✨ ¿Cuánto cuesta enfermar?

Predicción de Costos Médicos con Machine Learning

IRONHACK

Julio 2025

Núria Paredes

✨ Agenda

1. 🎯 Objetivo
2. 📊 Dataset Original
3. Metodología: Limpieza y Transformación
4. Análisis Exploratorio de Datos
5. 🔍 Identificación de Patrones
6. 💡 Hallazgos Relevantes
7. 🤖 Modelado Predictivo
   - 7.1 Modelos supervisados (numéricas)
   - 7.2 Modelos no supervisados (categóricas)

8 Conclusiones

Objetivo

Predicción de costos médicos y análisis de patrones hospitalarios a través de datos y aprendizaje automático.

Preguntas Clave:

- ¿Podemos predecir costos y la duración de estancia en los hospitales?
- ¿Qué influye en los costos hospitalarios?
- ¿Es diferente el costo por género?
- ¿Es diferente por enfermedad y género?
- ¿Qué aseguradora tiene más recaídas para una determinada enfermedad?

Contexto

- El sistema de salud de EEUU es complejo, fragmentado y mixto (público-privado) y dependiente del sector privado.
- Cobertura parcial, altos costos y desigualdades, en el acceso.
- Esperanza de vida: 77 años
- Avances como la Ley ACA han mejorado la cobertura, pero no la han universalizado.

Dataset original

- Fuente: https://www.kaggle.com/datasets/prasad22/healthcare-dataset
- 55500 registros, 15 variables.
- Var: Nombre, Edad, Género, Grupo Sanguíneo, Condición Médica, Fecha Admisión, Doctor, Hospital, Aseguradora, Monto Facturación, Numero Habitación, Tipo Admisión, Fecha Alta, Medicación, Resultados

SINTÉTICO (no representa complejidad real, es para fines educativos)

Variables en análisis

- Variable dependiente: Monto Facturación (a predecir/objetivo)
- Variables independientes: Resto

- Var_time: Fecha Admisión, Fecha Alta
- Var_num: Edad, Monto Facturación, Numero Habitación
    - Días de hospitalización
- Var_cat: Nombre, Género, Grupo Sanguíneo, Condición Médica, Doctor, Hospital, Aseguradora, Tipo Admisión, Medicación, Resultados Condición médica, Aseguradora

- Variables descartadas:
    - Nombre, Numero Habitación (administrativas)
    - Doctor, Hospital, (alta cardinalidad)

1. Objetivo
2. Dataset Original
3. **Metodología: Limpieza y Transformación**
4. Análisis Exploratorio de Datos
5. Identificación de Patrones
6. Hallazgos Relevantes
7. Modelado Predictivo
    - 7.1 Modelos supervisados (numéricas)
    - 7.2 Modelos no supervisados (categóricas)
8. Conclusiones

Limpieza y Transformación

- Manipulación de registros faltantes, duplicados

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 55500 entries, 0 to 55499
Data columns (total 15 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Name                55500 non-null  object
 1   Age                 36865 non-null  float64
 2   Gender              36865 non-null  object
 3   Blood Type          36865 non-null  object
 4   Medical Condition   36865 non-null  object
 5   Date of Admission   36865 non-null  object
 6   Doctor              36865 non-null  object
 7   Hospital            36865 non-null  object
 8   Insurance Provider  36865 non-null  object
 9   Billing Amount      36865 non-null  float64
 10  Room Number         36865 non-null  float64
 11  Admission Type      36865 non-null  object
 12  Discharge Date      36865 non-null  object
 13  Medication          36865 non-null  object
 14  Test Results        36865 non-null  object
dtypes: float64(3), object(12)
memory usage: 6.4+ MB
```

```python
duplicados = df.duplicated().sum() #Contar el número total de filas duplicadas (exceptuando la primera ocurrencia)
print(f'El número de duplicados es {duplicados}')
# tenemos 369 duplicados

```

#36496

```
El número de duplicados es 369
```

```python
df['Date of Admission'] = pd.to_datetime(df['Date of Admission'], errors='coerce')
df['Discharge Date'] = pd.to_datetime(df['Discharge Date'], errors='coerce')

# Encuentra inconsistencias lógicas: alta antes de ingreso
inconsistencias = df[df['Discharge Date'] < df['Date of Admission']]

# Mostrar resultados
print(f"Registros con fechas inconsistentes: {len(inconsistencias)}")
print(inconsistencias[['Date of Admission', 'Discharge Date']])
```

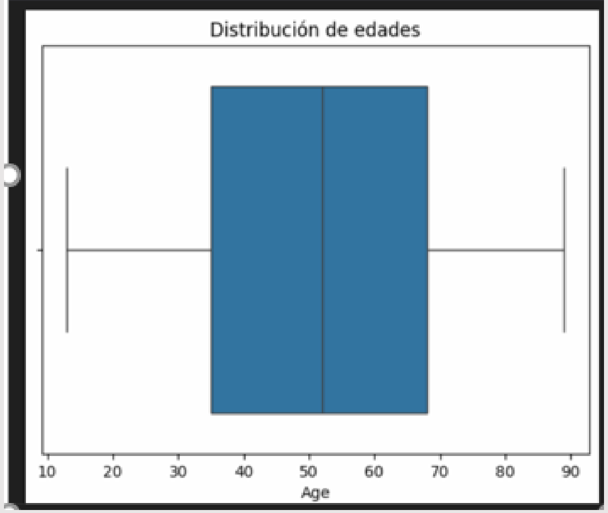
Python

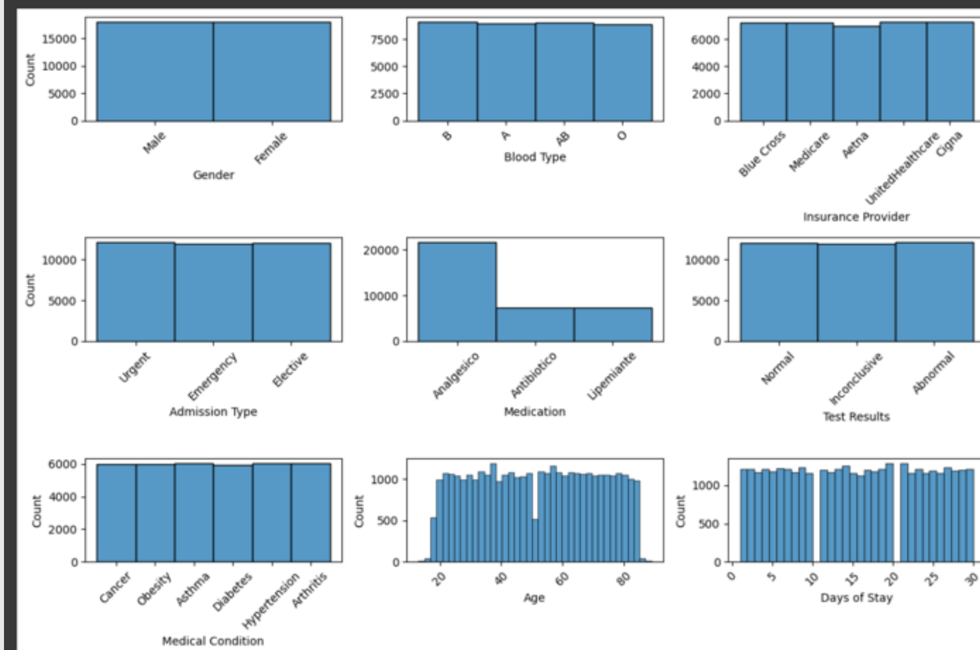
```
Registros con fechas inconsistentes: 0
Empty DataFrame
Columns: [Date of Admission, Discharge Date]
Index: []
```

- Conversión a tipo adecuado de datos: fechas y categorías y registros inconsistentes

Outliers (variables numéricas)

```python
import seaborn as sns
import matplotlib.pyplot as plt

sns.boxplot(x=df['Age'])
plt.title('Distribución de edades')
plt.show()
```





Modificación de dataset con valores más realistas reales

```python
data.to_csv('mi_data.csv', index=False)
```

Dataset Modified

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 55500 entries, 0 to 55499
Data columns (total 15 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Name                55500 non-null  object
 1   Age                 55500 non-null  int64
 2   Gender              55500 non-null  object
 3   Blood Type          55500 non-null  object
 4   Medical Condition   55500 non-null  object
 5   Date of Admission   55500 non-null  object
 6   Doctor              55500 non-null  object
 7   Hospital            55500 non-null  object
 8   Insurance Provider  55500 non-null  object
 9   Billing Amount      55500 non-null  float64
 10  Room Number         55500 non-null  int64
 11  Admission Type      55500 non-null  object
 12  Discharge Date      55500 non-null  object
 13  Medication          55500 non-null  object
 14  Test Results        55500 non-null  object
dtypes: float64(1), int64(2), object(12)
memory usage: 6.4+ MB
```

	Age	Billing Amount	Room Number
count	55500.000000	55500.000000	55500.000000
mean	39.622468	25539.316097	301.134829
std	12.848535	14211.454431	115.243069
min	13.000000	-2008.492140	101.000000
25%	30.000000	13241.224652	202.000000
50%	39.000000	25538.069376	302.000000
75%	48.000000	37820.508436	401.000000
max	84.000000	52764.276736	500.000000

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

```
Name                  0
Age                   0
Gender                0
Blood Type            0
Medical Condition     0
Date of Admission     0
Doctor                0
Hospital              0
Insurance Provider    0
Billing Amount        0
Room Number           0
Admission Type        0
Discharge Date        0
Medication            0
Test Results          0
dtype: int64
```

```
duplicados_totales = df_unicos[df_unicos.duplicated(keep=False)].sum() #comprobamos que no quedan duplicad
print(" Filas duplicadas en general:")
print(f'El numero de duplicados en el dataframe es: \n{duplicados_totales}')
```

```
 Filas duplicadas en general:
El numero de duplicados en el dataframe es:
Name                  0
Age                   0
Gender                0
Blood Type            0
Medical Condition     0
Date of Admission     0
Doctor                0
Hospital              0
Insurance Provider    0
Billing Amount      0.0
Room Number           0
Admission Type        0
Discharge Date        0
Medication            0
Test Results          0
dtype: object
```

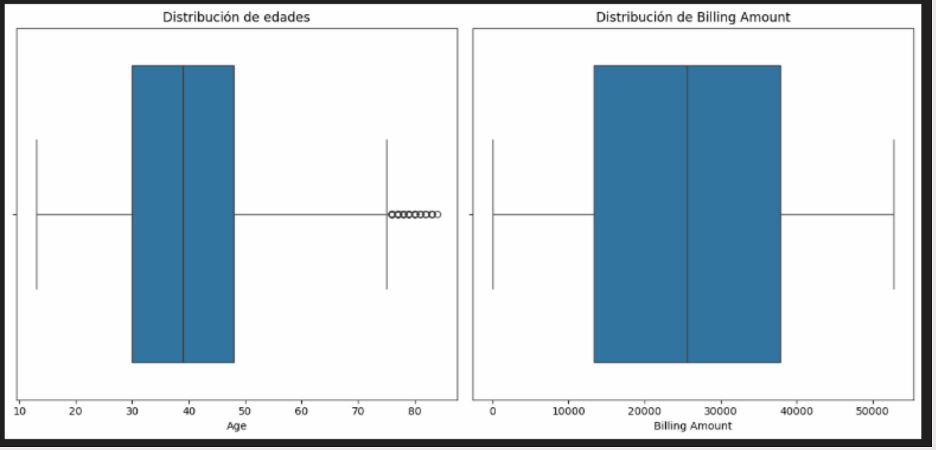
```
data= data[data['Billing Amount']>0]
data.info()
```

55392

Dataset Modified

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 55392 entries, 0 to 55391
Data columns (total 24 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Name                55392 non-null  object
 1   Age                 55392 non-null  int64
 2   Gender              55392 non-null  object
 3   Blood Type          55392 non-null  object
 4   Medical Condition   55392 non-null  object
 5   Date of Admission   55392 non-null  datetime64[ns]
 6   Doctor              55392 non-null  object
 7   Hospital            55392 non-null  object
 8   Insurance Provider  55392 non-null  object
 9   Billing Amount      55392 non-null  float64
 10  Room Number         55392 non-null  int64
 11  Admission Type      55392 non-null  object
 12  Discharge Date      55392 non-null  datetime64[ns]
 13  Medication          55392 non-null  object
 14  Test Results        55392 non-null  object
 15  Days of Stay        55392 non-null  int64
 16  AD_Mes              55392 non-null  category
 17  AD_DiaSemana        55392 non-null  category
 18  AD_Trimestre        55392 non-null  category
 19  AD_Anio             55392 non-null  category
...
 22  DD_Trimestre        55392 non-null  category
 23  DD_Anio             55392 non-null  category
dtypes: category(8), datetime64[ns](2), float64(1), int64(3), object(10)
memory usage: 7.2+ MB
```

```python
fig, axs = plt.subplots(1, 2, figsize=(12, 6))
sns.boxplot(x=data['Age'], ax=axs[0])
axs[0].set_title('Distribución de edades')

sns.boxplot(x=data['Billing Amount'], ax=axs[1])
axs[1].set_title('Distribución de Billing Amount')

plt.tight_layout()
plt.show()
#No hay outliers para
```




```
En la variable Age
Q1:30.0, Q3:48.0, IQR: 18.0
Limite inferior: 3.0
Limite superior: 75.0

Outliers using IQR method:
67       77
2074     77
2791     77
2825     77
2910     76
         ..
53318    76
53642    76
53812    76
54840    76
55185    79
Name: Age, Length: 155, dtype: int64

En la variable Billing Amount
Q1:13297.478681528502, Q3:37849.210062242, IQR: 24551.731380713492
Limite inferior: -23530.118389541734
Limite superior: 74676.80713331224

Outliers using IQR method:
Series([], Name: Billing Amount, dtype: float64)
```

Dataset Modified

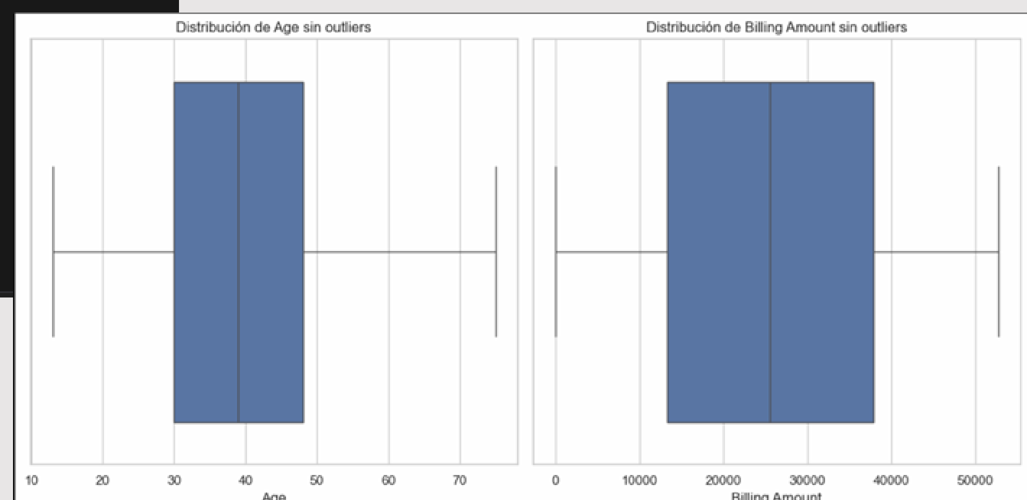
OUTLIERS

```python
quant_vars = ['Age', 'Billing Amount']

# Crear filtro para conservar filas sin outliers en ambas variables
filtro_sin_outliers = np.ones(len(data), dtype=bool)

for var in quant_vars:
    q1, q3 = np.percentile(data[var], [25, 75])
    iqr = q3 - q1
    lim_inferior = q1 - 1.5 * iqr
    lim_superior = q3 + 1.5 * iqr

    filtro_sin_outliers &= (data[var] >= lim_inferior) & (data[var] <= lim_superior)

# Filtrar datos sin outliers
data_sin_outliers = data[filtro_sin_outliers]
data_sin_outliers = data #para consistencia de la propagación del código

# Graficar boxplots sin outliers
fig, axs = plt.subplots(1, 2, figsize=(12, 6))

for i, var in enumerate(quant_vars):
    sns.boxplot(x=data_sin_outliers[var], ax=axs[i])
    axs[i].set_title(f'Distribución de {var} sin outliers')

plt.tight_layout()
plt.show()
```

✓ 0.6s

55237

```
<class 'pandas.core.frame.DataFrame'>
Index: 55237 entries, 0 to 55391
Data columns (total 28 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Name                55237 non-null  object
 1   Age                 55237 non-null  int64
 2   Gender              55237 non-null  object
 3   Blood Type          55237 non-null  object
 4   Medical Condition   55237 non-null  object
 5   Date of Admission   55237 non-null  datetime64[ns]
 6   Doctor              55237 non-null  object
 7   Hospital            55237 non-null  object
 8   Insurance Provider  55237 non-null  object
 9   Billing Amount      55237 non-null  float64
 10  Room Number         55237 non-null  int64
 11  Admission Type      55237 non-null  object
 12  Discharge Date      55237 non-null  datetime64[ns]
 13  Medication          55237 non-null  object
 14  Test Results        55237 non-null  object
 15  Days of Stay        55237 non-null  int64
 16  AD_Mes              55237 non-null  category
 17  AD_DiaSemana        55237 non-null  category
 18  AD_Trimestre        55237 non-null  category
 19  AD_Anio             55237 non-null  category
...
 26  Num_cronicas        55237 non-null  int64
 27  Num_Cronicas        55237 non-null  int64
dtypes: category(9), datetime64[ns](2), float64(1), int64(6), object(10)
memory usage: 8.9+ MB
```



Dataset Modified

```
print(data['Gender'].unique())
print(data['Blood Type'].unique())
print(data['Medical Condition'].unique())

print(data['Doctor'].unique())
print(data['Hospital'].unique())

print(data['Insurance Provider'].unique())
print(data['Admission Type'].unique())
print(data['Medication'].unique())
print(data['Test Results'].unique())
```

```
['Male' 'Female']
['B-' 'A+' 'A-' 'O+' 'AB+' 'AB-' 'B+' 'O-']
['Arthritis' 'Diabetes' 'Obesity' 'Asthma' 'Hypertension' 'Cancer']
['Matthew Smith' 'Samantha Davies' 'Tiffany Mitchell' ... 'Deborah Sutton'
 'Mary Bartlett' 'Alec May']
['Sons and Miller' 'Kim Inc' 'Cook PLC' ... 'Guzman Jones and Graves,'
 'and Williams, Brown Mckenzie' 'Moreno Murphy, Griffith and']
['Blue Cross' 'Medicare' 'Aetna' 'UnitedHealthcare' 'Cigna']
['Elective' 'Emergency' 'Urgent']
['Aspirin' 'Paracetamol' 'Lipitor' 'Ibuprofen' 'Penicillin']
['Abnormal' 'Inconclusive' 'Normal']
```

Dataset Modified

- Agrupación de variables categóricas y creación de numéricas.

```
print(data['Gender'].unique())
print(data['Blood Type'].unique())
print(data['Medical Condition'].unique())

print(data['Doctor'].unique())
print(data['Hospital'].unique())

print(data['Insurance Provider'].unique())
print(data['Admission Type'].unique())
print(data['Medication'].unique())
print(data['Test Results'].unique())
```

```
['Male' 'Female']
['B-' 'A+' 'A-' 'O+' 'AB+' 'AB-' 'B+' 'O-']
['Arthritis' 'Diabetes' 'Obesity' 'Asthma' 'Hypertension' 'Cancer']
['Matthew Smith' 'Samantha Davies' 'Tiffany Mitchell' ... 'Deborah Sutton'
 'Mary Bartlett' 'Alec May']
['Sons and Miller' 'Kim Inc' 'Cook PLC' ... 'Guzman Jones and Graves,'
 'and Williams, Brown Mckenzie' 'Moreno Murphy, Griffith and']
['Blue Cross' 'Medicare' 'Aetna' 'UnitedHealthcare' 'Cigna']
['Elective' 'Emergency' 'Urgent']
['Aspirin' 'Paracetamol' 'Lipitor' 'Ibuprofen' 'Penicillin']
['Abnormal' 'Inconclusive' 'Normal']
```

```
grupo_sanguineo_map = { # Diccionario para mapear cada tipo a un grupo
    'A+': 'A',
    'A-': 'A',
    'B+': 'B',
    'B-': 'B',
    'AB+': 'AB',
    'AB-': 'AB',
    'O+': 'O',
    'O-': 'O'
}
data['Blood Type'] = data['Blood Type'].map(grupo_sanguineo_map) # sustituir la columna Blood Type por la
data
```

```
conteo_BT = data['Blood Type'].value_counts()
print(conteo_BT)
```

```
Blood Type
A     13892
B     13869
AB    13866
O     13765
Name: count, dtype: int64
```

```
medication_map = { # Diccionario para mapear cada tipo a un grupo
    'Aspirin': 'Analgesico',
    'Ibuprofen': 'Analgesico',
    'Paracetamol':'Analgesico',
    'Penicillin': 'Antibiotico',
    'Lipitor': 'Lipemiante',
  }
data['Medication'] = data['Medication'].map(medication_map) # sustituir la columna Medication por la nueva
data
```

```
conteo_M = data['Medication'].value_counts()
print(conteo_M)
```

```
Medication
Analgesico     41632
Lipemiante      8248
Antibiotico     5512
Name: count, dtype: int64
```

```
data['Days of Stay']=(data['Discharge Date']-data['Date of Admission']).dt.days
data = data.reset_index(drop=True) #el índice se reinicia (no afecta directamente al análisis pero puede causar confusión si haces
#operaciones que dependan de índices consecutivos (como .iloc).
data.info()
```

Variables Numéricas

- Estadística descriptiva:

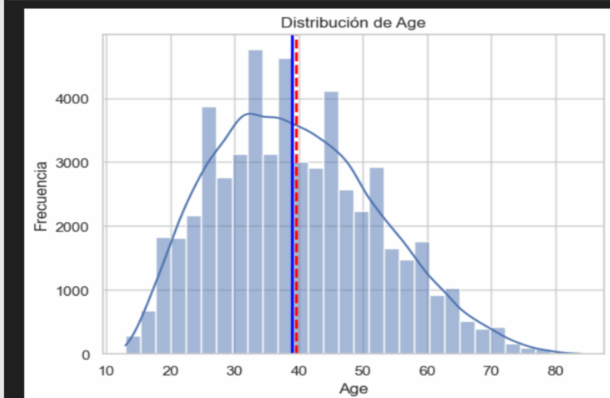
```python
quant_vars = ['Age','Days of Stay', 'Billing Amount']
for var in quant_vars:
    media = df_num[var].mean()
    devstd= df_num[var].std()
    mediana = df_num[var].median()
    varianza = df_num[var].var()
    moda= df_num[var].mode()[0]# Pandas tiene el método .mode() que devuelve una Serie (porque puede haber
    #más de una moda).Por eso, para imprimirla como número necesitas seleccionar la primera moda
    print(f'La variable: {var}, tiene una media de {media:.0f}')
    print(f'La variable: {var}, tiene una desviación estandard de {devstd:.0f}')
    print(f'La variable: {var}, tiene una mediana de {mediana:.0f}')
    print(f'La variable: {var}, tiene una varianza de {varianza:.0f}')
    print(f'La variable: {var}, tiene una moda de {moda:.0f}')
```

```
La variable: Age, tiene una media de 40
La variable: Age, tiene una desviación estandard de 13
La variable: Age, tiene una mediana de 39
La variable: Age, tiene una varianza de 165
La variable: Age, tiene una moda de 32
La variable: Days of Stay, tiene una media de 16
La variable: Days of Stay, tiene una desviación estandard de 9
La variable: Days of Stay, tiene una mediana de 15
La variable: Days of Stay, tiene una varianza de 75
La variable: Days of Stay, tiene una moda de 21
La variable: Billing Amount, tiene una media de 25590
La variable: Billing Amount, tiene una desviación estandard de 14179
La variable: Billing Amount, tiene una mediana de 25574
La variable: Billing Amount, tiene una varianza de 201034329
La variable: Billing Amount, tiene una moda de 69
```

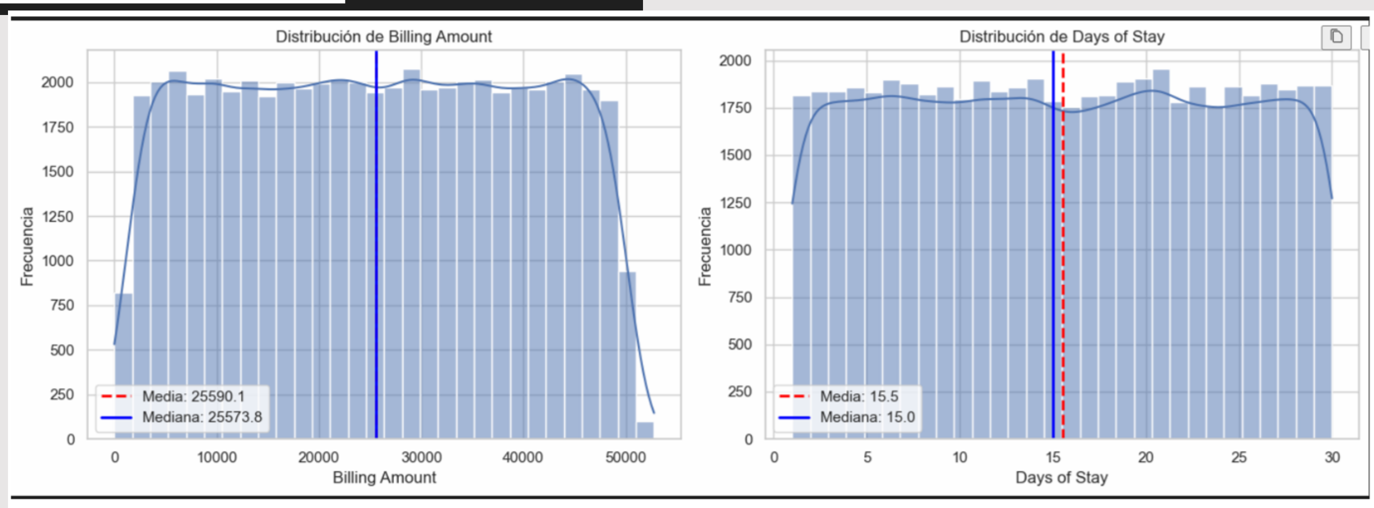
Variables Numéricas

```
mean_age = df_num['Age'].mean()
median_age = df_num['Age'].median()
sns.histplot(data=df_num, x='Age', kde=True, bins=30)
plt.axvline(mean_age, color='red', linestyle='--', linewidth=2, label=f'Media: {mean_age:.1f}')
plt.axvline(median_age, color='blue', linestyle='-', linewidth=2, label=f'Mediana: {median_age:.1f}')
plt.title('Distribución de Age')
plt.xlabel('Age')
plt.ylabel('Frecuencia')
plt.show()
```

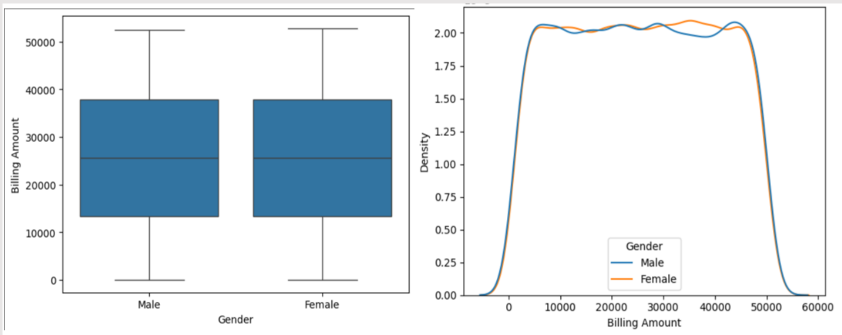
1.3s

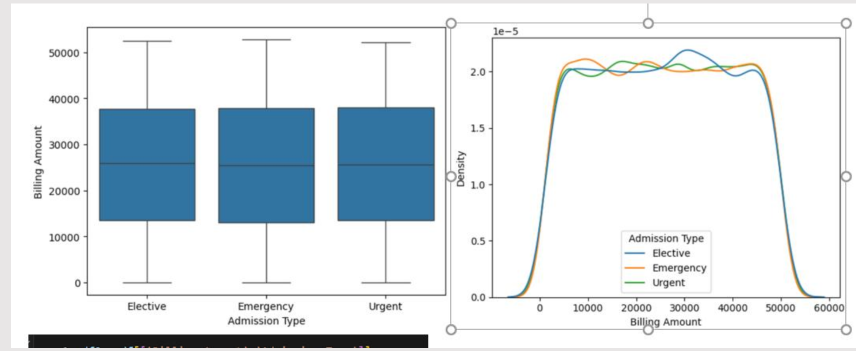


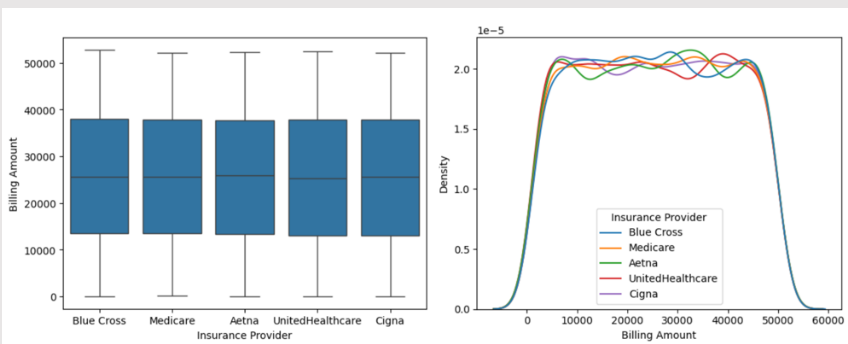
DISTRIBUCIÓN NO ES NORMAL

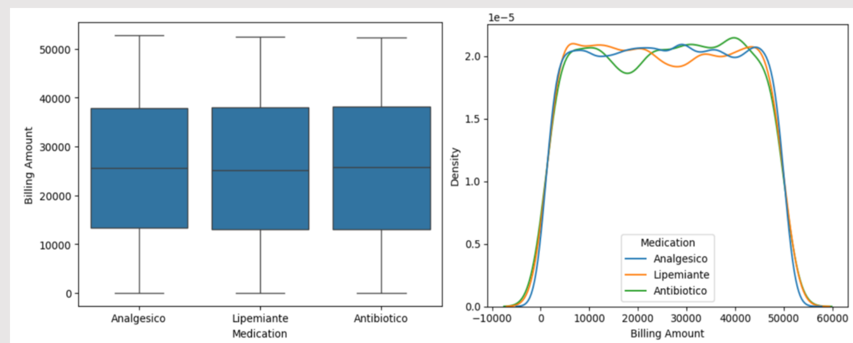


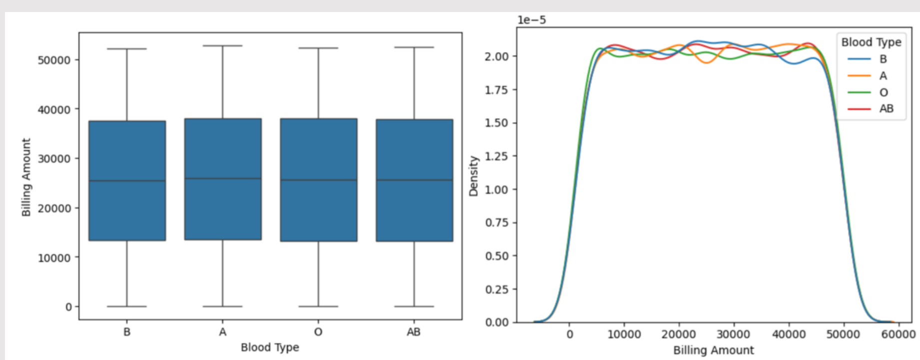
Variables Categóricas











Variables Categóricas

```python
df1 = df[['Billing Amount','Gender']]
df1['Gender'].value_counts(normalize=True)*100
```
✓ 0.0s

```
Gender
Female    52.691724
Male      47.308276
Name: proportion, dtype: float64
```

```python
df0 = df[['Billing Amount','Admission Type']]
df0['Admission Type'].value_counts(normalize=True)*100
```
✓ 0.1s

```
Admission Type
Emergency    45.158146
Urgent       34.972559
Elective     19.869295
Name: proportion, dtype: float64
```

```python
df2 = df[['Billing Amount','Insurance Provider']]
df2['Insurance Provider'].value_counts(normalize=True)*100
```
✓ 0.0s

```
Insurance Provider
Cigna               20.264659
Medicare            20.096765
UnitedHealthcare    20.049827
Blue Cross          19.936092
Aetna               19.652657
Name: proportion, dtype: float64
```

```python
df3 = df[['Billing Amount','Medication']]
df3['Medication'].value_counts(normalize=True)*100
```
✓ 0.0s

```
Medication
Analgesico     75.158868
Lipemiante     14.890237
Antibiotico     9.950895
Name: proportion, dtype: float64
```

```python
df['Medication'].value_counts(normalize=True)*100
```
✓ 0.0s

```
Medication
Paracetamol    29.873874
Aspirin        24.992793
Ibuprofen      20.286486
Lipitor        14.897297
Penicillin      9.949550
Name: proportion, dtype: float64
```

```python
df4 = df[['Billing Amount','Blood Type']]
df4['Blood Type'].value_counts(normalize=True)*100
```
✓ 0.0s

```
Blood Type
A     25.079434
B     25.037912
AB    25.032496
O     24.850159
Name: proportion, dtype: float64
```

- ~53% femenino
- Emergencia (mayoritaria)
- Aseguradora (uniforme)
- Medicación (no uniforme)
- Grupo sanguíneo (uniforme).

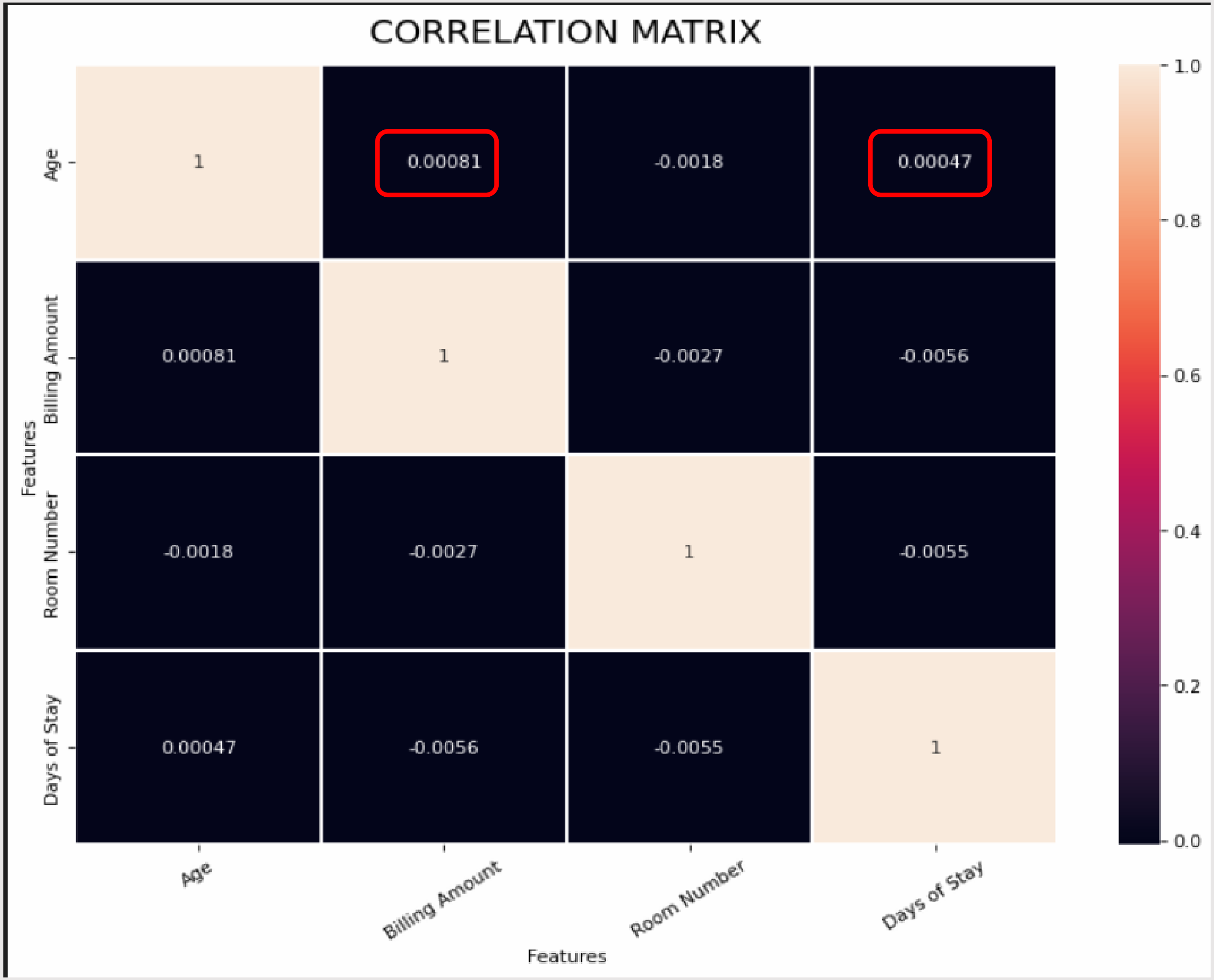
1. Objetivo
2. Dataset Original
3. Metodología: Limpieza y Transformación
4. Análisis Exploratorio de Datos
5. **Identificación de Patrones**
6. Hallazgos Relevantes
7. Modelado Predictivo
    7.1 Modelos supervisados (numéricas)
    7.2 Modelos no supervisados (categóricas)

8 Conclusiones

🔍 Identificación de Patrones

- Correlaciones entre variables numéricas
- Relaciones categóricas con numéricas
  - Categóricas vs objetivo (Billing Amount)
- Categóricas con categóricas (Chi2)
  - Categóricas vs género
- Numéricas con categóricas (t-test)
  - Numéricas vs género

Correlaciones entre variables Cuantitativas.



Relaciones entre variables categóricas con cuantitativas.

```python
var_cat = ['Gender','Blood Type','Medical Condition','Insurance Provider', 'Admission Type', 'Medication', 'Test Results']
for var in var_cat:
    contingency_table = pd.crosstab(df_num['Billing Amount'], df_cat[var])
    chi2, p, dof, expected = chi2_contingency(contingency_table)
    print(f'Variable: {var}')
    print(f'Chi2: {chi2:.4f}, p-value: {p:.4f}')
    if p < 0.05:
        print('→ Hay una relación significativa con el monto de facturación\n')
    else:
        print('→ No hay una relación significativa con l monto de facturación\n')
```

✓ 8.4s

```
...  Variable: Gender
     Chi2: 49890.0545, p-value: 0.5155
     → No hay una relación significativa con l monto de facturación

     Variable: Blood Type
     Chi2: 166176.0000, p-value: 0.0000
     → Hay una relación significativa con el monto de facturación

     Variable: Medical Condition
     Chi2: 249708.5500, p-value: 0.3917
     → No hay una relación significativa con l monto de facturación

     Variable: Insurance Provider
     Chi2: 221568.0000, p-value: 0.0000
     → Hay una relación significativa con el monto de facturación

     Variable: Admission Type
     Chi2: 99788.6045, p-value: 0.5149
     → No hay una relación significativa con l monto de facturación

     Variable: Medication
     Chi2: 99802.2130, p-value: 0.5028
     → No hay una relación significativa con l monto de facturación

     Variable: Test Results
     Chi2: 99775.9394, p-value: 0.5262
     → No hay una relación significativa con l monto de facturación
```

"Blood Type" y "Insurance Provider" mostraron una relación estadísticamente significativa ($p < 0.05$), lo que sugiere que el monto de facturación varia con los tipos de sangre y de aseguradora.

Relación entre variables categóricas: Chi2

Análisis de variables Categóricas vs Género

ENCODING O CATEGORIZACIÓN DEL GÉNERO

```
df_num2=data[['Gender','Age','Billing Amount', 'Days of Stay']]
df_num2['Gender']=df_num2['Gender'].map({'Male': 0, 'Female'
df_num2.head()
```

0.0s

	Gender	Age	Billing Amount	Days of Stay
0	0	45	18856.281306	2
1	0	38	33643.327287	6
2	0	49	27955.096079	15
3	1	29	37909.782410	30
4	1	62	14238.317814	20

```
var_cat = ['Blood Type','Medical Condition','Insurance Provider', 'Admission Type', 'Medication', 'T
for var in var_cat:
    contingency_table = pd.crosstab(df_cat['Gender'], df_cat[var])
    chi2, p, dof, expected = chi2_contingency(contingency_table)
    print(f'Variable: {var}')
    print(f'Chi2: {chi2:.4f}, p-value: {p:.4f}')
    if p < 0.05:
        print('→ Hay una relación significativa con el género\n')
    else:
        print('→ No hay una relación significativa con el género\n')
```

```
Variable: Blood Type
Chi2: 9.5383, p-value: 0.0229
→ Hay una relación significativa con el género

Variable: Medical Condition
Chi2: 10.3242, p-value: 0.0666
→ No hay una relación significativa con el género

Variable: Insurance Provider
Chi2: 4.3636, p-value: 0.3590
→ No hay una relación significativa con el género

Variable: Admission Type
Chi2: 4.4566, p-value: 0.1077
→ No hay una relación significativa con el género

Variable: Medication
Chi2: 4.6428, p-value: 0.0981
→ No hay una relación significativa con el género

Variable: Test Results
Chi2: 0.1699, p-value: 0.9186
→ No hay una relación significativa con el género

Variable: Doctor
Chi2: 40345.2403, p-value: 0.4014
→ No hay una relación significativa con el género

Variable: Hospital
Chi2: 39711.1196, p-value: 0.6415
→ No hay una relación significativa con el género
```

"Blood Type" mostró una relación estadísticamente significativa ($p < 0.05$), lo que sugiere que la distribución de tipos de sangre varía según el género.

Relación entre variables cuantitativas: t-test

Análisis de variables cuantitativas vs Género

```python
quant_vars = ['Age', 'Billing Amount','Days of Stay']
# Dividir en dos grupos por género
grupo1 = df_num2[df_num2['Gender'] == 'Male']
grupo2 = df_num2[df_num2['Gender'] == 'Female']

    # Aplicar t-test
for var in quant_vars:
    t_stat, p_val = ttest_ind(grupo1[var], grupo2[var], equal_var=False)
    print(f"Variable: {var}")
    print(f"T-student: {t_stat:.3f}, p-valor: {p_val:.3f}")

    if p_val < 0.05:
        print("→ Hay diferencia estadisticamente significativa entre los dos géneros.\n")
    else:
        print("→ No hay diferencia significativa entre los dos géneros.\n")
```

```
Variable: Age
T-student: nan, p-valor: nan
→ No hay diferencia significativa entre los dos géneros.

Variable: Billing Amount
T-student: nan, p-valor: nan
→ No hay diferencia significativa entre los dos géneros.

Variable: Days of Stay
T-student: nan, p-valor: nan
→ No hay diferencia significativa entre los dos géneros.
```

🔍 Identificación de Patrones

Relaciones detectadas:

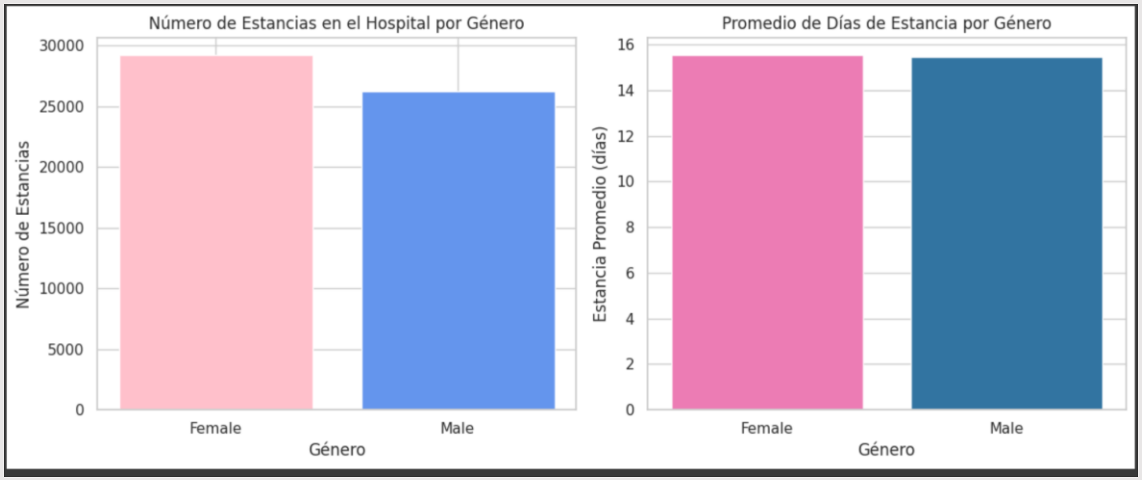
- Aseguradora ↔ Facturación

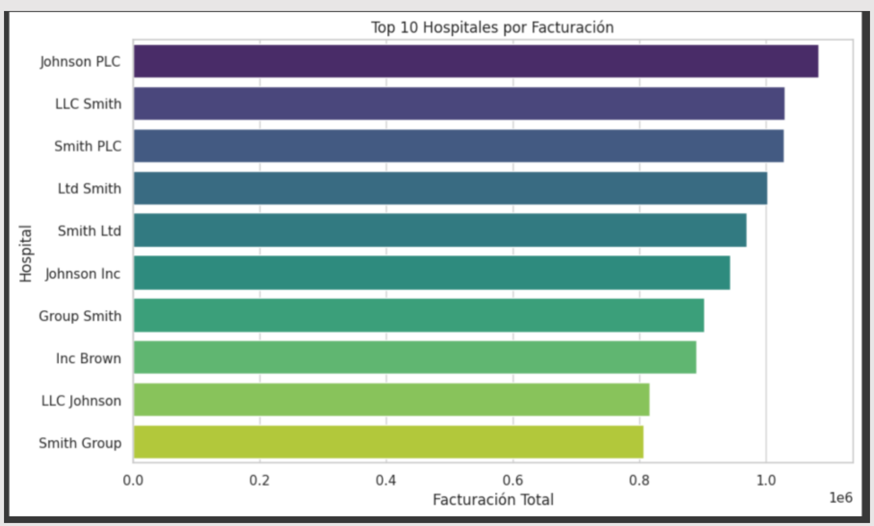
- Doctor ↔Tipo admisión, Aseguradora
- Hospital↔ Tipo admisión, Aseguradora

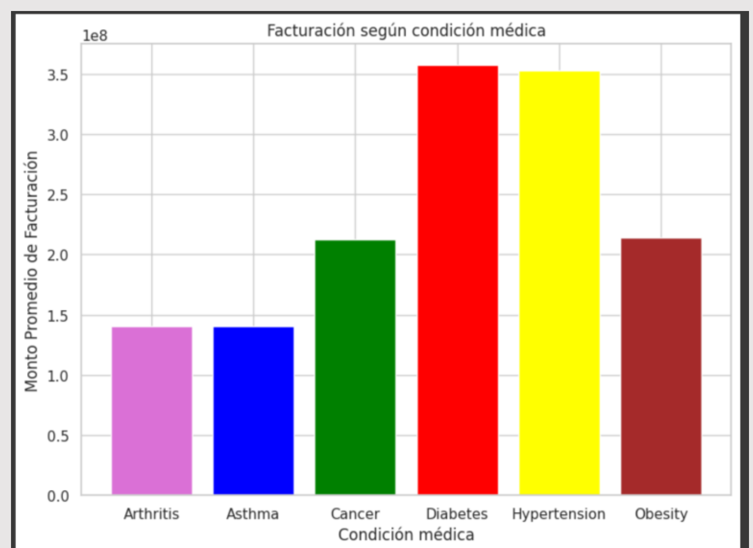
1. Objetivo
2. Dataset Original
3. Metodología: Limpieza y Transformación
4. Análisis Exploratorio de Datos
5. Identificación de Patrones
6. Hallazgos Relevantes
7. Modelado Predictivo
    7.1 Modelos supervisados (numéricas)
    7.2 Modelos no supervisados (categóricas)

8 Conclusiones

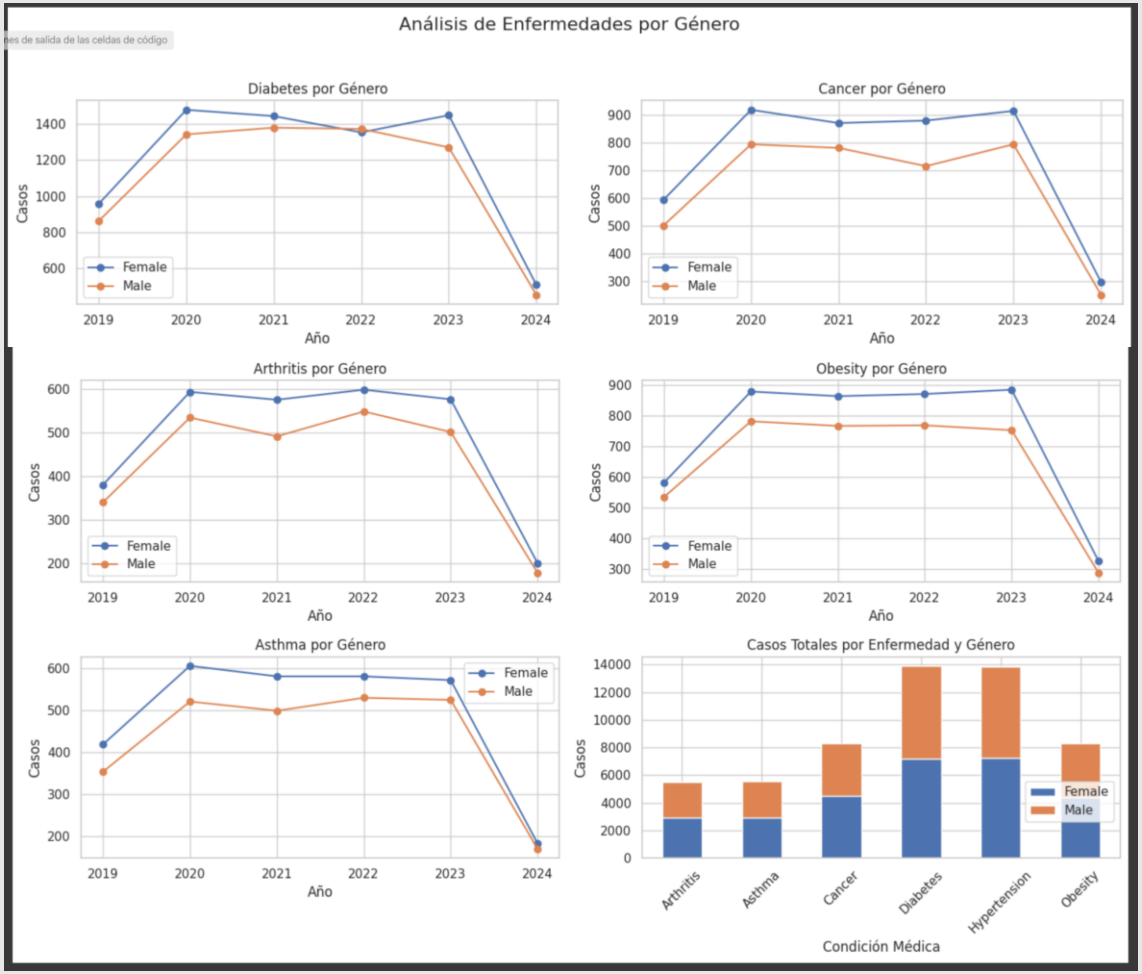
Hallazgos Relevantes





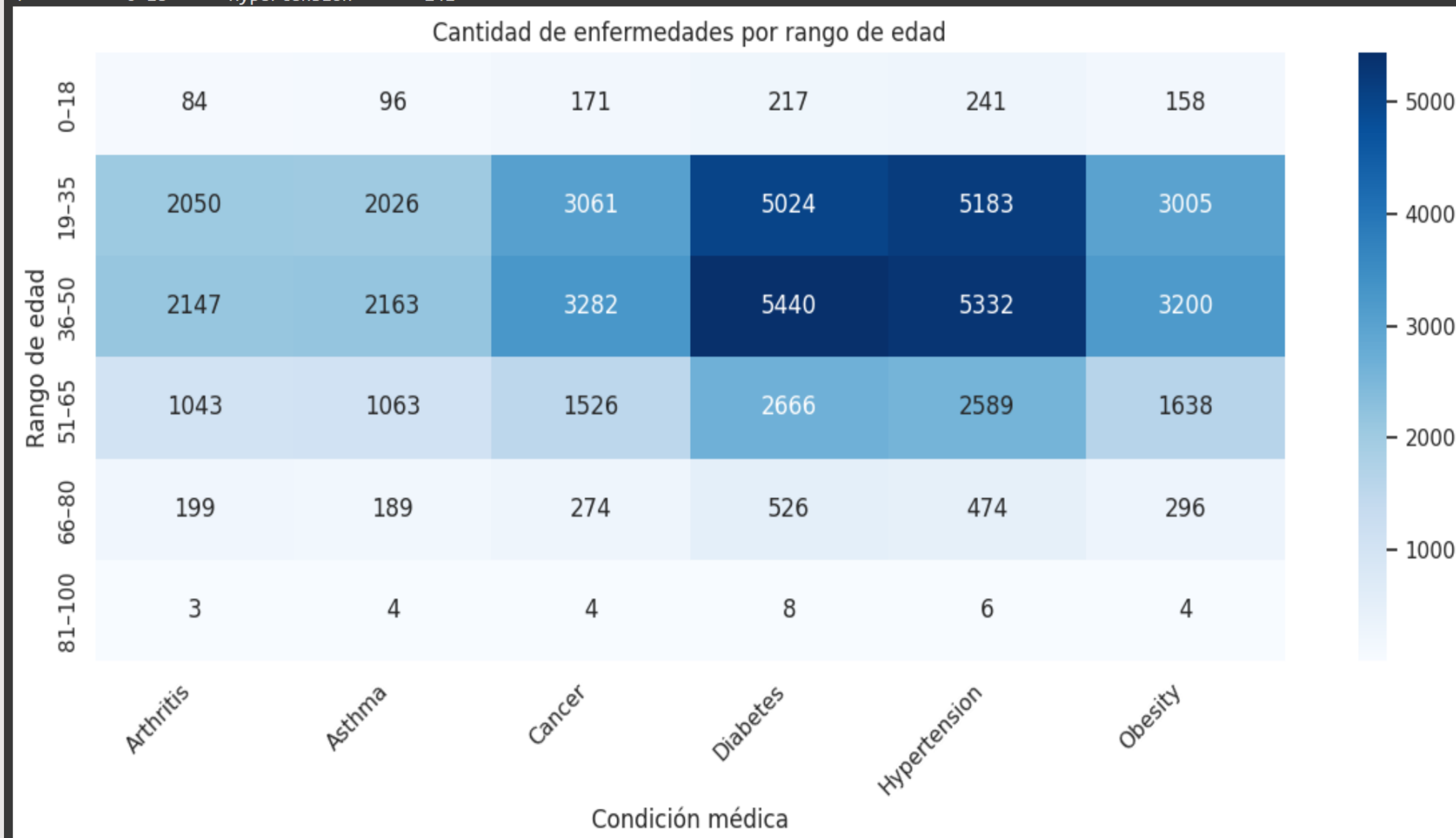


Hallazgos Relevantes

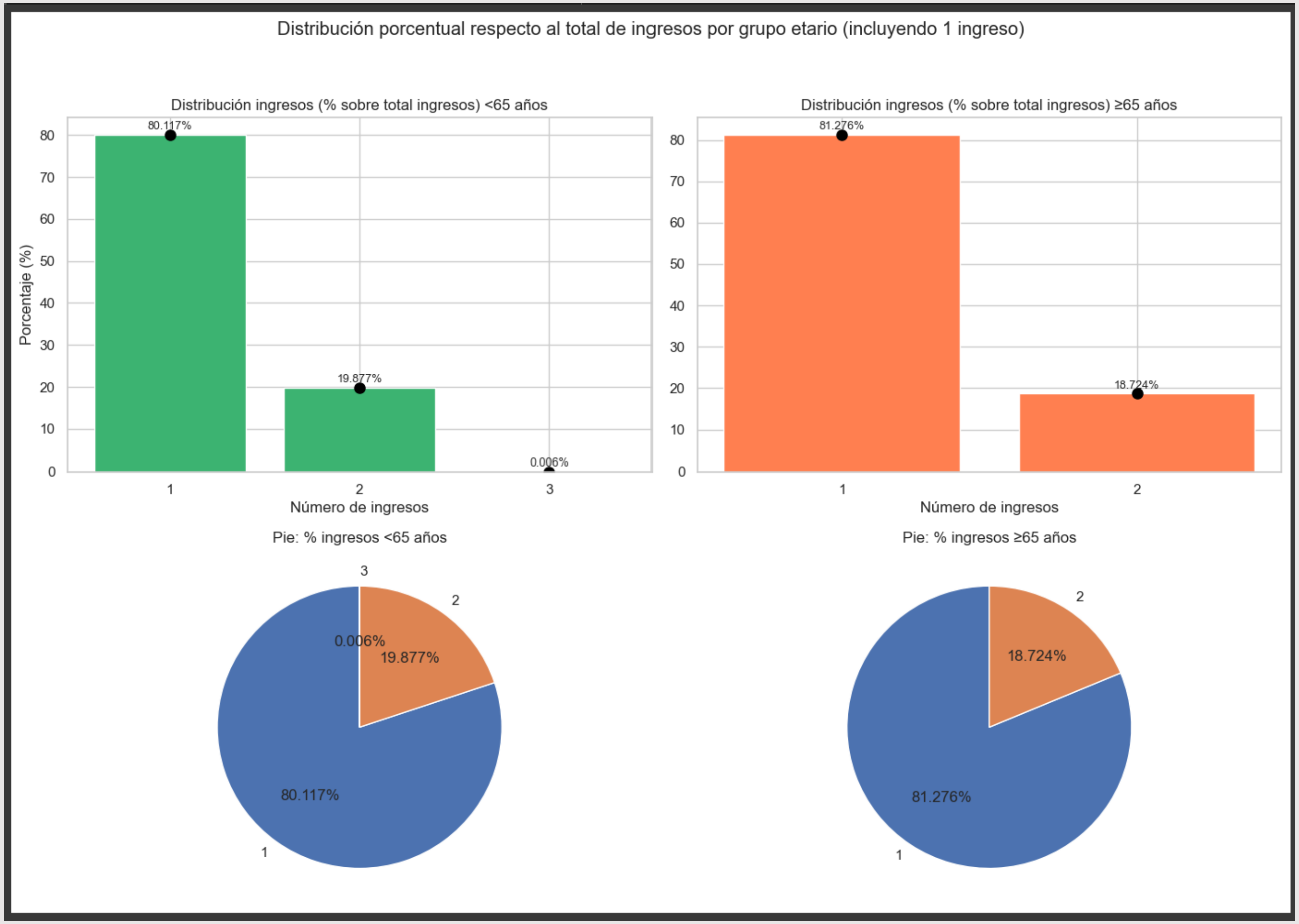


Hallazgos Relevantes

	Rango de Edad	Medical Condition	Cantidad
0	0-18	Arthritis	84
1	0-18	Asthma	96
2	0-18	Cancer	171
3	0-18	Diabetes	217
4	0-18	Hypertension	241

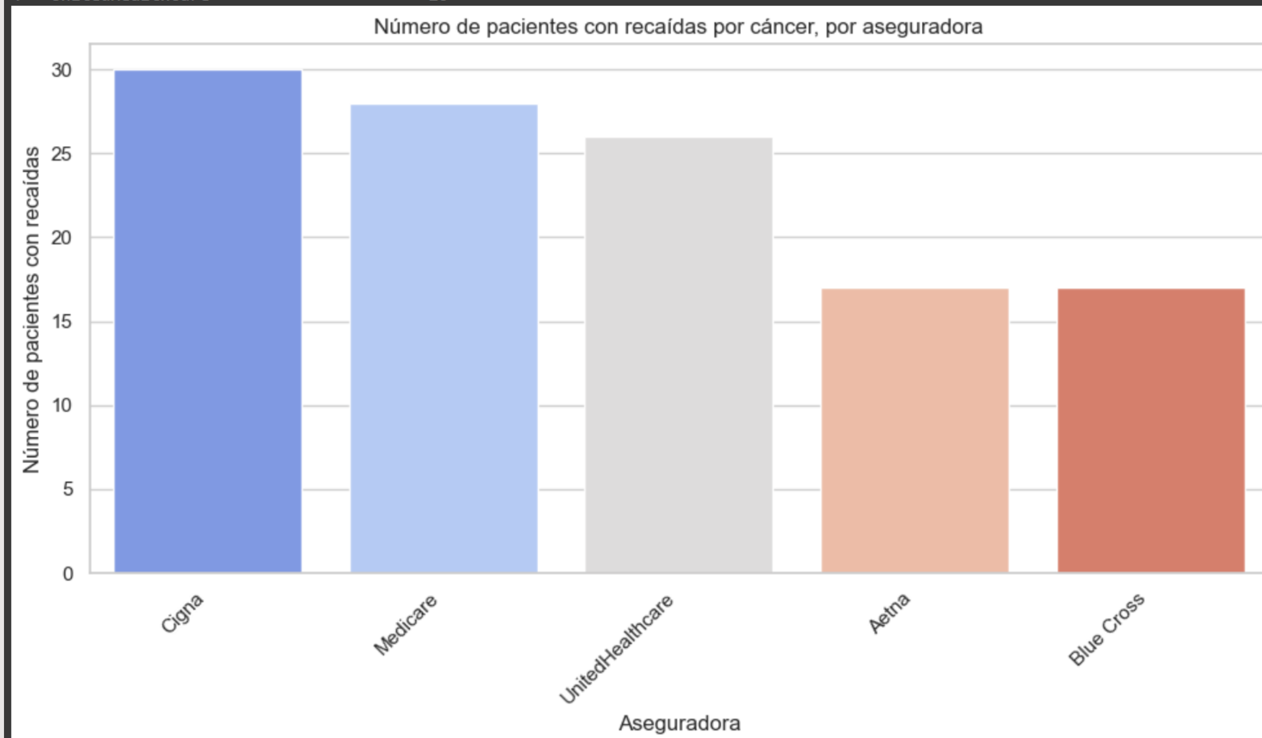


Hallazgos Relevantes

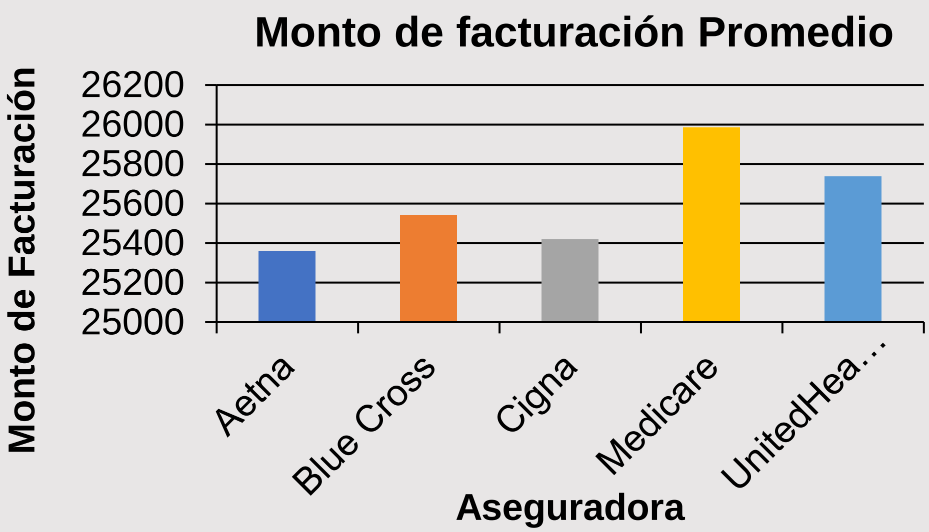


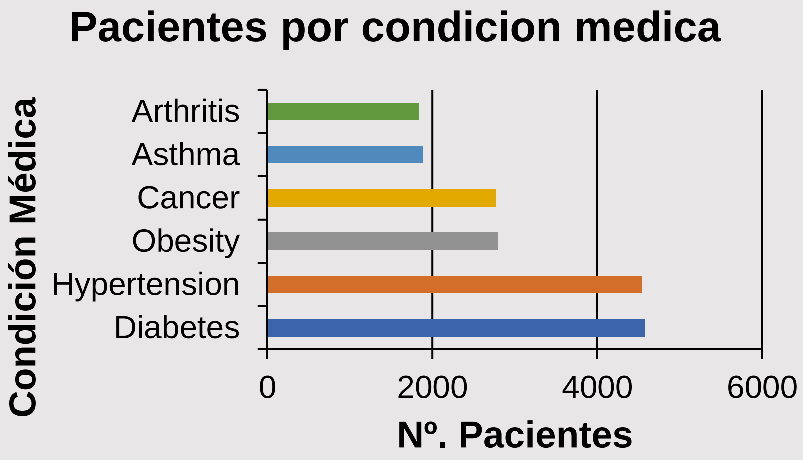
Hallazgos Relevantes

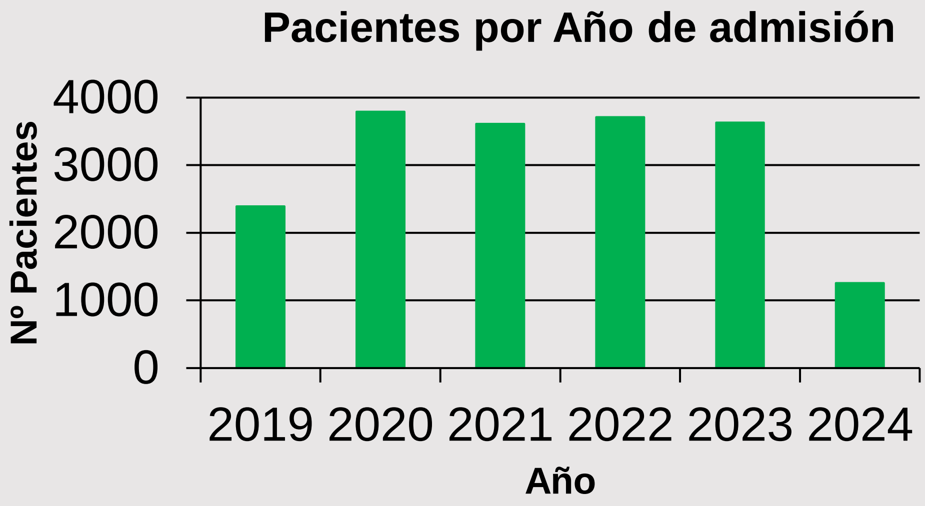
```
Número de pacientes con recaídas por cáncer: 117
  Insurance Provider  Pacientes con recaídas
0              Aetna                      17
1         Blue Cross                      17
2              Cigna                      30
3           Medicare                      28
4   UnitedHealthcare                      26
```



Hallazgos Relevantes







Hallazgos Relevantes

- Monto facturación por aseguradora
- Pacientes por año de admisión
- Pacientes por condición médica
- Estancia promedio con edad o género
- Admisiones de emergencia predominantes
- Factores que influyen en facturación
  - Aseguradoras ↔ Mayor facturación
  - Hospitales y doctores con altos montos

☑ Dataset Balanceado

```python
import pandas as pd
# Cargar el archivo original
df = pd.read_csv('mi_data.csv')
```

```
Distribución original:
Test Results  Gender  Admission Type
Normal        Female  Emergency         6673
              Male    Emergency         5873
              Female  Urgent            5044
              Male    Urgent            4553
Abnormal      Female  Emergency         3977
              Male    Emergency         3563
              Female  Urgent            3056
Normal        Female  Elective          2883
Abnormal      Male    Urgent            2754
Normal        Male    Elective          2667
Inconclusive  Female  Emergency         2640
              Male    Emergency         2288
              Female  Urgent            2078
              Male    Urgent            1887
Abnormal      Female  Elective          1739
              Male    Elective          1598
Inconclusive  Female  Elective          1097
              Male    Elective          1022
Name: count, dtype: int64
```

```
Distribución balanceada:
Test Results  Gender  Admission Type
Abnormal      Female  Elective          1022
                      Emergency         1022
                      Urgent            1022
              Male    Elective          1022
                      Emergency         1022
                      Urgent            1022
Inconclusive  Female  Elective          1022
                      Emergency         1022
                      Urgent            1022
              Male    Elective          1022
                      Emergency         1022
                      Urgent            1022
Normal        Female  Elective          1022
                      Emergency         1022
                      Urgent            1022
              Male    Elective          1022
                      Emergency         1022
                      Urgent            1022
Name: count, dtype: int64
```

☑ Dataset Balanceado

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18396 entries, 0 to 18395
Data columns (total 24 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Name                18396 non-null  object 
 1   Age                 18396 non-null  int64  
 2   Gender              18396 non-null  object 
 3   Blood Type          18396 non-null  object 
 4   Medical Condition   18396 non-null  object 
 5   Date of Admission   18396 non-null  object 
 6   Doctor              18396 non-null  object 
 7   Hospital            18396 non-null  object 
 8   Insurance Provider  18396 non-null  object 
 9   Billing Amount      18396 non-null  float64
 10  Room Number         18396 non-null  int64  
 11  Admission Type      18396 non-null  object 
 12  Discharge Date      18396 non-null  object 
 13  Medication          18396 non-null  object 
 14  Test Results        18396 non-null  object 
 15  Days of Stay        18396 non-null  int64  
 16  AD_Mes              18396 non-null  int64  
 17  AD_DiaSemana        18396 non-null  object 
 18  AD_Trimestre        18396 non-null  int64  
 19  AD_Anio             18396 non-null  int64  
...
 22  DD_Trimestre        18396 non-null  int64  
 23  DD_Anio             18396 non-null  int64  
dtypes: float64(1), int64(9), object(14)
memory usage: 3.4+ MB
```

✨ Agenda

1. Objetivo
2. Dataset Original
3. Metodología: Limpieza y Transformación
4. Análisis Exploratorio de Datos
5. Identificación de Patrones
6. Hallazgos Relevantes
7. Modelado Predictivo
   7.1 Modelos supervisados (numéricas)
   7.2 Modelos no supervisados (categóricas)
8. Conclusiones

Modelos Supervisados

Modelo	Descripción breve	Tarea principal
Regresión Lineal	Modelo simple para predecir valores continuos	Regresión
Regresión Logística	Clasificación binaria (sí/no, 0/1)	Clasificación
Árboles de Decisión	Divide los datos según reglas en forma de árbol	Clasificación / Regresión
Random Forest	Conjunto de árboles para mejorar precisión y reducir sobreajuste	Clasificación / Regresión
Gradient Boosting (XGBoost, etc)	Modelo por ensamble que mejora errores gradualmente	Clasificación / Regresión
Redes Neuronales	Modelos flexibles y potentes para tareas complejas	Clasificación / Regresión

Variables Numéricas

- Regresión Lineal

```
X = df[['Age','Days of Stay','Admission Type','Gender', 'Insurance Provider', 'Medication' ]]
X = pd.get_dummies(X)  # Codificación one-hot para variables categóricas
y = df['Billing Amount']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

# Evaluate the model
print("Mean Squared Error:", mean_squared_error(y_test, y_pred))
print("R2 Score:", r2_score(y_test, y_pred))
```

```
Mean Squared Error: 194410751.12432227
R2 Score: -0.0008445455789141132
```

Modelos Supervisados

- Variables numéricas:
- Random Forest

```python
X = df[['Age','Days of Stay','Admission Type','Gender', 'Insurance Provider', 'Medication' ]]
X = pd.get_dummies(X)  # Codificación one-hot para variables categóricas
y = df['Billing Amount']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Crear y entrenar el modelo Random Forest
rf_rg = RandomForestRegressor(n_estimators=150, max_depth=26, random_state=42)
rf_rg.fit(X_train, y_train)

# Predecir en el conjunto de prueba
y_pred = rf_rg.predict(X_test)
# Mètriques de regressió
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

# Mostrar resultats
print("MAE (Error absolut mitjà):", round(mae, 2))
print("MSE (Error quadràtic mitjà):", round(mse, 2))
print("RMSE (Arrel quadrada del MSE):", round(rmse, 2))
print("R² (Coeficient de determinació):", round(r2, 2))
```

```
MAE (Error absolut mitjà): 12903.07
MSE (Error quadràtic mitjà): 234122724.
RMSE (Arrel quadrada del MSE): 15301.07
R² (Coeficient de determinació): -0.21
```

Modelos Supervisados

- Variables numéricas:
- Red Neuronal

```
df=pd.read_csv('mi_data_balanceado.csv')
X = df[['Age','Days of Stay','Admission Type','Gender', 'Insurance Provider', 'Medication' ]]
X = pd.get_dummies(X)  # Codificación one-hot para variables categóricas
y = df['Billing Amount']
```

```
X_train , X_test , y_train , y_test = train_test_split(X,y,test_size = 0.2,random_state=777)
scaler = StandardScaler()
scaler.fit(X_train)
X_train , X_test = scaler.transform(X_train),scaler.transform(X_test)
rna = MLPRegressor(
    solver='adam',
    alpha=0.001,
    hidden_layer_sizes=(1000 , )
   )
modelo = rna.fit(X_train , y_train)
pred_rna = modelo.predict(X_test)
print(f"R2: {r2_score(y_test , pred_rna)}")
print(f"MAE: {mean_absolute_error(y_test , pred_rna)}")
for i , (w,b) in enumerate(zip(rna.coefs_ , rna.intercepts_)):
    print(f"Capa{i} -> capa{i+1}")
    print(f"Peso : {w.shape}")
    print(f"Bias : {b.shape}")
```

```
R2: -0.005093339458617452
MAE: 12211.184434815937
Capa0 -> capa1
Peso : (15, 1000)
Bias : (1000,)
Capa1 -> capa2
```

Modelos Supervisados

- Variables numéricas:
- XGBoost

```python
X = df[['Age', 'Days of Stay','Admission Type','Gender', 'Insurance Provider', 'Medication']]  # Eliminar la columna objectiu de les predictors
X = pd.get_dummies(X)  # Codificación one-hot para variables categóricas
y = df['Billing Amount']        # # Definir la variable objectivo;Corregido a una serie (no lista)
```

```python
X_train, X_test, y_train, y_test = train_test_split(X , y, random_state=777)
pipeline = Pipeline([
    ('sclaer' , StandardScaler()),
    ('xgb' , XGBRegressor( 'reg:squarederror' , random_state = 444))
])
param_grid ={
    'xgb__n_estimators':[100,200],
    'xgb__max_depth':[3,5],
    'xgb__learning_rate':[0.01 , 0.1],
    'xgb__subsample':[0.8,1],
    'xgb__colsample_bytree':[0.8,1]
}
cv = KFold(n_splits=5 , shuffle=True , random_state=444)
grid = GridSearchCV(pipeline , param_grid , cv = cv , scoring='neg_mean_squared_error')
grid.fit(X_train,y_train)
print(f"Mejor parametro :{grid.best_params_}")
y_pred = grid.predict(X_test)
print(f"MAE : {mean_absolute_error(y_test , y_pred)}")
print(f"MSE : {mean_squared_error(y_test , y_pred)}")
print(f"RMSE : {root_mean_squared_error(y_test , y_pred)}")
print(f"R2 : {r2_score(y_test , y_pred)}")
```

```
Mejor parametro :{'xgb__colsample_bytree': 0.8, 'xgb__learning_rate': 0.01, 'xgb__max_depth': 3, 'xgb__n_estimators': 100, 'xgb__subsample': 0.8}
MAE : 12227.222538237505
MSE : 199517821.1795032
RMSE : 14125.077740653436
R2 : -0.0013820993887629918
```

Resultados Predictivos

- Los modelos mostraron bajo rendimiento (R^2 negativo), indicando baja capacidad explicativa.

Método	Metricas	Metricas
Lineal Regresion	Mean Squared Error: 194410751.12432227	R2 Score: -0.0008445455789141132
RandonForest	MAE (Error absolut mitjà): 12903.07	MSE (Error quadràtic mitjà): 234122724.8
RandonForest	RMSE (Arrel quadrada del MSE): 15301.07	R^2 (Coeficient de determinació): -0.21
NeuralNetwork	MAE: 12211.184434815937	R2: -0.005093339458617452
XGBoost	MAE : 12227.222538237505	MSE : 199517821.1795032
XGBoost	RMSE : 14125.077740653436	R2 : -0.0013820993887629918

Ensemble de Modelos

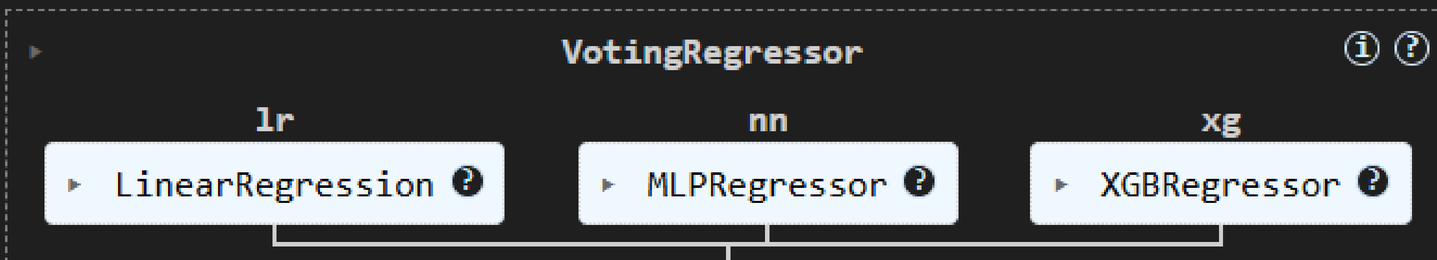
- Combinación de modelos (Lineal, Red Neuronal, XGBoost) para mejorar la predicción.

```python
df=pd.read_csv('mi_data_balanceado.csv')
X = df[['Age','Days of Stay','Admission Type','Gender', 'Insurance Provider', 'Medication' ]]
X = pd.get_dummies(X)  # Codificación one-hot para variables categóricas
y = df['Billing Amount']
X_train, X_test, y_train, y_test = train_test_split(X , y , test_size=0.2 , random_state=777)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test) #X_test_scaled = scaler.transform(X_test)        # Usa esos valores e
                                         #X_train_scaled = scaler.fit_transform(X_train) # Aprende media y std
kfold = KFold(n_splits=5 , shuffle=True , random_state=777)
lr = LinearRegression()
nn = MLPRegressor(max_iter=1000 , random_state=44)
xg = XGBRegressor(random_state=999)
param_grid_lr = {
    'fit_intercept':[True , False]
}
param_grid_xg ={
    'xgb__n_estimators':[100,200],
    'xgb__max_depth':[3,5],
    'xgb__learning_rate':[0.01 , 0.1],
    'xgb__subsample':[0.8,1],
    'xgb__colsample_bytree':[0.8,1]
}
param_grid_nn = {
    'hidden_layer_sizes': [(50,), (100,), (100, 50)],
    'learning_rate_init': [0.001, 0.01],
    'alpha': [0.0001, 0.001]
}
grid_lr = GridSearchCV(lr , param_grid_lr , cv = kfold , scoring='neg_mean_squared_error' , n_jobs=-1)
grid_nn = GridSearchCV(nn , param_grid_nn , cv = kfold , scoring='neg_mean_squared_error' , n_jobs=-1)
grid_xg = GridSearchCV(xg , param_grid_xg , cv = kfold , scoring='neg_mean_squared_error' , n_jobs=-1)
grid_lr.fit(X_train,y_train)
grid_nn.fit(X_train_scaled,y_train)
grid_xg.fit(X_train,y_train)
best_lr = grid_lr.best_estimator_
best_nn = grid_nn.best_estimator_
best_xg = grid_xg.best_estimator_
```

```
Lr (optimizada) (Validacion Cruzada en Entrenamiento) :
MSE (mean ± std): 200926905.5101 ± 1470305.6707
R² (mean ± std): -0.0016 ± 0.0022
R² ajustado (mean ± std): -0.0026 ± 0.0022

Lr (optimizada) (Conjunto prueba) :
MSE: 199073972.8847
R²: -0.0033
R² ajustado: -0.0074

NN (optimizada) (Validacion Cruzada en Entrenamiento) :
MSE (mean ± std): 201838828.3008 ± 1815215.4988
R² (mean ± std): -0.0062 ± 0.0040
R² ajustado (mean ± std): -0.0072 ± 0.0040

NN (optimizada) (Conjunto prueba) :
MSE: 199285083.9932
R²: -0.0044
R² ajustado: -0.0085

XG (optimizada) (Validacion Cruzada en Entrenamiento) :
MSE (mean ± std): 227276177.8781 ± 3129293.1101
R² (mean ± std): -0.1330 ± 0.0122
R² ajustado (mean ± std): -0.1341 ± 0.0122
...
Ensamblado (RL + NN + XG) (Conjunto prueba) :
MSE: 201896202.4164
R²: -0.0176
R² ajustado: -0.0217
```

Ensemble de Modelos

```
esemble.fit(X_train_scaled , y_train)
```




```
y_pred = esemble.predict(X_test_scaled[0].reshape(1,-1))
print(f"y_pred = {y_pred[0]:.4f}")
print(f"y_test = {y_test.iloc[0]:.4f}") # Accede por posición, no por etiqueta

```

```
y_pred = 23373.8224
y_test = 13692.4049
```

Ensemble de Modelos

```
# Diccionario para guardar métricas
resultados = {}
# Lista de modelos y nombres
modelos = [
    ("Regresión Lineal", best_lr, X_test),
    ("Red Neuronal", best_nn, X_test_scaled),
    ("XGBoost", best_xg, X_test),
    ("Ensamblado", esemble, X_test_scaled)
]
# Calcular métricas de cada modelo
for nombre, modelo, X_test_eval in modelos:
    y_pred = modelo.predict(X_test_eval)
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    n = len(y_test)
    p = X_test_eval.shape[1]
    r2_adj = 1 - (1 - r2) * (n - 1) / (n - p - 1)

    resultados[nombre] = {
        "MSE": mse,
        "R²": r2,
        "R² Ajustado": r2_adj
    }
# Convertir a DataFrame y ordenar por R²
df_resultados = pd.DataFrame(resultados).T
df_resultados = df_resultados.sort_values(by="R²", ascending=False)
print("\n Comparación final de modelos:")
print(df_resultados.round(4))
```

LIGERA MEJORA, PERO AÚN MUY LIMITADA

Comparación final de modelos:

	MSE	R^2	R^2 Ajustado
Regresión Lineal	1.990740e+08	-0.0033	-0.0074
Red Neuronal	1.992851e+08	-0.0044	-0.0085
Ensamblado	2.018962e+08	-0.0176	-0.0217
XGBoost	2.201454e+08	-0.1095	-0.1141

Agenda

1. Objetivo
2. Dataset Original
3. Metodología: Limpieza y Transformación
4. Análisis Exploratorio de Datos
5. Identificación de Patrones
6. Hallazgos Relevantes
7. Modelado Predictivo
   - 7.1 Modelos supervisados (numéricas)
   - 7.2 Modelos no supervisados (categóricas)

8 Conclusiones

🔎 Modelos No Supervisados

Modelo	**Descripción breve**	**Tarea principal**
K-Means/K-Modes	Agrupa datos en k grupos según similitud	Clustering
PCA (Análisis de Componentes Principales)	Reduce dimensiones conservando varianza	Reducción de Dimensión

Clustering K-Modes

```python
cat=['Admission Type','Gender', 'Insurance Provider', 'Medication' , 'Blood Type']
comb = set()
for i  in cat:
    for j in cat:
        if i < j:
            comb.add((i,j))

comb
```

```
{('Admission Type', 'Blood Type'),
 ('Admission Type', 'Gender'),
 ('Admission Type', 'Insurance Provider'),
 ('Admission Type', 'Medication'),
 ('Blood Type', 'Gender'),
 ('Blood Type', 'Insurance Provider'),
 ('Blood Type', 'Medication'),
 ('Gender', 'Insurance Provider'),
 ('Gender', 'Medication'),
 ('Insurance Provider', 'Medication')}
```

Clustering K-Modes

Variables Estudiadas

ADMISSION TYPE ⟷ GENDER

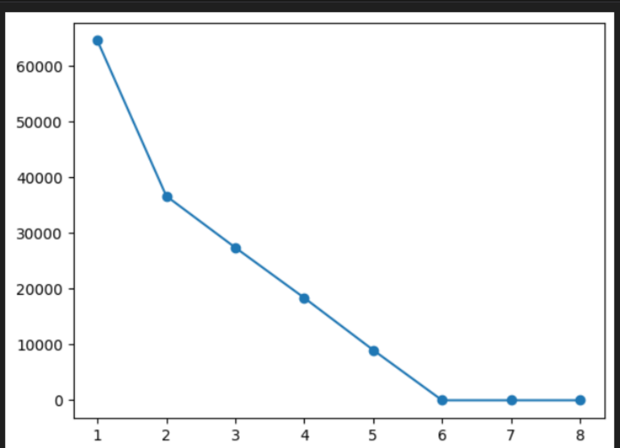
ADMISSION TYPE ⟷ INSURANCE PROVIDER

INSURANCE PROVIDER ⟷ GENDER

MEDICATION ⟷ GENDER

Clustering K-Modes

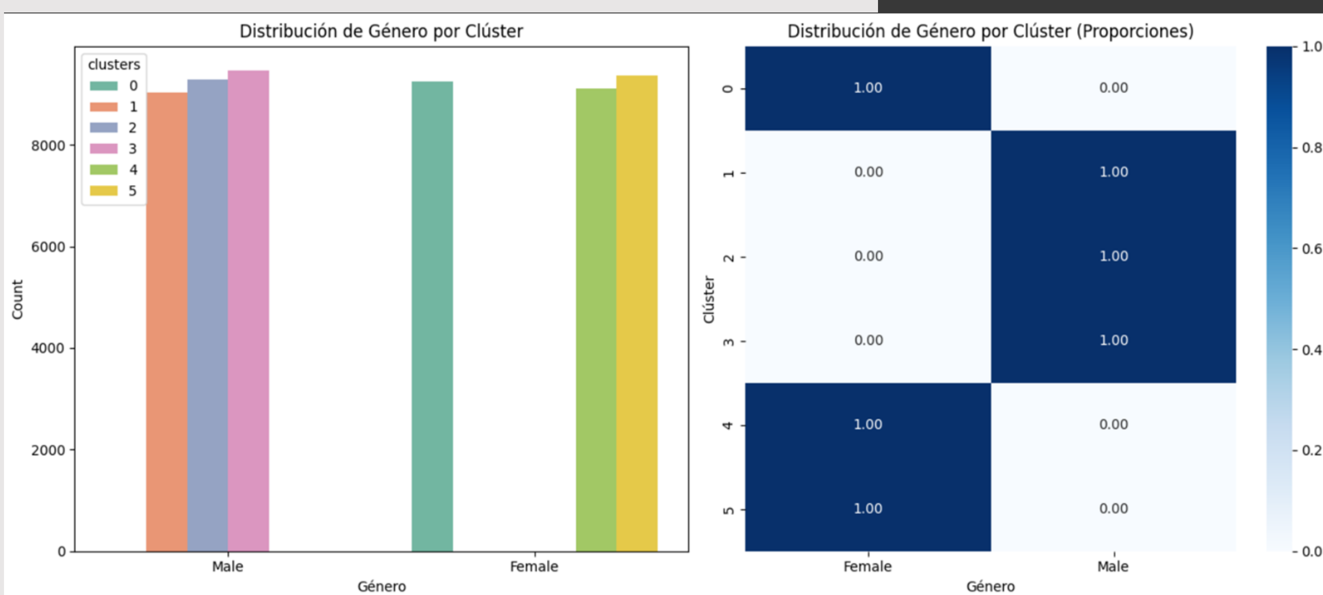
```
costes = []
df11 = df[['Admission Type' , 'Gender']]
num_clust = range(1,9)
for k in num_clust:
    km = KModes(n_clusters= k , init='Huang' , n_init=5 , verbose=0)
    km.fit(df11)
    costes.append(km.cost_)

plt.plot(num_clust , costes , marker='o')
plt.show()
```




```
#Probamos 6 clusters
num = 6
km = KModes(n_clusters= num , init='Huang' , n_init=5 , verbose = 1)
clusters = km.fit_predict(df11)

df11['clusters'] = clusters
```

Clustering K-Modes

```
df11.groupby('clusters').value_counts()

clusters  Admission Type  Gender
0         Urgent          Male      9468
1         Elective        Male      9281
2         Elective        Female    9374
3         Emergency       Female    9244
4         Urgent          Female    9108
5         Emergency       Male      9025
Name: count, dtype: int64
```

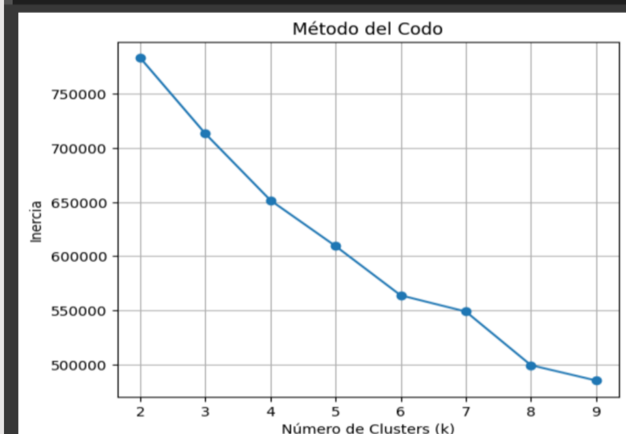


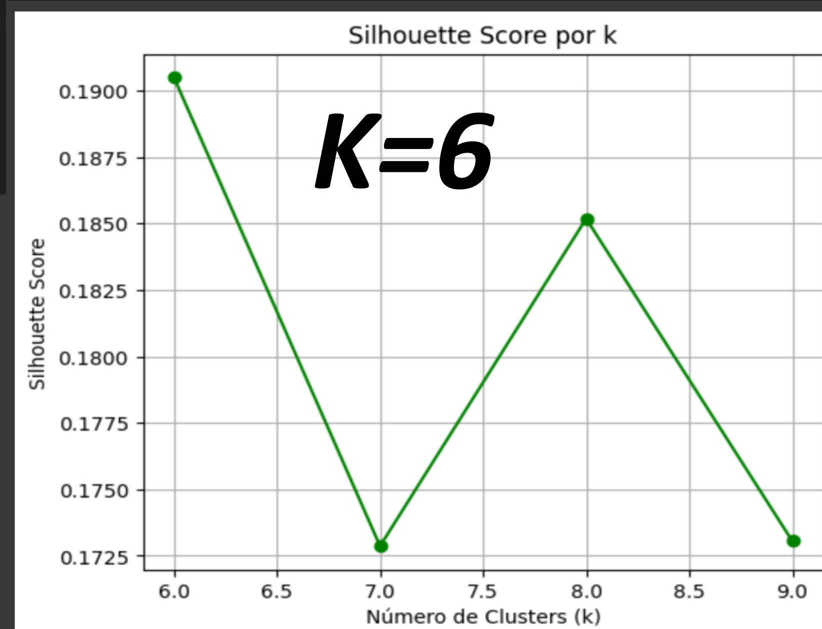
📌 PCA aplicado a los clusters generados por K-Modes

- Separación visual moderada entre clusters.

```python
# 1. Filtrado y preparación
df["Gender"] = df["Gender"].map({"Male": 0, "Female": 1})

# 2. Variables predictoras (sin la columna objetivo)
X = df[['Age', 'Days of Stay', 'Admission Type', 'Test Results', 'Insurance Provider', 'Medication']]
X = pd.get_dummies(X)

# 3. Escalar variables
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```python
# 4. Método del codo
inertias = []
K_range = range(2, 10)

for k in K_range:
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
    kmeans.fit(X_scaled)
    inertias.append(kmeans.inertia_)

plt.figure(figsize=(6, 5))
plt.plot(K_range, inertias, marker='o')
plt.title('Método del Codo')
plt.xlabel('Número de Clusters (k)')
plt.ylabel('Inercia')
plt.grid(True)
plt.show()
```

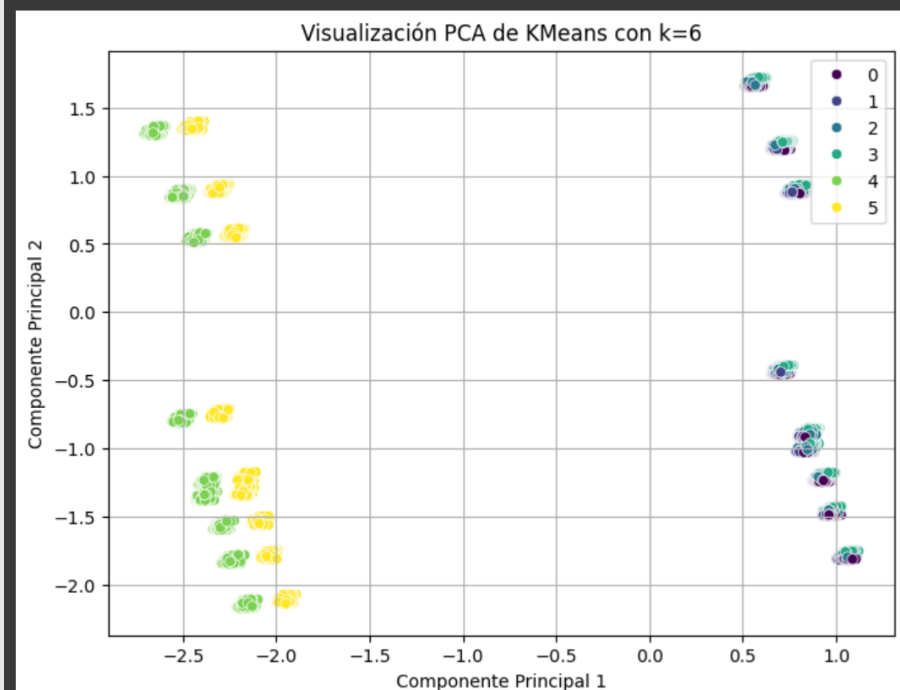




PCA aplicado a los clusters generados por K-Modes

- Creación de los clusters.

```
# 8. Visualización con PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

plt.figure(figsize=(8,6))
sns.scatterplot(x=X_pca[:,0], y=X_pca[:,1], hue=clusters, palette='viridis')
plt.title(f'Visualización PCA de KMeans con k={k}')
plt.xlabel('Componente Principal 1')
plt.ylabel('Componente Principal 2')
plt.grid(True)
plt.show()
```




```
df['Cluster'] = clusters
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 55392 entries, 0 to 55391
Data columns (total 25 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Name                55392 non-null  object
 1   Age                 55392 non-null  int64
 2   Gender              55392 non-null  int64
 3   Blood Type          55392 non-null  object
 4   Medical Condition   55392 non-null  object
 5   Date of Admission   55392 non-null  object
 6   Doctor              55392 non-null  object
 7   Hospital            55392 non-null  object
 8   Insurance Provider  55392 non-null  object
 9   Billing Amount      55392 non-null  float64
 10  Room Number         55392 non-null  int64
 11  Admission Type      55392 non-null  object
 12  Discharge Date      55392 non-null  object
 13  Medication          55392 non-null  object
 14  Test Results        55392 non-null  object
 15  Days of Stay        55392 non-null  int64
 16  AD_Mes              55392 non-null  int64
 17  AD_DiaSemana        55392 non-null  object
 18  AD_Trimestre        55392 non-null  int64
 19  AD_Anio             55392 non-null  int64
 20  DD_Mes              55392 non-null  int64
 21  DD_DiaSemana        55392 non-null  object
 22  DD_Trimestre        55392 non-null  int64
 23  DD_Anio             55392 non-null  int64
 24  Cluster             55392 non-null  int32
dtypes: float64(1), int32(1), int64(10), object(13)
memory usage: 10.4+ MB
```

PCA aplicado a los clusters generados por K-Modes

```
#Agrupar por Cluster y revisar medias de variables:
df.groupby('Cluster')[['Age', 'Days of Stay']].mean()
```

Cluster	Age	Days of Stay
0	39.544894	15.525323
1	39.791227	15.607622
2	39.738270	15.394805
3	39.582494	15.453357
4	39.506790	15.549952
5	39.341255	15.697569

```
#Ver proporciones de género por clúster:
pd.crosstab(df['Gender'], df['Cluster'], normalize='columns')
```

Gender \ Cluster	0	1	2	3	4	5
0	0.464239	0.478907	0.474178	0.47458	0.480238	0.461538
1	0.535761	0.521093	0.525822	0.52542	0.519762	0.538462

```
#Ver proporciones de tipo de admisión por clúster:
pd.crosstab(df['Admission Type'], df['Cluster'], normalize='columns')
```

Admission Type \ Cluster	0	1	2	3	4	5
Elective	0.202823	0.197148	0.199673	0.195564	0.197987	0.197569
Emergency	0.448820	0.453619	0.446086	0.451679	0.454413	0.464804
Urgent	0.348357	0.349233	0.354241	0.352758	0.347599	0.337627

Las proporciones son bastante equilibradas en todos los clusters

Los grupos de K-Modes no son útiles para entrenamiento, debido a su poca segmentación

Discusión

- Metodología robusta
    - utilidad educativa
- Análisis exhaustivo de relaciones entre variables.
- Dataset sintético
    - no refleja complejidad real
- Limitaciones estadísticas (Chi2 y regresión)
    - Variables sin relación con variable objetivo

Conclusiones

- Marco riguroso - hallazgos poco relevantes:
  - Correlaciones inexistentes
  - Grupo sanguíneo y Aseguradora relacionados con el objetivo
  - Grupo sanguíneo relacionado con el género (no es real)
  - Monto facturación/días de estancia/edad sin relación con el genero
  - Doctor y Hospital (relación significativa y alta cardinalidad)
  - Sin relación con la edad
  - Mujeres mayor núm. estancias (≠condición médica)
  - Estancia promedio similar (ambos géneros)
  - Diabetes y Hipertensión (mayor facturación)
  - Cigna mayor frecuencia de recaídas en cáncer, seguida de Medicare (estatal >65)
  - Blue Cross la que menos recaídas
  - Periodo con mayor núm ingresos 2020-2023
- Modelos de Predicción limitada sin datos reales

📌 Recomendaciones

- Aplicaciones futuras:
  - Aplicar a datasets reales
  - Incluir más variables clínicas
  - Crear dashboard predictivo hospitalario

Q&A

¡Gracias por vuestra atención!