



## Proyecto Integrador 1° Semestre

### Modelo predictivo de días estancia hospitalaria como herramienta para optimización de recursos

## Maestría en Ciencia de los Datos y Analítica

### Grupo 8 - Semestre 2024-2

- Gustavo Andrés Rubio Castillo
- Juan Pablo Bertel Morales
- Gustavo Adolfo Jerez Tous

### !!!Advertencia!!!

- Este notebook fue desarrollado en Google Colab PRO.

## ✓ CONFIGURACIÓN DE EJECUCIÓN

### ✓ Importante:

Por la dimensionalidad y cardinalidad del dataset en sus variables categóricas, fue necesario ejecutar este notebook en un ambiente Colab PRO, pues los recursos de memoria y procesador utilizados por los algoritmos de reducción de dimensionalidad desbordaban la capacidad de nuestros computadores personales. Se entiende que para efectos de despliegue en producción del modelo desarrollado esto no es necesario, pues la predicción es soportable por un servidor común dedicado a la ejecución de este tipo de modelos.

Instalar librerías que no vienen por defecto en Colab:

- pycaret: torneo de modelos
- prince: MCA

```
!pip install prince
!pip install pycaret
```

 [Show hidden output](#)

Silenciar warnings y otras alertas menores

```
%load_ext autoreload
%autoreload 2
```

```
import warnings
warnings.filterwarnings('ignore')
warnings.filterwarnings("ignore", category=DeprecationWarning)
```

```
import pandas as pd
import numpy as np
import plotly.express as px
```

```
import re
import pycaret
```

Cargar funciones utilitarias propias para visualizar distribución de variables

```
import func_utils as fu
```

## ✓ ETAPA 1 - ASEGURAR CALIDAD DEL DATASET

La etapa inicial del proceso, aquí se depura y estandariza el dataset de modelación para que pueda ser procesado más adelante y realizar los análisis estadísticos correspondientes.

### Pasos realizados

- Carga del dataset desde S3 bucket(o local)
- Definición de variables de modelación y variable de respuesta
- Separación de variables numéricas y categóricas para análisis
- Limpieza de texto en variables categóricas
- Eliminación de registros nulos y duplicados

### Resultado final

Dataset limpio y estandarizado para realizar análisis exploratorio de datos (EDA)

## ✓ Cargar el dataset desde AWS S3

```
!pip install boto3
```

 [Show hidden output](#)

Las claves de acceso son temporales, pues estamos restringidos por la licencia educativa para crear usuarios persistentes con credenciales permanentes. En caso de querer probar esta parte de la carga, favor pedirnos la creación de credenciales temporales.

```
#aws_access_key_id=ASIAQ4FS56V3QHBQZA6P
#aws_secret_access_key=COPSG9Q3CLAKz/ds0gH93ghVNpNroyX7ptLGx0g2
#aws_session_token=IQoJb3JpZ2luX2VjEAsaCXVzLXdlc3QtMiJHMEUCIGRucAtge3zx8yjjhB74HnBipQ6otLV1lqsYQqL+K01JAiEA3J0muf9wyuDghJ06uJf0A
```

```
import boto3
import pandas as pd
from io import BytesIO
```

```
def download_from_s3_with_temp_credentials(bucket_name, file_key, aws_access_key_id, aws_secret_access_key, aws_session_token):
```

```
    # Crear un cliente de S3 con credenciales temporales
    s3 = boto3.client(
        "s3",
        aws_access_key_id=aws_access_key_id,
        aws_secret_access_key=aws_secret_access_key,
        aws_session_token=aws_session_token,
    )
    try:
        # Descargar el archivo desde S3
        response = s3.get_object(Bucket=bucket_name, Key=file_key)

        # Leer el contenido del archivo Excel como DataFrame
        df = pd.read_excel(BytesIO(response["Body"].read()), engine="openpyxl")
        print(f"Archivo {file_key} descargado exitosamente.")
        return df
    except Exception as e:
        print(f"Error al descargar el archivo: {e}")
        return None
```

```
# Configuración
BUCKET_NAME = "proyectointegrador"
FILE_KEY_TO_DOWNLOAD = "zona_raw/dataset_estancia_hospitalaria.xlsx" # Cambia la extensión a .xlsx
```

```
# Credenciales temporales
AWS_ACCESS_KEY_ID = "ASIAQ4FS56V3QHBQZA6P"
AWS_SECRET_ACCESS_KEY = "COPSG9Q3CLakz/ds0gH93ghVNpNroyX7ptLGx0g2"
AWS_SESSION_TOKEN = "IQoJb3JpZ2luX2VjEAsaCXVzLXdlc3QtMiJHMEUCIGRucAtge3zx8yjjhB74HnBipQ6otLV1lqsYQqL+K01JAiEA3J0muf9wyuDghJ06uJf

# Descargar el archivo desde S3
data = download_from_s3_with_temp_credentials(
    BUCKET_NAME,
    FILE_KEY_TO_DOWNLOAD,
    AWS_ACCESS_KEY_ID,
    AWS_SECRET_ACCESS_KEY,
    AWS_SESSION_TOKEN.strip() # Elimina espacios en blanco adicionales
)

# Mostrar los datos descargados
if data is not None:
    data = data.drop(['fecha', 'unidad_x_edad', 'año', 'mes'],axis=1)
else:
    print("No se pudo descargar el dataset desde S3")

data.head()
```

↗ Archivo zona\_raw/dataset\_estancia\_hospitalaria.xlsx descargado exitosamente.

	estancia_en_uci	ir_cdm	ir_grd_base	nivel_de_complejidad	procedimiento_principal	diagnostico_principal	edad
0	0	01 - ENFERMEDADES Y TRASTORNOS DEL SISTEMA NER...	01426 - MH OTRAS ENFERMEDADES DEL SISTEMA NERV...	Baja Complejidad	89.39 - OTRAS MEDICIONES Y EXAMENES NO QUIRURG...	D43.2 - Tumor de comportamiento inierto o des...	5
1	0	01 - ENFERMEDADES Y TRASTORNOS DEL SISTEMA NER...	01101 - PH PROCEDIMIENTOS VASCULARES INTRACRAN...	Alta Complejidad	41.31 - BIOPSIA DE MEDULA OSEA	C69.2 - Tumor maligno de la retina	6
2	0	01 - ENFERMEDADES Y TRASTORNOS DEL SISTEMA NER...	01426 - MH OTRAS ENFERMEDADES DEL SISTEMA NERV...	Mediana Complejidad	-	I69.4 - Secuelas de accidente vascular encefal...	5
3	0	01 - ENFERMEDADES Y TRASTORNOS DEL SISTEMA NER...	01426 - MH OTRAS ENFERMEDADES DEL SISTEMA NERV...	Baja Complejidad	99.99 - OTRO PROCEDIMIENTO MISCELANEO NCOC	I67.8 - Otras enfermedades cerebrovasculares e...	6
4	0	01 - ENFERMEDADES Y TRASTORNOS DEL SISTEMA NER...	01130 - PH PROCEDIMIENTOS ESPINALES	Baja Complejidad	80.51 - ESCISION DE DISCO INTERVERTEBRAL	M51.1 - Trastornos de disco lumbar y otros, co...	4

## ↘ Cargar el dataset desde archivo local

```
data = pd.read_excel('INSUMOS/dataset_estancia_hospitalaria.xlsx', index_col = None)
#data['f_analisis'] = data['año']*100 + data['mes']
data = data.drop(['fecha', 'unidad_x_edad', 'año', 'mes'],axis=1)
data.shape
```

↗ (78052, 10)

```
data.head()
```

	estancia_en_uci	ir_cdm	ir_grd_base	nivel_de_complejidad	procedimiento_principal	diagnostico_principal	edad
0	0	01 - ENFERMEDADES Y TRASTORNOS DEL SISTEMA NER...	01426 - MH OTRAS ENFERMEDADES DEL SISTEMA NERV...	Baja Complejidad	89.39 - OTRAS MEDICIONES Y EXAMENES NO QUIRURG...	D43.2 - Tumor de comportamiento incierto o des...	5
1	0	01 - ENFERMEDADES Y TRASTORNOS DEL SISTEMA NER...	01101 - PH PROCEDIMIENTOS VASCULARES INTRACRAN...	Alta Complejidad	41.31 - BIOPSIA DE MEDULA OSEA	C69.2 - Tumor maligno de la retina	6
2	0	01 - ENFERMEDADES Y TRASTORNOS DEL SISTEMA NER...	01426 - MH OTRAS ENFERMEDADES DEL SISTEMA NERV...	Mediana Complejidad	-	I69.4 - Secuelas de accidente vascular encefal...	5
3	0	01 - ENFERMEDADES Y TRASTORNOS DEL SISTEMA NER...	01426 - MH OTRAS ENFERMEDADES DEL SISTEMA NERV...	Baja Complejidad	99.99 - OTRO PROCEDIMIENTO MISCELANEO NCOC	I67.8 - Otras enfermedades cerebrovasculares e...	6
4	0	01 - ENFERMEDADES Y TRASTORNOS DEL SISTEMA NER...	01130 - PH PROCEDIMIENTOS ESPINALES	Baja Complejidad	80.51 - ESCISION DE DISCO INTERVERTEBRAL	M51.1 - Trastornos de disco lumbar y otros, co...	4

### ✓ Definir las variables de modelación y respuesta

```
feats_train = ['ir_cdm', 'ir_grd_base', 'nivel_de_complejidad',
               'procedimiento_principal', 'diagnostico_principal', 'estancia_en_uci', 'edad',
               'costo_operativo_estimado', 'peso_ir_estimado']
```

```
feat_target = 'estancia_total'
```

```
print('Las variables de entrenamiento son', len(feats_train), ':')
print(feats_train)
```

```
print('\nLa variable de respuesta es: ', feat_target)
```

```
Las variables de entrenamiento son 9 :
['ir_cdm', 'ir_grd_base', 'nivel_de_complejidad', 'procedimiento_principal', 'diagnostico_principal', 'estancia_en_uci', 'edad', 'costo_operativo_estimado', 'peso_ir_estimado']

La variable de respuesta es: estancia_total
```

Se remueven las columnas innecesarias para que el procesamiento sea más directo

```
data = data[feats_train + [feat_target]]
data.head()
```

	ir_cdm	ir_grd_base	nivel_de_complejidad	procedimiento_principal	diagnostico_principal	estancia_en_uci	edad
0	01 - ENFERMEDADES Y TRASTORNOS DEL SISTEMA NER...	01426 - MH OTRAS ENFERMEDADES DEL SISTEMA NERV...	Baja Complejidad	89.39 - OTRAS MEDICIONES Y EXAMENES NO QUIRURG...	D43.2 - Tumor de comportamiento incierto o des...	0	5
1	01 - ENFERMEDADES Y TRASTORNOS DEL SISTEMA NER...	01101 - PH PROCEDIMIENTOS VASCULARES INTRACRAN...	Alta Complejidad	41.31 - BIOPSIA DE MEDULA OSEA	C69.2 - Tumor maligno de la retina	0	
2	01 - ENFERMEDADES Y TRASTORNOS DEL SISTEMA NER...	01426 - MH OTRAS ENFERMEDADES DEL SISTEMA NERV...	Mediana Complejidad		I69.4 - Secuelas de accidente vascular encefal...	0	5
3	01 - ENFERMEDADES Y TRASTORNOS DEL SISTEMA NER...	01426 - MH OTRAS ENFERMEDADES DEL SISTEMA NERV...	Baja Complejidad	99.99 - OTRO PROCEDIMIENTO MISCELANEO NCOC	I67.8 - Otras enfermedades cerebrovasculares e...	0	6
4	01 - ENFERMEDADES Y TRASTORNOS DEL SISTEMA NER...	01130 - PH PROCEDIMIENTOS ESPINALES	Baja Complejidad	80.51 - ESCISION DE DISCO INTERVERTEBRAL	M51.1 - Trastornos de disco lumbar y otros, co...	0	4

## ✓ Separar variables categóricas y numericas para facilitar el análisis

```

feats_categoricas = data[feats_train].select_dtypes(include=['object']).columns
feats_numericas = data[feats_train].select_dtypes(include=['number']).columns
print('Variables categoricas: ', list(feats_categoricas))
print('\nVariables numericas: ', list(feats_numericas))

```

```

Variables categoricas: ['ir_cdm', 'ir_grd_base', 'nivel_de_complejidad', 'procedimiento_principal', 'diagnostico_principal']
Variables numericas: ['estancia_en_uci', 'edad', 'costo_operativo_estimado', 'peso_ir_estimado']

```

```
data[feats_numericas].describe()
```

	estancia_en_uci	edad	costo_operativo_estimado	peso_ir_estimado
count	78052.000000	78052.000000	7.805200e+04	78052.000000
mean	1.458335	41.395711	1.743718e+07	1.820014
std	5.720490	27.512572	2.576580e+07	1.777316
min	0.000000	0.000000	2.296900e+02	0.000000
25%	0.000000	15.000000	4.607076e+06	0.636500
50%	0.000000	41.000000	9.537359e+06	1.360200
75%	0.000000	65.000000	1.942217e+07	2.404025

```
data[list(feats_numericas) + [feat_target]].corr()
```

	estancia_en_uci	edad	costo_operativo_estimado	peso_ir_estimado	estancia_total
estancia_en_uci	1.000000	0.000672	0.674788	0.527176	0.594789
edad	0.000672	1.000000	0.072235	0.069794	0.050194
costo_operativo_estimado	0.674788	0.072235	1.000000	0.497779	0.816013
peso_ir_estimado	0.527176	0.069794	0.497779	1.000000	0.424316

## ▼ Remover texto no informativo de las variables categoricas

```
import unicodedata


def remover_texto_no_informativo(text):
    return re.sub(r'^.*? -', '', text)

def remover_tildes_y_simbolos(text):
    # Normalize the input string into a decomposed form (NFD)
    nfkd_form = unicodedata.normalize('NFD', text)
    # Remove characters that are non-spacing marks (diacritical marks)
    return ''.join([char for char in nfkd_form if unicodedata.category(char) != 'Mn'])

def remover_espacios_y_minusculas(text):
    return re.sub(r'\s+', ' ', text).strip().lower()

for f in feats_categoricas:
    data[f] = data[f].apply(remover_texto_no_informativo)
    data[f] = data[f].apply(remover_tildes_y_simbolos)
    data[f] = data[f].apply(remover_espacios_y_minusculas)

data.head()
```



	ir_cdm	ir_grd_base	nivel_de_complejidad	procedimiento_principal	diagnostico_principal	estancia_en_uci	edad	cost
0	enfermedades y trastornos del sistema nervioso	mh otras enfermedades del sistema nervioso	baja complejidad	otras mediciones y exámenes no quirúrgicos	tumor de comportamiento incierto o desconocido...	0	96	
1	enfermedades y trastornos del sistema nervioso	ph procedimientos vasculares intracraneales	alta complejidad	biopsia de médula ósea	tumor maligno de la retina	0	4	
2	enfermedades y trastornos del sistema nervioso	mh otras enfermedades del sistema nervioso	mediana complejidad		secuelas de accidente vascular encefálico, no ...	0	57	
3	enfermedades y trastornos del sistema nervioso	mh otras enfermedades del sistema nervioso	baja complejidad	otro procedimiento misceláneo ncoc	otras enfermedades cerebrovasculares específicas...	0	66	
4	enfermedades y trastornos del sistema nervioso	ph procedimientos espinales	baja complejidad	escisión de disco intervertebral	trastornos de disco lumbar y otros, con radicu...	0	45	

```
data.sample(5)
```



	ir_cdm	ir_grd_base	nivel_de_complejidad	procedimiento_principal	diagnostico_principal	estancia_en_uci	edad
8155	enfermedades y trastornos de oído, nariz, boca...	mh enfermedades orales y dentales	baja complejidad	otras mediciones y exámenes no quirúrgicos	gingivostomatitis y faringoamigdalitis herpética	0	7
60911	enfermedades y trastornos del aparato urinario	mh neoplasia de riñón y tracto urinario e insu...	baja complejidad	e.m. muestra de otro sitio cultivo	otras insuficiencias renales agudas	0	62
51339	enfermedades y trastornos sistema musculoesque...	ph procedimientos sobre tejidos blandos	baja complejidad	bursectomía	bursitis del olecranon	0	63
1230	enfermedades y trastornos del sistema nervioso	ph craneotomía	mediana complejidad	otras craneotomías	pérdida de líquido cefalorraquídeo	0	71
55105	enfermedades y trastornos de piel, tejido subcutáneo...	mh traumatismo de piel, tejido subcutáneo y mama	baja complejidad	radiografía, otra y no especificada	herida de otras partes de la pierna	0	38

### Validar registros nulos

Hay una limitación en numpy para reconocer string vacías como un dato nulo(NaN) por lo que toca remover estas filas manualmente

```
data = data[(data['ir_cdm'] != '')
            & (data['ir_grd_base'] != '')
            & (data['nivel_de_complejidad'] != '')
            & (data['procedimiento_principal'] != '')
            & (data['diagnostico_principal'] != '')]
```

data.shape



(76258, 10)

```
data = data.reset_index(drop=True)
data.head()
```



	ir_cdm	ir_grd_base	nivel_de_complejidad	procedimiento_principal	diagnostico_principal	estancia_en_uci	edad	cost
0	enfermedades y trastornos del sistema nervioso	mh otras enfermedades del sistema nervioso	baja complejidad	otras mediciones y exámenes no quirúrgicos	tumor de comportamiento incierto o desconocido...	0	96	
1	enfermedades y trastornos del sistema nervioso	ph procedimientos vasculares intracraneales	alta complejidad	biopsia de médula ósea	tumor maligno de la retina	0	4	
2	enfermedades y trastornos del sistema nervioso	mh otras enfermedades del sistema nervioso	baja complejidad	otro procedimiento misceláneo ncoc	otras enfermedades cerebrovasculares específicas...	0	66	
3	enfermedades y trastornos del sistema nervioso	ph procedimientos espinales	baja complejidad	escisión de disco intervertebral	trastornos de disco lumbar y otros, con radicul...	0	45	
4	enfermedades y trastornos del sistema nervioso	ph procedimientos de derivación ventricular	alta complejidad	derivación ventricular a localización extracra...	meningitis por hemófilos	10	48	

### Validar registros duplicados

```
# Revisar valores duplicados en el dataset
filas_duplicadas = data.duplicated().sum()
print(f'# de registros duplicados: {filas_duplicadas}')
```

```
↗ # de registros duplicados: 0
```

## ✓ ETAPA 2 - ANÁLISIS EXPLORATORIO DE DATOS

Con el dataset depurado se avalizan las variables para entender mejor el comportamiento esperado durante el entrenamiento y los resultados obtenidos del proceso. En la etapa de exploración se busca hacer un análisis estadístico descriptivo de todas las variables disponibles, desde un enfoque univariado (completitud, tendencia central, significancia, distribución) y multivariado (Correlación total y parcial, variabilidad, explicabilidad).

### Pasos realizados

- Visualización inicial
  - Scatter plot en relación con la variable de respuesta
  - Boxplot e histogramas de distribución
- Estadísticos y gráficas descriptivas complementarias
  - Análisis de frecuencias
  - Medidas de tendencia y dispersión
  - Pruebas de normalidad
  - Funciones de densidad y normalidad
- Correlaciones

### Resultado final

Comprensión de los datos disponibles e insights para la transformación de características

## ✓ Feats numéricas

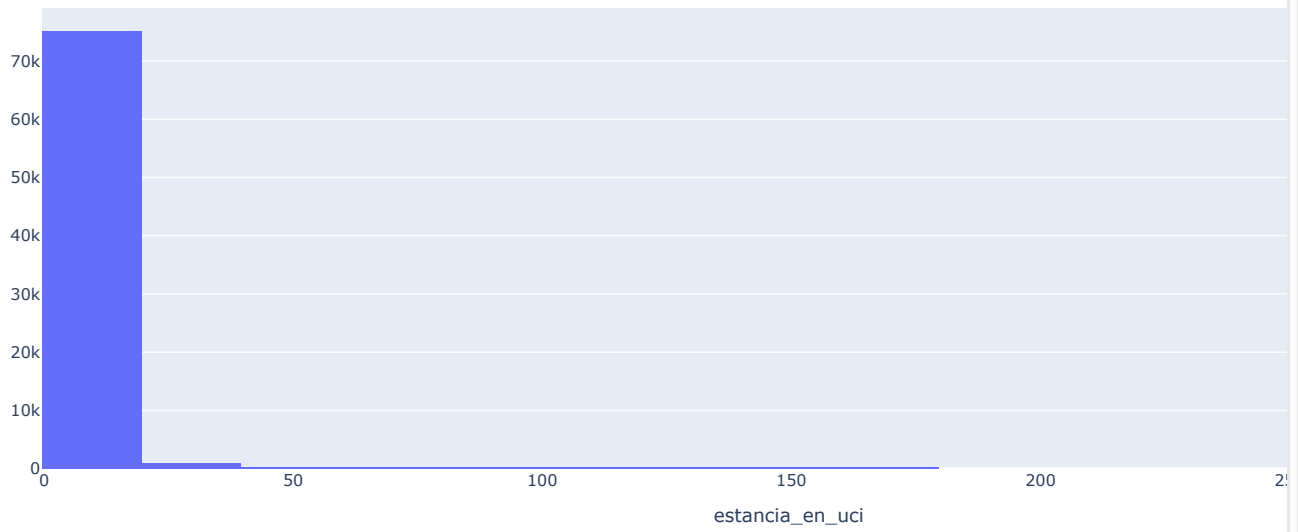
### ✓ Boxplot e histogramas interactivos iniciales

```
for f in feats_numericas:
    df = data.groupby([f]).agg(conteo=(feat_target, 'count')).reset_index()
    df['perc'] = round((df['conteo'] / df['conteo'].sum())*100, 1)
    df = df.sort_values(by='perc', ascending=False)
    fig = px.histogram(df, x = f, y = 'conteo', text_auto=True, title = f)
    fig.show()
```

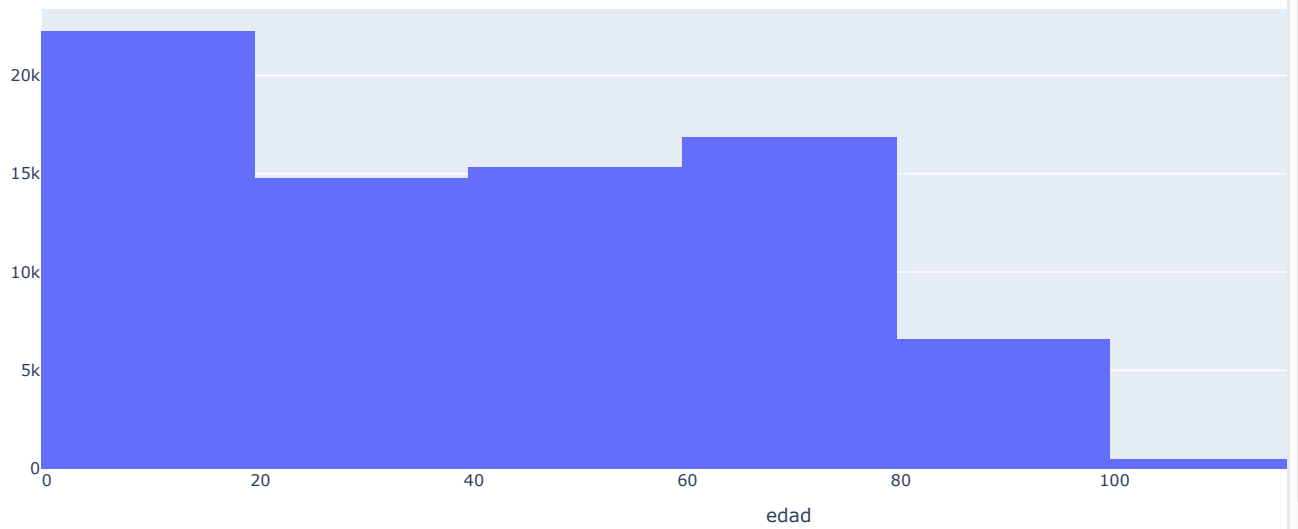




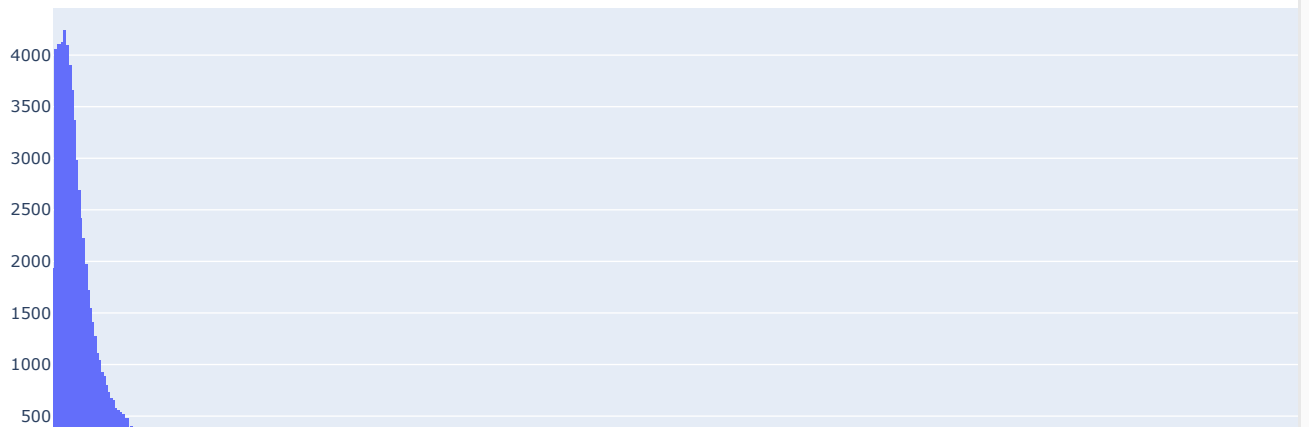
estancia\_en\_uci



edad

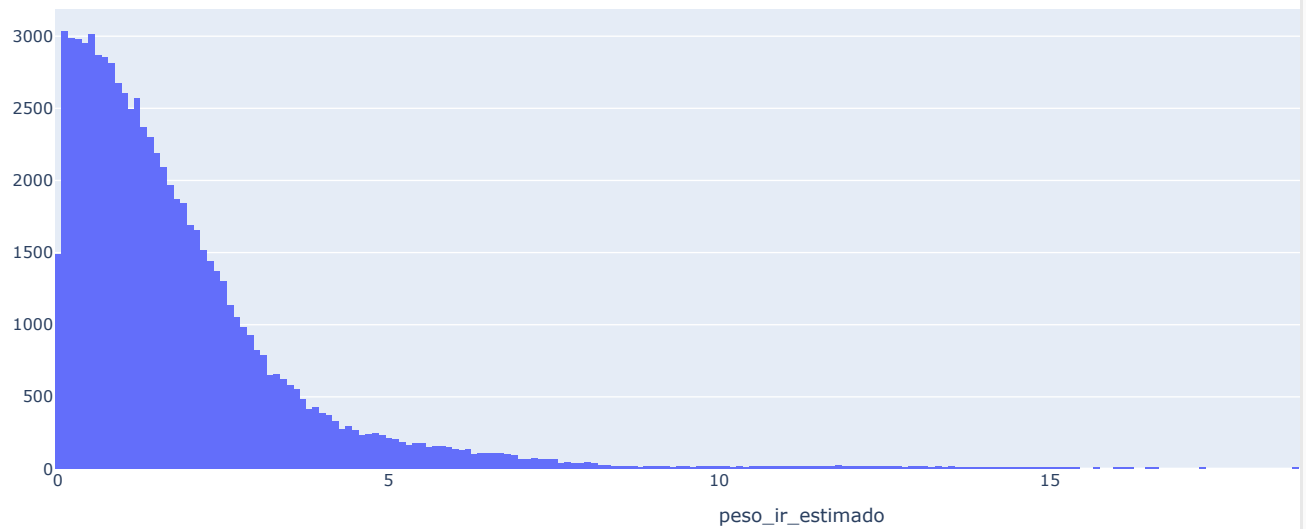


costo\_operativo\_estimado





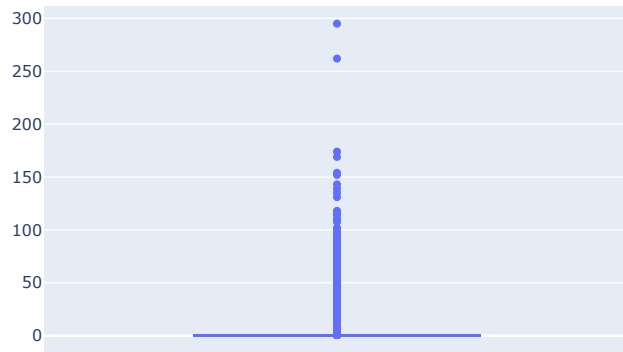
peso\_ir\_estimado



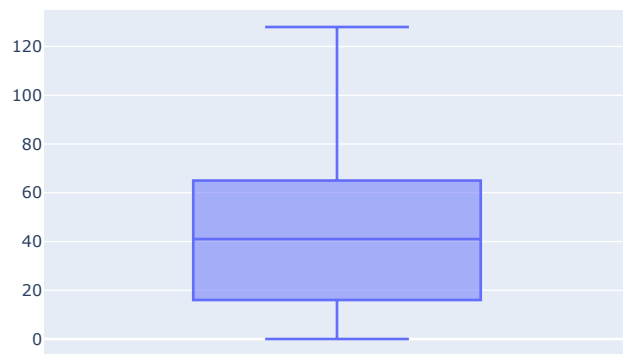
```
for f in feats_numericas:  
    fig = px.box(data, y=f, width=600, height=400)  
    fig.update_layout(title_text = f)  
    fig.show()
```



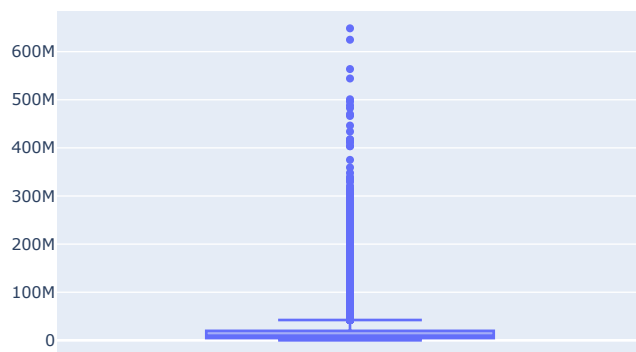
estancia\_en\_uci



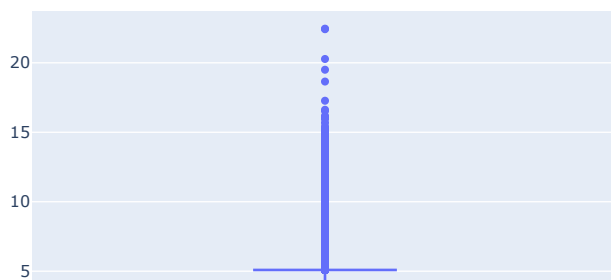
edad

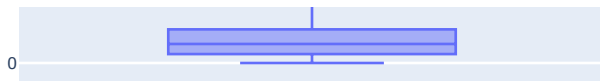


costo\_operativo\_estimado



peso\_ir\_estimado





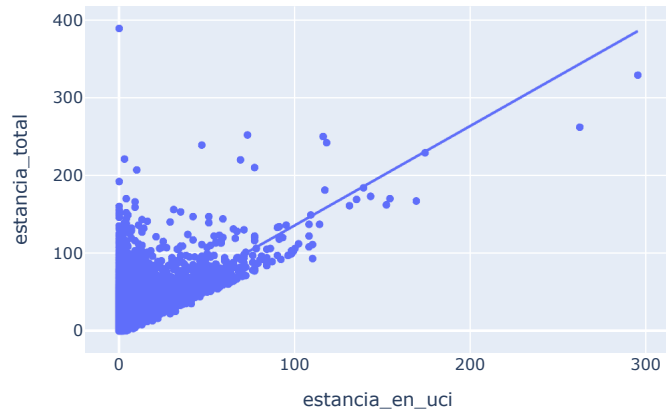
### ▼ Relación con la variable de respuesta

```
from plotly.subplots import make_subplots
fig = make_subplots(rows=2, cols=2)

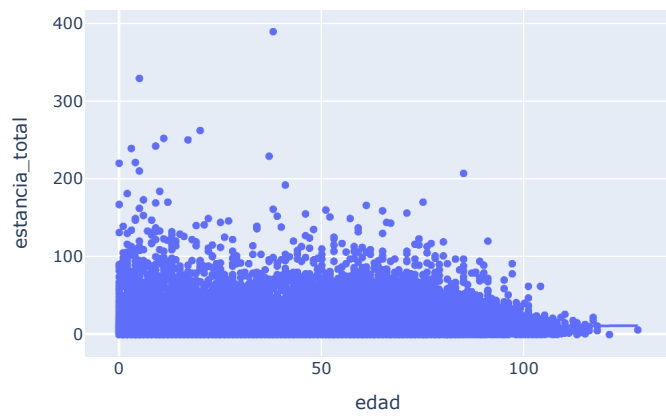
for f in feats_numericas:
    fig = px.scatter(data, x = f, y = feat_target, trendline="ols", width=600, height=400)
    fig.update_layout(title_text = f)
fig.show()
```



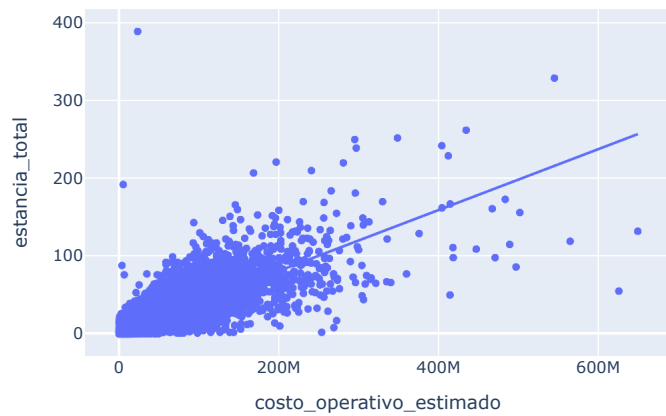
estancia\_en\_uci



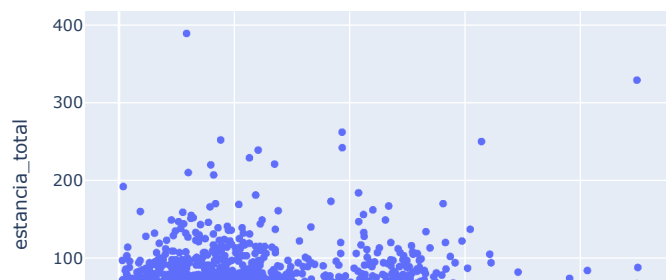
edad

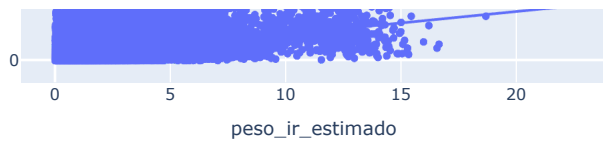


costo\_operativo\_estimado



peso\_ir\_estimado





### ▼ Estadísticos y descriptivas

```
for f in feats_numericas:  
    print('\n\t\t',str.upper(f), '\n')  
    fu.inter_uncond_descrp_num_var(f, col = data[f])
```



ESTANCIA\_EN\_UCI

## Estadísticos - estancia\_en\_uci

Frecuencia

TC y Posición

Dispersión y Forma

Normalidad

## Conteo de registros

Frec.Abs. Frec.Rel. Frec.Rel.Acum.

Con dato 76,258 100.00% 100.00%

## Gráficos descriptivos - estancia\_en\_uci

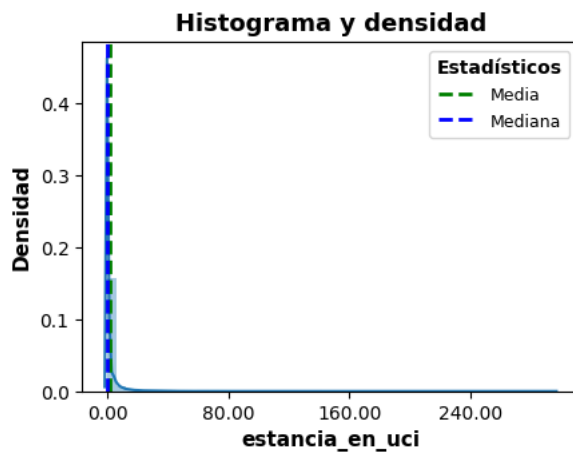
Histograma

Caja y bigotes

Violín

Densidad Norm.

QQ Norm.



EDAD

## Estadísticos - edad

Frecuencia

TC y Posición

Dispersión y Forma

Normalidad

## Conteo de registros

Frec.Abs. Frec.Rel. Frec.Rel.Acum.

Con dato 76,258 100.00% 100.00%

## Gráficos descriptivos - edad

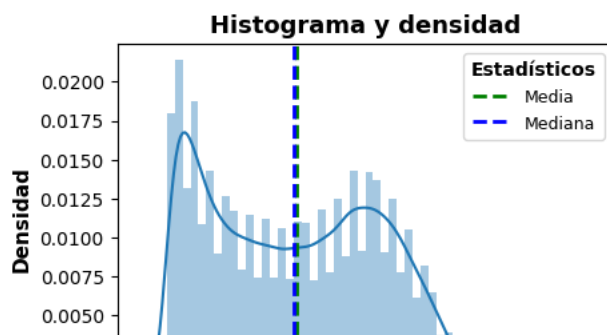
Histograma

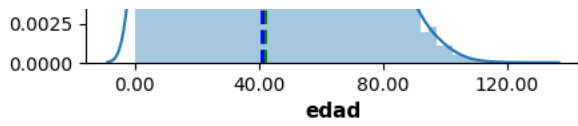
Caja y bigotes

Violín

Densidad Norm.

QQ Norm.





COSTO\_OPERATIVO\_ESTIMADO

## Estadísticos - costo\_operativo\_estimado

Frecuencia

TC y Posición

Dispersión y Forma

Normalidad

## Conteo de registros

Frec.Abs. Frec.Rel. Frec.Rel.Acum.

Con dato 76,258 100.00% 100.00%

## Gráficos descriptivos - costo\_operativo\_estimado

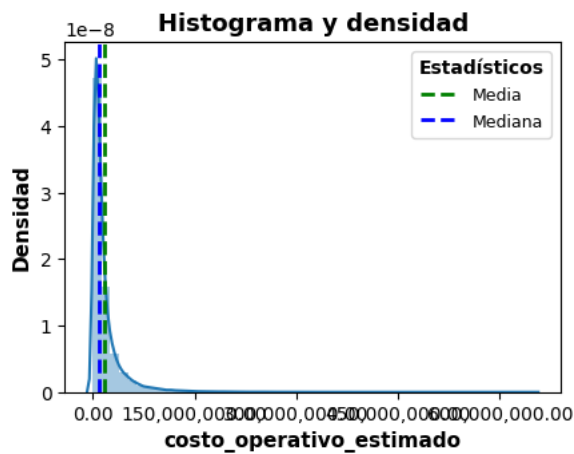
Histograma

Caja y bigotes

Violín

Densidad Norm.

QQ Norm.



PESO\_IR\_ESTIMADO

## Estadísticos - peso\_ir\_estimado

Frecuencia

TC y Posición

Dispersión y Forma

Normalidad

## Conteo de registros

Frec.Abs. Frec.Rel. Frec.Rel.Acum.

Con dato 76,258 100.00% 100.00%

## Gráficos descriptivos - peso\_ir\_estimado

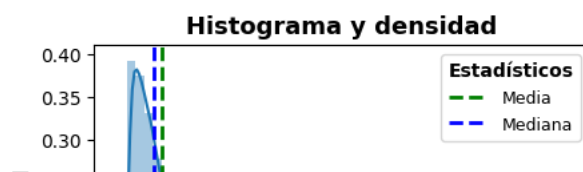
Histograma

Caja y bigotes

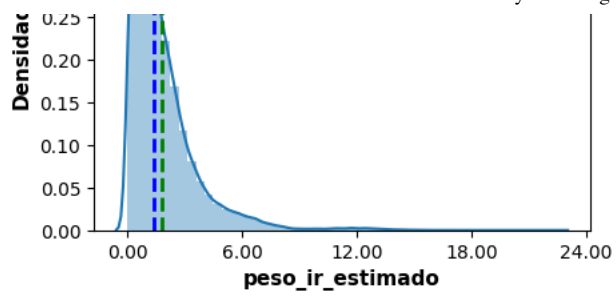
Violín

Densidad Norm.

QQ Norm.

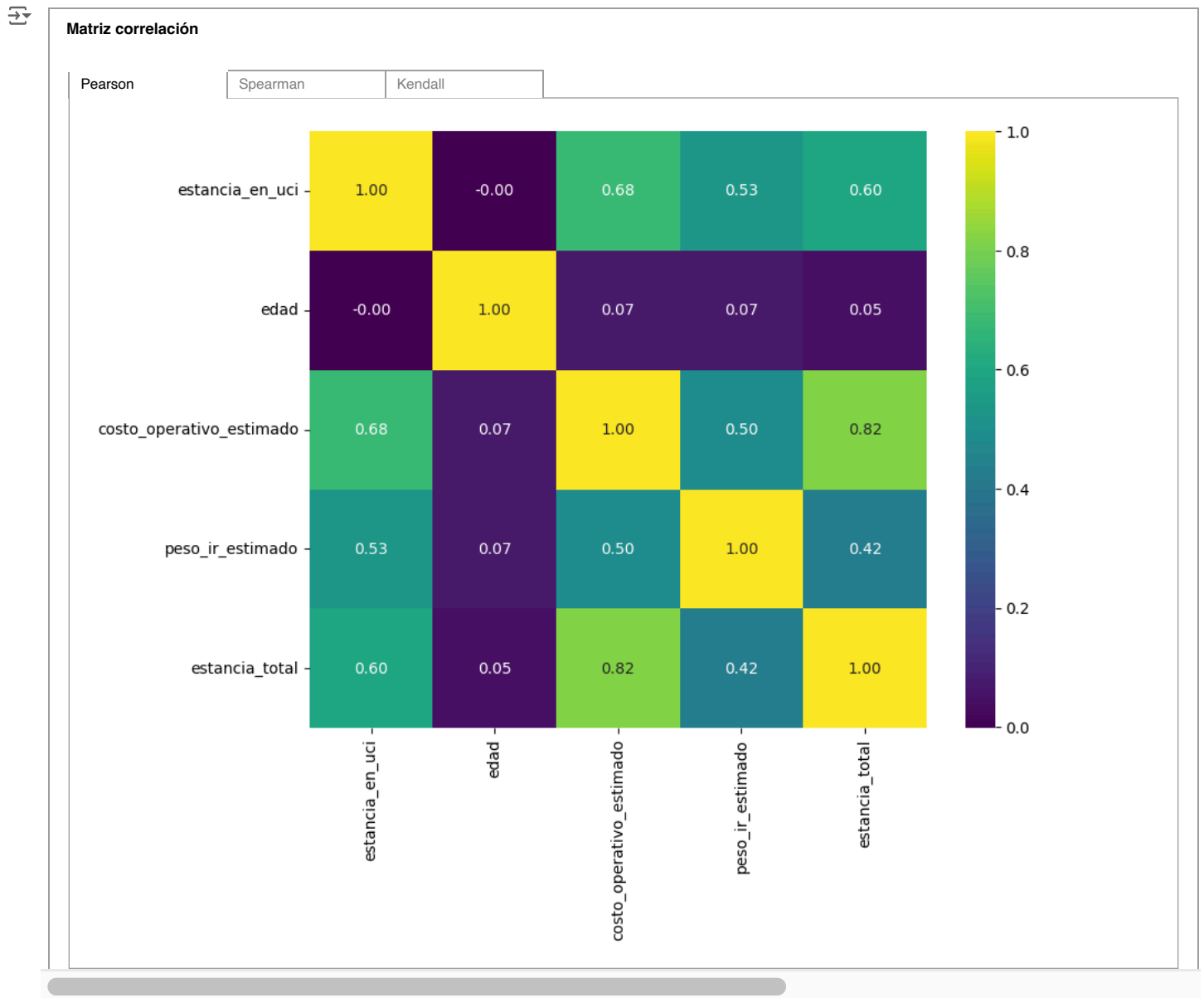






## Correlaciones

```
feats = list(feats_numericas) + [feat_target]
fu.print_corr_var(data[feats])
```



## Feats categóricas

### Test ANOVA de significancia sobre la relación con la variable de respuesta

```
from scipy.stats import f_oneway

for f in feats_categoricas:
    categories = data[f].unique()
    groups = [data[data[f] == category][feat_target] for category in categories]

    f_stat, p_value = f_oneway(*groups)
    print(f"\nANOVA p-value de la variable {f}: {p_value}")
```

```
ANOVA p-value de la variable ir_cdm: 0.0
ANOVA p-value de la variable ir_grd_base: 0.0
```

ANOVA p-value de la variable nivel\_de\_complejidad: 0.0

ANOVA p-value de la variable procedimiento\_principal: 0.0

ANOVA p-value de la variable diagnostico\_principal: 0.0

Como las p-values son significativos ( $p < 0.05$ ), se infiere que hay relación entre las variables categoricas y la de respuesta

### > Promedio de target por categoria

[ ] ↪ 1 cell hidden

### ▼ Estadísticos y descriptivas

```
for f in feats_categoricas:
    print('\n\t\t',str.upper(f), '\n')
    fu.inter_uncond_descrp_cat_var(f, col = data[f])
```



IR\_CDM

Estadísticos - ir\_cdm

Frecuencia

Conteo de registros				Conteo de frecuencias			
	Frec.Abs.	Frec.Rel.	Frec.Rel.Acum.		Frec.Abs.	Frec.Rel.	Frec.Rel.Acum.
Con				Categorías			
dato	76,258	100.00%	100.00%	enfermedades y trastornos del aparato respiratorio	13,408	17.58%	17.58%
				enfermedades y trastornos del aparato circulatorio	10,094	13.24%	30.82%
				enfermedades y trastornos sistema musculoesqueletico y tejido conectivo	7,528	9.87%	40.69%
				enfermedades y trastornos del aparato digestivo	7,522	9.86%	50.55%
				enfermedades v trastornos del sistema nervioso	6,049	7.93%	58.49%

Gráficos descriptivos - ir\_cdm



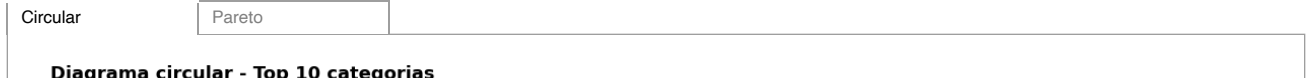
IR\_GRD\_BASE

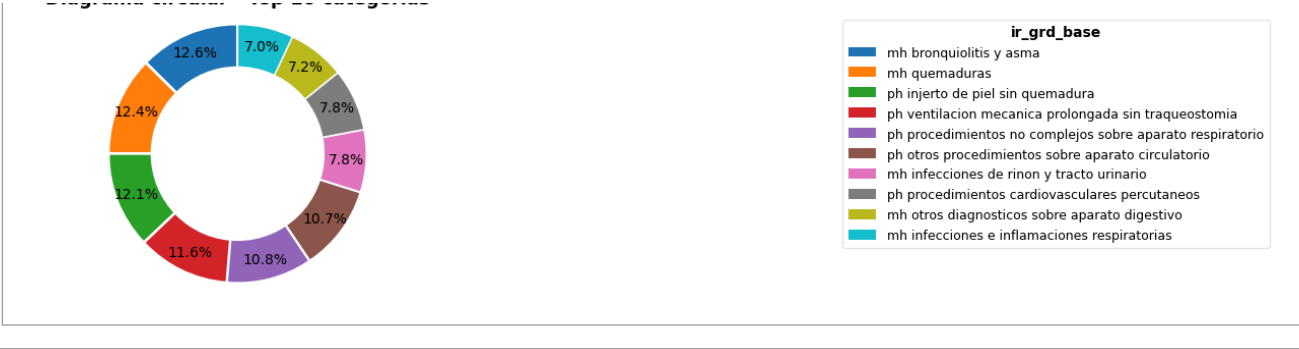
Estadísticos - ir\_grd\_base

Frecuencia

Conteo de registros				Conteo de frecuencias			
	Frec.Abs.	Frec.Rel.	Frec.Rel.Acum.		Frec.Abs.	Frec.Rel.	Frec.Rel.Acum.
Con				Categorías			
dato	76,258	100.00%	100.00%	mh bronquiolitis y asma	2,414	3.17%	3.17%
				mh quemaduras	2,376	3.12%	6.28%
				ph injerto de piel sin quemadura	2,311	3.03%	9.31%
				ph ventilacion mecanica prolongada sin traqueostomia	2,226	2.92%	12.23%
				ph procedimientos no complejos sobre aparato respiratorio	2,065	2.71%	14.94%
				ph otros procedimientos sobre aparato circulatorio	2,059	2.70%	17.64%

Gráficos descriptivos - ir\_grd\_base



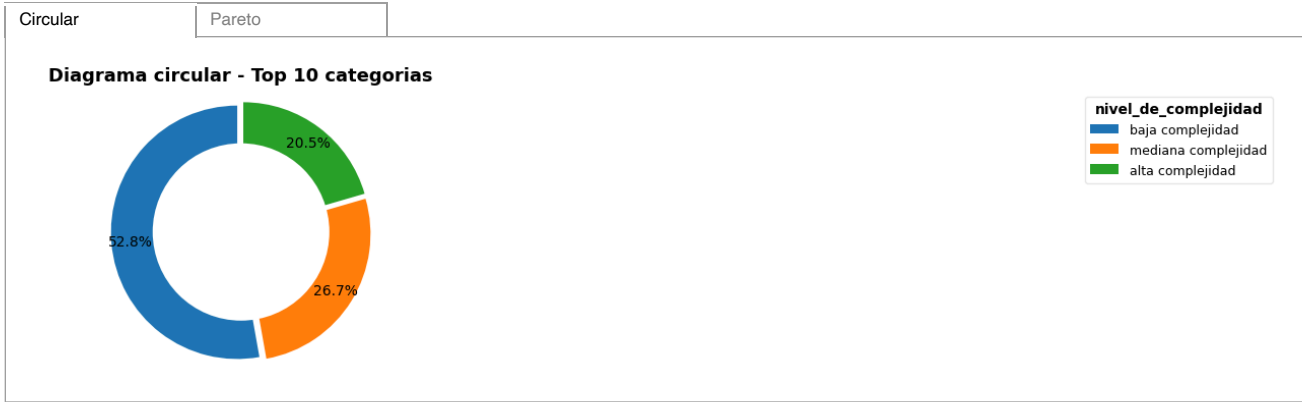


NIVEL\_DE\_COMPLEJIDAD

Estadísticos - nivel\_de\_complejidad

Frecuencia			
Conteo de registros		Conteo de frecuencias	
Frec.Abs. Frec.Rel. Frec.Rel.Acum.		Frec.Abs. Frec.Rel. Frec.Rel.Acum.	
Con dato 76,258 100.00% 100.00%		Categorías	
		baja complejidad 40,270 52.81% 52.81%	
		mediana complejidad 20,343 26.68% 79.48%	
		alta complejidad 15,645 20.52% 100.00%	

Gráficos descriptivos - nivel\_de\_complejidad



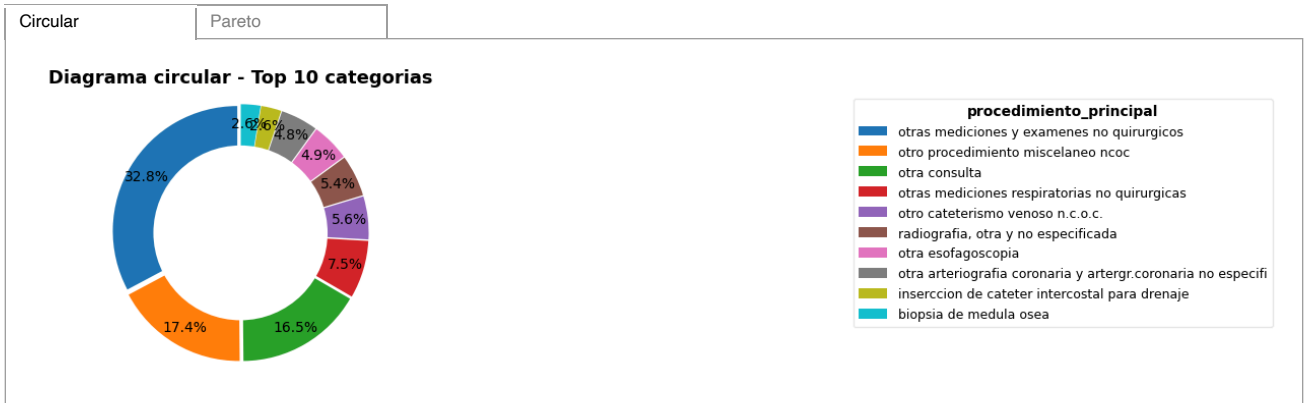
PROCEDIMIENTO\_PRINCIPAL

Estadísticos - procedimiento\_principal

Frecuencia			
Conteo de registros		Conteo de frecuencias	
Frec.Abs. Frec.Rel. Frec.Rel.Acum.		Frec.Abs. Frec.Rel. Frec.Rel.Acum.	
Con dato 76,258 100.00% 100.00%		Categorías	
		otras mediciones y examenes no quirurgicos 12,785 16.77% 16.77%	
		otro procedimiento miscelaneo ncoc 6,776 8.89% 25.65%	

otra consulta	6,435	8.44%	34.09%
otras mediciones respiratorias no quirurgicas	2,913	3.82%	37.91%
otro cateterismo venoso n.c.o.c.	2,169	2.84%	40.75%
radiografia, otra y no especificada	2,002	2.74%	42.50%

Gráficos descriptivos - procedimiento\_principal



DIAGNOSTICO\_PRINCIPAL

Estadísticos - diagnostico\_principal

Frecuencia

Con dato

Frec.Abs.	Frec.Rel.	Frec.Rel.Acum.	Categorías	Frec.Abs.	Frec.Rel.	Frec.Rel.Acum.
76,258	100.00%	100.00%	quemaduras que afectan menos del 10% de la superficie del cuerpo	1,851	2.43%	2.43%
			infeccion de vias urinarias, sitio no especificado	1,298	1.70%	4.13%
			infarto agudo del miocardio, sin otra especificacion	1,042	1.37%	5.50%
			enfermedad aterosclerotica del corazon	985	1.29%	6.79%
			insuficiencia cardiaca congestiva	965	1.27%	8.05%
			uso emergente de u07.1	740	0.98%	9.04%

Gráficos descriptivos - diagnostico\_principal



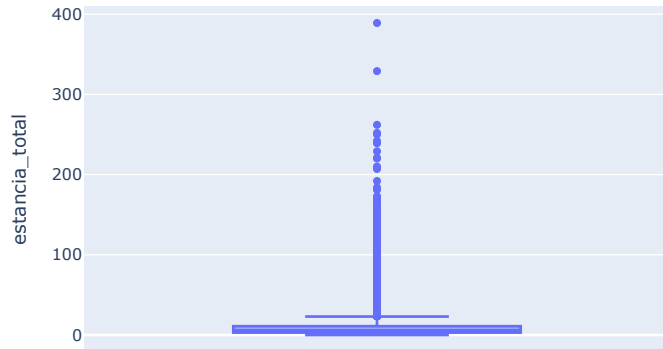


## ✓ Feat target

```
fig = px.box(data, y=feat_target, width=600, height=400)
fig.update_layout(title_text = feat_target)
fig.show()
```



estancia\_total



## ✓ ETAPA 3 - FEATURE ENGINEERING

Una vez se han analizado las variables de modelación, se procede a realizar las transformaciones necesarias para reducir el sesgo durante el entrenamiento y generar confianza sobre la evaluación de los modelos obtenidos.

### Pasos realizados

- Identificar outliers en variables numéricas
  - Evaluar distancia de Mahalanobis
  - Aplicar clustering DBSCAN
- Identificar outliers en variables categóricas
  - Aplicar moda de variable a las categorías potenciales outliers
  - Aplicar Multiple Correspondence Analysis (MCA)
  - Evaluar distancia de Hamming
- Consolidar y remover outliers del dataset
- Acotar (Winsorizing) variable de respuesta para mitigar sesgos por outliers

### Resultado final

Dataset sin outliers para modelación

```
from sklearn.preprocessing import RobustScaler, StandardScaler, MinMaxScaler
from sklearn.metrics import silhouette_score, davies_bouldin_score
from scipy.spatial.distance import cdist
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.neighbors import NearestNeighbors
from sklearn.cluster import DBSCAN
from plotly.subplots import make_subplots
import seaborn as sns
import plotly.express as px
import matplotlib.pyplot as plt
```

## ✓ 3.1 Identificar outliers de feats numéricas

```
data_numerica = data[feats_numericas]
print(data_numerica.shape)
```



```
data_numerica.head()
```

```
(76258, 4)
```

	estancia_en_uci	edad	costo_operativo_estimado	peso_ir_estimado
0	0	96	1540005.10	0.3326
1	0	4	23759810.59	4.0919
2	0	66	4997120.16	1.7552
3	0	45	8939942.81	0.9393

## ▼ Distancia de Mahalanobis

```
data_numerica.describe()
```

```
(76258, 4)
```

	estancia_en_uci	edad	costo_operativo_estimado	peso_ir_estimado
count	76258.000000	76258.000000	7.625800e+04	76258.000000
mean	1.484880	41.637795	1.763403e+07	1.833033
std	5.777062	27.489393	2.588949e+07	1.788586
min	0.000000	0.000000	2.296900e+02	0.000000
25%	0.000000	16.000000	4.671039e+06	0.641700
50%	0.000000	41.000000	9.670476e+06	1.369100
75%	0.000000	65.000000	1.969744e+07	2.420525

```
data_numerica = data_numerica.values
scaler = RobustScaler()
data_numerica_std = scaler.fit_transform(data_numerica)
data_numerica_std.shape
```

```
(76258, 4)
```

```
punto_central = np.mean(data_numerica_std, axis=0)
punto_central
```

```
array([1.48488027, 0.01301623, 0.52997104, 0.26080872])
```

```
pca = PCA(n_components=2)
pca_result = pca.fit_transform(data_numerica_std)
```

```
matriz_covarianza = np.cov(data_numerica_std, rowvar=False)
matriz_covarianza
```

```
array([[ 3.33744452e+01, -5.64849368e-03,  6.73906231e+00,
         3.06541570e+00],
       [-5.64849368e-03,  3.14730004e-01,  6.71193038e-02,
         3.79227245e-02],
       [ 6.73906231e+00,  6.71193038e-02,  2.96850207e+00,
         8.64154924e-01],
       [ 3.06541570e+00,  3.79227245e-02,  8.64154924e-01,
         1.01100422e+00]])
```

```
covarianza_inversa = np.linalg.inv(matriz_covarianza)
covarianza_inversa
```

```
array([[ 0.06111498,  0.0360978 , -0.11345044, -0.08968607],
       [ 0.0360978 ,  3.21858559, -0.11677109, -0.13036938],
       [-0.11345044, -0.11677109,  0.65983488, -0.21562573],
       [-0.08968607, -0.13036938, -0.21562573,  1.45024426]])
```

```
mahalanobis = []
```

```
for i, x in enumerate(data_numerica_std):
```

```

    distancia = (x - punto_central).T.dot(covarianza_inversa).dot(x - punto_central)
    mahalanobis.append(distancia)
mahalanobis = np.array(mahalanobis)

```

```

from scipy.stats import chi2

```

```

corte = chi2.ppf(0.95, data_numerica_std.shape[1])

```

```

outliers_feats_num_mahalanobis = np.where(mahalanobis > corte )[0]

```

```

print('Index de outliers de variables numericas')
print(outliers_feats_num_mahalanobis)

```

```

#print(data_numerica[mahalanobis > corte , :])

```

```

↪ Index de outliers de variables numericas
[ 50  72  83 ... 75369 76216 76219]

```

```

corte

```

```

↪ 9.487729036781154

```

```

outliers = mahalanobis > np.sqrt(corte)
outliers

```

```

↪ array([ True,  True, False, ..., False,  True, False])

```

```

len(outliers_feats_num_mahalanobis)

```

```

↪ 4450

```

```

mahalanobis[outliers_feats_num_mahalanobis]

```

```

↪ array([12.37103595, 37.96343939, 11.47825781, ...,  9.71938187,
        15.53970406, 39.12334896])

```

```

pca_result[0]

```

```

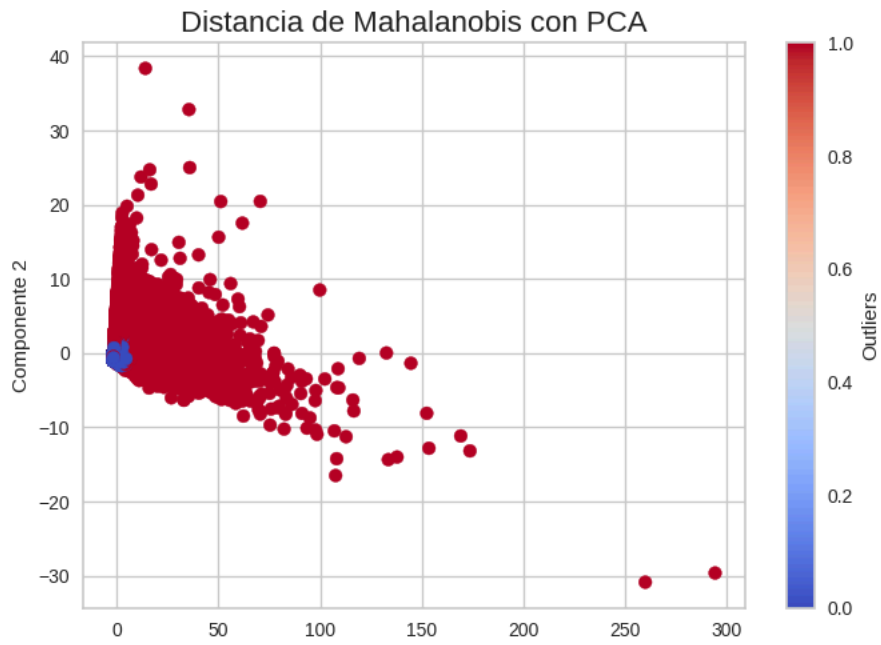
↪ array([-1.74563676, -0.81360753])

```

```

plt.scatter(pca_result[:, 0], pca_result[:, 1], c=outliers, cmap='coolwarm', marker='o')
plt.title("Distancia de Mahalanobis con PCA", fontsize=16)
plt.xlabel('Componente 1')
plt.ylabel('Componente 2')
plt.colorbar(label="Outliers")
plt.show()

```



## DBSCAN

```
scaler = StandardScaler()
df_scaled = scaler.fit_transform(data[feats_numericas])
df_scaled = pd.DataFrame(df_scaled, columns=feats_numericas)
df_scaled.head()
```



	estancia_en_uci	edad	costo_operativo_estimado	peso_ir_estimado
0	-0.257032	1.977583	-0.621647	-0.838899
1	-0.257032	-1.369184	0.236614	1.262943
2	-0.257032	0.886246	-0.488113	-0.043517
3	-0.257032	0.122310	-0.335818	-0.499690

```
neighbors = NearestNeighbors(n_neighbors=5)
neighbors_fit = neighbors.fit(df_scaled)
distances, indices = neighbors_fit.kneighbors(df_scaled)
```

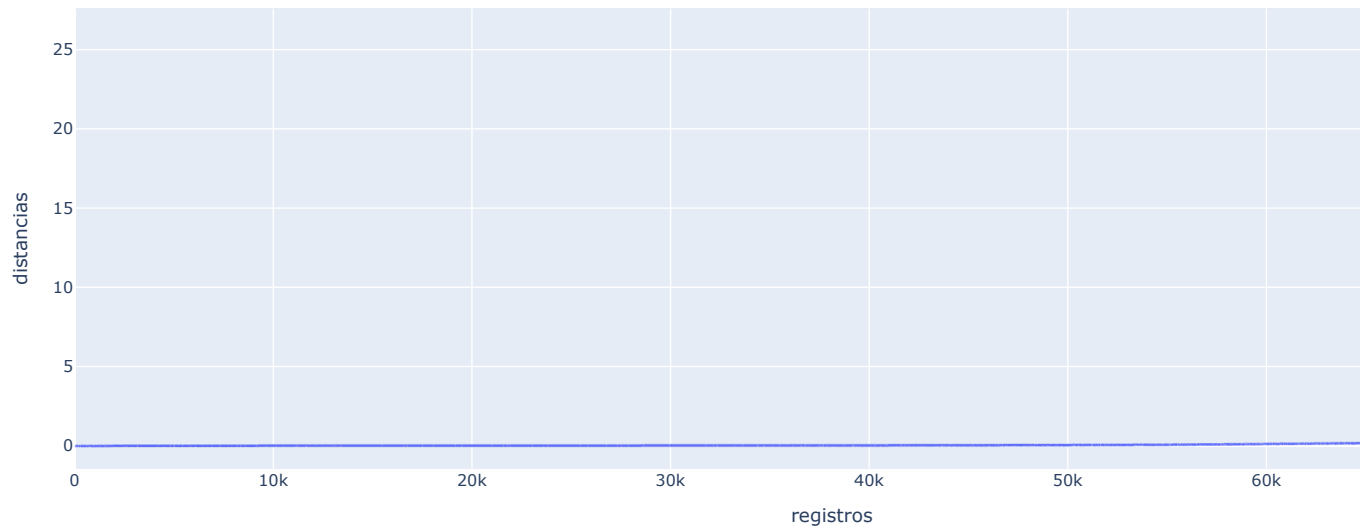
```
sorted_distances = np.sort(distances[:, -1], axis=0)
```

```
df_k = pd.DataFrame(sorted_distances).reset_index()
df_k.columns = ['registros', 'distancias']
```

```
fig = px.line(df_k, x="registros", y="distancias", title="Distancias al 5° vecino más cercano")
fig.show()
```



### Distancias al 5° vecino más cercano



De la gráfica anterior se infiere que el codo se presenta cuando  $k=1$

```
dbscan = DBSCAN(eps=1, min_samples=5, n_jobs=-1)
df_scaled['Outlier'] = dbscan.fit_predict(df_scaled)

df_scaled['Outlier'] = df_scaled['Outlier'] == -1
df_scaled.groupby('Outlier')['edad'].count()
```



edad	
Outlier	
False	75852
True	406

```
# Reduce the data to 2D using PCA for visualization
pca = PCA(n_components=2)
X_pca = pca.fit_transform(df_scaled)

outliers = df_scaled[df_scaled['Outlier'] == True]

# Add the PCA components to the dataframe
df_scaled['PCA1'] = X_pca[:, 0]
df_scaled['PCA2'] = X_pca[:, 1]

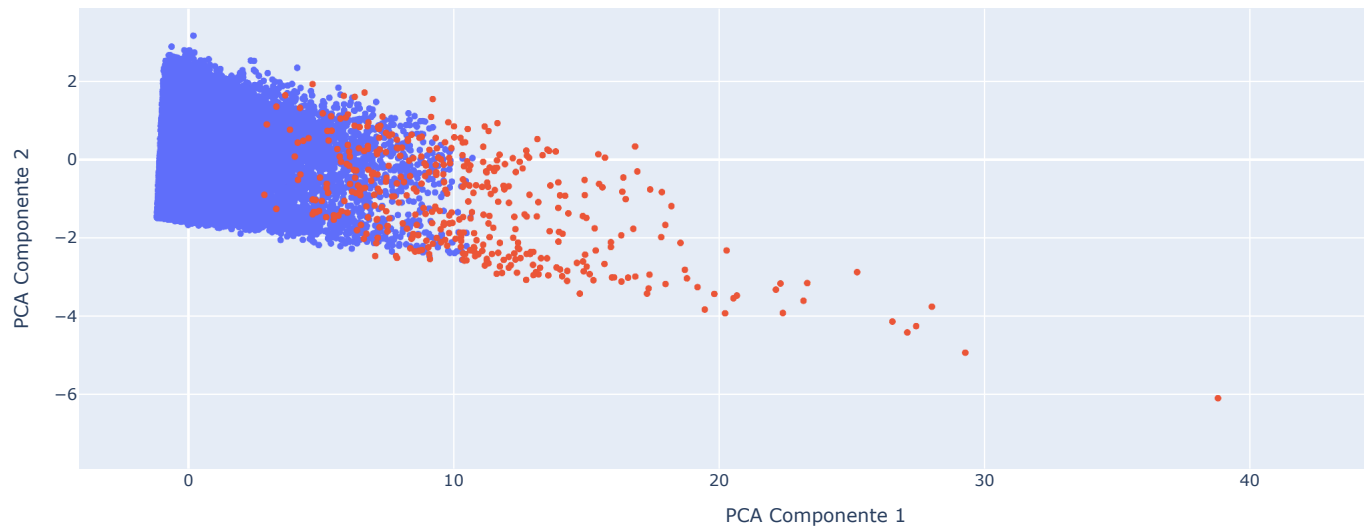
fig = px.scatter(df_scaled, x='PCA1', y='PCA2', color='Outlier',
                 title='DBSCAN Clustering (Proyección PCA )',
                 labels={'PCA1': 'PCA Componente 1', 'PCA2': 'PCA Componente 2'},
                 color_continuous_scale='Viridis',
                 category_orders={'Cluster': [-1, 0, 1, 2]},
                 )

# Add hover information (e.g., point index and cluster label)
fig.update_traces(marker=dict(size=5), hoverinfo='x+y+text',
                  hovertext=df_scaled.index.astype(str) + '<br>Cluster: ' + df_scaled['Outlier'].astype(str))

# Show the plot
fig.show()
```



### DBSCAN Clustering (Proyección PCA )



```
df = df_scaled.loc[df_scaled.Outlier == True].reset_index()
outliers_feats_num_dbscan = list(df['index'])
print(len(outliers_feats_num_dbscan))
```

406

- Se valida la correcta aplicación de DBSCAN con el score Silhouette que evidencia una buena separación de grupos de datos para facilitar la identificación de outliers

```
silhouette = silhouette_score(df_scaled, df_scaled['Outlier'])
db_index = davies_bouldin_score(df_scaled, df_scaled['Outlier'])

print("Resultados de DBSCAN:")
print(f"Coeficiente de Silueta: {silhouette}")
print(f"Índice de Davies-Bouldin: {db_index}\n")
```

Resultados de DBSCAN:  
Coeficiente de Silueta: 0.8264040136567306  
Índice de Davies-Bouldin: 0.5889918342636833

### 3.2 Identificar outliers de feats categóricas

```
for f in feats_categoricas:
    categories = data[f].unique()
    print('La variable',f,'tiene categorias distintas:', len(categories))
```

La variable ir\_cdm tiene categorias distintas: 22  
La variable ir\_grd\_base tiene categorias distintas: 254  
La variable nivel\_de\_complejidad tiene categorias distintas: 3  
La variable procedimiento\_principal tiene categorias distintas: 1069  
La variable diagnostico\_principal tiene categorias distintas: 3870

### Reemplazando moda en categorías con poca frecuencia

```
def reemplazar_categorias_poca_frecuencia(df, feats_agrupar, threshold=1):
    for f in feats_agrupar:
        conteos_categorias = df[f].value_counts()
        moda = conteos_categorias.idxmax()
        #top5 = conteos_categorias.head(5)
```

```

categorias_outliers = conteos_categorias[conteos_categorias <= threshold].index
#df[f] = df[f].apply(lambda x: top5.sample(1).idxmax() if x in categorias_outliers else x)
df[f] = df[f].apply(lambda x: moda if x in categorias_outliers else x)

```

```

return df

```

```

feats_agrupar = ['ir_grd_base', 'procedimiento_principal', 'diagnostico_principal']
data = reemplazar_categorias_poca_frecuencia(data, feats_agrupar, threshold=3)

```

```

for f in feats_categoricas:
    categories = data[f].unique()
    print('La variable', f, 'tiene categorias distintas:', len(categories))

```

```

↗ La variable ir_cdm tiene categorias distintas: 22
La variable ir_grd_base tiene categorias distintas: 247
La variable nivel_de_complejidad tiene categorias distintas: 3
La variable procedimiento_principal tiene categorias distintas: 652
La variable diagnostico_principal tiene categorias distintas: 1995

```

```

data_categorica = data[feats_categoricas]
print(data_categorica.shape)
data_categorica.head()

```

```

↗ (76258, 5)

```

	ir_cdm	ir_grd_base	nivel_de_complejidad	procedimiento_principal	diagnostico_principal
0	enfermedades y trastornos del sistema nervioso	mh otras enfermedades del sistema nervioso	baja complejidad	otras mediciones y exámenes no quirurgicos	tumor de comportamiento incierto o desconocido...
1	enfermedades y trastornos del sistema nervioso	ph procedimientos vasculares intracraneales	alta complejidad	biopsia de medula osea	tumor maligno de la retina
2	enfermedades y trastornos del sistema nervioso	mh otras enfermedades del sistema nervioso	baja complejidad	otro procedimiento miscelaneo ncoc	otras enfermedades cerebrovasculares especific...
	enfermedades y	ph procedimientos			trastornos de disco lumbar y otros

## ✓ Multiple Correspondence Analysis (MCA) - Feats categóricas

### ✓ MCA con One-Hot encoding

```

oh_data_cat = pd.get_dummies(data_categorica)
oh_data_cat.shape

```

```

↗ (76258, 2919)

```

```

import prince

```

```

mca = prince.MCA(
    n_components=2,
    n_iter=3,
    copy=True,
    check_input=True,
    engine='sklearn',
    random_state=42,
    one_hot=False
)

```

```

mca = mca.fit(oh_data_cat)

```

```

mca.eigenvalues_summary

```

```

↗
      eigenvalue  % of variance  % of variance (cumulative)
component
0          0.683         0.12%          0.12%

```

```
mca_coordenadas = mca.transform(oh_data_cat)
print(mca_coordenadas.head())
```

```
↗
```

	0	1
0	-0.409548	0.355293
1	-0.431348	0.706937
2	-0.516768	0.468471
3	0.860031	0.459029
4	0.234757	0.498882

✓ Se valida la correcta aplicación de MCA con el score Silhouette que evidencia una buena separación de grupos de datos para facilitar la identificación de outliers

```
kmeans = KMeans(n_clusters=3)
clusters = kmeans.fit_predict(mca_coordenadas)
```

```
silhouette = silhouette_score(mca_coordenadas, clusters)
print(f"Silhouette Score: {silhouette}")
```

```
↗ Silhouette Score: 0.6069958275284768
```

```
centroide = mca_coordenadas.mean(axis=0)
```

```
distancias = cdist(mca_coordenadas, [centroide], metric='euclidean')
corte = np.mean(distancias) + 2 * np.std(distancias)
outliers_feats_cat_OH = np.where(distancias > corte)[0]
```

```
df_mca = list(zip(mca_coordenadas.iloc[:, 0], mca_coordenadas.iloc[:, 1], distancias[:, 0]))
df_mca = pd.DataFrame(df_mca, columns = [0,1,'distancia'])
df_mca.head()
```

```
↗
```

	0	1	distancia
0	-0.409548	0.355293	0.542183
1	-0.431348	0.706937	0.828143
2	-0.516768	0.468471	0.697505
3	0.860031	0.459029	0.974864

```
def plot_PC_outliers(df_mca, outliers, x, y):
    fig = make_subplots()
    fig.add_trace(
        px.scatter(df_mca, x = x, y = y, hover_data = ['distancia']).data[0]
    )
    fig['data'][0]['showlegend']=True
    fig['data'][0]['name']='Aceptados'

    fig.add_trace(
        px.scatter(df_mca.iloc[outliers:], x = x, y = y, hover_data = ['distancia'], color_discrete_sequence=['red']).data[0]
    )
    fig['data'][1]['showlegend']=True
    fig['data'][1]['name']='Outliers'

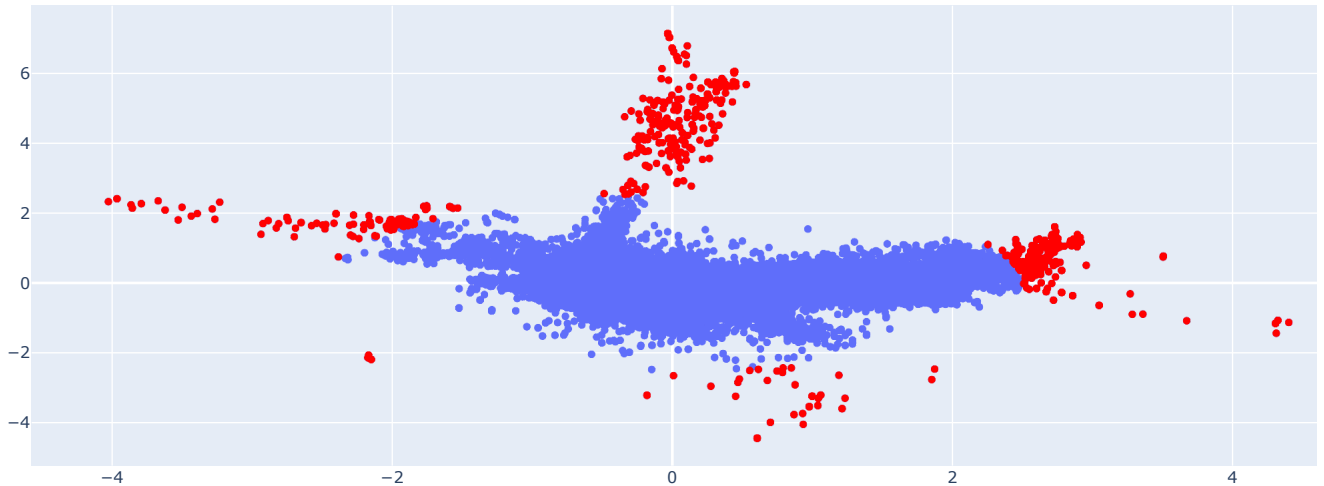
    fig.update_layout(title_text = f'PC{x} vs PC{y}')

    fig.show()

plot_PC_outliers(df_mca, outliers_feats_cat_OH, 0, 1)
```



PC0 vs PC1



### ▼ MCA con factorización

```
factorized_data_cat = data_categorica.apply(lambda col: pd.factorize(col)[0])
factorized_data_cat.shape
```

```
(76258, 5)
```

Se suma 1 al encoding para garantizar la división al momento de ajustar MCA. Esto no afecta el resultado.

```
factorized_data_cat = factorized_data_cat + 1
factorized_data_cat.head()
```

```
ir_cdm ir_grd_base nivel_de_complejidad procedimiento_principal diagnostico_principal
0      1          1          1          1          1          1
1      1          2          2          2          2          2
2      1          1          1          3          3          3
3      1          3          1          4          4          4
```

```
import prince
```

```
mca = prince.MCA(
    n_components=2,
    n_iter=10,
    copy=True,
    check_input=True,
    engine='sklearn',
    random_state=42,
    one_hot=False
)
```

```
mca = mca.fit(factorized_data_cat)
```

```
mca.eigenvalues_summary
```



↗

	eigenvalue	% of variance	% of variance (cumulative)
component			
0	0.218	61.67%	61.67%

```
mca_coordenadas = mca.transform(factorized_data_cat)
print(mca_coordenadas.head())
```

↗

	0	1
0	0.748909	1.035695
1	0.789773	0.852219
2	0.896402	0.324286
3	0.849697	0.530767
4	0.847616	0.552439

✓ Se valida la correcta aplicación de MCA con el score Silhouette que evidencia una buena separación de grupos de datos para facilitar la identificación de outliers

```
from sklearn.cluster import KMeans
```

```
kmeans = KMeans(n_clusters=3)
clusters = kmeans.fit_predict(mca_coordenadas)
```

```
silhouette = silhouette_score(mca_coordenadas, clusters)
print(f"Silhouette Score: {silhouette}")
```

↗ Silhouette Score: 0.5737581285609583

```
from scipy.spatial.distance import cdist
```

```
centroide = mca_coordenadas.mean(axis=0)
```

```
distancias = cdist(mca_coordenadas, [centroide], metric='euclidean')
corte = np.mean(distancias) + (3 * np.std(distancias)) # Regla 3-sigma que cubre el 99% de la distribución
outliers_feats_cat_fact = np.where(distancias > corte)[0]
```

```
corte
```

↗ 2.0408802356239644

```
len(outliers_feats_cat_fact)
```

↗ 2561

```
df_mca = list(zip(mca_coordenadas.iloc[:, 0], mca_coordenadas.iloc[:, 1], distancias[:, 0]))
df_mca = pd.DataFrame(df_mca, columns = [0,1, 'distancia'])
df_mca.head()
```

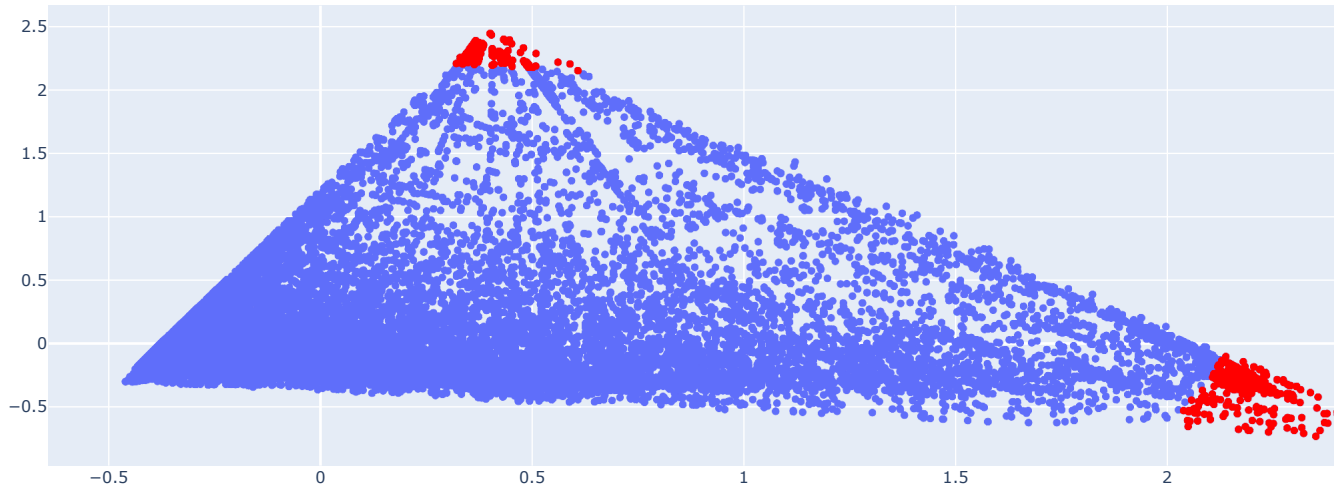
↗

	0	1	distancia
0	0.748909	1.035695	1.075105
1	0.789773	0.852219	0.962943
2	0.896402	0.324286	0.803928
3	0.849697	0.530767	0.824408

```
plot_PC_outliers(df_mca, outliers_feats_cat_fact, 0, 1)
```



PC0 vs PC1



### > DBSCAN - Distancia de Hamming

[ ] ↪ 6 cells hidden

### ✓ 3.3 Remover outliers calculados

### > Cargar indices de outliers calculados previamente

▶ ↪ 1 cell hidden

### ✓ Consolidar y exportar indices de outliers

```
print('Outliers de variables numéricas - Mahalanobis:', len(outliers_feats_num_mahalanobis))
print('\nOutliers de variables numéricas - DBSCAN:', len(outliers_feats_num_dbscan))
print('\nOutliers de variables categóricas - One Hot:', len(outliers_feats_cat_OH))
print('\nOutliers de variables categóricas - Factorización:', len(outliers_feats_cat_fact))
```

```
Outliers de variables numéricas - Mahalanobis: 4450
Outliers de variables numéricas - DBSCAN: 406
Outliers de variables categóricas - One Hot: 2092
Outliers de variables categóricas - Factorización: 2561
```

```
from itertools import chain
```

```
outliers = list(chain(outliers_feats_num_dbscan, outliers_feats_cat_fact, outliers_feats_cat_OH))
outliers = list(set(outliers))
data_sin_outliers = data.drop(data.index[outliers])
print('\nCantidad de outliers total removidos:', len(outliers))
```



Cantidad de outliers total removidos: 5030

```
perc = 1 - data_sin_outliers.shape[0] / data.shape[0]
print("\nRegistros del dataset inicial: ", data.shape[0])
```

```
print("\nRegistros del dataset sin outliers: ", data_sin_outliers.shape[0])
print("\nPorcentaje de datos removidos como outliers: "+"{:.1%}".format(perc))
```



Registros del dataset inicial: 76258

Registros del dataset sin outliers: 71228

Porcentaje de datos removidos como outliers: 6.6%

## ✓ Validar aplicación adecuada de outliers

```
for f in feats_numericas:
    fig = px.box(data_sin_outliers, y=f, width=600, height=400)
    fig.update_layout(title_text = f)
    fig.show()
```



Show hidden output

## ✓ 3.4 Acotar variable de respuesta por encima de percentil 99 (Winsorizing)

```
target_p99 = np.percentile(data_sin_outliers[feat_target], 99)
data_sin_outliers[feat_target] = data_sin_outliers[feat_target].clip(upper=target_p99)
data_sin_outliers.shape
```



(71228, 10)

```
data_sin_outliers[feat_target].describe(percentiles=[.01, .1, .2, .25, .4, .5, .75, .9, .95, .99])
```



	estancia_total
count	71228.000000
mean	9.174426
std	9.826989
min	0.000000
1%	0.000000
10%	2.000000
20%	3.000000
25%	3.000000
40%	5.000000
50%	6.000000
75%	11.000000
90%	21.000000
95%	30.000000
99%	54.000000
max	54.000000

## › Ejecutar ÚNICAMENTE cuando se actualicen los outliers para exportar a CSV

[ ] ↪ 1 cell hidden

## ✓ ETAPA 4 - PREPARACIÓN DE LOS DATOS DE MODELACIÓN

Con el dataset depurado de outliers se puede proceder a la preparación final de datos para el proceso de modelación.

### Pasos realizados

- Particionar el dataset en conjuntos de train y test

- Aplicar target encoding regularizado a las variables categóricas
- Estandarizar los datos codificados para reducir la varianza de entrenamiento
- Validar multicolinealidad del dataset final de entrenamiento

## Resultado final

Dataset listo para modelar

```
from sklearn.model_selection import train_test_split, GridSearchCV, RandomizedSearchCV
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.metrics import accuracy_score, confusion_matrix, roc_auc_score, precision_score, recall_score, roc_curve
from sklearn.preprocessing import StandardScaler
import category_encoders as ce
```

### ✓ 4.1 Particionar, codificar y estandarizar el dataset

```
feats_train = list(data_sin_outliers.columns)
feats_train.remove(feats_target)
feats_train
```

```
['ir_cdm',
 'ir_grd_base',
 'nivel_de_complejidad',
 'procedimiento_principal',
 'diagnostico_principal',
 'estancia_en_uci',
 'edad',
 'costo_operativo_estimado',
 'peso_ir_estimado']
```

```
#Definir variables dependientes e independientes
X = data_sin_outliers[feats_train]
y = data_sin_outliers[feats_target]
```

```
#Generar particiones
X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    test_size=0.1,
                                                    stratify=y,
                                                    random_state=42)
```

```
perc_train = X_train.shape[0] / data_sin_outliers.shape[0]
print("Porcentaje de datos en partición train: {:.1%}".format(perc_train)+" - registros: "+str(X_train.shape[0]))
```

```
perc_test = X_test.shape[0] / data_sin_outliers.shape[0]
print("Porcentaje de datos en partición test: {:.1%}".format(perc_test)+" - registros: "+str(X_test.shape[0]))
```

```
# Codificar variables categoricas de entrenamiento
smoothing = 1
encoder = ce.TargetEncoder(cols=list(feats_categoricas), smoothing=smoothing)
X_train_encoded = encoder.fit_transform(X_train, y_train)
X_test_encoded = encoder.transform(X_test)
```

```
# Estandarizar datos de entrenamiento
scaler = StandardScaler()
#scaler = MinMaxScaler()
#scaler = RobustScaler()
X_train_std = scaler.fit_transform(X_train_encoded[feats_train])
X_test_std = scaler.transform(X_test_encoded[feats_train])
```

```
↳ Porcentaje de datos en partición train: 90.0% - registros: 64105
Porcentaje de datos en partición test: 10.0% - registros: 7123
```

```
df = X_test
null_mask = df.isnull().any(axis=1)
null_rows = df[null_mask]
null_rows
```


```
↳ ir_cdm ir_grd_base nivel_de_complejidad procedimiento_principal diagnostico_principal estancia_en_uci edad costo_oper
```

## ✓ 4.2 Validar multicolinealidad en data de entrenamiento

```
from statsmodels.stats.outliers_influence import variance_inflation_factor

vif_data = pd.DataFrame()
vif_data["Variable"] = X_train[feats_train].columns
vif_data["VIF"] = [variance_inflation_factor(X_train_std, i) for i in range(X_train_std.shape[1])]

vif_data
```



	Variable	VIF
0	ir_cdm	1.181699
1	ir_grd_base	2.939995
2	nivel_de_complejidad	2.304809
3	procedimiento_principal	1.592875
4	diagnostico_principal	1.322133
5	estancia_en_uci	2.112041
6	edad	1.038850
7	costo_operativo_estimado	2.187669

## ✓ ETAPA 5 - TORNEO DE SELECCIÓN DE MODELOS

En este punto, antes de iniciar el proceso de ajuste de mejor modelo, se realizan torneos de selección de modelos para definir cuales tipos de regresión son las que ofrecen mejores resultados para la estimación de la estancia hospitalaria. Del mejor torneo resultante, se toma el top 3 de modelos para luego proceder al ajuste de hiperparámetros y así llegar al modelo final.

### Pasos realizados


- Torneos de modelos variando configuraciones de:
  - Tipos de outliers removidos
  - Regularización del target encoding
  - Tipo de estandarización de variables
  - Aplicación de winsorizing

### Resultado final

Modelos seleccionados para ajuste de hiperparámetros

## ✓ Definir data final de entrenamiento para torneo de modelos

```
data_train = pd.DataFrame(X_train_std, columns = feats_train)
data_train[feat_target] = y_train.reset_index()[feat_target]
data_train.head()
```



	ir_cdm	ir_grd_base	nivel_de_complejidad	procedimiento_principal	diagnostico_principal	estancia_en_uci	edad	cost
0	-0.303008	0.628577	1.827805	0.073536	-0.071589	-0.316623	0.866342	
1	0.325284	0.142597	1.827805	-0.134502	0.365664	1.152745	1.808474	
2	1.326591	0.373973	0.066423	0.019791	-1.386132	-0.316623	-0.293206	
3	-0.633815	0.461289	-0.766410	-1.087831	2.676500	-0.316623	-1.488989	
4	0.981151	1.274282	1.827805	2.645030	-0.093765	-0.316623	1.373644	

## ✓ Configurar experimentos en el módulo de regresión de PyCaret