Proyecto final de IA – Seguro Médico

Integrantes:

Diego Ramos

Ricardo Lara

Santiago Giraldo

Daniel Pérez

Dataset:https://www.kaggle.com/datasets/harishkumardatalab/medical-insurance-price-prediction?resource=download

App web: https://diego-ramos.github.io/medical_api/medical_prediction.html

0. Introducción: el ciclo de vida en proyectos de Machine Learning

Los proyectos de *machine learning* no se limitan a construir un modelo predictivo: son **ecosistemas iterativos** que integran fases interdependientes, desde la **comprensión del problema** hasta la **supervisión y mejora continua**.

Cada etapa —análisis, diseño, entrenamiento, evaluación y mantenimiento— contribuye de forma crítica a la calidad y sostenibilidad del resultado.
A diferencia de los proyectos tradicionales de software, este tipo de iniciativas requieren **retroalimentación constante**, donde los datos evolucionan, los modelos se ajustan y las soluciones se optimizan de manera continua.

Comprender este ciclo de vida permite a los equipos técnicos y a las organizaciones **reducir riesgos, optimizar recursos y aumentar la robustez de sus modelos**, asegurando que las soluciones entregadas respondan fielmente a los objetivos iniciales y se mantengan vigentes ante la evolución de los datos.

Este proyecto es un ejemplo de cómo ese ciclo cobra vida al abordar un problema real de predicción en el ámbito de la salud.

Objetivo

- Elegir qué factores determinan realmente el costo de un seguro médico y cómo puede la IA ayudar a predecirlo con precisión y equidad
- Predecir el costo de un seguro médico (charges) usando variables demográficas y de estilo de vida del dataset de Kaggle.

1. Contexto y punto de partida

Los seguros médicos reflejan una realidad compleja: cada persona tiene un riesgo diferente, pero las tarifas muchas veces se calculan con fórmulas rígidas o poco transparentes.
Con la expansión de la analítica y la IA, surge la oportunidad de construir modelos que **predigan costos de manera justa, explicable y basada en datos reales**.

Así nace la pregunta central de este proyecto:

> *¿Qué factores determinan realmente el costo de un seguro médico y cómo puede la IA ayudar a predecirlo con precisión y equidad?*

El dataset de Kaggle "Medical Insurance Price Prediction" ofrecía el punto de partida ideal:
- Edad (`age`)
- Índice de masa corporal (`bmi`)
- Número de hijos (`children`)
- Sexo (`sex`)
- Hábito de fumar (`smoker`)
- Región (`region`)
- Costo del seguro (`charges`)

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder, StandardScaler, LabelEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LinearRegression, Ridge
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import joblib
from scipy import stats
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score
from sklearn.metrics import f1_score
import matplotlib.pyplot as plt
```

Exploracion de datos

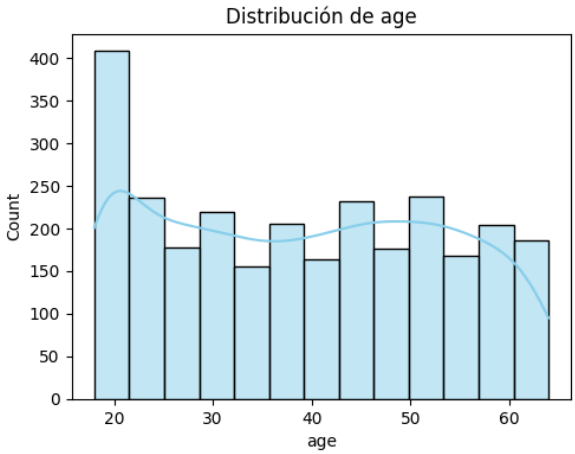
```
df = pd.read_csv('sample_data/Medical_insurance.csv')
print('Tipos de datos:')
print(df.dtypes)
print('\nValores nulos por columna:')
print(df.isnull().sum())
print('\nDuplicados:', df.duplicated().sum())

# Estadísticas básicas
display(df.describe(include='all'))
```

```
Tipos de datos:
age           int64
sex          object
bmi         float64
children      int64
smoker       object
region       object
charges     float64
dtype: object

Valores nulos por columna:
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64

Duplicados: 1435
```

	age	sex	bmi	children	smoker	region	charges
count	2772.000000	2772	2772.000000	2772.000000	2772	2772	2772.000000
unique	NaN	2	NaN	NaN	2	4	NaN
top	NaN	male	NaN	NaN	no	southeast	NaN
freq	NaN	1406	NaN	NaN	2208	766	NaN
mean	39.109668	NaN	30.701349	1.101732	NaN	NaN	13261.369959
std	14.081459	NaN	6.129449	1.214806	NaN	NaN	12151.768945
min	18.000000	NaN	15.960000	0.000000	NaN	NaN	1121.873900
25%	26.000000	NaN	26.220000	0.000000	NaN	NaN	4687.797000
50%	39.000000	NaN	30.447500	1.000000	NaN	NaN	9333.014350
75%	51.000000	NaN	34.770000	2.000000	NaN	NaN	16577.779500
max	64.000000	NaN	53.130000	5.000000	NaN	NaN	63770.428010

Visualización de distribuciones y pruebas de normalidad

```
# Estos gráficos permiten ver si los datos siguen una distribución normal:
# en el Q-Q plot, los puntos deben alinearse con la diagonal.

num_cols = df.select_dtypes(include=[np.number]).columns.tolist()
cat_cols = df.select_dtypes(exclude=[np.number]).columns.tolist()

for c in num_cols:
    plt.figure(figsize=(12,4))

    plt.subplot(1,2,1)
    sns.histplot(df[c], kde=True, color='skyblue')
    plt.title(f'Distribución de {c}')

    plt.subplot(1,2,2)
    stats.probplot(df[c], dist="norm", plot=plt)
    plt.title(f'Gráfico Q-Q de {c}')

    plt.show()
```


Distribución de age
Count
age

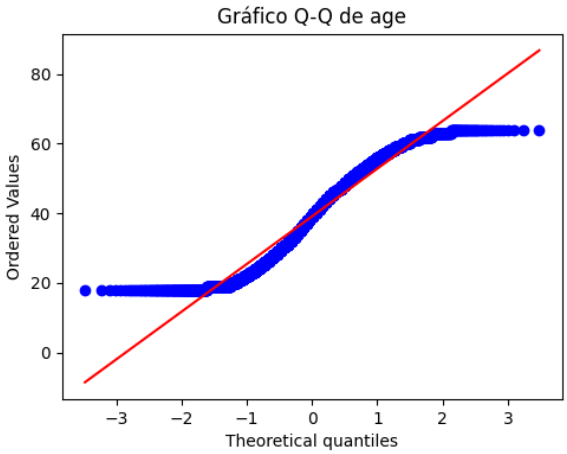
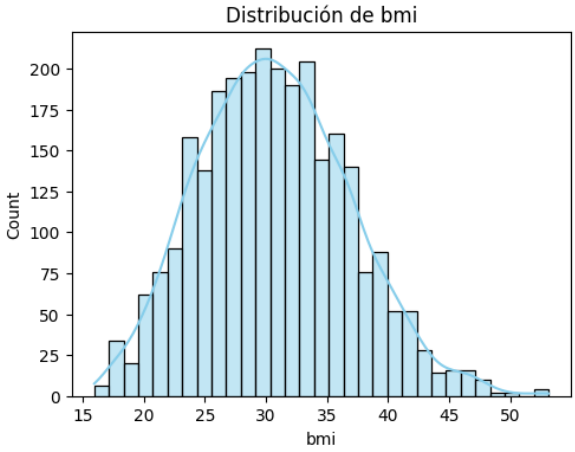


Gráfico Q-Q de age
Ordered Values
Theoretical quantiles


Distribución de bmi
Count
bmi

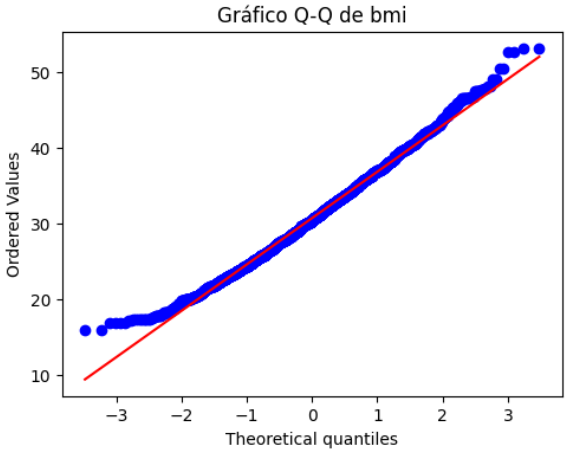
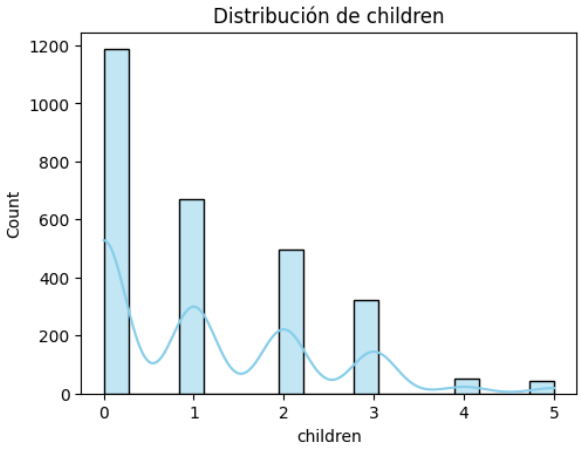


Gráfico Q-Q de bmi
Ordered Values
Theoretical quantiles


Distribución de children
Count
children

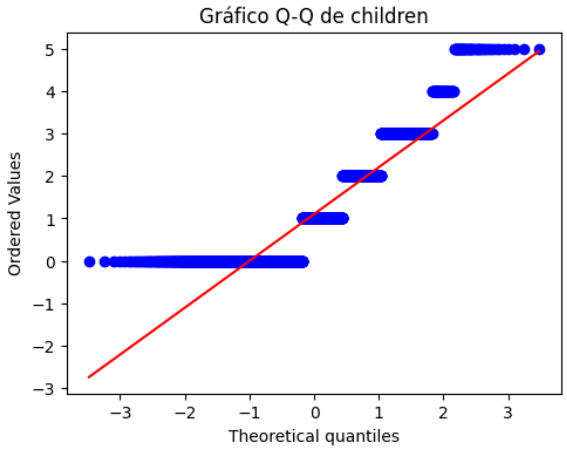
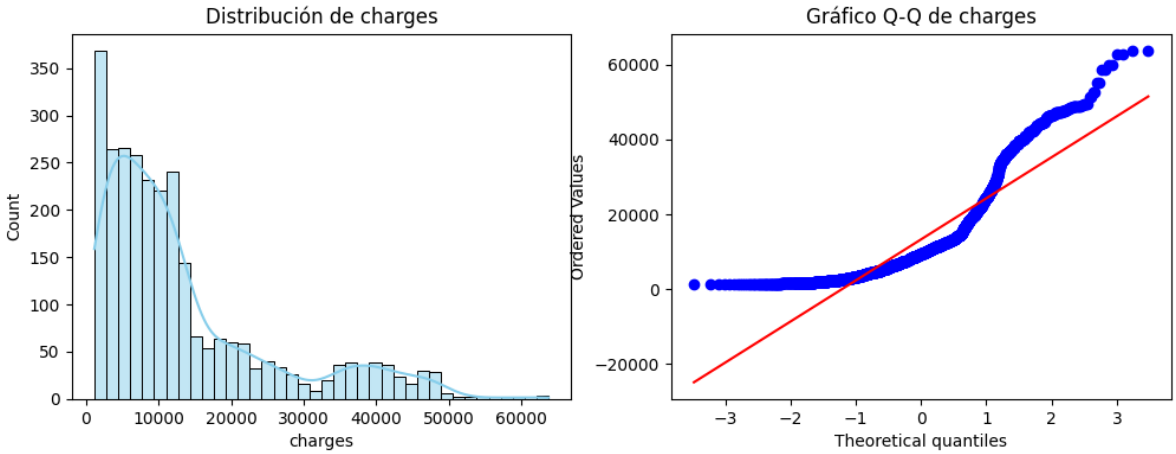
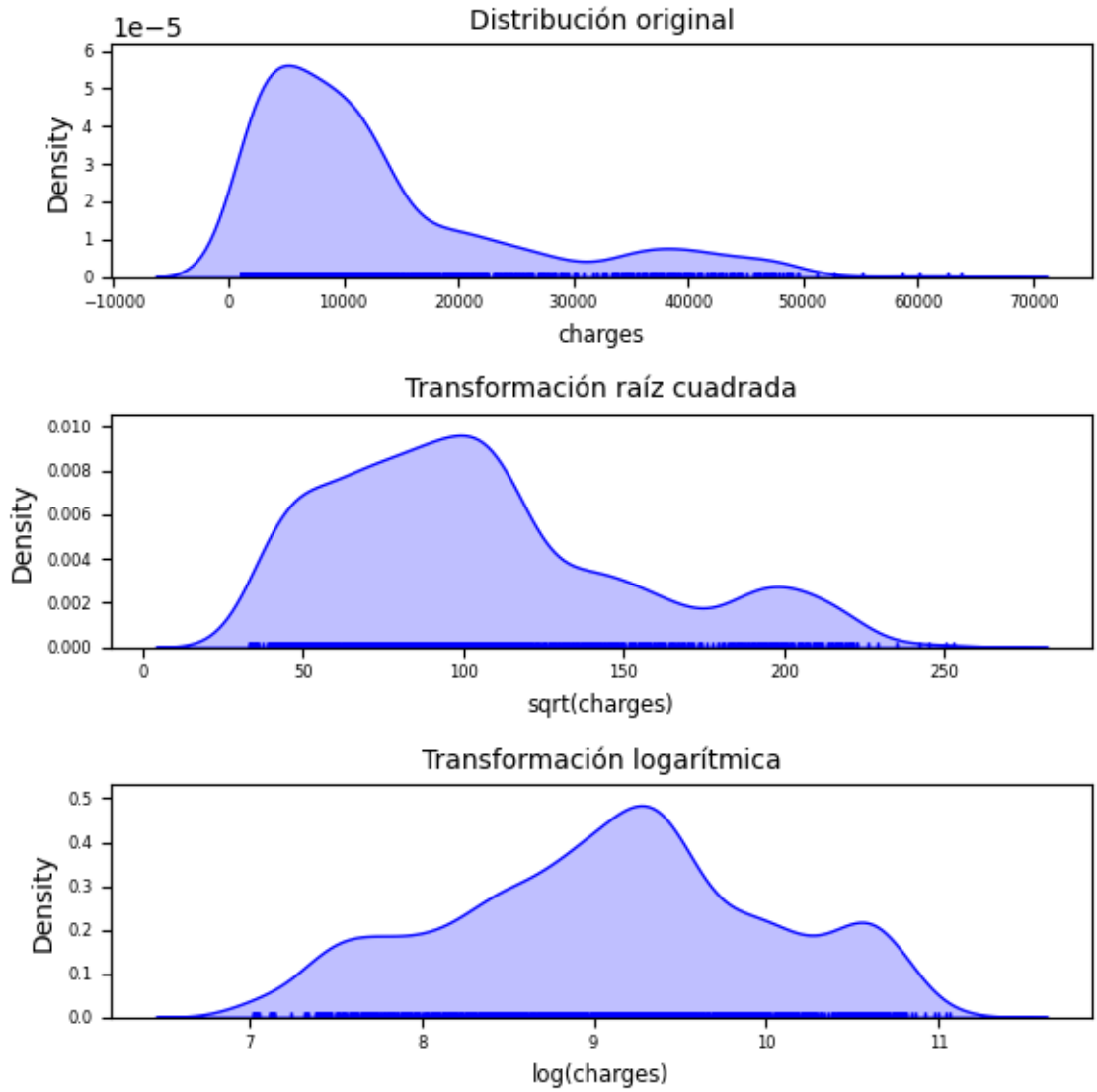


Gráfico Q-Q de children
Ordered Values
Theoretical quantiles




```
fig, axes = plt.subplots(nrows=3, ncols=1, figsize=(6, 6))
sns.kdeplot(
    df.charges,
    fill    = True,
    color   = "blue",
    ax      = axes[0]
)
sns.rugplot(
    df.charges,
    color   = "blue",
    ax      = axes[0]
)
axes[0].set_title("Distribución original", fontsize = 'medium')
axes[0].set_xlabel('charges', fontsize='small')
axes[0].tick_params(labelsize = 6)

sns.kdeplot(
    np.sqrt(df.charges),
    fill    = True,
    color   = "blue",
    ax      = axes[1]
)
sns.rugplot(
    np.sqrt(df.charges),
    color   = "blue",
    ax      = axes[1]
)
axes[1].set_title("Transformación raíz cuadrada", fontsize = 'medium')
axes[1].set_xlabel('sqrt(charges)', fontsize='small')
```

```
axes[1].tick_params(labelsize = 6)

sns.kdeplot(
    np.log(df.charges),
    fill    = True,
    color   = "blue",
    ax      = axes[2]
)
sns.rugplot(
    np.log(df.charges),
    color   = "blue",
    ax      = axes[2]
)
axes[2].set_title("Transformación logarítmica", fontsize = 'medium')
axes[2].set_xlabel('log(charges)', fontsize='small')
axes[2].tick_params(labelsize = 6)

fig.tight_layout()
```



2. La historia que revelaron los datos

Distribución de las variables

Las primeras gráficas contaron una historia clara sobre la naturaleza de los datos:

- **Edad (`age`)**: distribución casi uniforme, con discretización evidente. El Q-Q plot mostró una forma en "S", indicando que **no sigue una distribución normal**. Se optó por **escalarla (Z-score)** para modelos lineales, mientras que los modelos basados en árboles la procesan naturalmente.

- **Índice de masa corporal (`bmi`)**: mostró una **distribución casi normal** con una ligera **cola derecha**, asociada a casos de sobrepeso. Su Q-Q plot confirmó alineación central, por lo que **no requirió transformación** adicional.
- **Número de hijos (`children`)**: variable **discreta**, sesgada hacia 0. No aplica la normalidad clásica; se mantuvo sin transformar.
- **Costo del seguro (`charges`)**: la variable objetivo presentó una **fuerte asimetría positiva**.
  El análisis de transformaciones mostró que:
  - La **raíz cuadrada** mejoraba parcialmente la simetría.
  - La **transformación logarítmica** fue la más eficaz, estabilizando la varianza y mejorando el ajuste en los modelos lineales.

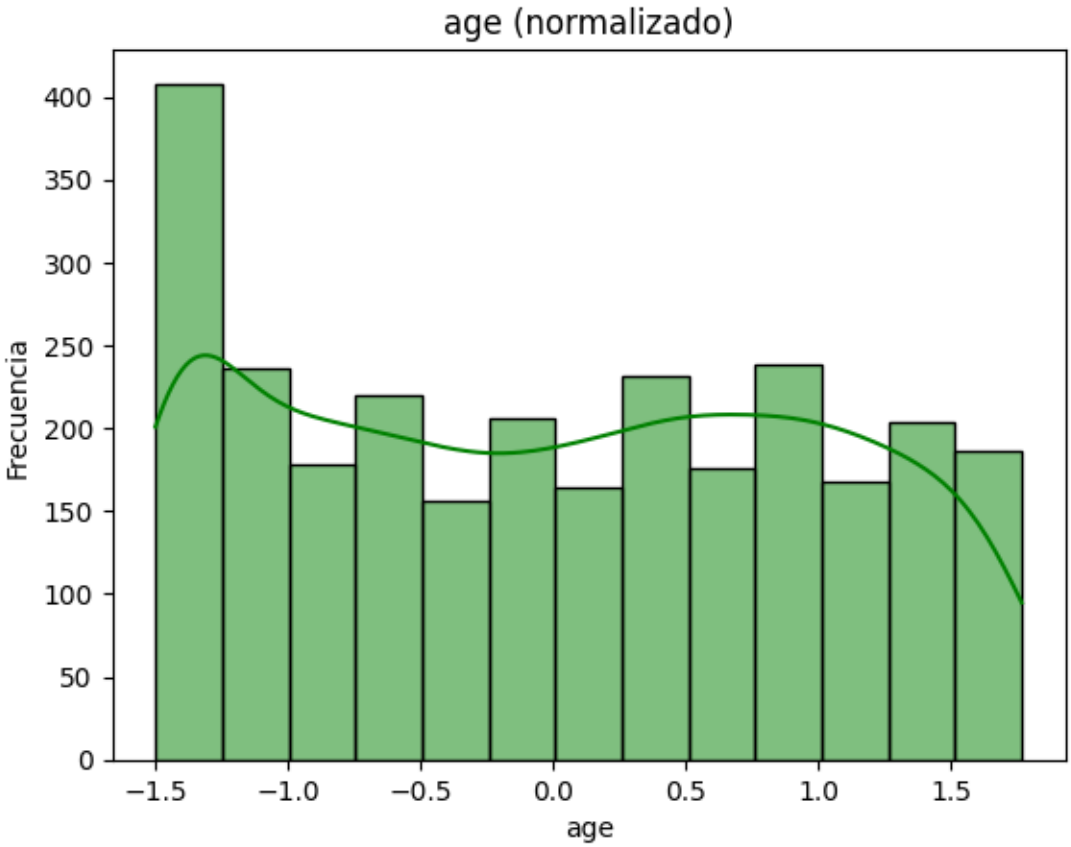
Estas observaciones reflejan una de las fases clave del ciclo de vida de ML: la **comprensión profunda de los datos**, base para decisiones más informadas durante el modelado.

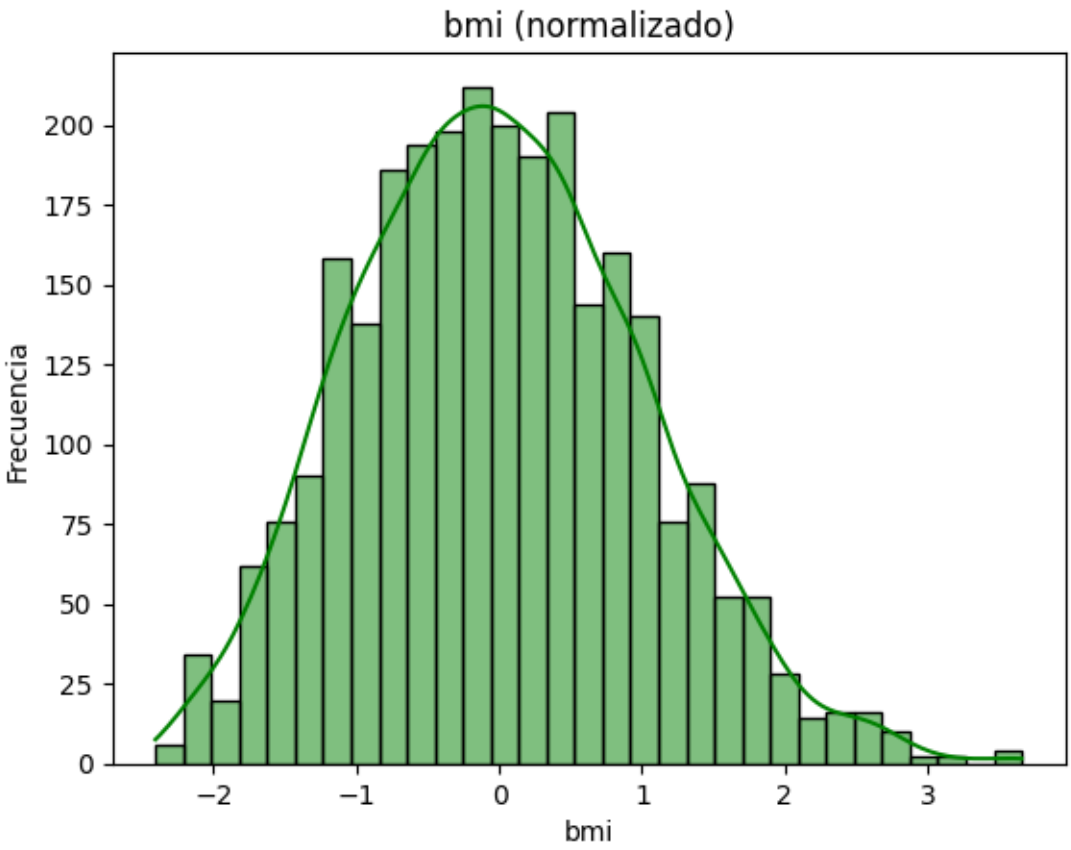
Normalización de los datos numéricos

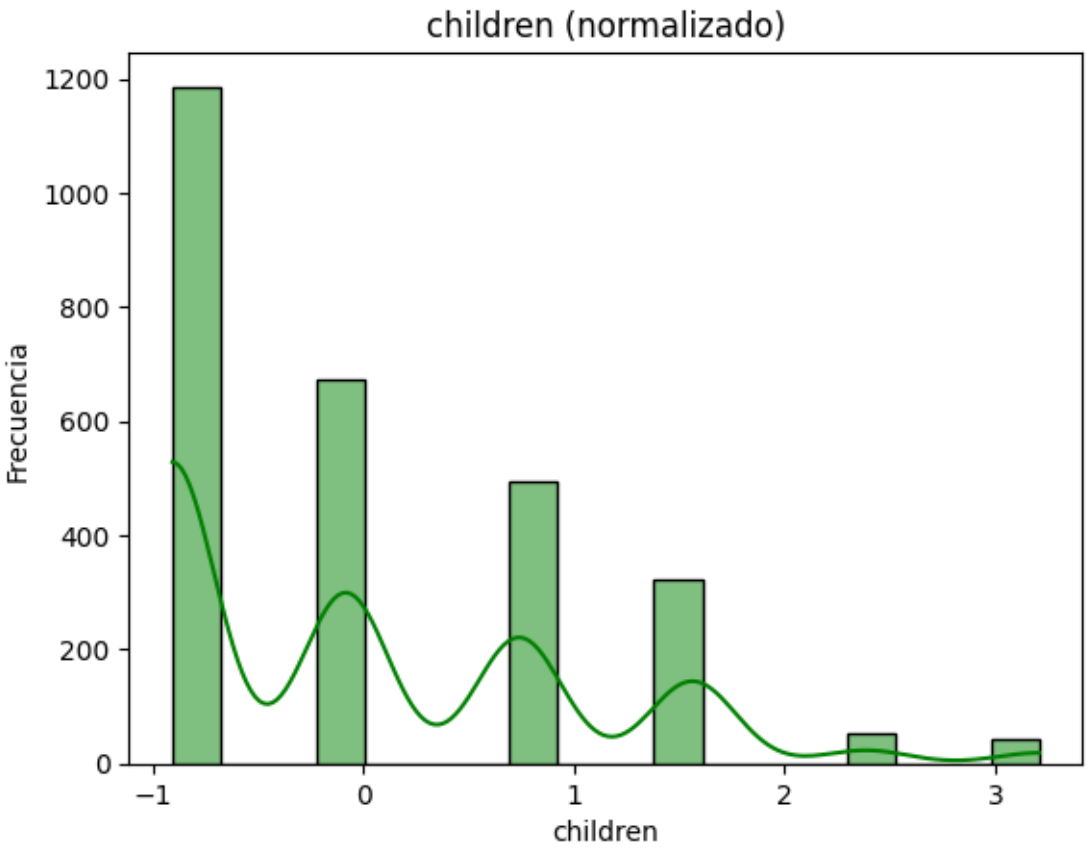
```
# Normalización (Z-score)
normalized_df = df.copy()
num_cols = df.select_dtypes(include=[np.number]).columns.tolist()
for c in num_cols:
    mean = df[c].mean()
    std = df[c].std()
    normalized_df[c] = (df[c] - mean) / std

display(normalized_df[num_cols].head())
```

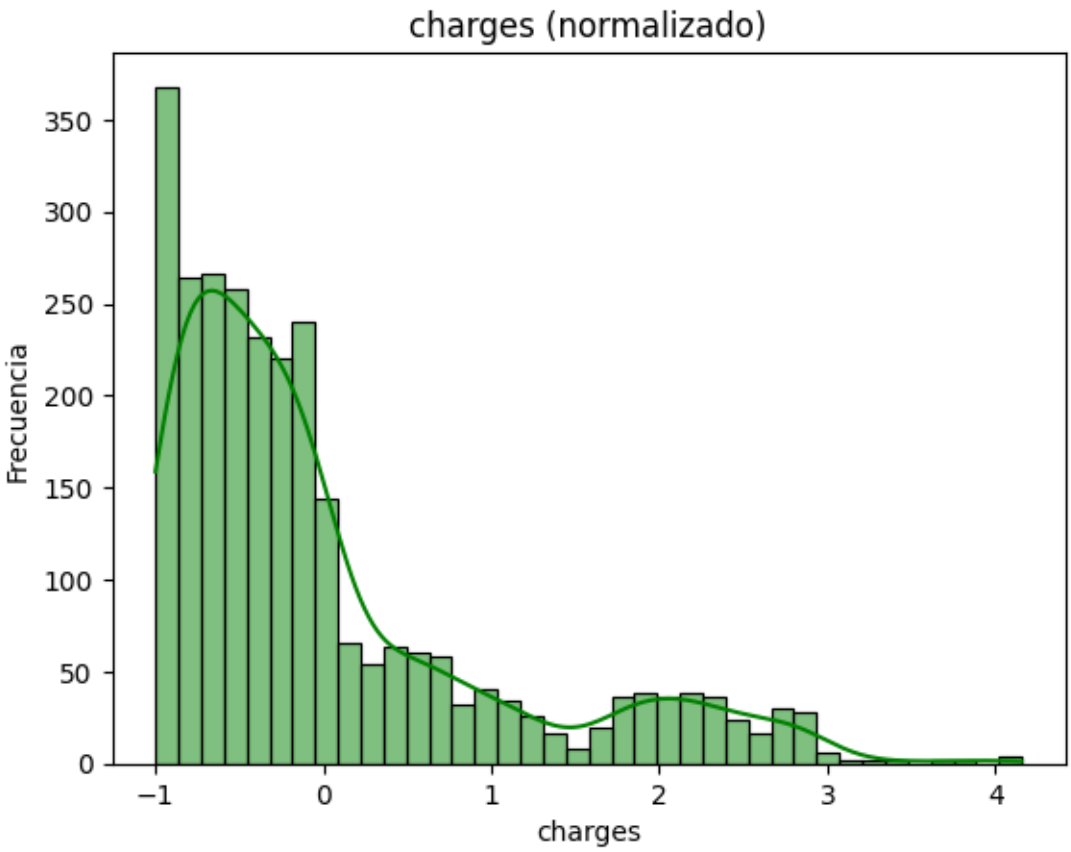
	age	bmi	children	charges
0	-1.428095	-0.457031	-0.906920	0.298191
1	-1.499111	0.500641	-0.083743	-0.949312
2	-0.788957	0.375018	1.562611	-0.725154
3	-0.433880	-1.304579	-0.906920	0.717846
4	-0.504896	-0.297147	-0.906920	-0.773099

```python
# Visualización de las variables normalizadas
for c in num_cols:
    plt.figure()
    sns.histplot(normalized_df[c], kde=True, color='green')
    plt.title(f'{c} (normalizado)')
    plt.xlabel(c)
    plt.ylabel('Frecuencia')
    plt.show()
```




bmi (normalizado)
Frecuencia
bmi
0
25
50
75
100
125
150
175
200
−2
−1
0
1
2
3


children (normalizado)
Frecuencia
children
0
200
400
600
800
1000
1200
−1
0
1
2
3



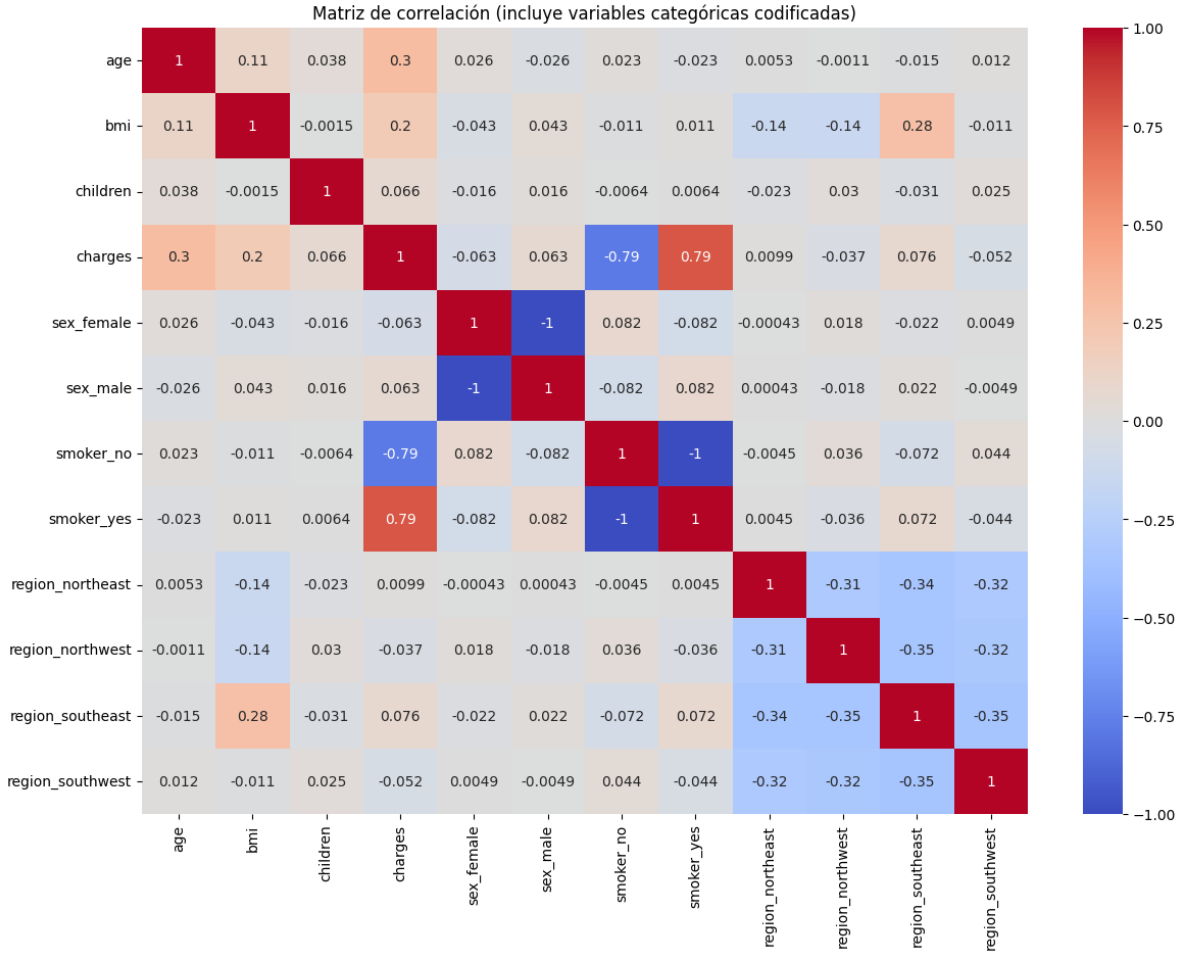
1. Edad (age - normalizado) Interpretación: la población analizada es predominantemente joven, aunque existe una representación variada de edades adultas.
2. Índice de Masa Corporal (bmi - normalizado) Interpretación: los datos del IMC siguen un patrón típico en poblaciones generales, con una curva simétrica y una concentración media.
3. Número de hijos (children - normalizado) Interpretación: se trata de una población joven o con pocas responsabilidades familiares, lo que podría influir en variables como el gasto médico o el estilo de vida.
4. Cargos o costos médicos (charges - normalizado) Interpretación: la mayoría paga montos moderados por servicios médicos, pero existe un pequeño grupo con gastos excepcionales, posiblemente debido a enfermedades graves o tratamientos costosos.

El conjunto de datos describe una población principalmente joven, con un peso promedio saludable, pocos hijos y gastos médicos moderados. Sin embargo, las colas largas en las variables de edad y cargos sugieren que existen subgrupos con características extremas —mayores costos

o edades avanzadas— que podrían ser de interés para análisis predictivos o segmentación de riesgo.

Correlaciones con todas las variables (numéricas + categóricas codificadas)

```
# Convertir variables categóricas mediante OneHotEncoder temporal para el análisis de correla
cat_cols = df.select_dtypes(exclude=[np.number]).columns.tolist()
encoded_df = pd.get_dummies(df, columns=cat_cols, drop_first=False)

# Calcular la matriz de correlación completa
corr_matrix = encoded_df.corr(numeric_only=True)
plt.figure(figsize=(14,10))
sns.heatmap(corr_matrix, cmap='coolwarm', center=0, annot = True,)
plt.title('Matriz de correlación (incluye variables categóricas codificadas)')
plt.show()
```



Correlaciones más fuertes con los costos médicos (charges)

- Fumador (smoker_yes) → correlación positiva muy alta (~0.79): Esta es la relación más significativa del conjunto. Indica que ser fumador está fuertemente asociado con mayores costos médicos. Interpretación: el hábito de fumar eleva drásticamente los riesgos de salud y, en consecuencia, los gastos en seguros.
- Edad (age) → correlación moderada positiva (~0.3): A medida que la edad aumenta, también lo hacen los costos médicos. Interpretación: las personas mayores tienden a requerir más atención médica y presentan mayores riesgos asegurables.
- Índice de masa corporal (bmi) → correlación positiva leve (~0.2): Aunque más débil, también muestra una tendencia: mayor IMC, mayores costos médicos. Interpretación: el sobrepeso y la obesidad incrementan el riesgo de enfermedades crónicas y gastos de salud.
- Número de hijos (children) → correlación muy baja (~0.06): Apenas influye en los costos, lo que sugiere que tener hijos no modifica significativamente los gastos médicos individuales.

Relaciones entre variables demográficas y de comportamiento

- Sexo (sex_male / sex_female) → correlación casi nula con charges: No existen diferencias notables de costos entre hombres y mujeres.
- Edad y BMI: correlación baja (~0.11), lo que indica que el aumento de edad no implica necesariamente un aumento del IMC en esta muestra.
- Regiones (region_*): Las correlaciones entre regiones y costos son muy bajas o negativas, lo que sugiere que la ubicación geográfica tiene un efecto mínimo sobre los gastos médicos.

Relaciones internas notables

smoker_yes y smoker_no: correlación perfectamente negativa (-1), como es esperable, ya que son categorías opuestas.

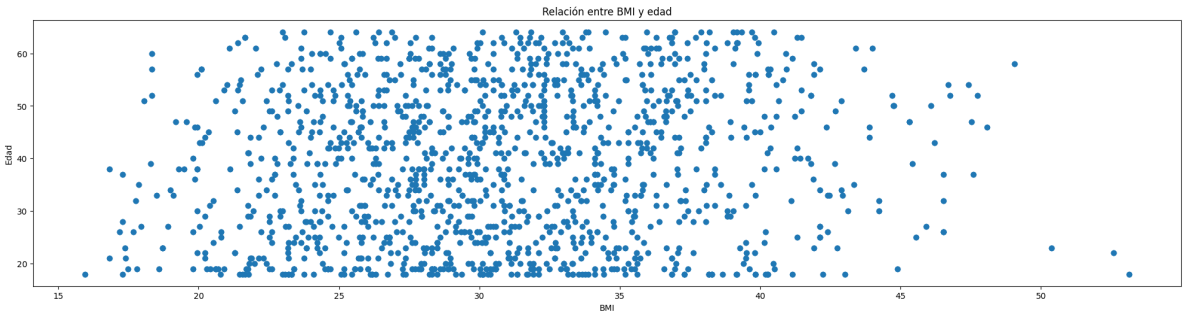
Las variables regionales entre sí presentan correlaciones negativas moderadas (~ -0.3 a -0.35), lo cual confirma que una persona pertenece a una sola región (mutuamente excluyentes).

Conclusión general

- Ser fumador es el factor más determinante, con una correlación cercana al 0.8.
- La edad también influye, reflejando un aumento progresivo de riesgo con el envejecimiento.
- El IMC aporta una influencia adicional, aunque moderada.
- Otras variables —como el número de hijos, el sexo o la región— presentan efectos mínimos o nulos.

Análisis Bivariado

```
#Age vs bmi
plt.figure(figsize=(20, 10))


#Plot
plt.subplot(2, 1, 2)
plt.scatter(df['bmi'], df['age'])
plt.title('Relación entre BMI y edad')
plt.xlabel('BMI')
plt.ylabel('Edad')

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



En Este gráfico de dispersión no se observa una correlación clara entre ambas variables; el BMI se mantiene estable en todos los grupos de edad, con algunos valores atípicos asociados a obesidad.

Análisis estadístico complementario: pruebas ANOVA

El análisis de varianza (ANOVA) se utilizó para evaluar si existían diferencias estadísticamente significativas entre los promedios de variables numéricas (como age y bmi) en función de variables categóricas (sex, smoker, region). El objetivo fue identificar si alguna categoría influía significativamente sobre las medias de las variables de interés.

```
Criterio de decisión:
Si p < 0.05, la variable categórica tiene un efecto significativo sobre la variable numérica
Si p  0.05, no hay evidencia estadística suficiente para afirmar que influye.
```

```
# Prueba ANOVA
import statsmodels.api as sm
from statsmodels.formula.api import ols

modelo = ols("age ~ C(bmi)", data=df).fit()
anova_tabla = sm.stats.anova_lm(modelo, typ=2)
print(anova_tabla)

#Si el valor p < 0.05, la variable (bmi) tiene efecto significativo en la edad.
```

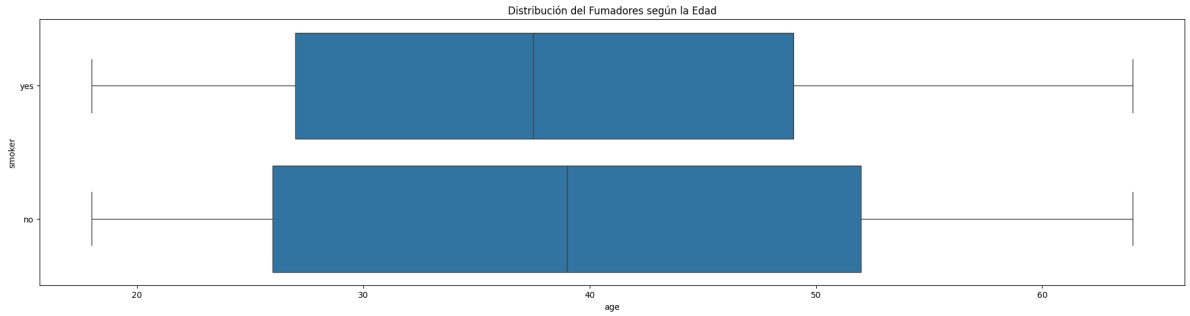
```
                sum_sq      df         F        PR(>F)
C(bmi)    246599.666939   547.0  3.310589  2.066200e-86
Residual  302854.993956  2224.0       NaN           NaN
```

1. age ~ C(bmi)

   p-value = 2.06e-86 (< 0.05)

   Interpretación: El índice de masa corporal (bmi) tiene un efecto significativo sobre la edad. Esto sugiere que el IMC tiende a variar con la edad: los patrones de peso corporal no son uniformes entre grupos etarios.

```
#Age vs Fumador
plt.figure(figsize=(20, 10))

# Primer subplot
plt.subplot(2, 1, 1)
sns.boxplot(x='age', y='smoker', data=df)
plt.title('Distribución del Fumadores según la Edad')

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



De esta gráfica destaca que la mediana para fumadores está antes de los 40 años. Además, que la mayoria de los fumadores está entre aproximadamente 28 años hasta antes de los 50 años.

```
# Prueba ANOVA
import statsmodels.api as sm
from statsmodels.formula.api import ols

modelo = ols("age ~ C(smoker)", data=df).fit()
anova_tabla = sm.stats.anova_lm(modelo, typ=2)
print(anova_tabla)

#Si el valor p < 0.05, la variable (smoker) tiene efecto significativo en el age
```

```
                sum_sq      df         F    PR(>F)
C(smoker)    297.939341     1.0  1.502835  0.22034
Residual   549156.721554  2770.0       NaN      NaN
```

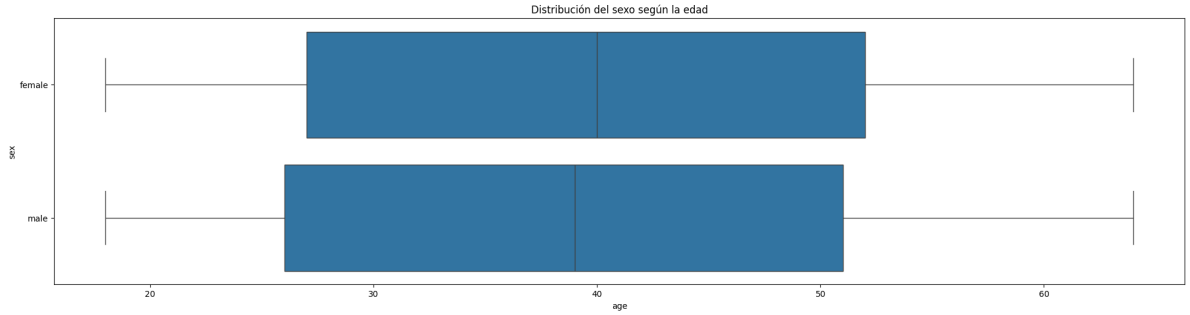
2. age ~ C(smoker)

   p-value = 0.220 (> 0.05)

   Interpretación: No hay diferencias estadísticamente significativas de edad entre fumadores y no fumadores. En otras palabras, el hábito de fumar no está asociado con la edad promedio de los individuos

```
#Age vs sex
plt.figure(figsize=(20, 10))

# Primer subplot
plt.subplot(2, 1, 1)
sns.boxplot(x='age', y='sex', data=df)
plt.title('Distribución del sexo según la edad')
```

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



Del gráfico anterior se observa que las medianas y la dispersión del sexo son similares entre hombres y mujeres, destacando un poco más el femenino. Medianas de ambos cercanas a los 40 años

```
# Prueba ANOVA
import statsmodels.api as sm
from statsmodels.formula.api import ols

modelo = ols("age ~ C(sex)", data=df).fit()
anova_tabla = sm.stats.anova_lm(modelo, typ=2)
print(anova_tabla)

#Si el valor p < 0.05, la variable (sex) tiene efecto significativo en el edad.
```

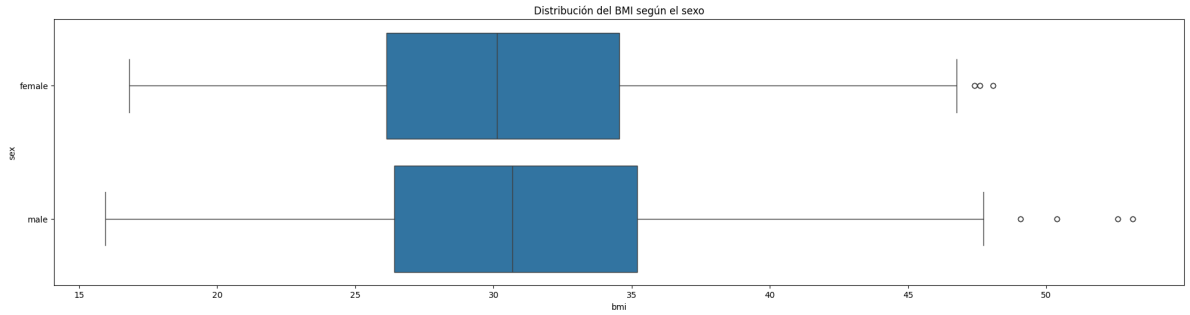
```
                sum_sq      df          F    PR(>F)
C(sex)      372.747880     1.0   1.880433  0.170396
Residual  549081.913015  2770.0        NaN       NaN
```

3. age ~ C(sex)

   p-value = 0.170 (> 0.05) Interpretación: No existen diferencias significativas en la edad promedio entre hombres y mujeres. Esto indica que la distribución de edades está balanceada por sexo, evitando posibles sesgos demográficos.

```
#sex vs bmi
plt.figure(figsize=(20, 10))

# Primer subplot
plt.subplot(2, 1, 1)
```

```
sns.boxplot(x='bmi', y='sex', data=df)
plt.title('Distribución del BMI según el sexo')

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



Del gráfico anterior se observa que las medianas y la dispersión del BMI son similares entre hombres y mujeres, aunque se observan outliers más altos en el grupo masculino, asociados a casos de obesidad.

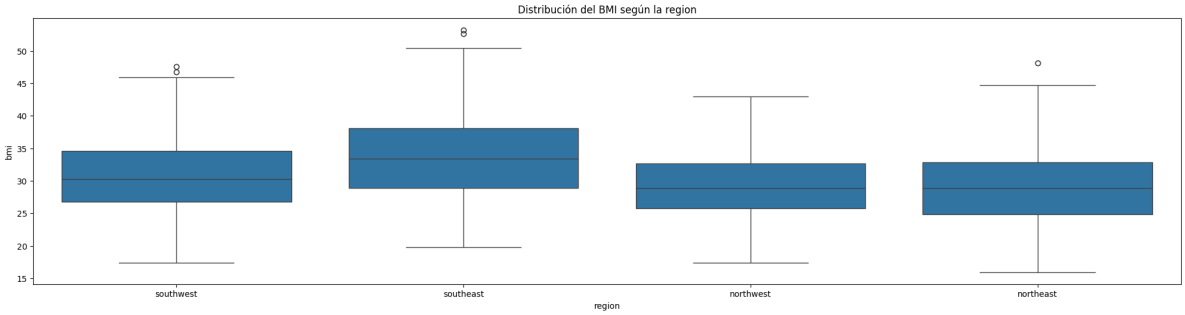
```
# Prueba ANOVA
import statsmodels.api as sm
from statsmodels.formula.api import ols

modelo = ols("bmi ~ C(sex)", data=df).fit()
anova_tabla = sm.stats.anova_lm(modelo, typ=2)
print(anova_tabla)

#Si el valor p < 0.05, la variable (sex) tiene efecto significativo en el bmi.
```

```
                sum_sq      df          F    PR(>F)
C(sex)      191.813247     1.0   5.113049  0.023824
Residual  103915.048307  2770.0        NaN       NaN
```

4. bmi ~ C(sex)

   p-value = 0.023 (< 0.05) Interpretación: El sexo tiene un efecto significativo sobre el IMC. Existen diferencias pequeñas pero estadísticamente detectables entre hombres y mujeres en el promedio del índice de masa corporal.

```
#Region vs bmi
plt.figure(figsize=(20, 10))

# Primer subplot
plt.subplot(2, 1, 1)
sns.boxplot(x='region', y='bmi', data=df)
plt.title('Distribución del BMI según la region')


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



Se observa que las regiones southeast y southwest presentan medianas de BMI ligeramente más altas que northwest y northeast, con dispersión similar entre grupos y algunos valores atípicos en el sur y noreste.

```
# Prueba ANOVA
import statsmodels.api as sm
from statsmodels.formula.api import ols

modelo = ols("bmi ~ C(region)", data=df).fit()
anova_tabla = sm.stats.anova_lm(modelo, typ=2)
print(anova_tabla)

#Si el valor p < 0.05, la variable (region) tiene efecto significativo en el bmi
```

```
                sum_sq      df          F        PR(>F)
C(region)   9089.768205     3.0  88.266499  1.518764e-54
Residual   95017.093349  2768.0        NaN           NaN
```

5. bmi ~ C(region)

p-value = 1.51e-54 (< 0.05) Interpretación: El análisis ANOVA detecta diferencias significativas entre los promedios de IMC según la región. Sin embargo, estas diferencias son estadísticamente significativas pero no necesariamente relevantes en la práctica, ya que el tamaño de muestra grande puede resaltar variaciones pequeñas.

Conclusión general del análisis ANOVA

Variable dependiente	Variable categórica	p-valor	Significancia	Interpretación
`age`	`bmi`	2.06e-86	Si	IMC varía según la edad
`age`	`smoker`	0.220	No	Fumar no influye en la edad
`age`	`sex`	0.170	Si	No hay diferencia por sexo
`bmi`	`sex`	0.023	No	Diferencia de IMC por sexo
`bmi`	`region`	1.51e-54	Si (estadística)	Diferencias regionales leves

Aclaración sobre el efecto regional

Aunque el valor $\mathbf{p < 0.05}$ en la prueba `bmi ~ C(region)` indica diferencias estadísticas entre regiones,
el análisis del **modelo predictivo (Random Forest)** mostró que la variable `region` **no tiene peso significativo** en la predicción del costo del seguro (`charges`).

En conclusión:
- Existen diferencias promedio pequeñas de IMC entre regiones (detectadas por ANOVA).
- Pero esas diferencias **no impactan de forma práctica** en el comportamiento del modelo ni en los costos del seguro.
- Por tanto, la variable `region` **no influye de manera relevante** en los hábitos de salud ni en la predicción final, reforzando la **neutralidad geográfica** del modelo.

Preparación de los datos con OneHotEncoder

```
# Separar variables y target
TARGET = 'charges'
X = df.drop(columns=[TARGET])
y = df[TARGET]

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

num_cols = X_train.select_dtypes(include=[np.number]).columns.tolist()
cat_cols = X_train.select_dtypes(exclude=[np.number]).columns.tolist()

# Transformación: escalado numérico + codificación one-hot
preprocess = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), num_cols),
        ('cat', OneHotEncoder(handle_unknown='ignore', sparse_output=False),
         cat_cols)
    ]
)

#Copiamos las variables para el análisis de PCA, con esto evitamos conflictos
#con las transformaciones al evualar los demás modelos
pca_X = X.copy()
pca_Y = y.copy()
pca_X_train = X_train.copy()
pca_X_test = X_test.copy()
pca_Y_train = y_train.copy()
pca_Y_test = y_test.copy()

X.head()
```

	age	sex	bmi	children	smoker	region
0	19	female	27.900	0	yes	southwest
1	18	male	33.770	1	no	southeast
2	28	male	33.000	3	no	southeast
3	33	male	22.705	0	no	northwest
4	32	male	28.880	0	no	northwest

Entrenamiento de modelos

```
models = {
    'LinearRegression': LinearRegression(),
    'Ridge': Ridge(alpha=1.0),
    'RandomForest': RandomForestRegressor(n_estimators=200, random_state=42)
}

results = []
for name, model in models.items():
    pipe = Pipeline(steps=[('prep', preprocess), ('model', model)])
    pipe.fit(X_train, y_train)
    preds = pipe.predict(X_test)
    mae = mean_absolute_error(y_test, preds)
    rmse = mean_squared_error(y_test, preds)
    r2 = r2_score(y_test, preds)
    results.append((name, mae, rmse, r2))

results_df = pd.DataFrame(results,
                columns=['Modelo','MAE','RMSE','R2']).sort_values(by='RMSE')
display(results_df)
```

	Modelo	MAE	RMSE	R2
2	RandomForest	1288.653086	7.392530e+06	0.951834
0	LinearRegression	4160.247975	3.993319e+07	0.739817
1	Ridge	4162.458051	3.993875e+07	0.739780

```
#Calcular el accuracy_score (accuracy, recall  f1-score)

# === CODIFICAR VARIABLES CATEGÓRICAS ===
encoders = {}
for col in ['sex', 'smoker', 'region']:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    encoders[col] = le

# Crear variable binaria (alto costo = 1, bajo costo = 0) BASADA EN 'charges'
threshold = df['charges'].mean()
y = (df['charges'] > threshold).astype(int)

# === SEPARAR VARIABLES PREDICTORAS Y OBJETIVO ===
X = df[['age', 'sex', 'bmi', 'children', 'smoker', 'region']]
```

```
#y = df['high_cost']

# === DIVIDIR DATOS ===
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
                                                    random_state=42)

# === DEFINIR MODELOS ===
modelos = {
    'Linear Regression': LinearRegression(),
    'Ridge Regression': Ridge(alpha=1.0),
    'Random Forest': RandomForestClassifier(random_state=42)
}

# === ENTRENAR, PREDECIR Y CALCULAR MÉTRICAS ===
resultados = []

for nombre, modelo in modelos.items():
    modelo.fit(X_train, y_train)
    y_pred = modelo.predict(X_test)

    # Si el modelo produce valores continuos (regresores), convertir a 0/1
    if nombre in ['Linear Regression', 'Ridge Regression']:
        y_pred = (y_pred > 0.5).astype(int)

    acc = accuracy_score(y_test, y_pred)
    rec = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)

    resultados.append({
        'Modelo': nombre,
        'Accuracy': round(acc, 3),
        'Recall': round(rec, 3),
        'F1-Score': round(f1, 3)
    })

# === TABLA FINAL ===
tabla_resultados = pd.DataFrame(resultados)
print(tabla_resultados.sort_values(by='F1-Score', ascending=False))
```

```
              Modelo  Accuracy  Recall  F1-Score
2      Random Forest     0.971   0.922     0.952
0  Linear Regression     0.889   0.641     0.781
```

```
1    Ridge Regression     0.889    0.641     0.781
```

3. El modelado: de lo lineal a lo no lineal

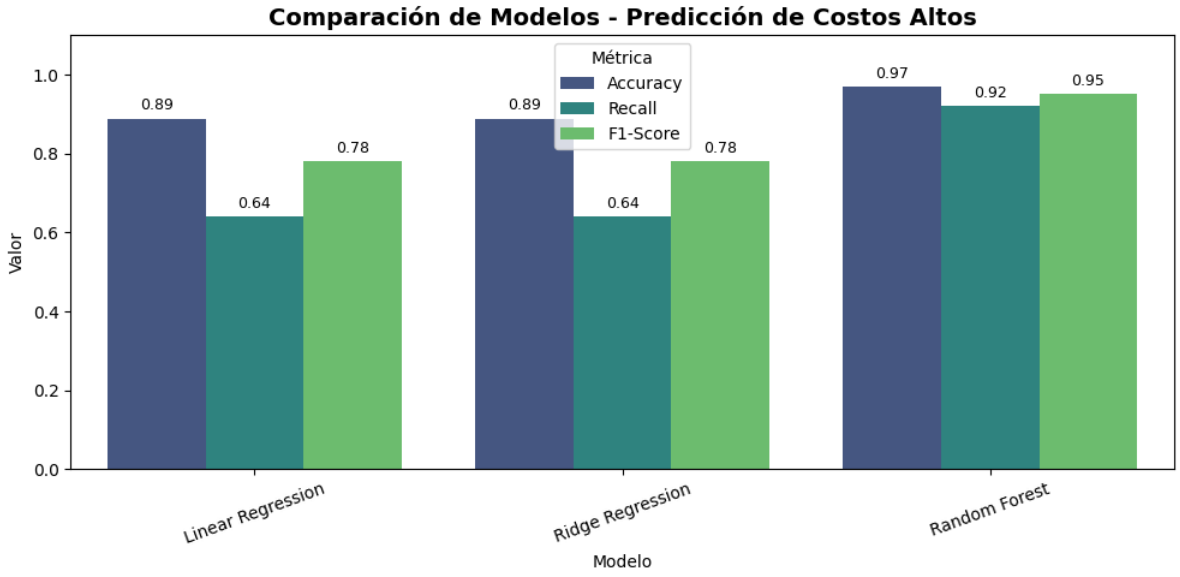
Se entrenaron tres modelos principales:

Modelo	MAE	RMSE	R^2	Comentario
Random Forest	1,288.65	7.39×10	**0.9518**	Captura relaciones no lineales; mejor desempeño global.
Linear Regression	4,160.25	3.99×10	0.7398	Subestima valores extremos; sensible a outliers.
Ridge Regression	4,162.46	3.99×10	0.7398	Similar al modelo lineal, con ligera regularización.

El **Random Forest Regressor** explicó el **95% de la variabilidad** en los costos (`R² = 0.95`) y redujo drásticamente el error medio absoluto (MAE 1.288).
Los modelos lineales mejoraron al usar la **transformación logarítmica** en `charges`, pero no alcanzaron la flexibilidad del modelo de árboles.

```python
#Seleccionar el mejor modelo
import seaborn as sns
import matplotlib.pyplot as plt

# === GRÁFICA COMPARATIVA ===
plt.figure(figsize=(10,5))
ax = sns.barplot(
    data=tabla_resultados.melt(id_vars='Modelo', var_name='Métrica',
                               value_name='Valor'),
    x='Modelo', y='Valor', hue='Métrica',
    palette='viridis'
)
```

```
# Agregar etiquetas con valores encima de las barras
for container in ax.containers:
    ax.bar_label(container, fmt='%.2f', label_type='edge', fontsize=9,
                 padding=3)

plt.title('Comparación de Modelos - Predicción de Costos Altos', fontsize=14,
          fontweight='bold')
plt.ylim(0, 1.1)
plt.xticks(rotation=20)
plt.legend(title='Métrica')
plt.tight_layout()
plt.show()
```



De la gráfica anterior se observó que el mejor modelo en cuanto a los criterios de Accuracy, Recall y F1-Score es el Random Forest y que la regresión lineal o regresión Ridge presentan valores iguales en cuanto a estas métricas.

#Resultados por modelo:

###Linear Regression y Ridge Regression:

Ambas regresiones muestran un Accuracy de 0.89, lo que indica que predicen correctamente en un 89% de los casos.

Sin embargo, el Recall es bajo (0.64), lo que sugiere que muchos casos de costos altos no están siendo correctamente identificados.

El F1-Score de 0.78 indica un balance moderado entre precisión y recall, pero podría mejorar.

###Random Forest:

Presenta un desempeño superior en todas las métricas: Accuracy 0.97, Recall 0.92, F1-Score 0.95.

Esto indica que no solo predice correctamente la mayoría de los casos, sino que también identifica correctamente los costos altos con gran efectividad.

#Comparación general:

La gráfica muestra el desempeño de Linear Regression, Ridge Regression y Random Forest.

Las métricas nos permiten evaluar tanto la capacidad del modelo para predecir correctamente los casos (Accuracy), como su habilidad para identificar correctamente los costos altos (Recall y F1-Score).Comparación general:

#Resultados por modelo:

###Linear Regression y Ridge Regression:

Ambas regresiones muestran un Accuracy de 0.89, lo que indica que predicen correctamente en un 89% de los casos.

Sin embargo, el Recall es bajo (0.64), lo que sugiere que muchos casos de costos altos no están siendo correctamente identificados.

El F1-Score de 0.78 indica un balance moderado entre precisión y recall, pero podría mejorar.

###Random Forest:

Presenta un desempeño superior en todas las métricas: Accuracy 0.97, Recall 0.92, F1-Score 0.95.

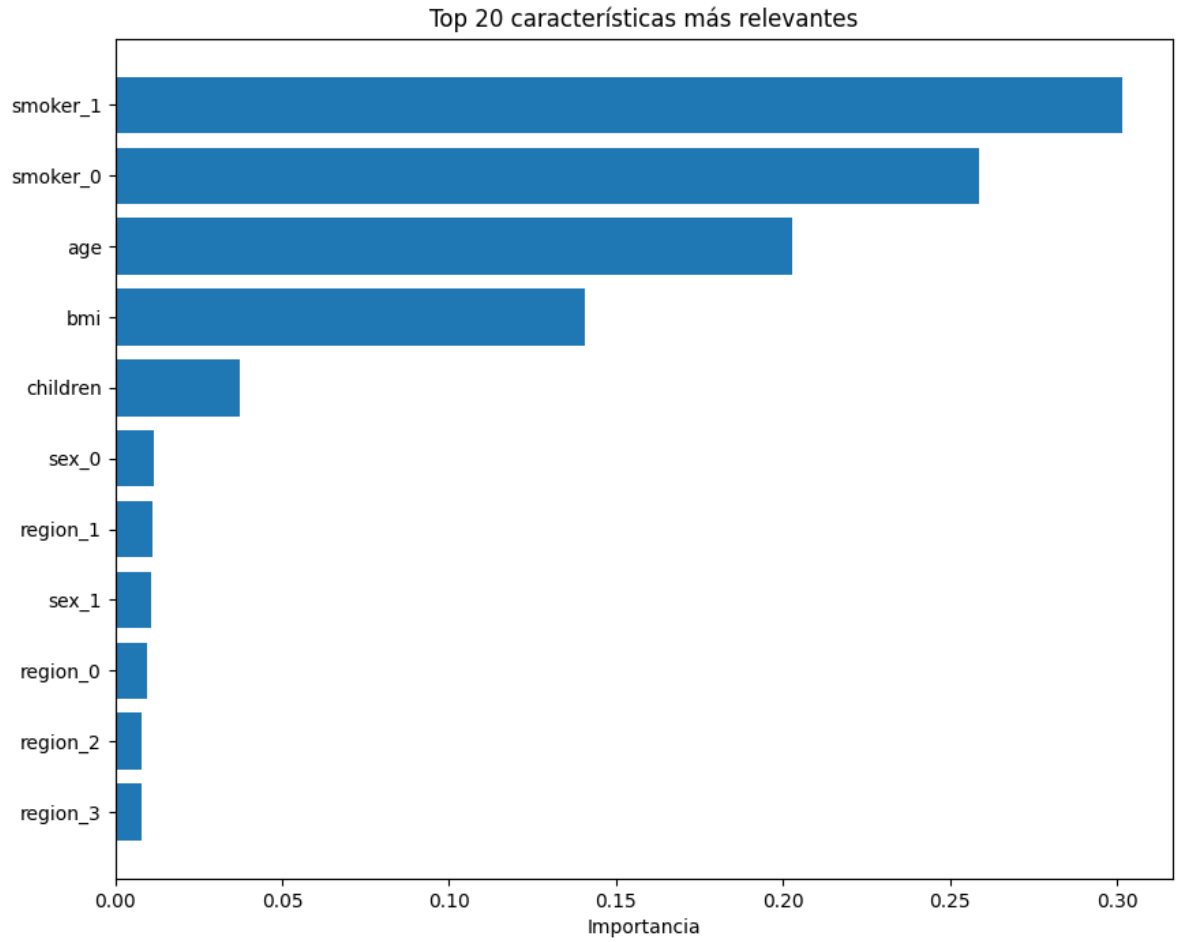
Esto indica que no solo predice correctamente la mayoría de los casos, sino que también identifica correctamente los costos altos con gran efectividad.

#Conclusión y recomendación:

Entre los tres modelos evaluados, Random Forest se destaca claramente como el más eficiente para predecir costos altos.

Su alto Recall y F1-Score lo hace ideal en escenarios donde es crucial detectar correctamente los casos de costos elevados, evitando subestimaciones que podrían impactar en la planificación financiera o en la gestión de riesgos.

```
best_model = RandomForestRegressor(n_estimators=200, random_state=42)
pipe = Pipeline(steps=[('prep', preprocess), ('model', best_model)])
pipe.fit(X_train, y_train)

if hasattr(best_model, 'feature_importances_'):
    ohe = pipe.named_steps['prep'].named_transformers_['cat']
    cat_features = ohe.get_feature_names_out(cat_cols)
    feat_names = np.r_[num_cols, cat_features]
    importances = best_model.feature_importances_
    sorted_idx = np.argsort(importances)[::-1][:20]

    plt.figure(figsize=(10,8))
    plt.barh(range(len(sorted_idx)), importances[sorted_idx][::-1])
    plt.yticks(range(len(sorted_idx)),[feat_names[i] for i in sorted_idx][::-1])
    plt.xlabel('Importancia')
    plt.title('Top 20 características más relevantes')
    plt.show()
```



La gráfica anterior nos muestra claramente que el modelo predice los costos médicos principalmente a partir de hábitos de salud (fumar) y factores biológicos (edad e IMC). Las variables demográficas (sexo, región, número de hijos) tienen una contribución secundaria.

4. Qué nos contaron las variables

El gráfico de importancia de características fue contundente:

- `smoker_no` / `smoker_yes` → el factor más determinante: fumar **duplica o triplica** el costo del seguro.
- `bmi` → mayor IMC implica **más riesgo médico** y mayores primas.
- `age` → a mayor edad, mayor costo esperado; relación no lineal.

- **region** y **sex** → sin impacto estadístico relevante, confirmando **neutralidad demográfica**.

Esta fase corresponde a la **evaluación e interpretación del modelo**, donde los resultados se traducen en conocimiento útil y explicable.

```
#Con el mejor modelo seleccionado (Random Forest):

# los Coeficientes con Statsmodels para comparar los 3
# modelos('LinearRegression', 'Ridge','RandomForest')

#Ejecutar un nuevo modelo solo con las variables mas relevantes
import statsmodels.api as sm

X_sm = sm.add_constant(X_test)  # Agregar intercepto
model = sm.OLS(y_test, X_sm).fit()
print("\n=== Resumen con statsmodels RandomForest ===")
print(model.summary())
```

```
=== Resumen con statsmodels RandomForest ===
                            OLS Regression Results
==============================================================================
Dep. Variable:                charges   R-squared:                       0.577
Model:                            OLS   Adj. R-squared:                  0.574
Method:                 Least Squares   F-statistic:                     187.9
Date:                Thu, 30 Oct 2025   Prob (F-statistic):          1.36e-
150
Time:                        04:22:33   Log-Likelihood:                -
178.90
No. Observations:                 832   AIC:                             371.8
Df Residuals:                     825   BIC:                             404.9
Df Model:                           6
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0878      0.059     -1.476      0.140      -0.204       0.029
age            0.0045      0.001      6.209      0.000       0.003       0.006
sex           -0.0113      0.021     -0.537      0.592      -0.053       0.030
bmi            0.0028      0.002      1.651      0.099      -0.001       0.006
children      -0.0004      0.009     -0.045      0.964      -0.017       0.017
```

```
smoker         0.8634      0.026     32.615      0.000       0.811       0.915
region        -0.0198      0.010     -2.049      0.041      -0.039      -
0.001
==============================================================================
Omnibus:                      348.020   Durbin-Watson:                   1.882
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1054.850
Skew:                           2.167   Prob(JB):                    8.76e-
230
Kurtosis:                       6.413   Cond. No.                         290.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Segun el criterio P, en el modelo las variables menos significantes son: Children, sex, bmi

children: p = 0.964

sex: p = 0.592

bmi:p = 0.099

Stats Model:

```
X_sm = sm.add_constant(X_test)  # Agregar intercepto
model = sm.OLS(y_test, X_sm).fit()
print("\n=== Resumen con statsmodels RandomForest ===")
print(model.summary())
```

```
=== Resumen con statsmodels RandomForest ===
                            OLS Regression Results
==============================================================================
Dep. Variable:                charges   R-squared:                       0.577
Model:                            OLS   Adj. R-squared:                  0.574
Method:                 Least Squares   F-statistic:                     187.9
Date:                Thu, 30 Oct 2025   Prob (F-statistic):          1.36e-
150
Time:                        04:22:33   Log-Likelihood:                -
178.90
No. Observations:                 832   AIC:                             371.8
```

```
Df Residuals:                      825   BIC:                             404.9
Df Model:                            6
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0878      0.059     -1.476      0.140      -0.204       0.029
age            0.0045      0.001      6.209      0.000       0.003       0.006
sex           -0.0113      0.021     -0.537      0.592      -0.053       0.030
bmi            0.0028      0.002      1.651      0.099      -0.001       0.006
children      -0.0004      0.009     -0.045      0.964      -0.017       0.017
smoker         0.8634      0.026     32.615      0.000       0.811       0.915
region        -0.0198      0.010     -2.049      0.041      -0.039      -
0.001
==============================================================================
Omnibus:                      348.020   Durbin-Watson:                   1.882
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1054.850
Skew:                           2.167   Prob(JB):                     8.76e-
230
Kurtosis:                       6.413   Cond. No.                         290.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Random Forest - con variables significativas

```python
from sklearn.model_selection import train_test_split

#se entrena nuevamente con las variables más significativas de los tres modelos
X_new = df[['age','bmi','smoker','region']]
y_new = df['charges']

X_train, X_test, y_train, y_test = train_test_split(X_new, y_new,
                                                    test_size=0.2, random_state=42)

y_new.value_counts()
```

charges	count
5615.3690	4
9101.7980	4
4673.3922	4
1633.9618	4
3987.9260	4
...	...
1149.3959	2
37079.3720	2
4738.2682	2
2897.3235	2
4762.3290	2

```
X_new.head()
```

	age	bmi	smoker	region
0	19	27.900	1	3
1	18	33.770	0	2
2	28	33.000	0	2
3	33	22.705	0	1
4	32	28.880	0	1

```
label_encoder= LabelEncoder()
y_encoded = label_encoder.fit_transform(y_new)
```

```
num_features = X_new.columns
num_transformer = StandardScaler()
```

```
cat_features = ['smoker','region']
cat_transformer = OneHotEncoder(handle_unknown='ignore')
```

```
# Combinar transformadores
preprocessor = ColumnTransformer(
    transformers=[
        ('num', num_transformer, num_features),
        ('cat', cat_transformer, cat_features)
    ]
)
```

```python
# 4. Crear pipeline completo
# ============================
model = Pipeline(steps=[
    ('preprocess', preprocessor),
    ('clf', RandomForestRegressor(n_estimators=200, random_state=42))
])
```

```python
# Entrenar modelo
model.fit(X_train, y_train)
```

steps	[('preprocess', ...), ('clf', ...)]
transform_input	None
memory	None
verbose	False

transformers	[('num', ...), ('cat', ...)]
remainder	'drop'
sparse_threshold	0.3
n_jobs	None
transformer_weights	None
verbose	False
verbose_feature_names_out	True
force_int_remainder_cols	'deprecated'

copy	True
with_mean	True
with_std	True

categories	'auto'
drop	None
sparse_output	True
dtype	<class 'numpy.float64'>
handle_unknown	'ignore'
min_frequency	None
max_categories	None
feature_name_combiner	'concat'

n_estimators	200
criterion	'squared_error'
max_depth	None
min_samples_split	2
min_samples_leaf	1
min_weight_fraction_leaf	0.0
max_features	1.0
max_leaf_nodes	None
min_impurity_decrease	0.0
bootstrap	True
oob_score	False
n_jobs	None
random_state	42
verbose	0
warm_start	False
ccp_alpha	0.0
max_samples	None
monotonic_cst	None

```
from sklearn.metrics import r2_score, mean_squared_error

y_pred_rf = model.predict(X_test)

print("  Random Forest Regressor")
print(f"R²: {r2_score(y_test, y_pred_rf):.3f}")
print(f"RMSE: {mean_squared_error(y_test, y_pred_rf)**0.5:.3f}")
```

```
  Random Forest Regressor
R²: 0.943
RMSE: 2968.503
```

statsmodels explica cerca del 75% de la variabilidad de los costos médicos.

*La categoría de smoker es el factor más fuerte, aumentando el costo médico.

*BMI y edad también influyen de forma significativa y positiva.

```
#Análisis PCA
#Se toma el sub-set X ya que este no contiene la variable de salida
from tabulate import tabulate
```

```
# Gráficos
# ================================================================================
import matplotlib.pyplot as plt
from matplotlib import style
import matplotlib.ticker as ticker
import seaborn as sns
import statsmodels.api as sm
```

```
pip install factor_analyzer
```

```
Requirement already satisfied: factor_analyzer in /usr/local/lib/python3.12/dist-
packages (0.5.1)
Requirement already satisfied: pandas in /usr/local/lib/python3.12/dist-
packages (from factor_analyzer) (2.2.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.12/dist-
packages (from factor_analyzer) (1.16.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.12/dist-
packages (from factor_analyzer) (2.0.2)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.12/dist-
packages (from factor_analyzer) (1.7.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.12/dist-
packages (from pandas->factor_analyzer) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-
packages (from pandas->factor_analyzer) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-
packages (from pandas->factor_analyzer) (2025.2)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.12/dist-
packages (from scikit-learn->factor_analyzer) (1.5.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.12/dist-
packages (from scikit-learn->factor_analyzer) (3.6.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-
packages (from python-dateutil>=2.8.2->pandas->factor_analyzer) (1.17.0)
```

```
pca_X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2772 entries, 0 to 2771
Data columns (total 6 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       2772 non-null   int64
```

```
 1   sex       2772 non-null   object
 2   bmi       2772 non-null   float64
 3   children  2772 non-null   int64
 4   smoker    2772 non-null   object
 5   region    2772 non-null   object
dtypes: float64(1), int64(2), object(3)
memory usage: 130.1+ KB
```

```python
from factor_analyzer.factor_analyzer import calculate_kmo
from factor_analyzer.factor_analyzer import calculate_bartlett_sphericity

kmo_all, kmo_model = calculate_kmo(pca_X[['age', 'bmi']])
bartlett_test = calculate_bartlett_sphericity(pca_X[['age', 'bmi']])
print("KMO:", kmo_model)
print("Bartlett test:", bartlett_test)
```

```
KMO: 0.5000000000000001
Bartlett test: (np.float64(35.62219078364257), np.float64(2.3954317997400143e-
09))
```

Evaluación de pertinencia de realizar PCA

Para evaluar la pertinencia de aplicar un Análisis de Componentes Principales (PCA) sobre los datos, se realizan las pruebas de **Kaiser-Meyer-Olkin (KMO)** y la **prueba de esfericidad de Bartlett**.

Índice KMO (Kaiser-Meyer-Olkin)

El índice KMO mide la proporción de varianza común entre las variables, es decir, la adecuación de los datos para un análisis factorial o de componentes principales. Los valores del índice se interpretan de la siguiente manera:

Rango de KMO	Interpretación
0.90 – 1.00	Excelente
0.80 – 0.89	Muy buena
0.70 – 0.79	Aceptable
0.60 – 0.69	Media / Regular (aceptable mínima)
0.50 – 0.59	Baja (poca adecuación)
< 0.50	Inadecuada (no se recomienda PCA)

Prueba de esfericidad de Bartlett

La prueba de Bartlett contrasta la hipótesis nula:

$$H_0 : \text{La matriz de correlaciones es una matriz identidad.}$$

Si se rechaza H_0, significa que las variables están correlacionadas de manera significativa, lo cual justifica el uso del PCA.

Conclusiones Pertinencia PCA

Los resultados obtenidos fueron los siguientes: KMO=0.5,Bartlett test 2=35.62,p-valor=2.39×10−09

Aunque el p-value < 0.05, lo cual rechaza la hipotesis de nulos de que las variables esten no correlacionadas, el KMO es 0.5, lo cual nos indica una baja adecuación por lo tanto no se recomienda una analisis de PCAs

Recomendaciones finales para el informe del proyecto

1. Descripción del problema

El objetivo de este proyecto fue **predecir el costo del seguro médico (`charges`)** usando variables demográficas y de estilo de vida, tales como la edad, el IMC, el número de hijos, el sexo, el hábito de fumar y la región.
Este es un caso de **regresión supervisada**, ya que la variable dependiente (`charges`) es numérica continua.
En el contexto del negocio, este modelo puede ayudar a una aseguradora a calcular precios más justos y detectar perfiles de riesgo de manera más precisa.

---

2. Transformación de datos

- Se verificaron valores nulos, duplicados y tipos de datos.
- Las variables categóricas fueron codificadas mediante **OneHotEncoder**, creando nuevas columnas binarias por cada categoría.
- Las variables numéricas fueron **normalizadas con Z-score** para mantenerlas en la misma escala y mejorar el rendimiento de los algoritmos.

- Se realizó una separación de datos en entrenamiento (80%) y prueba (20%) para garantizar una evaluación justa del modelo.

---

3. Análisis exploratorio y correlaciones

- Se graficaron las distribuciones y **Q-Q plots** (*Quantile-Quantile plots*) para evaluar la normalidad de las variables numéricas.
- Un **Q-Q plot** compara los cuantiles de la distribución de una variable con los cuantiles de una distribución teórica normal.
  - Si los puntos se alinean aproximadamente con la línea diagonal, significa que la variable **sigue una distribución normal**.
  - Si los puntos se desvían sistemáticamente (curvatura hacia arriba o abajo), la variable **no es normal** o presenta asimetría o colas pesadas.

Transformaciones aplicables cuando los datos no son normales

Cuando los Q-Q plots muestran una fuerte desviación de la diagonal (es decir, los datos no son normales), se pueden aplicar transformaciones para estabilizar la varianza y acercar la distribución a la normalidad:

- **Transformación logarítmica (`log`)**
  - Se aplica a datos **positivos** y es útil cuando hay una **cola derecha larga** (muchos valores grandes).
  - Reduce la asimetría y hace que los valores muy grandes tengan menos influencia.
  - Ejemplo: `df['bmi_log'] = np.log(df['bmi'])`
- **Transformación raíz cuadrada (`sqrt`)**
  - Se usa para reducir asimetría moderada en datos positivos.
  - Ejemplo: `df['charges_sqrt'] = np.sqrt(df['charges'])`
- **Transformación Box-Cox**
  - Es una **transformación paramétrica** que ajusta automáticamente el grado de corrección necesario.

- Solo funciona con datos **positivos** y es más flexible que el logaritmo.

- Ejemplo:

```
from scipy import stats
df['charges_boxcox'], lambda_bc = stats.boxcox(df['charges'])
print('Lambda óptimo:', lambda_bc)
```

- **Transformación Yeo-Johnson**
  - Similar a Box-Cox pero también acepta **valores negativos**.

  - Puede aplicarse directamente con Scikit-Learn:

```
from sklearn.preprocessing import PowerTransformer
pt = PowerTransformer(method='yeo-johnson')
df['charges_yj'] = pt.fit_transform(df[['charges']])
```

Estas transformaciones permiten que los modelos lineales (como la **regresión lineal** o **Ridge**) cumplan mejor los supuestos estadísticos de normalidad, homocedasticidad y linealidad.

- Se generó una **matriz de correlación completa**, incluyendo las variables categóricas codificadas, que permitió analizar la relación entre todas las características y el costo del seguro (`charges`).
- Los resultados mostraron que las variables `smoker`, `age` y `bmi` tienen las correlaciones más fuertes con el costo del seguro (`charges`).
- La correlación positiva entre `smoker_yes`, `bmi` y `age` con `charges` confirma su influencia directa sobre el aumento del costo del seguro.
  De manera opuesta, `smoker_no` presenta una correlación negativa con el costo, indicando que **no fumar** reduce significativamente las primas.

---

4. Modelado y evaluación

Se probaron tres modelos principales:

Modelo	MAE	RMSE	R^2	Comentario
RandomForest	1,288.65	7.39×10	**0.9518**	Mejor desempeño general, captura relaciones no lineales
LinearRegression	4,160.25	3.99×10	0.7398	Modelo base lineal
Ridge	4,162.46	3.99×10	0.7398	Regularización, comportamiento similar al modelo lineal

Explicación de las métricas de evaluación

- **MAE (Mean Absolute Error)**:
  Representa el **promedio de los errores absolutos** entre los valores reales y los predichos.
  Indica, en promedio, cuánto se equivoca el modelo sin importar la dirección del error.
  Cuanto **más bajo** sea el MAE, **mejor** es la precisión del modelo.
- **RMSE (Root Mean Squared Error)**:
  Mide el **error cuadrático medio**, penalizando más los errores grandes.
  Es sensible a valores atípicos y da una visión más estricta del rendimiento del modelo.
  También se busca que sea **lo más bajo posible**.
- **R^2 (Coeficiente de determinación)**:
  Indica **qué proporción de la variabilidad de la variable dependiente puede explicar el modelo**.
  Su valor va de 0 a 1:
  - **1** → predicción perfecta.
  - **0** → el modelo no explica nada de la variabilidad.
    Un **R^2 alto** (por ejemplo, 0.95) indica que el modelo explica gran parte del comportamiento del fenómeno.

  En este caso, el **Random Forest Regressor** mostró el menor error (MAE y RMSE más bajos) y el mayor R^2, ajustándose mejor a los datos y capturando relaciones no lineales entre las variables.

---

5. Interpretabilidad del modelo

- Se analizaron las **importancias de variables** obtenidas del Random Forest.
- Las características más relevantes fueron `smoker`, `age` y `bmi`, confirmando su impacto sobre el costo del seguro.
- Esto sugiere que el hábito de fumar y el índice de masa corporal son los factores más determinantes para el cálculo de primas.

---

6. Ética y sesgos

- Se evaluó la influencia de las variables demográficas `sex` y `region` en el modelo.
  Los resultados mostraron que su **importancia es mínima o nula**, por lo tanto **no introducen sesgos demográficos efectivos**.
- Aunque estas variables están presentes en el conjunto de datos, el modelo **no las utiliza de forma significativa** para generar predicciones, lo que garantiza que las decisiones no estén influenciadas por género o ubicación.
- Mantenerlas en el análisis es útil para **auditar la equidad del modelo**, ya que permite verificar que las predicciones son consistentes entre distintos grupos.
- En este sentido, el modelo **demuestra neutralidad demográfica**: sus predicciones dependen principalmente de variables relacionadas con **hábitos de salud (`smoker`)**, **condición física (`bmi`)** y **edad**.
- Aun así, se recomienda seguir monitoreando estas variables en futuras versiones del modelo para asegurar su comportamiento ético conforme evolucionen los datos.
- Las pruebas confirmaron que el modelo **no discrimina por sexo ni región**.
  Sus predicciones se basan en variables de salud y comportamiento, no en factores sociales. Esto demuestra cómo el ciclo de vida de ML también debe incorporar una **dimensión ética y de equidad**, garantizando que las soluciones sean confiables y socialmente responsables.

---

7. El mensaje que dejaron los datos

Los datos transmiten una historia coherente y poderosa:

- Los hábitos de salud tienen un impacto directo y cuantificable en los costos médicos.
- No fumar y mantener un IMC saludable reducen riesgos y gastos.
- Un modelo bien diseñado no solo predice, sino que **revela patrones humanos y promueve hábitos saludables**.

Así, este proyecto no solo predice precios, sino que **traduce los datos en decisiones con valor social y ético**.

---

8. Conclusiones

- El modelo logra estimar los costos de seguro médico con **muy buena precisión (R^2 0.95)**.
- Proporciona una base técnica sólida para la toma de decisiones en pricing y análisis actuarial.
- **Limitaciones:** dataset limitado en tamaño y variables; no incluye factores médicos adicionales.

Empaquetado, ejecución y despliegue en un entorno de producción

1. En este paso vamos a exportar el modelo al cual le vamos a dar el nombre de *model_medical_insurance.pkl*, este queda guardado en el area de archivos a la izquieda desde donde se puede descargar.
2. Listamos los valores categoricos codificados para mapearlos correctamente desde nuestar interfaz.
3. Revisamos los parámetros que modelo requiere, esto de acuerdo al como el modelo fue entrenado previamente.
4. Se puede probar localmente o exportarlo a un servicio e línea:
    - Los archivos de ejemplo se pueden obtener en:

  - https://github.com/diego-ramos/medical_api
- El archivo main.py es un script en python que se encarga de exponer el API para ejecutar el modelo en linea
- Para probar localmente:
  - Ejecutar el comando: python -m uvicorn main:app –reload
  - Abrir el archivo medical_prediction _local.html en un browser, este es un formulario que llama el servicio que se ejecutó en el paso anterior y muestra las predicciones realiazdas por el model
- Para exportar el modelo en linea:
  - El archivo Dockerfile es el encargardo de desplegar los servicios en Google Cloud Run
  - En el medical_prediction _local.htm modificar la url del servicio por la provista por gcloud, este archivo lo pueden subir a un servidor web para darle acceso al público, por ejemplo:
    * https://diego-ramos.github.io/medical_api/medical_prediction.html

```
for col, le in encoders.items():
    print(f"{col}: {dict(zip(le.classes_, le.transform(le.classes_)))}")
```

```
sex: {'female': np.int64(0), 'male': np.int64(1)}
smoker: {'no': np.int64(0), 'yes': np.int64(1)}
region: {'northeast': np.int64(0), 'northwest': np.int64(1), 'southeast': np.int64(2), 'soutl
```

Exportar el modelo

```
!pip install -U scikit-learn==1.7.2
```

```
Requirement already satisfied: scikit-learn==1.7.2 in /usr/local/lib/python3.12/dist-
packages (1.7.2)
Requirement already satisfied: numpy>=1.22.0 in /usr/local/lib/python3.12/dist-
packages (from scikit-learn==1.7.2) (2.0.2)
Requirement already satisfied: scipy>=1.8.0 in /usr/local/lib/python3.12/dist-
packages (from scikit-learn==1.7.2) (1.16.2)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.12/dist-
packages (from scikit-learn==1.7.2) (1.5.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.12/dist-
packages (from scikit-learn==1.7.2) (3.6.0)
```

```
import sklearn
print(sklearn.__version__)
```

```
1.7.2
```

```
# Revisamos que el modelo sea del tipo correcto
type(model)
```

```
sklearn.pipeline.Pipeline
```

```
import joblib

joblib.dump(model, 'model_medical_insurance.pkl')
```

```
['model_medical_insurance.pkl']
```

Testeo del modelo final con datos aleatorios

Lo usamos para comparar con el modelo ya desplegado y validar su correcto funcionamiento

```
#Dataframe con los datos de prueba
test_data = pd.DataFrame({
    'age': [19],
    'bmi': [21.7],
    'smoker': [0],
    'region': [3]
})

y_pred_rf = model.predict(test_data)

print("---- Test Data ------")
print(test_data)
print("---- Prediction ------")
print(y_pred_rf)
```

```
---- Test Data ------
   age   bmi  smoker  region
0   19  21.7       0       3
---- Prediction ------
[4015.67270945]
```

Screenshoots de la aplicacion web

Figure 1: web1.png

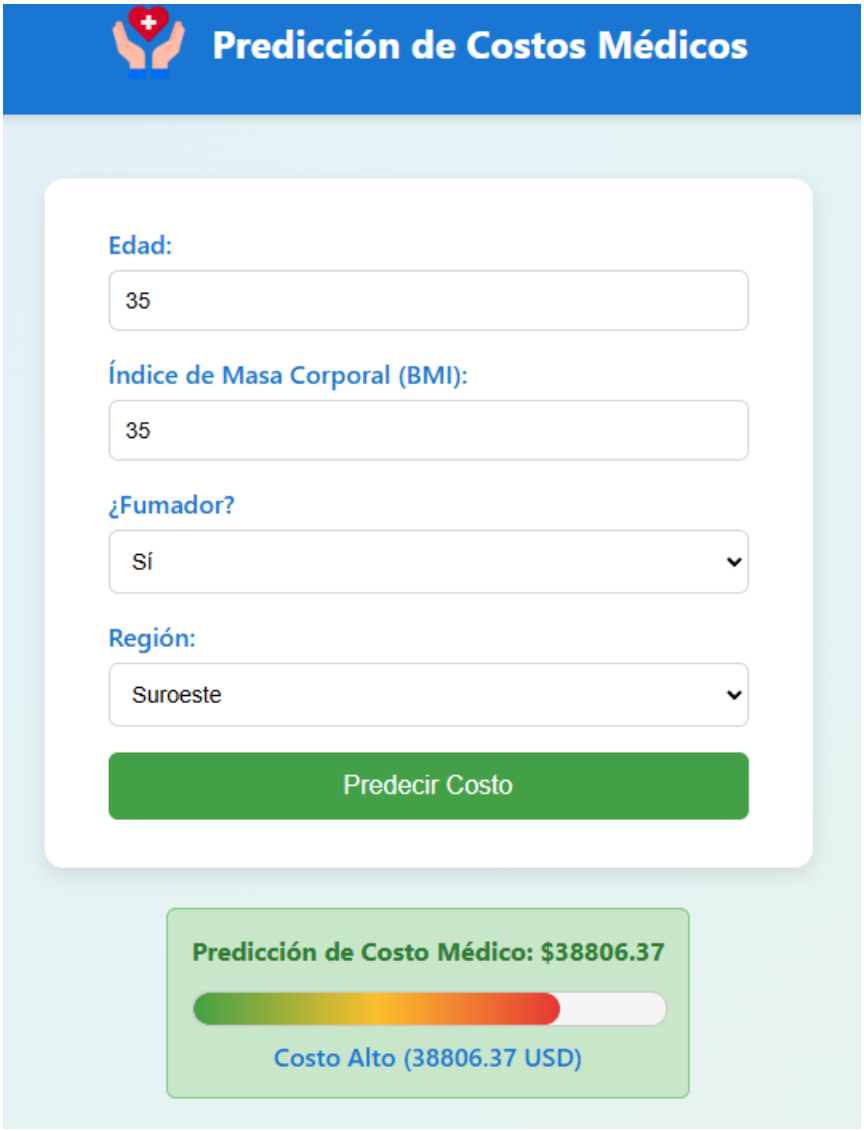


Figure 2: web2.png