

## › Programa Ejemplo para Realizar Aprendizaje Supervisado

### Etapa 1: Entendimiento de los datos

Antes de comenzar incluso a preprocesar los datos, debemos conocer las características del conjunto de datos que tenemos.

Como mínimo debemos saber:

- Cuántos registros hay?
- Cuantos atributos tiene cada registro?
- De qué tipo son los atributos?

Además:

- Obtener una medida de centralidad para cada atributo
- Obtener una medida de dispersión o desviación para los atributos que ésto tenga sentido
- Calcular la matriz de correlación de los atributos de entrada para identificar atributos redundantes

Como paso previo al preprocesamiento de los datos, debemos:

- Detectar si hay datos faltantes (determinar en qué columnas y cuantos datos faltan en un mismo registro)
- Detectar si hay datos atípicos
- Detectar si hay desbalance entre clases

Notar que en esta etapa no hacemos ninguna modificación sobre los datos, sólo estamos conociéndolos a fondo para saber con qué material contamos para trabajar.

[ ] ↪ 21 cells hidden

### Fin del programa

## ✓ Telco Customer Churn

### Etapa 1: Entendimiento y Análisis exploratorio de los datos (EDA)

```
import kagglehub
```

```
# Download latest version
```

```
path = kagglehub.dataset_download("blastchar/telco-customer-churn")
```

```
print("Path to dataset files:", path)
```

⚠ Warning: Looks like you're using an outdated `kagglehub` version (installed: 0.3.7), please consider upgrading to the latest version (0.3.8).  
Path to dataset files: /root/.cache/kagglehub/datasets/blastchar/telco-customer-churn/versions/1

```
csv_file_path = path + '/WA_Fn-UseC_-Telco-Customer-Churn.csv' # Or the actual filename
```

```
data = pd.read_csv(csv_file_path)
```

## Exploración Básica:

### a) Número de muestras, datos faltantes, atributos y tipos de variables de cada atributo

```
print(f'Número de muestras: {data.shape[0]}')
print(f'Número de atributos: {data.shape[1]}')
data.shape
```

```
➜ Número de muestras: 7043
    Número de atributos: 21
    (7043, 21)
```

```
data.head(7043)
```

➜

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSupport	StreamingTV
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No	No	No
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes	No	No
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No	No	No
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes	Yes	No
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No	No	No
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
7038	6840-RESVB	Male	0	Yes	Yes	24	Yes	Yes	DSL	Yes	...	Yes	Yes	Yes
7039	2234-XADUH	Female	0	Yes	Yes	72	Yes	Yes	Fiber optic	No	...	Yes	No	Yes

```
data.info()
```

```
➜ <class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype
---  -
 0   customerID      7043 non-null   object
 1   gender          7043 non-null   object
 2   SeniorCitizen   7043 non-null   int64
 3   Partner         7043 non-null   object
 4   Dependents      7043 non-null   object
 5   tenure          7043 non-null   int64
 6   PhoneService    7043 non-null   object
```

```

7 MultipleLines      7043 non-null object
8 InternetService    7043 non-null object
9 OnlineSecurity     7043 non-null object
10 OnlineBackup      7043 non-null object
11 DeviceProtection  7043 non-null object
12 TechSupport       7043 non-null object
13 StreamingTV       7043 non-null object
14 StreamingMovies   7043 non-null object
15 Contract          7043 non-null object
16 PaperlessBilling  7043 non-null object
17 PaymentMethod     7043 non-null object
18 MonthlyCharges    7043 non-null float64
19 TotalCharges      7043 non-null object
20 Churn             7043 non-null object
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB

```

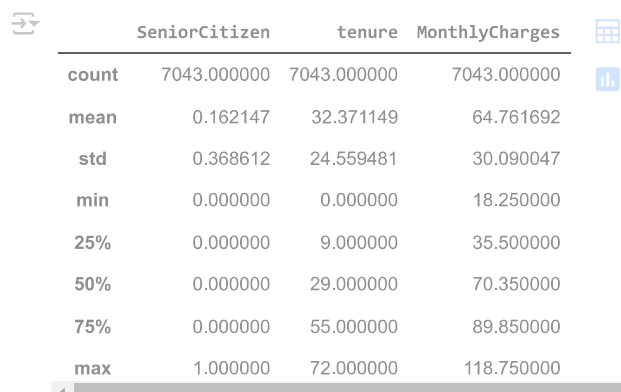
## ✓ b) Medidas de Centralidad y Dispersión:

1. **Centralidad:** Calcula la media, mediana y moda para cada atributo numérico.
2. **Dispersión:** Obtén la desviación estándar, varianza, rango intercuartílico, etc., para entender la variabilidad de los datos.

```

# Para atributos numéricos
data.describe()

```



	SeniorCitizen	tenure	MonthlyCharges
count	7043.000000	7043.000000	7043.000000
mean	0.162147	32.371149	64.761692
std	0.368612	24.559481	30.090047
min	0.000000	0.000000	18.250000
25%	0.000000	9.000000	35.500000
50%	0.000000	29.000000	70.350000
75%	0.000000	55.000000	89.850000
max	1.000000	72.000000	118.750000

Con base en lo anterior, se destacan las siguientes observaciones:

- **Observación 1:** Para fines del modelado, el *ID del cliente (Customer ID)* no tiene ningún valor predictivo y se descartará.
- **Observación 2:** Las variables categóricas deben codificarse como numéricas para el modelado.
- **Observación 3:** Se requiere transformar el atributo *TotalCharges* a un formato numérico para su posterior procesamiento.

## ✓ c) Detección de Datos Faltantes

```

# Datos faltantes
print("Datos Faltantes por atributo")
data.isnull().sum()

```


 Datos Faltantes por atributo

	0
customerID	0
gender	0
SeniorCitizen	0
Partner	0
Dependents	0
tenure	0
PhoneService	0
MultipleLines	0
InternetService	0
OnlineSecurity	0
OnlineBackup	0
DeviceProtection	0
TechSupport	0
StreamingTV	0
StreamingMovies	0
Contract	0
PaperlessBilling	0
PaymentMethod	0
MonthlyCharges	0
TotalCharges	0
Churn	0

dtype: int64

## ▼ Análisis Exploratorio de los Datos (EDA)

```
data.info()
print(data.isnull().sum())
```



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
#   Column              Non-Null Count  Dtype
---  -
0   customerID          7043 non-null   object
1   gender               7043 non-null   object
2   SeniorCitizen        7043 non-null   int64
3   Partner              7043 non-null   object
4   Dependents           7043 non-null   object
5   tenure               7043 non-null   int64
```

```

6  PhoneService      7043 non-null object
7  MultipleLines     7043 non-null object
8  InternetService   7043 non-null object
9  OnlineSecurity    7043 non-null object
10 OnlineBackup      7043 non-null object
11 DeviceProtection  7043 non-null object
12 TechSupport       7043 non-null object
13 StreamingTV       7043 non-null object
14 StreamingMovies   7043 non-null object
15 Contract          7043 non-null object
16 PaperlessBilling  7043 non-null object
17 PaymentMethod     7043 non-null object
18 MonthlyCharges    7043 non-null float64
19 TotalCharges      7043 non-null object
20 Churn             7043 non-null object

```

```
dtypes: float64(1), int64(2), object(18)
```

```
memory usage: 1.1+ MB
```

```
customerID      0
```

```
gender          0
```

```
SeniorCitizen   0
```

```
Partner         0
```

```
Dependents      0
```

```
tenure          0
```

```
PhoneService    0
```

```
MultipleLines   0
```

```
InternetService 0
```

```
OnlineSecurity  0
```

```
OnlineBackup    0
```

```
DeviceProtection 0
```

```
TechSupport     0
```

```
StreamingTV     0
```

```
StreamingMovies 0
```

```
Contract        0
```

```
PaperlessBilling 0
```

```
PaymentMethod   0
```

```
MonthlyCharges  0
```

```
TotalCharges    0
```

```
Churn           0
```

```
dtype: int64
```

## ✎ Boxplots (diagramas de cajas y bigotes) para las variables *Tenure*, *MonthlyCharges* y *TotalCharges*

Convertiremos a *TotalCharges* a un formato numérico para su posterior procesamiento

```
data['TotalCharges'] = pd.to_numeric(data['TotalCharges'], errors='coerce')
```

```
print(data['TotalCharges'].dtype)
```

```
float64
```

```
# Visualización de outliers en atributos numéricos
```

```
plt.figure(figsize=(15,3))
```

```
for i, col in enumerate(['tenure', 'MonthlyCharges', 'TotalCharges']):
```

```
    plt.subplot(1, 3, i + 1)
```

```
    # Convertimos la columna TotalCharges a numérica antes de calcular la mediana
```

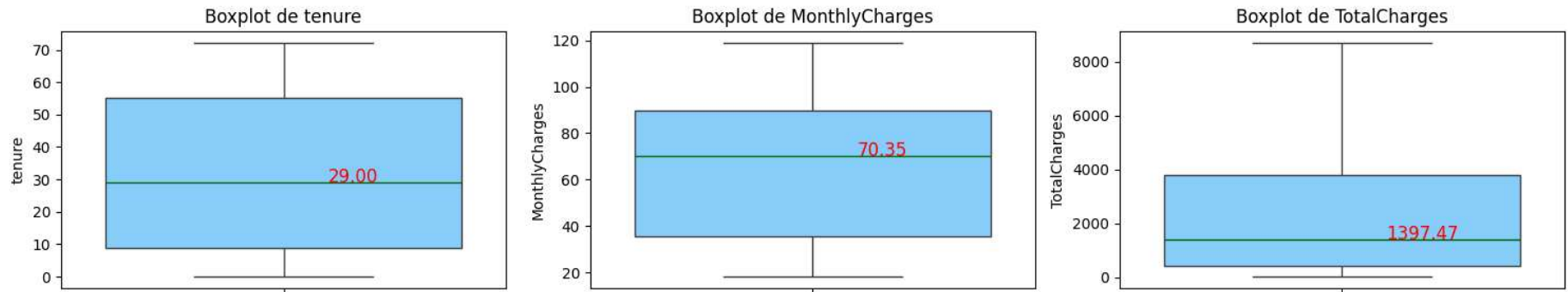
```
    #data[col] = pd.to_numeric(data[col], errors='coerce')
```

```

bp = sns.boxplot(y=data[col], medianprops={'color': 'darkgreen'},
                 patch_artist=True,
                 boxprops={'facecolor': 'lightskyblue'})
median_val = data[col].median()
plt.text(0.1, median_val, f'{median_val:.2f}',
        fontsize=12, color='red')

plt.title(f'Boxplot de {col}')
plt.tight_layout()
plt.show()

```



## ✓ Análisis de los Diagramas de Caja y Datos Estadísticos

### Interpretación General

#### Antigüedad (Tenure)

- **Distribución Sesgada a la Derecha:**

El diagrama de caja para la antigüedad probablemente muestra una distribución asimétrica con una cola extendida hacia la derecha. Esto indica que la mayoría de los clientes tiene un periodo corto de relación con la compañía, mientras que un grupo menor permanece por tiempos considerablemente más largos.

- **Amplio Rango:**

La caja abarca un rango considerable, lo que sugiere una alta variabilidad en la duración de la relación con los clientes.

- **Valores Atípicos:**

Se pueden observar algunos valores atípicos en el extremo superior, representando a aquellos clientes con una permanencia excepcionalmente larga.

#### Cargos Mensuales (MonthlyCharges)

- **Distribución Relativamente Simétrica:**

El diagrama de caja de los cargos mensuales es más simétrico en comparación con el de la antigüedad, lo que sugiere que la mayoría de los clientes paga montos similares mensualmente.

- **Rango Moderado:**

Los valores para los cargos mensuales presentan un rango menos extenso, indicando menor dispersión en comparación con la antigüedad.

- **Posibles Valores Atípicos:**

Puede haber algunos valores atípicos en ambos extremos (inferior y superior), señalando la existencia de clientes con cargos inusualmente bajos o altos.

#### Cargos Totales (TotalCharges)

- **Distribución Sesgada a la Derecha:**

Similar a la antigüedad, el diagrama de caja para los cargos totales muestra una asimetría hacia la derecha. Esto implica que la mayoría de los clientes acumula cargos totales relativamente bajos, mientras que un pequeño grupo alcanza montos elevados.

- **Rango Amplio:**

El rango de los cargos totales es el más extenso entre las tres variables, reflejando una gran variabilidad en los montos acumulados.

- **Valores Atípicos:**

Es posible identificar valores atípicos en el extremo superior, correspondientes a clientes con acumulaciones de cargos excepcionalmente altas.

#### Posibles Implicaciones

- **Antigüedad:**

La distribución sesgada a la derecha sugiere una mayor tasa de abandono en las etapas iniciales, que disminuye conforme los clientes permanecen más tiempo. Por ello, retener a los clientes a largo plazo es fundamental para la estabilidad de la compañía.

- **Cargos Mensuales:**

La distribución relativamente simétrica indica que la estructura de precios es equilibrada y se ajusta a una base de clientes diversa.

- **Cargos Totales:**

La asimetría observada en los cargos totales evidencia una relación directa entre la antigüedad del cliente y el monto acumulado en cargos. Los clientes con mayor permanencia tienden a generar cargos totales más elevados.

```
import numpy as np
from scipy.stats import norm

def graficarHistogramaDistNormal(column, title, subplot, data):
    plt.subplot(subplot)
    # Filter out non-finite values
    filtered_data = data[column][~np.isfinite(data[column])]

    plt.hist(filtered_data, bins=20, density=True, alpha=0.6, label='Histograma')

    # Calcular la media y desviación estándar (using filtered data)
    mu, std = norm.fit(filtered_data)

    # Generar valores para la distribución normal teórica
    xmin, xmax = plt.xlim()
    x = np.linspace(xmin, xmax, 100)
    y = norm.pdf(x, mu, std)
```

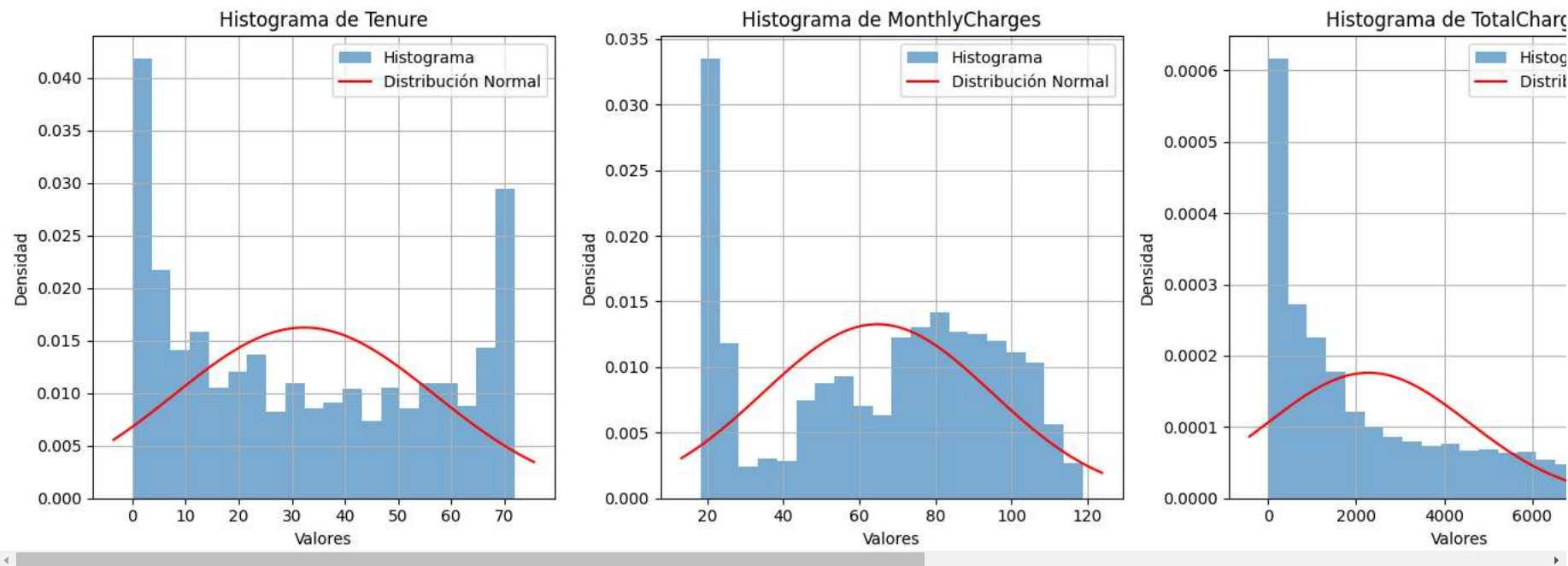
```
plt.plot(x, y, 'r-', label='Distribución Normal')

plt.xlabel('Valores')
plt.ylabel('Densidad')
plt.title(title)
plt.legend()
plt.grid(True)

plt.figure(figsize=(15, 5))

graficarHistogramaDistNormal('tenure', 'Histograma de Tenure', 131, data)
graficarHistogramaDistNormal('MonthlyCharges', 'Histograma de MonthlyCharges', 132, data)
graficarHistogramaDistNormal('TotalCharges', 'Histograma de TotalCharges', 133, data)

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



Los histogramas proporcionados ofrecen una visión clara de la distribución de las variables y es posible decir que:

## ✓ Antigüedad (Tenure)

### • \*\* Distribución: \*\*

La distribución de **Tenure** muestra una ligera **asimetría positiva**, con una cola derecha más larga. Esto indica que hay un mayor número de clientes con un período de suscripción más corto, pero también existe un grupo significativo de clientes de larga duración.



- **\*\* Implicaciones:\*\***

Esta distribución sugiere la presencia de diferentes **segmentos de clientes**:

- **Clientes a corto plazo:** Probablemente más volátiles, con una mayor tasa de abandono.
- **Clientes a largo plazo:** De mayor valor, con mayor probabilidad de permanencia.

## Cargos Mensuales (MonthlyCharges)

- **\*\* Distribución:\*\***

La distribución de **MonthlyCharges** se aproxima a una **distribución normal**, con una ligera **asimetría positiva**. Esto indica que la mayoría de los clientes paga un cargo mensual similar, aunque hay un grupo que paga tarifas más altas.

- **\*\* Implicaciones:\*\***

- La estrategia de precios de la compañía ha sido efectiva en atraer a una base de clientes con tarifas promedio.
- Existe un segmento de clientes dispuestos a pagar más por **servicios adicionales**.

## Cargos Totales (TotalCharges)

- **Distribución:**

La distribución de **TotalCharges** presenta una **asimetría positiva pronunciada**, con una cola derecha muy larga. Esto indica que la mayoría de los clientes tiene **cargos acumulados bajos**, mientras que un pequeño grupo ha generado **cargos significativamente altos**.

- **Implicaciones:**

- Existe una gran **variabilidad en el consumo de servicios** entre los clientes.
- Un pequeño grupo de clientes contribuye a una **parte importante de los ingresos**.

```
data.info()
print(data.isnull().sum())
```



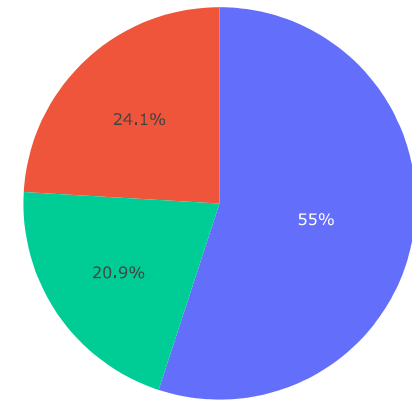
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
#   Column                Non-Null Count  Dtype
---  -
0   customerID            7043 non-null  object
1   gender                 7043 non-null  object
2   SeniorCitizen          7043 non-null  int64
3   Partner                7043 non-null  object
4   Dependents             7043 non-null  object
5   tenure                 7043 non-null  int64
6   PhoneService           7043 non-null  object
7   MultipleLines           7043 non-null  object
8   InternetService        7043 non-null  object
9   OnlineSecurity          7043 non-null  object
10  OnlineBackup            7043 non-null  object
11  DeviceProtection       7043 non-null  object
12  TechSupport            7043 non-null  object
13  StreamingTV            7043 non-null  object
14  StreamingMovies         7043 non-null  object
15  Contract               7043 non-null  object
16  PaperlessBilling        7043 non-null  object
17  PaymentMethod          7043 non-null  object
18  MonthlyCharges         7043 non-null  float64
19  TotalCharges           7032 non-null  float64
20  Churn                  7043 non-null  object
```

```
dtypes: float64(2), int64(2), object(17)
memory usage: 1.1+ MB
customerID      0
gender          0
SeniorCitizen   0
Partner         0
Dependents      0
tenure          0
PhoneService    0
MultipleLines   0
InternetService 0
OnlineSecurity  0
OnlineBackup    0
DeviceProtection 0
TechSupport     0
StreamingTV     0
StreamingMovies 0
Contract        0
PaperlessBilling 0
PaymentMethod   0
MonthlyCharges  0
TotalCharges    11
Churn           0
dtype: int64
```

```
import plotly.express as px
```

```
contract_counts = data['Contract'].value_counts().reset_index()
contract_counts.columns = ['Contract', 'Count']
```

```
fig = px.pie(data_frame=contract_counts, values='Count', names='Contract')
fig.show()
```



```
payment_method_counts = data['PaymentMethod'].value_counts().reset_index()
payment_method_counts.columns = ['PaymentMethod', 'Count']
```

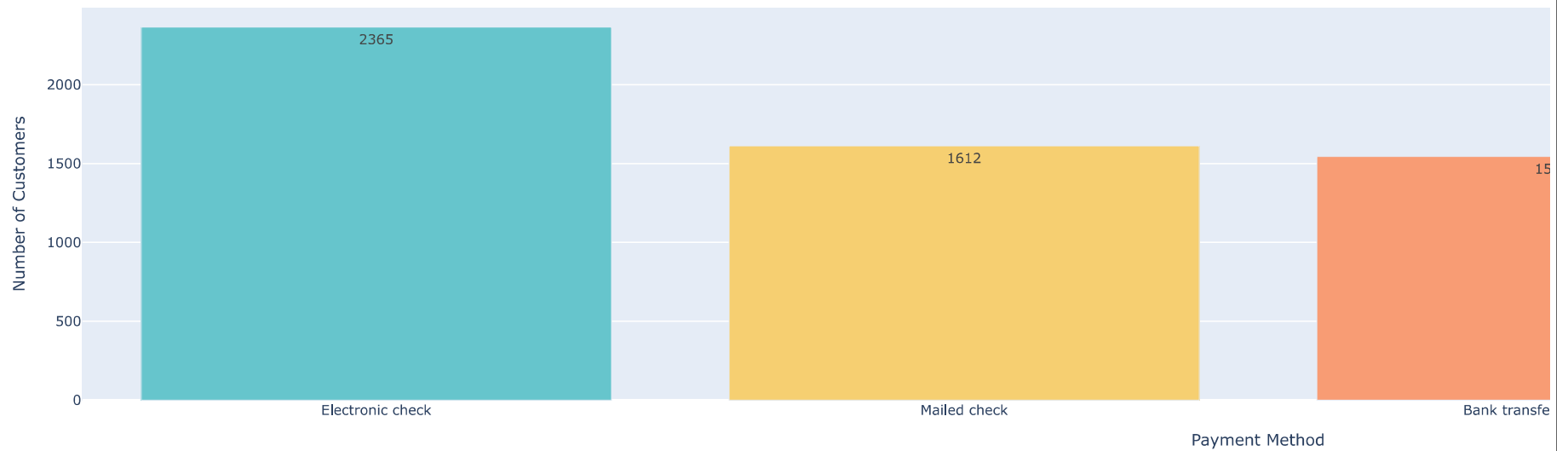
```
fig = px.bar(payment_method_counts,
             x='PaymentMethod',
             y='Count',
             title='Distribution of Payment Methods',
             color='PaymentMethod',
             color_discrete_sequence=px.colors.qualitative.Pastel,
             text='Count',
             labels={'Count': 'Number of Customers'})
```

```
fig.update_layout(
    xaxis_title='Payment Method',
    yaxis_title='Number of Customers',
    showlegend=True
)
```

```
fig.show()
```



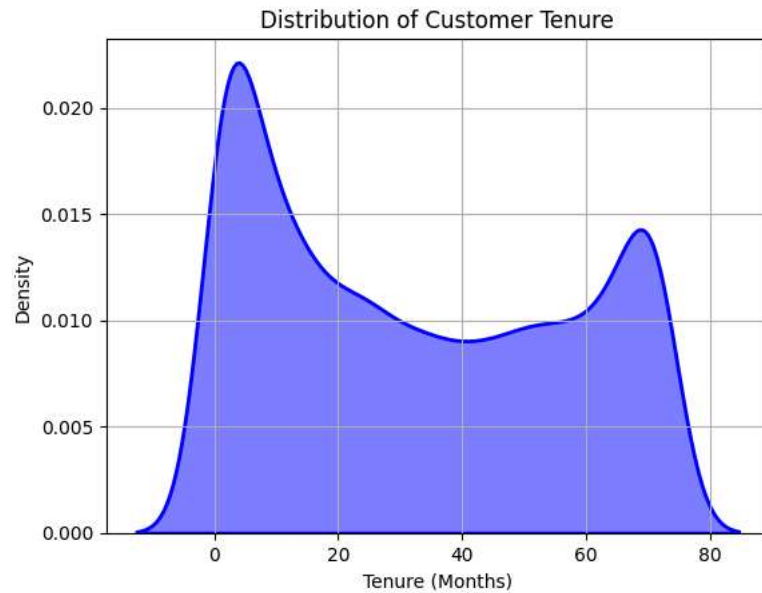
Distribution of Payment Methods



```
sns.kdeplot(data['tenure'],
            color='blue',
            fill=True,
            alpha=0.5,
            linewidth=2)

plt.title('Distribution of Customer Tenure')
plt.xlabel('Tenure (Months)')
plt.ylabel('Density')
plt.grid(True)

plt.show()
```



#### ✓ Matriz de Correlación:

Calculamos la matriz de correlación para identificar posibles relaciones entre los atributos. Esto ayudará a detectar variables redundantes o fuertemente correlacionadas.

```
# Eliminamos la columna innecesario de customerID
data = data.drop(['customerID'], axis = 1)
```





















