Pre-procesamiento de datos

Bernardo Morales Ramos

Ciencia de Datos

Limpieza de datos

1. San Francisco Permits

Objetivo del Dataset: Ver la discrepancia entre demanda y suministro en la industria inmobiliaria debido a retrasos en otorgar permisos de construcción

Llenar Street Suffix con la moda de la columna

```
[210]
street_suffix_mode = df['Street Suffix'].mode()[0]
print("Mode is:", street_suffix_mode)
df['Street Suffix'] = df['Street Suffix'].fillna(street_suffix_mode)
```

```
... Mode is: St
```

Asignamos los valores faltantes de la columna "Description" como "Not specified"

```
4]
missing_description = "Not specified"
df.fillna({"Description": missing_description}, inplace=True)
```

```
percentages = df['Unit'].value_counts(normalize=True) * 100
percentages
```

```
Unit
0.0       72.627972
1.0        4.450626
2.0        1.862343
3.0        1.519726
101.0      1.238170
             ...
717.0      0.003392
1016.0     0.003392
432.0      0.003392
229.0      0.003392
68.0       0.003392
Name: proportion, Length: 660, dtype: float64
```

Dado el objetivo del dataset, elminar las filas con valores nulos en Issued Date

```
15]
df = df.dropna(subset=['Issued Date'])
```

Limpieza de datos

Marcar los valores nulos en Completed Date como una variable global fuera del rango normal de fechas. Asumo esto significa que la obra aún no termina

```
not_completed_date = pd.to_datetime('2021-01-01')
df.fillna({"Completed Date": not_completed_date}, inplace=True)
```

17]

En 'First Construction Document Date'. Qué porcentage de valores es el mismo que 'Issued Date'

```
print((df['First Construction Document Date'] == df["Issued Date"]).sum() / df.shape[0] * 100)
```

9]

```
98.35562078712763
```

Debido a la inmensa mayoría de coincidencias, llenar los datos faltantes de First Construction Document Date con Issued Date

```
df.fillna({"First Construction Document Date": df["Issued Date"]}, inplace=True)
```

9]

La columna "Structural Notification" sólo tiene los valores 'Y' y nulo. Asumiremos que los valores nulos en realidad son un 'N'

```
df['Structural Notification'].unique()
```

0]

```
array([nan, 'Y'], dtype=object)
```

```
df.fillna({'Structural Notification': 'N'}, inplace=True)
```

1]

Si no se ha completado (Completed Date = not_completed_date)

```
stories_for_not_completed = 0
df.loc[df['Completed Date'] == not_completed_date, 'Number of Existing Stories'] = stories_for_not_completed
```

1]

De los valores nulos que queden, asignar los valores de Number of Proposed Stories que no sean nulos en los valores nulos de Number of Existing Stories

```
df['Number of Existing Stories'] = df['Number of Existing Stories'].fillna(df['Number of Proposed Stories'])
```

0]

Para los valores nulos que queden, asignarle una variable global

```
number_of_stories_mode = 1
df['Number of Existing Stories'] = df['Number of Existing Stories'].fillna(number_of_stories_mode)
```

1]

Limpieza de datos

1. Singapore Airbnb

Objetivo del Dataset: Qué features afectan al precio?

```
df.isnull().sum()
```

```
id                                  0
name                                2
host_id                             0
host_name                           0
neighbourhood_group                 0
neighbourhood                       0
latitude                            0
longitude                           0
room_type                           0
price                               0
minimum_nights                      0
number_of_reviews                   0
last_review                      2758
reviews_per_month                2758
calculated_host_listings_count      0
availability_365                    0
dtype: int64
```

No existen datos de "Reviews per Month" o "Last Review" cuando no existen reviews. Es decir, el número de reviews es 0

```
df[df['reviews_per_month'].isna() & df['number_of_reviews'] > 0]
```

Python

id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	last_review	reviews_per_month	calculated_host_listin

Considerando el propósito del dataset y que las reviews podrían no ser demasiado relevantes para ese propósito, se llenan los valores faltantes de la siguiente forma:

Declarar valor global para los valores faltantes de "Name"

Asignar a los valores faltantes de "Reviews per Month": 0

```
name_not_found = "Name not found"
null_date_value = "2000-01-01"

df.fillna({ "name": name_not_found, "reviews_per_month": 0, "last_review": null_date_value}, inplace=True)
```

Python

Limpieza de datos

1. Heterogeneity Activity Recognition

Objetivo del conjunto de Datasets: Juntar todos los datasets en uno solo

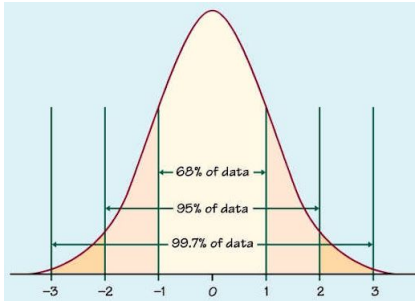
```python
# Define the directory containing the CSV files
directory = 'heterogeneity+activity+recognition'

# Initialize an empty list to store DataFrames
dataframes = []

# Iterate over all files and folders in the directory
for root, dirs, files in os.walk(directory):
    for file in files:
        # Check if the file is a CSV file
        if file.endswith('.csv'):
            print(f'Reading file: {file}')

            # Construct the full file path
            file_path = os.path.join(root, file)

            # Read the CSV file into a DataFrame
            # Use dtype=str to read all columns as strings initially
            df = pd.read_csv(file_path, dtype=str)

            # Check if the "Model" column exists
            if 'Model' not in df.columns:
                # Add a new "Model" column with the file name (without .csv extension)
                df['Model'] = os.path.splitext(file)[0]

            # Append the DataFrame to the list
            dataframes.append(df)

# Concatenate all DataFrames into a single DataFrame, aligning columns by name
merged_df = pd.concat(dataframes, ignore_index=True, sort=False)

# Save the merged DataFrame to a new CSV file (optional)
merged_df.to_csv('merged_output.csv', index=False)

# Print the merged DataFrame (optional)
print(merged_df)
```

```
Reading file: Watch_gyroscope.csv
Reading file: Watch_accelerometer.csv
Reading file: Phones_accelerometer.csv
                 Index   Arrival_Time   Creation_Time         x         y  \
0        1398347018283  1398347018283       -0.722009  9.785926 -1.082794
1        1398347018293  1398347018293       -0.729322  9.776669 -1.091828
2        1398347018303  1398347018303       -0.732950  9.782205 -1.082679
3        1398347018313  1398347018313       -0.727329  9.763894 -1.082577
4        1398347018323  1398347018323       -0.718202  9.765773 -1.082663
...                ...            ...             ...       ...       ...
36349807        129048  1424778553315  92263781761000  1.379043  0.153227
36349808        129049  1424778553346  92263812248000  1.379043  0.153227
36349809        129050  1424778553366  92263832267000   1.53227  0.153227
36349810        129051  1424778553386  92263852409000   1.53227       0.0
36349811        129052  1424778553395  92263861839000  1.379043       0.0

                z       Model Sensor_Time User        Device    gt
0             NaN     x-sense         NaN  NaN           NaN   NaN
1             NaN     x-sense         NaN  NaN           NaN   NaN
2             NaN     x-sense         NaN  NaN           NaN   NaN
3             NaN     x-sense         NaN  NaN           NaN   NaN
4             NaN     x-sense         NaN  NaN           NaN   NaN
...           ...         ...         ...  ...           ...   ...
36349807  9.959755  samsungold         NaN    i  samsungold_2  bike
36349808  9.806528  samsungold         NaN    i  samsungold_2  bike
36349809  9.806528  samsungold         NaN    i  samsungold_2  bike
36349810  9.959755  samsungold         NaN    i  samsungold_2  bike
36349811  9.959755  samsungold         NaN    i  samsungold_2  bike

[36349812 rows x 11 columns]
```

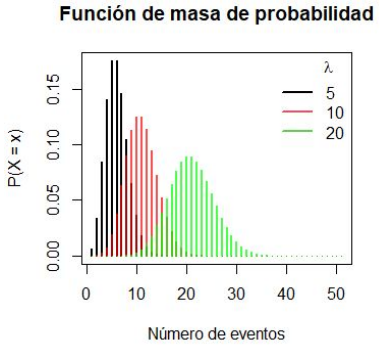
Normalización y Estandarización

La **normalización** es un proceso que escala los datos a un rango específico, generalmente entre 0 y 1, o a veces entre -1 y 1

La **estandarización**, también conocida como escala de puntuación z, ajusta los datos para que tengan una media de cero y una desviación estándar de uno.



La normalización es especialmente útil cuando la **distribución de los datos es desconocida** o **no sigue una distribución gaussiana**. La estandarización debería usarse en caso contrario.



Normalización y Estandarización

Normalización Min-Max (0 a 1):
Escala los datos entre 0 y 1, utilizando la fórmula $\frac{X - X_{min}}{X_{max} - X_{min}}$

Si se sabe que los datos deben estar dentro de un rango específico (por ejemplo, imágenes con valores de píxeles entre 0 y 255), la normalización Min-Max es ideal.

Ideal para **algoritmos basados en distancias**, como k-Nearest Neighbors (k-NN)
Puede ser muy sensible a outliers

Estandarización Z-Score (Puntuación Z):
Transforma los datos para que tengan una media de 0 y una desviación estándar de 1, utilizando la fórmula $\frac{X - \mu}{\sigma}$

Ideal para algoritmos que **asumen normalidad** (una distribución normal en sus datos), como la regresión lineal y Gaussian Naive Bayes.

La estandarización es una transformación lineal de los datos, es decir, **no cambia la distribución de los datos originales**.

Reducción de dimensionalidad (Feature selection)

Dispersion ratio

Este método se basa en la idea de que las características con una mayor dispersión o variabilidad en sus valores tienden a ser más informativas para la tarea de modelado.

Cómo funciona matemáticamente

La dispersión se calcula normalmente usando una métrica estadística que mide la variabilidad. Algunas fórmulas comunes para evaluar la dispersión incluyen:

1. **Coeficiente de variación (CV)**: $$CV = \frac{\sigma}{\mu}$$

2. **AM / GM:**

$$Dispersion\ Ratio = \frac{Media\ Aritmética\ (AM)}{Media\ Geométrica\ (GM)}$$

$$AM = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad GM = \sqrt[n]{\prod_{i=1}^{n} x_i}$$

Este ratio siempre es 1 (por la desigualdad AM GM), y valores más altos indican mayor dispersión. Si el valor es cercano a 1 indica baja dispersión, si el valor es alto, indica alta dispersión.

3. **Entropía**:

$$H(X) = -\sum_{i=1}^{n} P(x_i) log_2 P(x_i) \qquad P(x_i) = \frac{Número\ de\ ocurrencias\ de\ x_i}{Número\ total\ de\ ocurrencias}$$

Reducción de dimensionalidad (Feature selection) Dataset 1

Dispersion Ratio

Para variables numéricas:

$$\text{Ratio de Dispersión} = \frac{\sigma}{\mu}$$

```
# Calcular el Ratio de Dispersión para variables numéricas
epsilon = 1e-9
dispersion_ratios = df.select_dtypes(include=np.number).std() / (df.select_dtypes(include=np.number).mean() + epsilon)

# Mostrar los resultados
print("Ratio de Dispersión (Numéricas):")
print(dispersion_ratios)
```

✓ 0.0s

```
Ratio de Dispersión (Numéricas):
id                      0.576944
Age                     0.189979
Academic Pressure       0.439787
Work Pressure         102.284903
CGPA                    0.192096
Study Satisfaction      0.462372
Job Satisfaction       65.191907
Work/Study Hours        0.518045
Depression              0.841410
dtype: float64
```

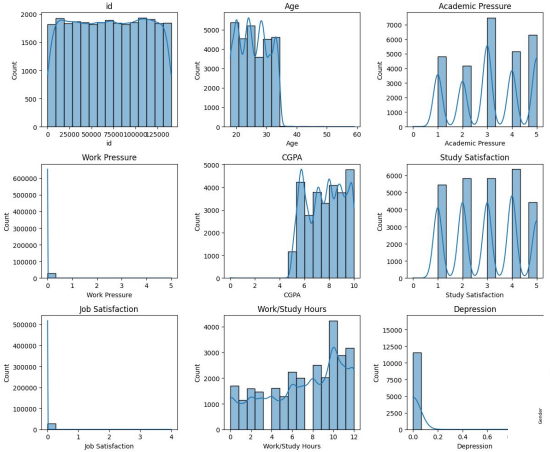
Para variables categóricas:

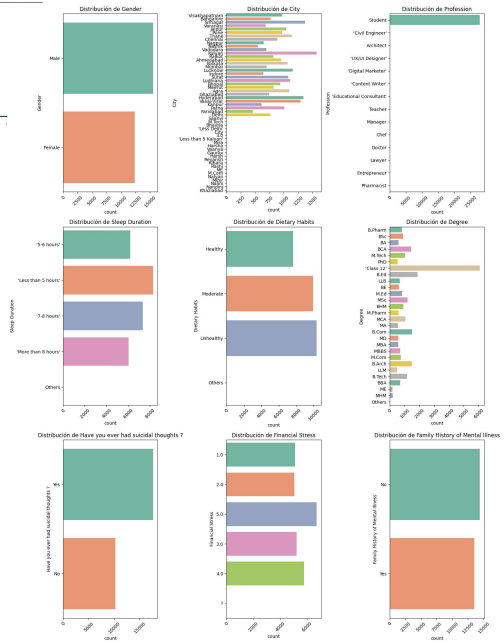
$$\text{Frecuencia Relativa (Entropía)} : H(X) = -\sum_{i=1}^{n} P(x_i)\log_2 P(x_i)$$

```
# Calcular la Entropía para variables categóricas
def categorical_entropy(series):
    probs = series.value_counts(normalize=True)
    return entropy(probs, base=2)

entropy_values = df.select_dtypes(include='object').apply(categorical_entropy)

print("\nEntropía (Categóricas):")
print(entropy_values)
```

✓ 0.0s

```
Entropía (Categóricas):
Gender                                 0.990532
City                                   4.860974
Profession                             0.016137
Sleep Duration                         1.993971
Dietary Habits                         1.577836
Degree                                 4.313886
Have you ever had suicidal thoughts ?  0.948491
Financial Stress                       2.314590
Family History of Mental Illness       0.999258
dtype: float64
```

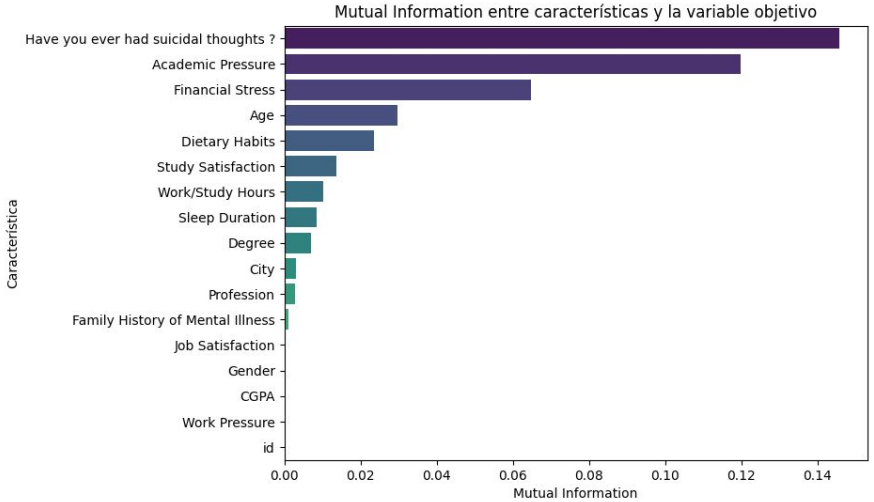


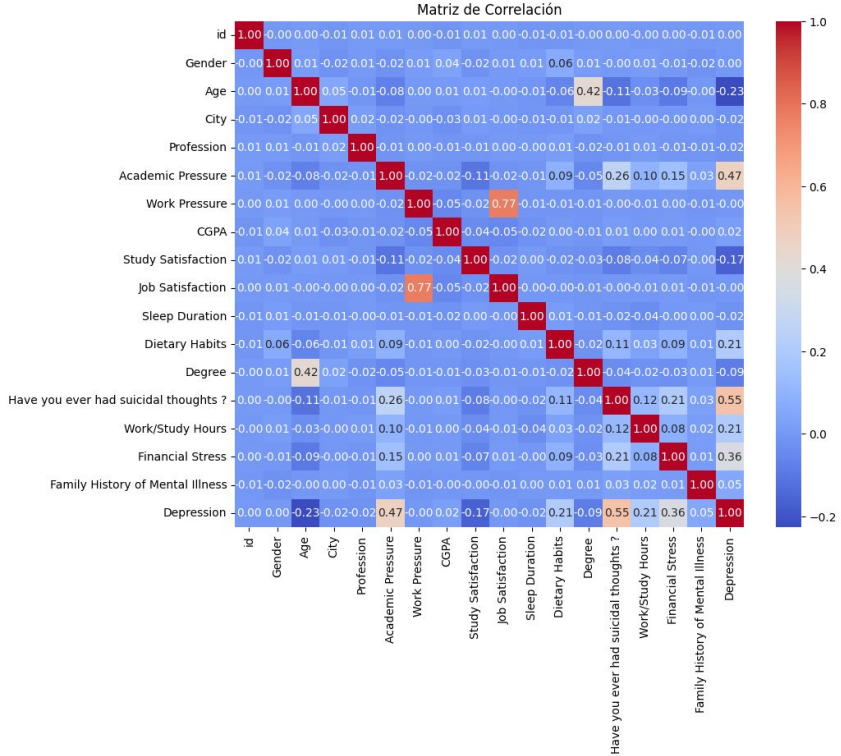


Reducción de dimensionalidad (Feature selection) Dataset 1

Mutual Information

Variable objetivo: Depression





Reducción de dimensionalidad (Feature selection) Dataset 2

Dispersion Ratio

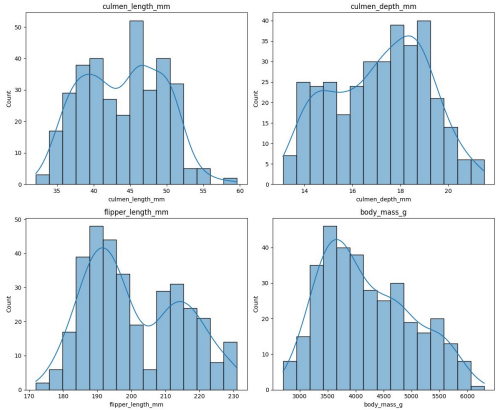
Para variables numéricas:

$$\text{Ratio de Dispersión} = \frac{\sigma}{\mu}$$

```
# Calcular el Ratio de Dispersión para variables numéricas
epsilon = 1e-9
dispersion_ratios = df.select_dtypes(include=np.number).std() / (df.select_dtypes(include=np.number).mean() + epsilon)

# Mostrar los resultados
print("Ratio de Dispersión (Numéricas):")
print(dispersion_ratios)
```

[4] ✓ 0.0s

```
Ratio de Dispersión (Numéricas):
culmen_length_mm     0.124302
culmen_depth_mm      0.115140
flipper_length_mm    0.069988
body_mass_g          0.190862
dtype: float64
```



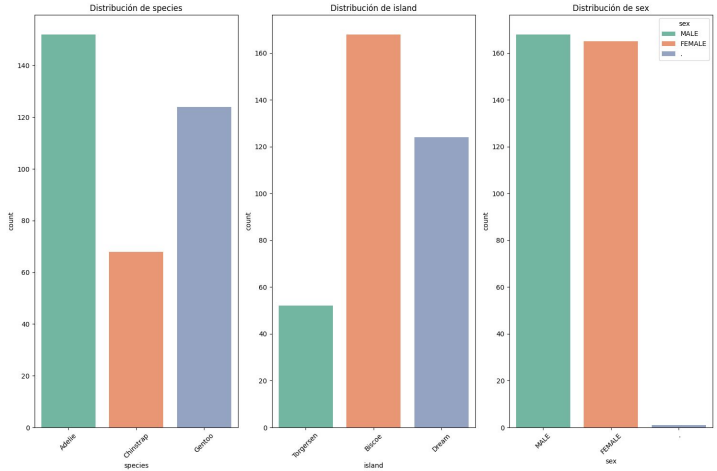
Para variables categóricas:

$$\text{Frecuencia Relativa (Entropía)} : H(X) = -\sum_{i=1}^{n} P(x_i) \log_2 P(x_i)$$

```
# Calcular la Entropía para variables categóricas
def categorical_entropy(series):
    probs = series.value_counts(normalize=True)
    return entropy(probs, base=2)

entropy_values = df.select_dtypes(include='object').apply(categorical_entropy)

print("\nEntropía (Categóricas):")
print(entropy_values)
```

✓ 0.0s

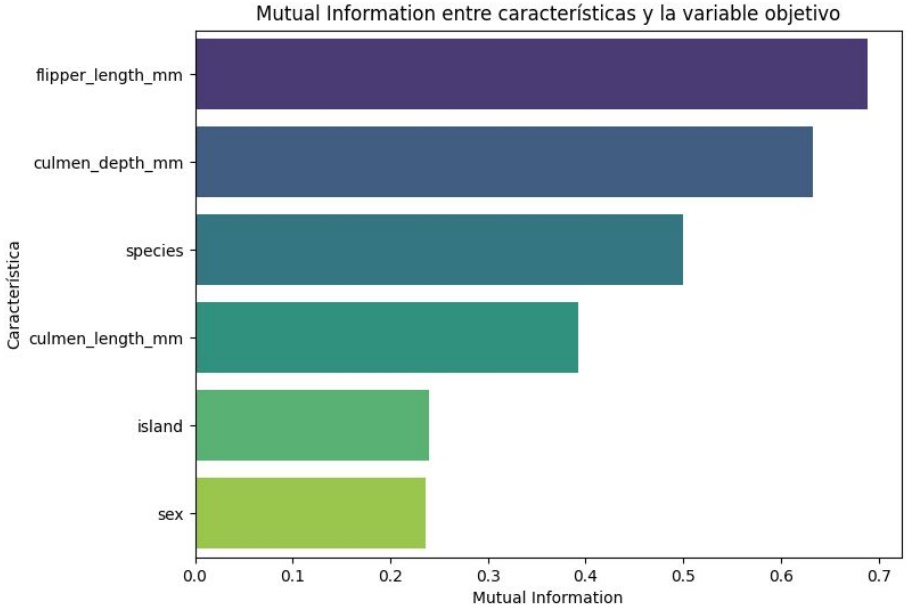
```
Entropía (Categóricas):
species    1.513611
island     1.447624
sex        1.026362
dtype: float64
```

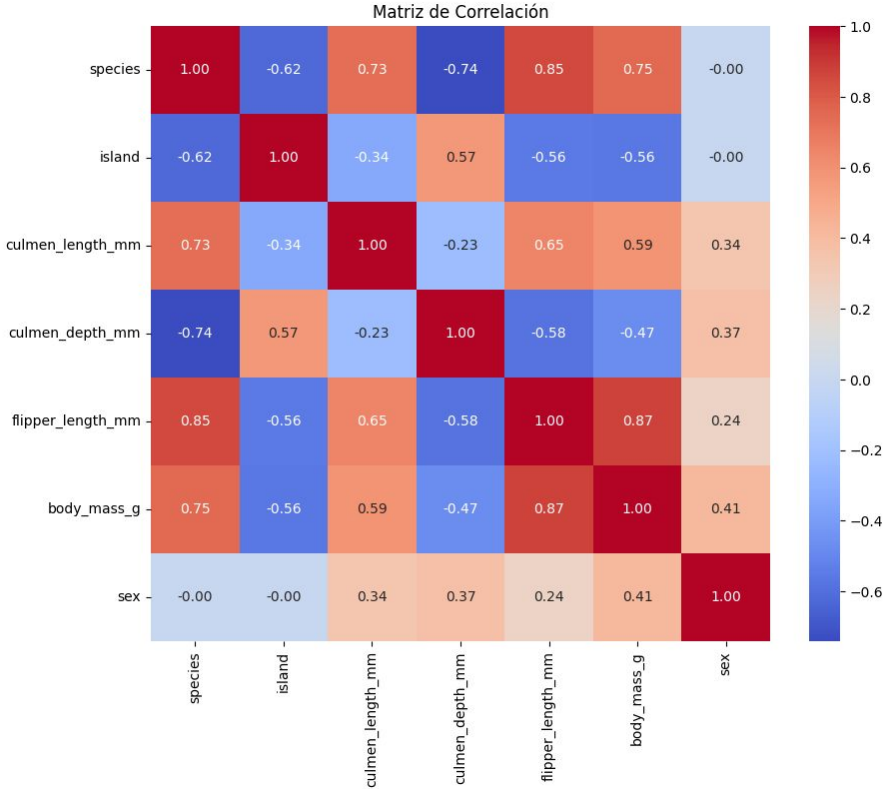


Reducción de dimensionalidad (Feature selection) Dataset 2

Mutual Information

Variable objetivo: body_mass_g





Colinealidad

Situación en la que **dos variables independientes** (predictoras) en un modelo de regresión **están altamente correlacionadas entre sí**. Esto significa que una variable puede ser expresada como una combinación lineal de la otra, lo que impide que el modelo prediga de forma independiente el valor de la variable dependiente

¿Porqué es ideal identificarlo para eliminarlo?
Inestabilidad del Modelo: Esto significa que pequeños cambios en los datos o la inclusión/exclusión de variables pueden provocar grandes cambios en los resultados del modelo, lo que reduce su confiabilidad y precisión

Dificultad para Interpretar Coeficientes: Por ejemplo, dos variables altamente correlacionadas pueden tener coeficientes de signos opuestos, lo que complica entender su efecto real sobre la variable dependiente

¿Cómo se identifica la colinealidad?
Matriz de Correlaciones: Una matriz de correlaciones es una herramienta efectiva para visualizar la fuerza de las relaciones entre variables. Los valores de correlación van de -1 a 1, donde 1 y -1 indican una correlación perfecta positiva o negativa, respectivamente.

Factor de Inflación de la Varianza (VIF): Es una estadística que mide cuánto se infla la varianza de un coeficiente de regresión estimado debido a la multicolinealidad. La multicolinealidad ocurre cuando hay una correlación entre las variables predictoras en un modelo de regresión.
VIF = 1: No hay correlación
1 < VIF < 5: Correlación moderada; generalmente aceptable
VIF >=5: Indica un potencial problema de multicolinealidad
VIF >= 10: Indica multicolinealidad seria que tal vez requiera más investigación

Colinealidad

Dataset 1 (Depresión)

```python
# Function to calculate VIF
def calculate_vif(X):
    X = sm.add_constant(X) # intercept, regression line to fit better to the data

    vif_data = pd.DataFrame()
    vif_data["feature"] = X.columns
    vif_data["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
    return vif_data

# Select features for VIF calculation (excluding target variable if any)
X = df.drop("Depression", axis=1)

# Calculate VIF
vif_result = calculate_vif(X)

# Display VIF results
print(vif_result)

# Check for multicollinearity
for index, row in vif_result.iterrows():
    if row['VIF'] >= 10:
        print(f"Multicollinearity detected in {row['feature']} with VIF: {row['VIF']}")
```

✓ 0.1s

	feature	VIF
0	const	3007.509733
1	id	1.000625
2	Gender	1.007800
3	Age	1.244356
4	City	1.004535
5	Profession	1.001452
6	Academic Pressure	1.107618
7	Work Pressure	2.464141
8	CGPA	1.009162
9	Study Satisfaction	1.022491
10	Job Satisfaction	2.464990
11	Sleep Duration	1.003038
12	Dietary Habits	1.027989
13	Degree	1.222253
14	Have you ever had suicidal thoughts ?	1.135263
15	Work/Study Hours	1.024124
16	Financial Stress	1.069664
17	Family History of Mental Illness	1.001984

```
Multicollinearity detected in const with VIF: 3007.5097331255133
```

Dataset 2 (Pingüinos)

```python
# Function to calculate VIF
def calculate_vif(X):
    X = sm.add_constant(X) # intercept, regression line to fit better to the data

    vif_data = pd.DataFrame()
    vif_data["feature"] = X.columns
    vif_data["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
    return vif_data

# Select features for VIF calculation (excluding target variable if any)
X = df.drop("body_mass_g", axis=1)

# Calculate VIF
vif_result = calculate_vif(X)

# Display VIF results
print(vif_result)

# Check for multicollinearity
for index, row in vif_result.iterrows():
    if row['VIF'] >= 10:
        print(f"Multicollinearity detected in {row['feature']} with VIF: {row['VIF']}")
```

✓ 0.0s

	feature	VIF
0	const	1176.444997
1	species	12.851194
2	island	1.782744
3	culmen_length_mm	4.254406
4	culmen_depth_mm	4.756765
5	flipper_length_mm	4.780388
6	sex	1.978636

```
Multicollinearity detected in const with VIF: 1176.4449966187638
Multicollinearity detected in species with VIF: 12.851193592698468
```