Validaci n y selecci n de la columna de resultado para los siguientes gr ficos ---
        available_columns = df.columns.tolist()

        # Filtrar solo columnas num ricas para el resultado por defecto, si es posible
        opciones_resultado = [col for col in available_columns if pd.api.types.is_numeric_dtype(df[col])]

        if not opciones_resultado:
            st.error("No se encontraron columnas num ricas para usar como 'Resultado de la entrevista'. Algunos gr ficos no podr n generarse.")
            col_resultado_comun = None
        else:
            col_resultado_comun = st.selectbox(
                "Selecciona la columna de **Resultado de la entrevista** (para gr ficos 2, 3 y 4)",
                opciones_resultado, # Limitar a columnas num ricas
                key="resultado_select_comun"
            )
            # Asegurarse de que la columna seleccionada sigue siendo num rica (por si el usuario elige otra cosa en un set distinto)
            if not pd.api.types.is_numeric_dtype(df[col_resultado_comun]):
                st.warning("La columna de Resultado seleccionada no es num rica. Por favor, selecciona una columna num rica.")
                col_resultado_comun = None # Invalidar si no es num rica

        if col_resultado_comun: # Solo proceder si hay una columna de resultado num rica v lida
            # --- 2. Gr fico de l nea: Estrato vs Resultado ---
            st.subheader("2        Influye   el estrato social en el resultado?")

            col_estrato = st.selectbox("Selecciona la columna de Estrato Social", available_columns, key="estrato_select")

```

```
131             st.info(f"DEBUG: Columna 'Estrato Social' seleccionada: '{
    col_estrato}'")
132             st.info(f"DEBUG: Tipo de dato de '{col_estrato}': {df[
    col_estrato].dtype}")
133             st.info(f"DEBUG: Columna 'Resultado' seleccionada: '{
    col_resultado_comun}'")
134             st.info(f"DEBUG: Tipo de dato de '{col_resultado_comun}': {
    df[col_resultado_comun].dtype}")
135
136             if col_estrato not in df.columns:
137                 st.error("La columna de estrato seleccionada no es
    v lida.")
138             else:
139                 if not pd.api.types.is_numeric_dtype(df[col_estrato])
    and not pd.api.types.is_categorical_dtype(df[col_estrato]):
140                     try:
141                         df[col_estrato] = df[col_estrato].astype('
    category')
142                         st.info(f"DEBUG: Columna '{col_estrato}'
    convertida a tipo 'category' para agrupaci n.")
143                     except Exception as e:
144                         st.error(f"Error al intentar convertir '{
    col_estrato}' a categor a: {e}. Esto podr a causar problemas en la
    agrupaci n.")
145
146                 df_line = df[[col_estrato, col_resultado_comun]].dropna
    ()
147
148                 if df_line.empty:
149                     st.warning(f"No hay datos disponibles para '{
    col_estrato}' y '{col_resultado_comun}' despu s de eliminar valores
    faltantes.")
150                 else:
151                     if df_line[col_estrato].nunique() < 2:
152                         st.warning(f"La columna '{col_estrato}' tiene
    muy pocos valores  nicos  ({df_line[col_estrato].nunique()}) para un
    an lisis significativo. Necesitas al menos 2 grupos.")
153                     else:
154                         try:
155                             st.info(f"DEBUG: Tipo de df_line['{
    col_estrato}'] antes de groupby: {df_line[col_estrato].dtype}")
156                             st.info(f"DEBUG: Shape de df_line['{
    col_estrato}'] antes de groupby: {df_line[col_estrato].shape}")
157
158                             df_grouped = df_line.groupby(col_estrato)[
    col_resultado_comun].mean().reset_index()
159
160                             fig2 = px.line(df_grouped, x=col_estrato, y=
    col_resultado_comun,
161                                            markers=True, title=f"
    Tendencia del promedio de {col_resultado_comun} seg n {col_estrato}"
    )
162                             st.plotly_chart(fig2, use_container_width=
    True)
163                             st.markdown(f"""
164                             **Interpretaci n:** Este gr fico de l nea
     muestra c mo el valor promedio de **'{col_resultado_comun}'**
165                              var a a trav s de las diferentes
```

```
    categor as de **'{col_estrato}'**.
166                         Puedes observar tendencias o patrones en los
     resultados a medida que cambias de estrato.
167                         """)
168                     except Exception as e:
169                         st.error(f"Error al generar el gr fico de
    l nea: {e}. Esto podr a deberse a datos inesperados en la columna
    de agrupaci n, o si la columna '{col_estrato}' tiene una estructura
    compleja (ej. MultiIndex).")
170                         st.info(f"Aseg rate de que la columna '{
    col_estrato}' sea un tipo de dato simple y 1-dimensional (categ rico
    , texto, o num rico discreto).")
171
172
173             # --- 3. Gr fico de caja: Regi n vs Resultado ---
174             st.subheader("3        Hay  diferencias por regi n?")
175             col_region = st.selectbox("Selecciona la columna de Regi n"
    , available_columns, key="region_select")
176
177             if col_region not in df.columns:
178                 st.error("La columna de regi n seleccionada no es
    v lida.")
179             else:
180                 if not pd.api.types.is_numeric_dtype(df[col_region]) and
     not pd.api.types.is_categorical_dtype(df[col_region]):
181                     try:
182                         df[col_region] = df[col_region].astype('category
    ')
183                         st.info(f"DEBUG: Columna '{col_region}'
    convertida a tipo 'category' para agrupaci n.")
184                     except Exception as e:
185                         st.error(f"Error al intentar convertir '{
    col_region}' a categor a: {e}. Esto podr a causar problemas en la
    agrupaci n.")
186
187                 df_box = df[[col_region, col_resultado_comun]].dropna()
188
189                 if df_box.empty:
190                     st.warning(f"No hay datos disponibles para '{
    col_region}' y '{col_resultado_comun}' despu s de eliminar valores
    faltantes.")
191                 else:
192                     if df_box[col_region].nunique() < 2:
193                         st.warning(f"La columna '{col_region}' tiene muy
     pocos valores  nicos  ({df_box[col_region].nunique()}) para un
    an lisis significativo. Necesitas al menos 2 grupos.")
194                     else:
195                         fig3 = px.box(df_box, x=col_region, y=
    col_resultado_comun,
196                                       title=f"Distribuci n de {
    col_resultado_comun} por {col_region}")
197                         st.plotly_chart(fig3, use_container_width=True)
198                         st.markdown(f"""
199                         **Interpretaci n:** Los diagramas de caja
    muestran la distribuci n de **'{col_resultado_comun}'**
200                         para cada categor a de **'{col_region}'**.
    Puedes identificar:
201                         - **Mediana:** La l nea central de la caja.
```

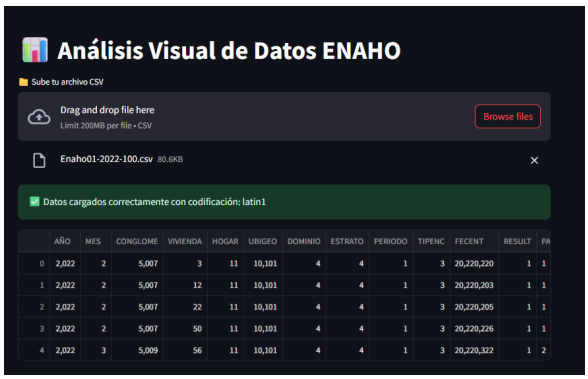
```
202                         - **Rango intercuart lico (IQR):** La altura de
    la caja (donde se concentra el 50% central de los datos).
203                         - **Bigotes:** La dispersi n de los datos fuera
    del IQR.
204                         - **Puntos (Outliers):** Valores at picos que
    est n significativamente fuera del rango.
205                         Esto te ayuda a ver r pidamente si hay
    diferencias en la tendencia central y la dispersi n entre regiones.
206                         """)
207
208             # --- 4. Visualizaci n Combinada: Mapa de Calor de Medias
    ---
209             st.subheader("4       Mapa de Calor de Medias por Dos
    Categor as")
210             st.markdown("Este gr fico muestra c mo el promedio de una
    variable num rica se distribuye entre las combinaciones de dos
    variables categ ricas (o discretas).")
211
212             col_heatmap_x = st.selectbox("Selecciona la columna para el
    Eje X (Categor a 1)", available_columns, key="heatmap_x")
213             col_heatmap_y = st.selectbox("Selecciona la columna para el
    Eje Y (Categor a 2)", available_columns, key="heatmap_y")
214             col_heatmap_value = st.selectbox("Selecciona la columna
    num rica para el valor (Ej. Resultado, Ingreso)", opciones_resultado
    , key="heatmap_value")
215
216             if col_heatmap_x and col_heatmap_y and col_heatmap_value:
217                 if col_heatmap_x not in df.columns or col_heatmap_y not
    in df.columns or col_heatmap_value not in df.columns:
218                     st.error("Una de las columnas seleccionadas para el
    mapa de calor no es v lida.")
219                 elif not pd.api.types.is_numeric_dtype(df[
    col_heatmap_value]):
220                     st.warning("La columna de valor para el mapa de
    calor debe ser num rica.")
221                 else:
222                     # Convertir a categor a si no lo son ya, para
    asegurar un comportamiento de agrupaci n predecible
223                     temp_df = df[[col_heatmap_x, col_heatmap_y,
    col_heatmap_value]].copy()
224                     if not pd.api.types.is_numeric_dtype(temp_df[
    col_heatmap_x]) and not pd.api.types.is_categorical_dtype(temp_df[
    col_heatmap_x]):
225                         temp_df[col_heatmap_x] = temp_df[col_heatmap_x].
    astype('category')
226                     if not pd.api.types.is_numeric_dtype(temp_df[
    col_heatmap_y]) and not pd.api.types.is_categorical_dtype(temp_df[
    col_heatmap_y]):
227                         temp_df[col_heatmap_y] = temp_df[col_heatmap_y].
    astype('category')
228
229                     pivot_table = temp_df.pivot_table(index=
    col_heatmap_y, columns=col_heatmap_x, values=col_heatmap_value,
    aggfunc='mean')
230
231                     if not pivot_table.empty:
232                         fig_heatmap_custom = px.imshow(
233                             pivot_table,
```

```
                            text_auto=".2f",
                            aspect="auto",
                            color_continuous_scale=px.colors.sequential.Viridis,
                            title=f"Promedio de {col_heatmap_value} por {col_heatmap_y} y {col_heatmap_x}"
                        )
                        fig_heatmap_custom.update_xaxes(side="top")
                        fig_heatmap_custom.update_layout(height=600, width=800)
                        st.plotly_chart(fig_heatmap_custom, use_container_width=True)
                        st.markdown(f"""
                        **Interpretaci n:** Este mapa de calor muestra el promedio de **'{col_heatmap_value}'**
                        para cada combinaci n  nica  de **'{col_heatmap_x}'** y **'{col_heatmap_y}'**.
                        Los colores m s intensos (o m s claros, dependiendo de la escala) indican promedios m s altos (o bajos).
                        Es  til  para identificar r pidamente combinaciones de categor as con resultados particulares.
                        """)
                    else:
                        st.warning("No hay suficientes datos para generar el mapa de calor con las columnas seleccionadas despu s de la agrupaci n.")
            else:
                st.info("Selecciona tres columnas (dos categ ricas/discretas y una num rica) para generar el mapa de calor de medias.")


            st.markdown("    **Interpretaci n General**: Puedes ajustar los campos para personalizar el an lisis  seg n  tu CSV.")

    except Exception as e:
        st.error(f"Ocurri  un error inesperado al procesar los datos o generar gr ficos: {e}")
        st.info("Aseg rate de que tu archivo CSV tenga el formato esperado y que las columnas seleccionadas sean apropiadas para el tipo de gr fico.")
else:
    st.info("Por favor, sube un archivo CSV para comenzar.")
```

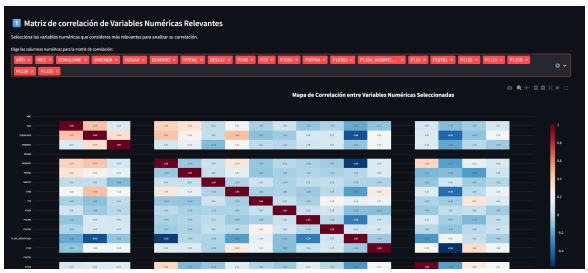
Listing 3: Tercera Aplicación

3.2. Capturas de Pantalla

(a) [Descripción de la imagen 1]



(b) [Descripción de la imagen 2]



(a) [Descripción de la imagen 3]

(b) [Descripción de la imagen 4]

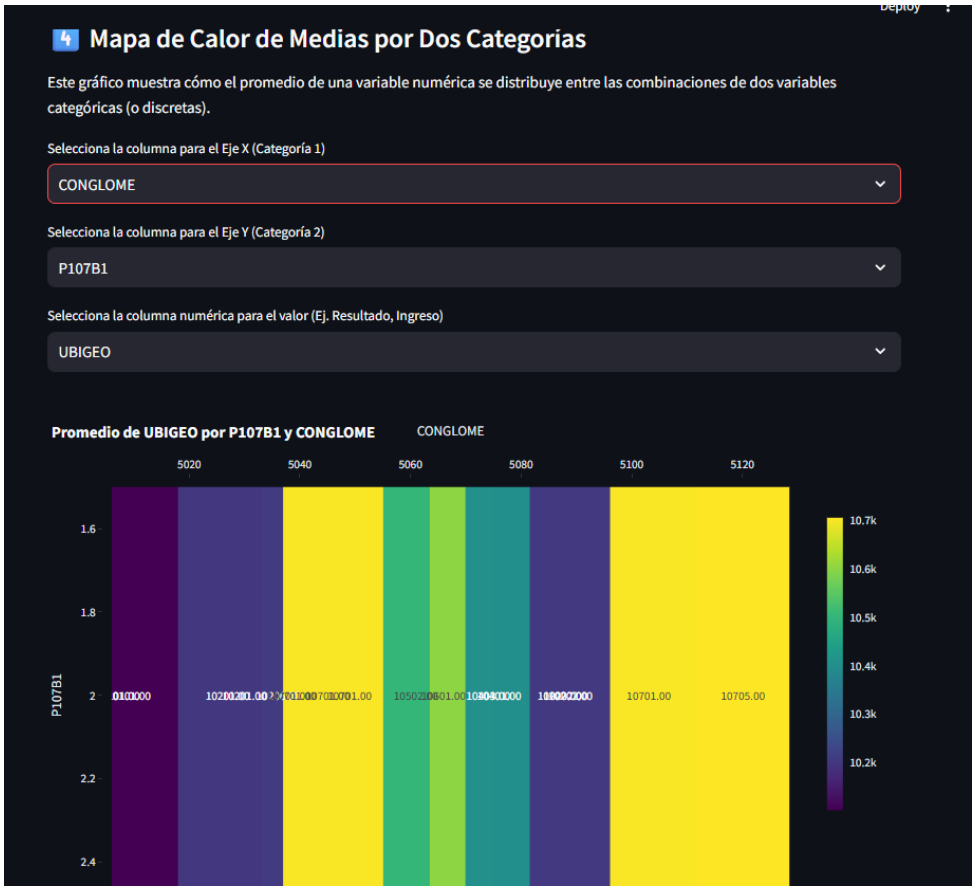


Figura 9: [Descripción de la imagen 5]

3.3. Análisis Técnico

Aspectos técnicos destacados:

- **Framework Streamlit:** Permite crear rápidamente una **interfaz web interactiva** a partir de código Python, simplificando el desarrollo de la UI.
- **Gestión de Datos con Pandas:** Utiliza **Pandas** para un manejo eficiente de DataFrames, incluyendo carga robusta (multi-codificación), limpieza y preprocesamiento de datos.
- **Visualizaciones Interactivas con Plotly:** Emplea **Plotly Express y Graph Objects** para generar gráficos dinámicos y explorables, mejorando la interacción del usuario con los datos.
- **Manejo Robusto de Errores y Flexibilidad:** Incluye lógica para **adaptarse a diversos CSVs** (limpieza de columnas, detección de tipos de datos) y proporciona retroalimentación clara ante errores.
- **Arquitectura Basada en Componentes:** La aplicación se construye a partir de **componentes modulares de Streamlit**, lo que facilita la adición o modificación de nuevas funcionalidades de análisis.

4. Análisis de Gastos en Servicios Básicos con MANOVA

Propósito: Permitir a los usuarios **analizar diferencias significativas en patrones de gasto multivariados** (varios servicios básicos a la vez) entre distintos grupos (ej. estratos socioeconómicos), ofreciendo una plataforma interactiva para la ejecución e interpretación del MANOVA.

Funcionalidades principales:

- **Carga Flexible de Datos:** Admite archivos CSV con manejo automático de múltiples codificaciones y separadores, o permite usar datos de ejemplo sintéticos.
- **Preprocesamiento Automatizado:** Realiza limpieza de nombres de columnas y validación de datos para asegurar la idoneidad para MANOVA.
- **Análisis MANOVA Interactivo:** Ejecuta el Análisis Multivariado de Varianza y presenta sus resultados clave (ej. Traza de Pillai, p-valor).
- **Visualizaciones Exploratorias y de Diagnóstico:** Ofrece gráficos (pastel, barras, violín) y métricas para entender la distribución de datos y verificar supuestos.
- **Interpretación Guiada y Post-Hoc:** Proporciona explicaciones claras de los resultados del MANOVA y realiza análisis ANOVA individuales por variable para un detalle adicional.

Características técnicas:

- **Framework Streamlit:** Facilita la creación de una **interfaz web intuitiva** y de fácil despliegue en Python.
- **Gestión de Datos con Pandas:** Utiliza **Pandas** para la manipulación, validación y preparación robusta de los datos.
- **Análisis Estadístico con Statsmodels y SciPy:** Implementa el MANOVA con ‘statsmodels.multivariate.manova‘ y un método de respaldo con ‘scipy.stats‘ para mayor robustez.
- **Visualización Interactiva con Plotly Express:** Genera **gráficos dinámicos** (mapas de calor, barras, violín) que permiten la exploración de los datos.
- **Manejo Inteligente de Errores:** Ofrece **mensajes contextuales** (errores, advertencias) para guiar al usuario ante problemas en los datos o el análisis.

4.1. Código Fuente

```
import streamlit as st
import pandas as pd
import numpy as np
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
from scipy.stats import multivariate_normal
from scipy import stats
```

```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from statsmodels.multivariate.manova import MANOVA
import warnings
warnings.filterwarnings('ignore')

# Configuraci n de la p gina
st.set_page_config(
    page_title="Storytelling: An lisis MANOVA de Servicios B sicos",
    page_icon="      ",
    layout="wide",
    initial_sidebar_state="expanded"
)

# T tulo principal
st.title("       Storytelling con MANOVA: An lisis de Gastos en
   Servicios B sicos")
st.markdown("---")

# Funci n para generar datos sint ticos (como ejemplo)
@st.cache_data
def generar_datos_ejemplo():
    """Genera datos sint ticos como ejemplo"""
    np.random.seed(42)
    n_hogares = 300

    # Definir grupos (estratos socioecon micos)
    grupos = np.random.choice(['Bajo', 'Medio', 'Alto'], n_hogares, p
   =[0.4, 0.4, 0.2])

    # Generar gastos base seg n grupo
    gastos_base = {
        'Bajo': {'agua': 50, 'luz': 80, 'gas': 40, 'internet': 30, '
   telefono': 25},
        'Medio': {'agua': 80, 'luz': 150, 'gas': 70, 'internet': 60, '
   telefono': 45},
        'Alto': {'agua': 120, 'luz': 250, 'gas': 100, 'internet': 100, '
   telefono': 70}
    }

    # Crear DataFrame
    data = []
    for i, grupo in enumerate(grupos):
        base = gastos_base[grupo]
        # Agregar variabilidad realista
        agua = np.random.normal(base['agua'], base['agua'] * 0.3)
        luz = np.random.normal(base['luz'], base['luz'] * 0.4)
        gas = np.random.normal(base['gas'], base['gas'] * 0.35)
        internet = np.random.normal(base['internet'], base['internet'] *
    0.2)
        telefono = np.random.normal(base['telefono'], base['telefono'] *
    0.25)

        # Asegurar valores positivos
        agua = max(20, agua)
        luz = max(30, luz)
        gas = max(15, gas)
```

```
        internet = max(10, internet)
        telefono = max(10, telefono)

        data.append({
            'hogar_id': i+1,
            'grupo': grupo,
            'agua': round(agua, 2),
            'luz': round(luz, 2),
            'gas': round(gas, 2),
            'internet': round(internet, 2),
            'telefono': round(telefono, 2),
            'total': round(agua + luz + gas + internet + telefono, 2)
        })

    return pd.DataFrame(data)

# Funci n para validar datos
def validar_datos(df):
    """Valida que los datos sean apropiados para MANOVA"""
    errores = []

    if df is None or df.empty:
        errores.append("El archivo est  vac o")
        return errores

    # Verificar que tenga al menos 3 filas
    if len(df) < 10:
        errores.append("Se necesitan al menos 10 observaciones para un
    an lisis confiable")

    # Verificar columnas num ricas
    columnas_numericas = df.select_dtypes(include=[np.number]).columns
    if len(columnas_numericas) < 2:
        errores.append("Se necesitan al menos 2 variables num ricas
    para MANOVA")

    return errores

# Funci n para limpiar nombres de columnas
def limpiar_nombres_columnas(df):
    """Limpia los nombres de columnas para evitar problemas con MANOVA
    """
    df_limpio = df.copy()

    # Diccionario para mapear nombres originales a nombres limpios
    mapeo_nombres = {}

    for col in df_limpio.columns:
        # Limpiar el nombre: solo letras, n meros y guiones bajos
        nombre_limpio = ''.join(c if c.isalnum() or c == '_' else '_'
    for c in str(col))
        # Asegurar que empiece con letra
        if nombre_limpio and not nombre_limpio[0].isalpha():
            nombre_limpio = 'var_' + nombre_limpio
        # Evitar nombres vac os
        if not nombre_limpio:
            nombre_limpio = f'variable_{len(mapeo_nombres)}'

```

```
114         mapeo_nombres[col] = nombre_limpio
115
116     # Renombrar columnas
117     df_limpio = df_limpio.rename(columns=mapeo_nombres)
118
119     return df_limpio, mapeo_nombres
120
121 # Funci n para realizar MANOVA
122 def realizar_manova(df, variables_dependientes, variable_grupo):
123     """Realiza el an lisis MANOVA"""
124     try:
125         # Limpiar nombres de columnas
126         df_limpio, mapeo = limpiar_nombres_columnas(df)
127
128         # Mapear nombres de variables a nombres limpios
129         variables_limpias = [mapeo[var] for var in
    variables_dependientes]
130         grupo_limpio = mapeo[variable_grupo]
131
132         # Verificar que tengamos suficientes datos
133         if len(df_limpio) < 10:
134             return None, "Se necesitan al menos 10 observaciones para un
     an lisis confiable"
135
136         # Verificar que cada grupo tenga suficientes observaciones
137         conteo_grupos = df_limpio[grupo_limpio].value_counts()
138         if conteo_grupos.min() < 3:
139             return None, f"Cada grupo debe tener al menos 3
    observaciones. Grupo con menos datos: {conteo_grupos.min()}"
140
141         # Eliminar filas con valores faltantes
142         df_limpio = df_limpio.dropna(subset=variables_limpias + [
    grupo_limpio])
143
144         if len(df_limpio) == 0:
145             return None, "No quedan datos despu s de eliminar valores
    faltantes"
146
147         # Crear f rmula para MANOVA
148         formula = f"{' + '.join(variables_limpias)} ~ {grupo_limpio}"
149
150         # Realizar MANOVA
151         manova = MANOVA.from_formula(formula, data=df_limpio)
152         resultado = manova.mv_test()
153
154         return resultado, None
155
156     except Exception as e:
157         # Intentar m todo alternativo con scipy
158         try:
159             return realizar_manova_scipy(df, variables_dependientes,
    variable_grupo)
160         except Exception as e2:
161             return None, f"Error en MANOVA: {str(e)}. Error alternativo:
     {str(e2)}"
162
163 # Funci n alternativa usando scipy
164 def realizar_manova_scipy(df, variables_dependientes, variable_grupo):
```

```
    """M todo alternativo para MANOVA usando scipy"""
    from scipy.stats import f

    # Preparar datos
    grupos = df[variable_grupo].unique()
    n_grupos = len(grupos)

    if n_grupos < 2:
        return None, "Se necesitan al menos 2 grupos para el an lisis"

    # Calcular matrices necesarias para MANOVA
    X = df[variables_dependientes].values
    y = df[variable_grupo].values

    # Calcular medias por grupo
    medias_grupo = {}
    for grupo in grupos:
        mask = y == grupo
        medias_grupo[grupo] = X[mask].mean(axis=0)

    # Media global
    media_global = X.mean(axis=0)

    # Matriz de suma de cuadrados entre grupos (H)
    H = np.zeros((len(variables_dependientes), len(
variables_dependientes)))
    for grupo in grupos:
        mask = y == grupo
        n_grupo = mask.sum()
        diff = medias_grupo[grupo] - media_global
        H += n_grupo * np.outer(diff, diff)

    # Matriz de suma de cuadrados dentro de grupos (E)
    E = np.zeros((len(variables_dependientes), len(
variables_dependientes)))
    for grupo in grupos:
        mask = y == grupo
        X_grupo = X[mask]
        for i in range(len(X_grupo)):
            diff = X_grupo[i] - medias_grupo[grupo]
            E += np.outer(diff, diff)

    # Calcular estad sticos
    try:
        # Pillai's Trace
        eigenvals = np.linalg.eigvals(H @ np.linalg.inv(E))
        pillai_trace = np.sum(eigenvals / (1 + eigenvals))

        # Aproximaci n F para Pillai's Trace
        p = len(variables_dependientes)
        df_hyp = p * (n_grupos - 1)
        df_err = len(df) - n_grupos

        # Asegurar que df_err - p + 1 no sea cero o negativo
        den_df_pillai = max(1, df_err - p + 1)

        # Evitar divisi n por cero para el c lculo de f_stat si
pillai_trace es 1 (separaci n perfecta)
```

```
220         if (1 - pillai_trace) == 0:
221             f_stat = np.inf
222         else:
223             f_stat = (pillai_trace / (p * (n_grupos - 1))) * ((
    den_df_pillai) / (1 - pillai_trace))
224 
225         # Manejar casos donde f_stat podr a ser muy grande o infinito
226         if np.isinf(f_stat):
227             p_valor = 0.0 # Altamente significativo
228         elif np.isnan(f_stat):
229             p_valor = np.nan # No se puede calcular
230         else:
231             p_valor = 1 - f.cdf(f_stat, df_hyp, den_df_pillai)
232 
233         # Crear DataFrame de resultados similar al de statsmodels
234         resultado = pd.DataFrame({
235             'Value': [pillai_trace],
236             'Num DF': [df_hyp],
237             'Den DF': [den_df_pillai],
238             'F Value': [f_stat],
239             'Pr > F': [p_valor]
240         }, index=[variable_grupo])
241 
242         return resultado, None
243 
244     except Exception as e:
245         return None, f"Error en c lculo alternativo: {str(e)}"
246 
247 # Sidebar para configuraci n
248 st.sidebar.header("       Configuraci n del An lisis")
249 
250 # Opci n para cargar datos
251 opcion_datos = st.sidebar.radio(
252     "  Cmo   quieres cargar los datos?",
253     ["       Subir mi archivo CSV", "       Usar datos de ejemplo"]
254 )
255 
256 # Cargar datos
257 df = None
258 if opcion_datos == "       Subir mi archivo CSV":
259     st.sidebar.subheader("       Carga tu archivo CSV")
260     archivo_csv = st.sidebar.file_uploader(
261         "Selecciona tu archivo CSV",
262         type=['csv'],
263         help="El archivo debe contener columnas num ricas para gastos y
    una columna categ rica para agrupar"
264     )
265 
266     # Opciones adicionales para la carga
267     with st.sidebar.expander("       Opciones avanzadas de carga"):
268         separador = st.selectbox(
269             "Separador del CSV:",
270             [",", ";", "\t", "|"],
271             help="Car cter que separa las columnas"
272         )
273 
274         decimal = st.selectbox(
275             "Separador decimal:",
```

```
            [".", ","],
            help="Car cter usado para decimales"
        )

    if archivo_csv is not None:
        try:
            # Intentar diferentes codificaciones
            codificaciones = ['utf-8', 'latin-1', 'iso-8859-1', 'cp1252', 'utf-16']
            df = None
            codificacion_exitosa = None

            for encoding in codificaciones:
                try:
                    # Resetear el puntero del archivo
                    archivo_csv.seek(0)
                    df = pd.read_csv(archivo_csv, encoding=encoding, sep=separador, decimal=decimal)
                    codificacion_exitosa = encoding
                    break
                except UnicodeDecodeError:
                    continue
                except Exception as e:
                    # Si es otro tipo de error, intentar con la siguiente codificaci n
                    continue

            if df is not None:
                st.sidebar.success(f"    Archivo cargado: {len(df)} filas")
                st.sidebar.info(f"       Codificaci n detectada: {codificacion_exitosa}")
            else:
                st.sidebar.error("    No se pudo leer el archivo con ninguna codificaci n est ndar")
                st.sidebar.markdown("""
                **Posibles soluciones:**
                1. Guarda el archivo como CSV UTF-8 desde Excel
                2. Usa un editor de texto para cambiar la codificaci n
                3. Verifica que el archivo sea realmente un CSV
                """)

        except Exception as e:
            st.sidebar.error(f"    Error al cargar el archivo: {str(e)}")
            st.sidebar.markdown("""
            **Posibles causas:**
            - Formato de archivo incorrecto
            - Archivo corrupto
            - Separadores no est ndar
            """)
    else:
        st.info("       Por favor, sube tu archivo CSV usando el panel lateral")
        st.markdown("""
        ###        Formato requerido del CSV:

        Tu archivo debe tener:
```

```
        - **Una columna categ rica** para agrupar (ej: 'grupo', '
    estrato', 'region')
        - **Varias columnas num ricas** con los gastos en servicios (ej
    : 'agua', 'luz', 'gas', etc.)

        **Ejemplo de estructura:**
        ```
        grupo,agua,luz,gas,internet,telefono
        Bajo,45.50,85.20,35.10,25.00,20.50
        Medio,78.30,145.80,65.40,55.20,42.10
        Alto,115.60,245.90,95.80,95.50,68.30
        ```

        ###        Problemas comunes y soluciones:

        **Si tienes problemas de codificaci n:**
        - Abre tu archivo en Excel
        - Ve a "Archivo"     "Guardar como"
        - Selecciona "CSV UTF-8 (delimitado por comas)"
        - Guarda con ese formato

        **Si tienes separadores diferentes:**
        - Usa las opciones avanzadas en el panel lateral
        - Prueba con punto y coma (;) si tu archivo viene de Excel en
    espa ol

        **Si tienes decimales con coma:**
        - Cambia el separador decimal en las opciones avanzadas
        """)

        # Mostrar ejemplo de archivo descargable
        st.markdown("###        Descargar archivo de ejemplo")
        ejemplo_data = {
            'grupo': ['Bajo', 'Bajo', 'Medio', 'Medio', 'Alto', 'Alto'],
            'agua': [45.50, 48.20, 78.30, 82.10, 115.60, 120.80],
            'luz': [85.20, 90.10, 145.80, 150.20, 245.90, 260.30],
            'gas': [35.10, 38.50, 65.40, 68.20, 95.80, 100.10],
            'internet': [25.00, 28.30, 55.20, 58.90, 95.50, 98.20],
            'telefono': [20.50, 22.10, 42.10, 45.30, 68.30, 72.40]
        }
        ejemplo_df = pd.DataFrame(ejemplo_data)
        csv_ejemplo = ejemplo_df.to_csv(index=False)
        st.download_button(
            label="       Descargar CSV de ejemplo",
            data=csv_ejemplo,
            file_name="ejemplo_servicios_basicos.csv",
            mime="text/csv"
        )
else:
    df = generar_datos_ejemplo()
    st.sidebar.success("    Usando datos de ejemplo")

# Continuar solo si hay datos
if df is not None:
    # Validar datos
    errores = validar_datos(df)
    if errores:
        st.error("    Errores en los datos:")
```

```
381         for error in errores:
382             st.error(f"    {error}")
383         st.stop()
384 
385     # Configuraci n del an lisis
386     st.sidebar.subheader("       Configuraci n del An lisis")
387 
388     # Seleccionar columnas
389     columnas_disponibles = df.columns.tolist()
390     columnas_numericas = df.select_dtypes(include=[np.number]).columns.
    tolist()
391     columnas_categoricas = df.select_dtypes(include=['object', 'category
    ']).columns.tolist()
392 
393     # Variable de agrupaci n
394     if columnas_categoricas:
395         variable_grupo = st.sidebar.selectbox(
396             "Selecciona la variable de agrupaci n:",
397             columnas_categoricas,
398             help="Variable categ rica que define los grupos a comparar"
399         )
400     else:
401         st.sidebar.error("    No se encontraron columnas categ ricas")
402         st.stop()
403 
404     # Variables dependientes
405     if columnas_numericas:
406         variables_dependientes = st.sidebar.multiselect(
407             "Selecciona las variables de gasto:",
408             columnas_numericas,
409             default=columnas_numericas[:5] if len(columnas_numericas) >=
    5 else columnas_numericas,
410             help="Variables num ricas que representan los gastos en
    servicios"
411         )
412 
413         if len(variables_dependientes) < 2:
414             st.sidebar.error("    Selecciona al menos 2 variables de
    gasto")
415             st.stop()
416     else:
417         st.sidebar.error("    No se encontraron columnas num ricas")
418         st.stop()
419 
420     # Otras configuraciones
421     mostrar_datos = st.sidebar.checkbox("Mostrar datos crudos", value=
    False)
422     nivel_confianza = st.sidebar.slider("Nivel de confianza", 0.90,
    0.99, 0.95, 0.01)
423 
424     # Filtrar datos seg n selecci n
425     df_analisis = df[[variable_grupo] + variables_dependientes].copy()
426 
427     # Eliminar filas con valores faltantes
428     df_analisis = df_analisis.dropna()
429 
430     # Verificar que haya al menos 2 grupos
431     grupos_unicos = df_analisis[variable_grupo].unique()
```

```
    if len(grupos_unicos) < 2:
        st.error("    Se necesitan al menos 2 grupos diferentes para el
an lisis")
        st.stop()

    # Crear columna total si no existe
    if 'total' not in df_analisis.columns:
        df_analisis['total'] = df_analisis[variables_dependientes].sum(
axis=1)

    # Secci n 1: Introducci n y contexto
    st.header("       Cap tulo 1: El Contexto del Problema")
    st.markdown(f"""
    ### La Historia detr s de los N meros

    En este an lisis , exploraremos los patrones de gasto en servicios
b sicos entre diferentes grupos.
    Utilizaremos **{len(df_analisis)} observaciones** divididas en **{
len(grupos_unicos)} grupos** para entender si existen diferencias
significativas en el comportamiento de gasto.

    **Nuestra pregunta de investigaci n:**  Existen  diferencias
significativas en los patrones de gasto
    en servicios b sicos entre los diferentes grupos analizados?

    **Variables analizadas:**
    - **Variable de agrupaci n:** {variable_grupo}
    - **Variables de gasto:** {', '.join(variables_dependientes)}

    Para responder esta pregunta, utilizaremos **MANOVA (An lisis
Multivariado de Varianza)**,
    una t cnica estad stica que nos permite analizar m ltiples
variables dependientes simult neamente.
    """)

    if mostrar_datos:
        st.subheader("       Datos del An lisis")
        st.dataframe(df_analisis)

        # Estad sticas descriptivas
        st.subheader("       Estad sticas Descriptivas")
        stats_desc = df_analisis.groupby(variable_grupo)[
variables_dependientes].agg(['count', 'mean', 'std', 'min', 'max']).
round(2)
        st.dataframe(stats_desc)

    # Secci n 2: An lisis Exploratorio
    st.header("       Cap tulo 2: Explorando los Patrones de Gasto")

    col1, col2 = st.columns(2)

    with col1:
        st.subheader("Distribuci n por Grupo")
        conteo_grupos = df_analisis[variable_grupo].value_counts()
        fig_dist = px.pie(values=conteo_grupos.values, names=
conteo_grupos.index,
                          title=f"Distribuci n de Observaciones por {
variable_grupo}")
```

```
478         st.plotly_chart(fig_dist, use_container_width=True)
479
480     with col2:
481         st.subheader("Gasto Total Promedio por Grupo")
482         gasto_promedio = df_analisis.groupby(variable_grupo)['total'].
    mean().reset_index()
483         fig_bar = px.bar(gasto_promedio, x=variable_grupo, y='total',
484                          title="Gasto Total Promedio por Grupo",
485                          color=variable_grupo)
486         st.plotly_chart(fig_bar, use_container_width=True)
487
488     # An lisis por servicios individuales
489     st.subheader("       An lisis Detallado por Variable")
490
491     servicio_seleccionado = st.selectbox("Selecciona una variable para
    analizar:", variables_dependientes)
492
493     fig_violin = px.violin(df_analisis, x=variable_grupo, y=
    servicio_seleccionado,
494                           title=f"Distribuci n de {
    servicio_seleccionado}",
495                           color=variable_grupo)
496     st.plotly_chart(fig_violin, use_container_width=True)
497
498     # Informaci n de diagn stico
499     st.subheader("       Diagn stico de Datos")
500
501     col1, col2, col3 = st.columns(3)
502
503     with col1:
504         st.metric("Total de observaciones", len(df_analisis))
505         st.metric("Observaciones despu s de limpiar", len(df_analisis.
    dropna()))
506
507     with col2:
508         st.metric("N mero de grupos", len(grupos_unicos))
509         min_obs_grupo = df_analisis[variable_grupo].value_counts().min()
510         st.metric("M nimo obs. por grupo", min_obs_grupo)
511
512     with col3:
513         st.metric("Variables analizadas", len(variables_dependientes))
514         valores_faltantes = df_analisis[variables_dependientes].isnull()
    .sum().sum()
515         st.metric("Valores faltantes", valores_faltantes)
516
517     # Mostrar distribuci n por grupo
518     st.subheader("       Distribuci n de Observaciones por Grupo")
519     dist_grupos = df_analisis[variable_grupo].value_counts().sort_index
    ()
520     st.dataframe(dist_grupos.to_frame("Cantidad"))
521
522     # Advertencias
523     if min_obs_grupo < 3:
524         st.warning(f"       Algunos grupos tienen menos de 3
    observaciones. M nimo: {min_obs_grupo}")
525
526     if valores_faltantes > 0:
527         st.warning(f"       Hay {valores_faltantes} valores faltantes
```

```
    que ser n eliminados del an lisis")
528
529     # Verificar variabilidad
530     variabilidad = df_analisis[variables_dependientes].std()
531     variables_sin_variabilidad = variabilidad[variabilidad == 0].index.
    tolist()
532     if variables_sin_variabilidad:
533         st.warning(f"       Variables sin variabilidad: {', '.join(
    variables_sin_variabilidad)}")
534
535     # Mostrar primeras filas para verificar datos
536     with st.expander("       Vista previa de datos"):
537         st.dataframe(df_analisis.head(10))
538
539     # Secci n 3: An lisis MANOVA
540     st.header("       Cap tulo 3: El An lisis MANOVA")
541
542     st.markdown("""
543     ###    Qu    es MANOVA?
544
545     **MANOVA (Multivariate Analysis of Variance)** es una extensi n del
     ANOVA que nos permite:
546     - Analizar m ltiples variables dependientes simult neamente
547     - Controlar el error tipo I al hacer m ltiples comparaciones
548     - Detectar diferencias que podr an no ser evidentes en an lisis
    univariados
549
550     **Hip tesis de nuestro estudio:**
551     - ** H  :** No hay diferencias significativas en los gastos entre
    grupos
552     - ** H  :** Existen diferencias significativas en al menos una
    variable entre grupos
553     """)
554
555     # Realizar MANOVA
556     with st.spinner("Realizando an lisis MANOVA..."):
557         resultado_manova, error_manova = realizar_manova(df_analisis,
    variables_dependientes, variable_grupo)
558
559     if error_manova:
560         st.error(f"    Error en el an lisis MANOVA: {error_manova}")
561         st.markdown("""
562         **Posibles causas del error:**
563         - Nombres de columnas con caracteres especiales
564         - Muy pocas observaciones por grupo (m nimo 3 por grupo)
565         - Variables con poca variabilidad
566         - Problemas de multicolinealidad
567         - Valores faltantes en los datos
568
569         **Recomendaciones:**
570         1. Verifica que cada grupo tenga al menos 3 observaciones
571         2. Revisa que no haya valores faltantes
572         3. Aseg rate de que las variables num ricas tengan
    variabilidad
573         4. Simplifica los nombres de columnas (sin espacios ni
    caracteres especiales)
574         """)
575     else:
```

```
576         st.subheader("       Resultados del An lisis MANOVA")
577 
578         # Mostrar resultados
579         st.markdown("### Estad sticos de Prueba")
580         # Acceder a los DataFrames de los resultados (Pillai's Trace,
    Wilks' Lambda, etc.) desde el diccionario de resultados.
581         # El m todo 'mv_test()' usualmente retorna un diccionario con
    claves como 'Target', 'Intercept', etc.
582         # Estamos interesados en la tabla para la variable independiente
     (variable_grupo).
583 
584         # Esto obtendr  el DataFrame para tu variable independiente
585         if isinstance(resultado_manova, dict) and variable_grupo in
    resultado_manova:
586             resultado_df = resultado_manova[variable_grupo].round(4)
587         elif isinstance(resultado_manova, pd.DataFrame): # Para el caso
    de fallback de scipy
588             resultado_df = resultado_manova.round(4)
589         else:
590             st.warning("No se pudo formatear los resultados del MANOVA.
    Mostrando el objeto completo.")
591             st.write(resultado_manova)
592             resultado_df = pd.DataFrame() # Crear un DataFrame vac o
    para evitar errores posteriores
593 
594         if not resultado_df.empty:
595             st.dataframe(resultado_df)
596 
597             # Interpretaci n de resultados
598             st.markdown("###        Interpretaci n de los Resultados")
599 
600             # Obtener el p-valor (usando diferentes m todos seg n el
    resultado)
601             # Priorizar 'Pr > F' y manejar casos donde la estructura
    podr a ser diferente
602             p_valor_pillai = np.nan
603             if 'Pr > F' in resultado_df.columns:
604                 p_valor_pillai = resultado_df.loc[resultado_df.index[0],
     "Pr > F"]
605             elif 'P-value' in resultado_df.columns: # Para posibles
    cambios futuros u otras librer as
606                 p_valor_pillai = resultado_df.loc[resultado_df.index[0],
     "P-value"]
607             elif resultado_df.shape[1] > 0: # Intentar obtener la
     ltima  columna si no se encuentra 'Pr > F'
608                 p_valor_pillai = resultado_df.iloc[0, -1]
609 
610             # Asegurarse de que sea un float para la comparaci n
611             if isinstance(p_valor_pillai, (pd.Series, pd.DataFrame)):
612                 p_valor_pillai = p_valor_pillai.iloc[0]
613             try:
614                 p_valor_pillai = float(p_valor_pillai)
615             except (ValueError, TypeError):
616                 p_valor_pillai = np.nan # Si la conversi n falla,
    establecer a NaN
617 
618             if not np.isnan(p_valor_pillai) and p_valor_pillai < (1 -
    nivel_confianza):
```

```
619                 st.success(f"""
620                 **    RESULTADO SIGNIFICATIVO** (p = {p_valor_pillai:.4f
    })
621
622                 Con un nivel de confianza del {nivel_confianza*100}%,
    podemos **rechazar la hip tesis nula**.
623
624                 **Conclusi n:** Existen diferencias estad sticamente
    significativas en los patrones de gasto
625                 entre los diferentes grupos analizados.
626                 """)
627             elif not np.isnan(p_valor_pillai):
628                 st.warning(f"""
629                 **    RESULTADO NO SIGNIFICATIVO** (p = {p_valor_pillai
    :.4f})
630
631                 Con un nivel de confianza del {nivel_confianza*100}%, **
    no podemos rechazar la hip tesis nula**.
632
633                 **Conclusi n:** No hay evidencia suficiente para
    afirmar que existen diferencias significativas
634                 en los gastos entre los grupos analizados.
635                 """)
636             else:
637                 st.warning("No se pudo obtener un p-valor para la
    interpretaci n. Revise los resultados completos.")
638
639     # Secci n 4: An lisis Post-Hoc
640     st.header("       Cap tulo 4: An lisis Detallado por Variable")
641
642     st.markdown("""
643     Aunque MANOVA nos da una respuesta global, es importante entender **
    qu  variables espec ficas** muestran las mayores diferencias entre
    grupos.
644     """)
645
646     # An lisis univariado para cada variable
647     st.subheader("An lisis ANOVA Individual por Variable")
648
649     resultados_anova = []
650     for variable in variables_dependientes:
651         try:
652             # ANOVA para cada variable
653             grupos_variable = [df_analisis[df_analisis[variable_grupo]
    == grupo][variable].dropna()
654                                for grupo in grupos_unicos]
655             # Filtrar grupos vac os
656             grupos_variable = [grupo for grupo in grupos_variable if len
    (grupo) > 0]
657
658             # Asegurarse de que haya al menos 2 grupos no vac os y m s
     de 1 observaci n por grupo para un ANOVA significativo
659             if len(grupos_variable) >= 2 and all(len(g) > 1 for g in
    grupos_variable):
660                 f_stat, p_val = stats.f_oneway(*grupos_variable)
661
662                 resultados_anova.append({
663                     'Variable': variable,
```

```
                    'F-estad stico': round(f_stat, 4),
                    'p-valor': round(p_val, 4),
                    'Significativo': 'S  ' if p_val < (1 -
nivel_confianza) else 'No'
                })
            else:
                resultados_anova.append({
                    'Variable': variable,
                    'F-estad stico': 'N/A',
                    'p-valor': 'N/A',
                    'Significativo': 'Datos insuficientes'
                })
        except Exception as e:
            resultados_anova.append({
                'Variable': variable,
                'F-estad stico': 'Error',
                'p-valor': 'Error',
                'Significativo': 'Error'
            })

    df_anova = pd.DataFrame(resultados_anova)
    st.dataframe(df_anova)

    # Visualizaci n de diferencias entre grupos
    st.subheader("       Comparaci n Visual entre Grupos")

    # Crear gr fico de barras agrupadas
    n_vars = len(variables_dependientes)
    cols = 3
    rows = (n_vars + cols - 1) // cols

    fig_comparacion = make_subplots(
        rows=rows, cols=cols,
        subplot_titles=variables_dependientes,
        vertical_spacing=0.12,
        horizontal_spacing=0.08
    )

    for i, variable in enumerate(variables_dependientes):
        row = i // cols + 1
        col = i % cols + 1

        medias = df_analisis.groupby(variable_grupo)[variable].mean()

        fig_comparacion.add_trace(
            go.Bar(x=medias.index, y=medias.values,
                   name=variable,
                   showlegend=False),
            row=row, col=col
        )

    fig_comparacion.update_layout(height=200*rows, title_text="Gasto
Promedio por Variable y Grupo")
    st.plotly_chart(fig_comparacion, use_container_width=True)

    # Secci n 5: Conclusiones
    st.header("       Cap tulo 5: Conclusiones y Recomendaciones")

```

```
    st.markdown(f"""
    ###        Resumen Ejecutivo

    Basado en nuestro an lisis MANOVA de **{len(df_analisis)}
    observaciones**:

    **Principales Hallazgos:**
    """)

    # Calcular estad sticas descriptivas por grupo
    cols_metricas = st.columns(len(grupos_unicos))

    for i, grupo in enumerate(grupos_unicos):
        datos_grupo = df_analisis[df_analisis[variable_grupo] == grupo]
        with cols_metricas[i]:
            st.metric(f"Grupo: {grupo}", len(datos_grupo))
            st.metric(f"Gasto Promedio", f"${datos_grupo['total'].mean()
    :.2f}")
            st.metric(f"Desviaci n Est ndar", f"${datos_grupo['total
    '].std():.2f}")

    # Recomendaciones personalizadas
    st.markdown("""
    ###        Recomendaciones

    **Basado en el an lisis de tus datos:**

    1. **Si encontraste diferencias significativas:**
        - Investiga qu  factores causan estas diferencias
        - Considera estrategias diferenciadas por grupo
        - Desarrolla pol ticas segmentadas para cada grupo

    2. **Si no encontraste diferencias significativas:**
        - Los grupos muestran patrones similares de gasto
        - Puedes aplicar estrategias uniformes
        - Considera otros factores de segmentaci n

    3. **Para an lisis futuros:**
        - Incluye m s variables explicativas
        - Considera an lisis de series temporales
        - Eval a factores externos que puedan influir
    """)

    # Exportar resultados
    st.subheader("       Exportar Resultados")

    col1, col2 = st.columns(2)

    with col1:
        if st.button("       Descargar Estad sticas Descriptivas"):
            stats_export = df_analisis.groupby(variable_grupo)[
    variables_dependientes].agg(['count', 'mean', 'std']).round(2)
            csv = stats_export.to_csv()
            st.download_button(
                label="       Descargar CSV",
                data=csv,
                file_name="estadisticas_descriptivas.csv",
                mime="text/csv"
```

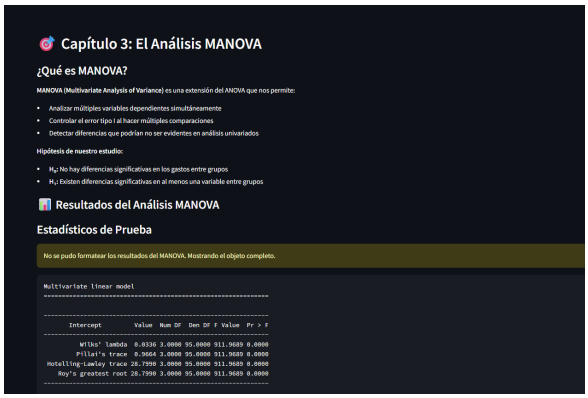
```
            )

    with col2:
        if st.button("       Descargar Resultados ANOVA"):
            csv_anova = df_anova.to_csv(index=False)
            st.download_button(
                label="       Descargar CSV",
                data=csv_anova,
                file_name="resultados_anova.csv",
                mime="text/csv"
            )

    # Secci n t cnica
    with st.expander("       Detalles T cnicos del An lisis"):
        st.markdown(f"""
        **Metodolog a Empleada:**
        - **T cnica:** MANOVA (Multivariate Analysis of Variance)
        - **Variables Dependientes:** {', '.join(variables_dependientes)
    }
        - **Variable Independiente:** {variable_grupo}
        - **N mero de grupos:** {len(grupos_unicos)}
        - **Tama o de muestra:** {len(df_analisis)}
        - **Estad stico Principal:** Pillai's Trace (m s robusto ante
    violaciones de supuestos)
        - **Software:** Python con librer as statsmodels, scipy, plotly

        **Supuestos del MANOVA:**
        - Normalidad multivariada
        - Homogeneidad de matrices de covarianza
        - Independencia de observaciones
        - Ausencia de outliers extremos

        **Grupos analizados:** {', '.join(grupos_unicos)}
        """)

# Footer
st.markdown("---")
st.markdown("*Desarrollado con  usando  Python y Streamlit para
    an lisis estad stico*")
```

Listing 4: Cuarta Aplicación

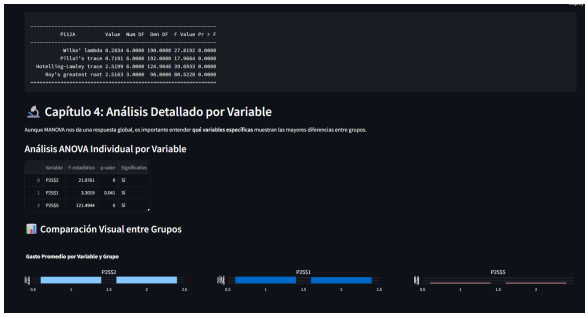
4.2. Capturas de Pantalla

(a) [Descripción de la imagen 1]

(b) [Descripción de la imagen 2]



(a) [Descripción de la imagen 3]



(b) [Descripción de la imagen 4]

Figura 12: [Descripción de la imagen 5]

4.3. Análisis Técnico

Aspectos técnicos destacados:

- **Framework Streamlit:** Permite crear rápidamente una **interfaz web interactiva** a partir de código Python, simplificando el desarrollo de la UI.
- **Gestión de Datos con Pandas:** Utiliza **Pandas** para un manejo eficiente de DataFrames, incluyendo carga robusta (multi-codificación), limpieza y preprocesamiento de datos.
- **Análisis Estadístico con Statsmodels y SciPy:** Emplea **statsmodels** para MANOVA y **scipy.stats** para ANOVA, asegurando robustez y flexibilidad en los cálculos.
- **Visualizaciones Interactivas con Plotly:** Genera **gráficos dinámicos** y explorables (barras, violín, pastel) que enriquecen la comprensión de los datos.
- **Manejo Robusto de Errores y Adaptabilidad:** Incluye validaciones y mensajes claros para guiar al usuario, además de adaptarse a diversas estructuras de CSV.