AnalisisExploratorioDatosSuicidio

March 24, 2026

1 Análisis Exploratorio de Datos de Suicidio

En este notebook se realiza un análisis exploratorio del dataset de suicidio: se importan y limpian los datos, se transforman variables categóricas y numéricas para su análisis, se revisan correlaciones y multicolinealidad bajo distintos escenarios, y finalmente se generan gráficas para responder preguntas clave sobre tendencias por generación, país y tiempo.

[2]:
```
!pip install skimpy pandas kagglehub scikit-learn matplotlib seaborn␣
↪statsmodels --quiet
```

Data Import

[3]:
```
from pathlib import Path
import pandas as pd
from skimpy import clean_columns # column names pip install skimpy
import seaborn as sns
import matplotlib.pyplot as plt
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

[4]:
```
data_path = Path.cwd().parent / "data" / "suicide_data.csv"
images_path = Path.cwd().parent / "reporte" / "images"
df = pd.read_csv(data_path)
```

2 Limpieza

[5]:
```
df = clean_columns(df)
print(df.columns)
nan_counts = df.isna().sum().sort_values(ascending=False)
print(nan_counts)
```

```
Index(['country', 'year', 'sex', 'age', 'suicides_no', 'population',
       'suicides_100k_pop', 'country_year', 'hdi_for_year', 'gdp_for_year_$',
       'gdp_per_capita_$', 'generation'],
      dtype='str')
hdi_for_year         19456
country                  0
year                     0
sex                      0
```

```
age                     0
suicides_no             0
population              0
suicides_100k_pop       0
country_year            0
gdp_for_year_$          0
gdp_per_capita_$        0
generation              0
dtype: int64
```

[6]:
```
df["gdp_for_year_$"] = df["gdp_for_year_$"].str.replace(",", "").
  ↪astype("Int64") # Reemplazar comas por puntos
print(df.dtypes)
```

```
country                  str
year                   int64
sex                      str
age                      str
suicides_no            int64
population             int64
suicides_100k_pop    float64
country_year             str
hdi_for_year         float64
gdp_for_year_$         Int64
gdp_per_capita_$       int64
generation               str
dtype: object
```

Columnas con nombres sencillos y no se encuentrar valores vacíos en ninguna otra columna.

3 Variable country-year

[7]:
```
combined = df["country"] + df["year"].astype(str)
match = combined.equals(df["country_year"])
print(match)
```

```
True
```

Por lo tanto, country es solo combinación de las otras 2.

4 Variable age

[8]:
```
combined = df["generation"] + df["age"].astype(str)
print(sorted(combined.unique()))
```

```
['Boomers25-34 years', 'Boomers35-54 years', 'Boomers55-74 years', 'G.I.
Generation55-74 years', 'G.I. Generation75+ years', 'Generation X15-24 years',
'Generation X25-34 years', 'Generation X35-54 years', 'Generation X5-14 years',
```

```
'Generation Z5-14 years', 'Millenials15-24 years', 'Millenials25-34 years',
'Millenials5-14 years', 'Silent35-54 years', 'Silent55-74 years', 'Silent75+
years']
```

Notamos que tenemos diferentes rangos de edades dentro de la misma generación. Haremos la nueva variable discreta usando la vriable combinada.

[9]:
```
orden = [
    "G.I. Generation55-74 years",
    "G.I. Generation75+ years",
    "Silent35-54 years",
    "Silent55-74 years",
    "Silent75+ years",
    "Boomers25-34 years",
    "Boomers35-54 years",
    "Boomers55-74 years",
    "Generation X5-14 years",
    "Generation X15-24 years",
    "Generation X25-34 years",
    "Generation X35-54 years",
    "Millenials5-14 years",
    "Millenials15-24 years",
    "Millenials25-34 years",
    "Generation Z5-14 years",
]

df["generation_age"] = pd.Categorical(
    combined,
    categories=orden,
    ordered=True
)

df["generation_age_code"] = df["generation_age"].cat.codes
print(df["generation_age_code"].value_counts())
```

```
generation_age_code
3     3206
6     3030
10    2682
4     2528
13    2528
12    2510
9     2114
1     2114
15    1470
5     1154
11     982
14     806
7      806
```

```
2      630
0      630
8      630
Name: count, dtype: int64
```

5 Transformaciones

[10]:
```
# sex a binaria
df["sex"] = df["sex"].map({"male": 1, "female": 0}).astype("int")
df["population"] = df["population"].astype("int")


# country a dummies
country_dummies = pd.get_dummies(
    df["country"],
    prefix="country",
    drop_first=True # Para evitar la trampa de las variables ficticias (dummy␣
 ↪variable trap)
)
df_cleaned = df.drop(columns=["country_year", "country", "generation", "age",␣
 ↪"generation_age"])
df_with_dummies = pd.concat([df_cleaned, country_dummies], axis=1)
df_with_dummies = clean_columns(df_with_dummies)
print(df_with_dummies.shape)
```

```
(27820, 109)
```

[11]:
```
print(df_cleaned.dtypes)
```

```
year                    int64
sex                     int64
suicides_no             int64
population              int64
suicides_100k_pop     float64
hdi_for_year          float64
gdp_for_year_$          Int64
gdp_per_capita_$        int64
generation_age_code      int8
dtype: object
```

6 Variable `hdi_for_year`

[12]:
```
df_without_na = df_cleaned.dropna()
print(df_without_na.shape)
df_without_hdi = df_cleaned.drop(columns=["hdi_for_year"])
print(df_without_hdi.shape)
```

```
(8364, 9)
(27820, 8)
```

Resúmen de dfs

- **df_with_dummies**: Contiene variables dummies para cada país (menos 1) (incluye `hdi_for_year`)
- **df_without_na**: Se quitaron aquellas observaciones con valores nulos en `hdi_for_year` (Sin `country`)
- **df_without_hdi**: Se eliminó la variable `hdi_for_year` (Sin `country`)

7 Análisis de Correlación

7.1 DF con dummies para países

[13]:
```
corr_matrix = df_with_dummies.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, cmap="coolwarm", center=0)
plt.title("Matriz de Correlación (Dummies)")
plt.tight_layout()
plt.savefig(images_path / "correlation_matrix_dummies.png")
plt.close()
```

7.2 DF sin valores vacios

[14]:
```
corr_matrix = df_without_na.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, cmap='coolwarm', center=0)
plt.title('Matriz de Correlación (Sin NA)')
plt.tight_layout()
plt.savefig(images_path / 'correlation_matrix_without_na.png')
plt.close()
```

[15]:
```
# Multicolinealidad
X_vif = pd.get_dummies(df_without_na, drop_first=True)
X_vif = X_vif.astype(float)

vif_data = pd.DataFrame()
vif_data["Variable"] = X_vif.columns
vif_data["VIF"] = [variance_inflation_factor(X_vif.values, i) for i in␣
 ↪range(X_vif.shape[1])]

print("Variance Inflation Factor (VIF):")
print(vif_data)
```

```
Variance Inflation Factor (VIF):
              Variable       VIF
0                 year  0.003008
1                  sex  1.249378
```

```
2          suicides_no  2.341593
3           population  3.223732
4    suicides_100k_pop  1.495324
5         hdi_for_year  2.689914
6       gdp_for_year_$  2.970946
7     gdp_per_capita_$  2.524035
8  generation_age_code  1.204561
```

7.3 DF sin hdi_for_year

[215]:
```
corr_matrix = df_without_hdi.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, cmap="coolwarm", center=0)
plt.title("Matriz de Correlación (Sin HDI)")
plt.tight_layout()
plt.savefig(images_path / "correlation_matrix_no_hdi.png", bbox_inches="tight")
plt.close()
```

[211]:
```
# Multicolinealidad sin HDI
X_vif = pd.get_dummies(df_without_hdi, drop_first=True)
X_vif = X_vif.astype(float)

vif_data = pd.DataFrame()
vif_data["Variable"] = X_vif.columns
vif_data["VIF"] = [variance_inflation_factor(X_vif.values, i) for i in␣
 ↪range(X_vif.shape[1])]

print("Variance Inflation Factor (VIF):")
print(vif_data)
```

```
Variance Inflation Factor (VIF):
              Variable       VIF
0                 year  0.000118
1                  sex  1.208082
2          suicides_no  1.897247
3           population  2.891489
4    suicides_100k_pop  1.483871
5       gdp_for_year_$  2.527749
6     gdp_per_capita_$  1.178311
7  generation_age_code  1.165086
```

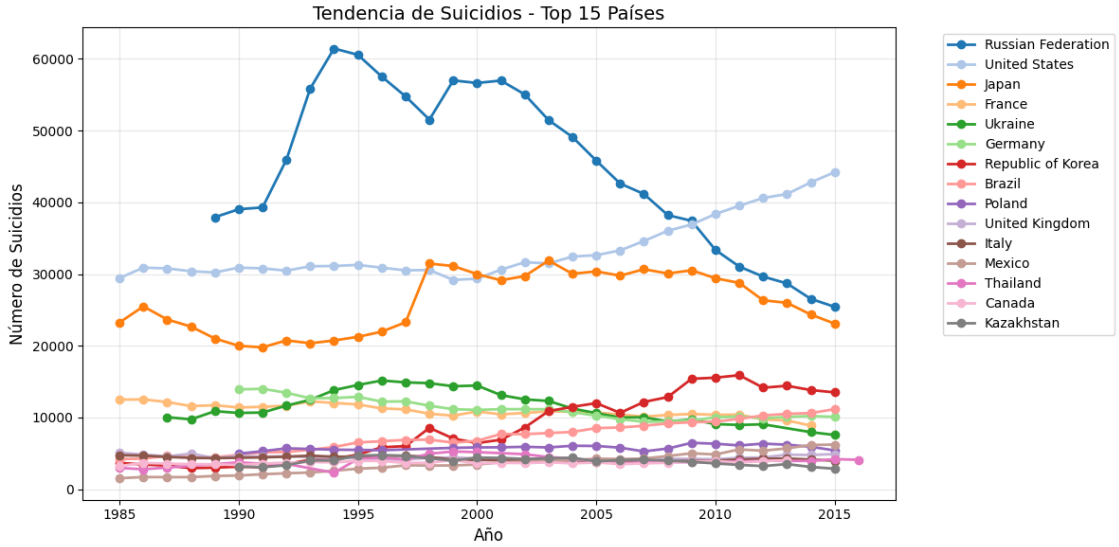
8 Gráficas para preguntas

[ ]:
```
suicides_by_gen_year = df.groupby(['age', 'year'])['suicides_no'].sum().
 ↪reset_index()

plt.figure(figsize=(12, 6))
```

```
for age in suicides_by_gen_year['age'].unique():
    gen_data = suicides_by_gen_year[suicides_by_gen_year['age'] == age]
    plt.plot(gen_data['year'], gen_data['suicides_no'], marker='o', label=age,␣
 ↪linewidth=2)

plt.xlabel('Año', fontsize=12)
plt.ylabel('Número de Suicidios', fontsize=12)
plt.title('Tendencia de Suicidios por Generación a través del Tiempo',␣
 ↪fontsize=14)
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.savefig(images_path / 'suicides_by_generation.png', dpi=300,␣
 ↪bbox_inches='tight')
plt.close()
```

[36]:
```
suicides_by_country_year = df.groupby(['country', 'year'])['suicides_no'].sum().
 ↪reset_index()

# Obtener top 15 países con más suicidios totales
top_15_countries = (
    suicides_by_country_year
    .groupby('country')['suicides_no']
    .sum()
    .sort_values(ascending=False)
    .head(15)
    .index
)

# Paleta de colores con 15+ colores distintos
colors = plt.cm.tab20(range(20))  # 20 colores distintos

plt.figure(figsize=(12, 6))
for i, country in enumerate(top_15_countries):
    country_data = suicides_by_country_year[suicides_by_country_year['country']␣
 ↪== country]
    plt.plot(country_data['year'], country_data['suicides_no'],
             marker='o', label=country, linewidth=2, color=colors[i])

plt.xlabel('Año', fontsize=12)
plt.ylabel('Número de Suicidios', fontsize=12)
plt.title('Tendencia de Suicidios - Top 15 Países', fontsize=14)
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.savefig(images_path / 'suicides_by_country.png', dpi=300,␣
 ↪bbox_inches='tight')
```

```
plt.show()
```



[ ]:

[219]:
```
# Transformar para agregar al reporte
! jupyter nbconvert 1.ipynb \
  --to latex \
  --TemplateExporter.exclude_preamble=True \
  --output-dir=converted_scripts/
# \pagestyle{empty}
# \AtBeginDocument{\let\maketitle\relax}
```

```
[NbConvertApp] WARNING | Config option `exclude_preamble` not recognized by
`LatexExporter`.  Did you mean one of: `exclude_input, exclude_input_prompt,
exclude_raw`?
[NbConvertApp] WARNING | Config option `exclude_preamble` not recognized by
`LatexExporter`.  Did you mean one of: `exclude_input, exclude_input_prompt,
exclude_raw`?
[NbConvertApp] Converting notebook 1.ipynb to latex
[NbConvertApp] Writing 43469 bytes to converted_scripts/1.tex
```