RELATÓRIO FINAL

Introdução

Propósito da análise

A evasão de clientes, ou churn, é um dos maiores desafios enfrentados por empresas de telecomunicações, e a Telecom X não é exceção. Recentemente, a empresa tem observado um aumento preocupante na perda de clientes, sem compreender as razões por trás desse comportamento. Diante desse cenário, este projeto foi desenvolvido com o objetivo de realizar uma análise de dados aprofundada para desvendar os fatores que levam os clientes a cancelarem seus serviços.

Utilizando o processo de ETL (Extract, Transform, Load), buscamos extrair, tratar e preparar os dados brutos, transformando-os em informações valiosas. A partir de uma análise exploratória (EDA), com o uso de ferramentas como Python e bibliotecas como Pandas, Seaborn e Matplotlib, procuraremos identificar padrões, tendências e as principais variáveis que impactam a retenção de clientes. O resultado deste trabalho servirá como base para a equipe de Data Science, que poderá, a partir desses insights, desenvolver modelos preditivos eficazes para mitigar a taxa de evasão e, consequentemente, impulsionar o crescimento e a sustentabilidade da Telecom X.

Estrutura do projeto e organização dos arquivos

O projeto se encontra no seguinte link no Github: https://github.com/thauanqs/DESAFIO_TELECOMX

Instruções para executar o notebook

Os requisitos estão listados em "requirements.txt" e podem ser instalados usando o comando: \

```
py -m pip install -r requirements.txt
```

Execução do trabalho

A seguir se dá como foi realizado o desenvolvimento do projeto.

ETL (Extração, Transformação e Carga)

Importação dos dados:

```
import pandas as pd
url = 'https://raw.githubusercontent.com/ingridcristh/challenge2-data-
science/refs/heads/main/TelecomX_Data.json'
dados = pd.read_json(url)
```

Transformação: Explorando as colunas do dataset e verificar seus tipos de dados:

```
dados.shape # exibe o numero de linhas/ colunas
dados.columns #exibe as colunas
dados.info() #exibe o tipo de dado de cada coluna
```

dados.info() retorna:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7267 entries, 0 to 7266
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   customerID  7267 non-null   object
 1   Churn       7267 non-null   object
 2   customer    7267 non-null   object
 3   phone       7267 non-null   object
 4   internet    7267 non-null   object
 5   account     7267 non-null   object
dtypes: object(6)
memory usage: 340.8+ KB
```

Temos o nome de cada coluna, a quantidade de dados não nulos existente na coluna em questão e o tipo de dado. Observamos que as 6 colunas possuem 7267 dados não nulos cada e que ambos os dados são do tipo “object”.

Usando o método describe() para calcular algumas estatísticas básicas dos dados.

	customerID	Churn	customer	phone	internet	account
count	7267	7267	7267	7267	7267	7267
unique	7267	3	891	3	129	6931
top	0002-ORFBO	No	{'gender': 'Male', 'SeniorCitizen': 0, 'Partne...	{'PhoneService': 'Yes', 'MultipleLines': 'No'}	{'InternetService': 'No', 'OnlineSecurity': 'N...	{'Contract': 'Month-to-month', 'PaperlessBilli...
freq	1	5174	223	3495	1581	6

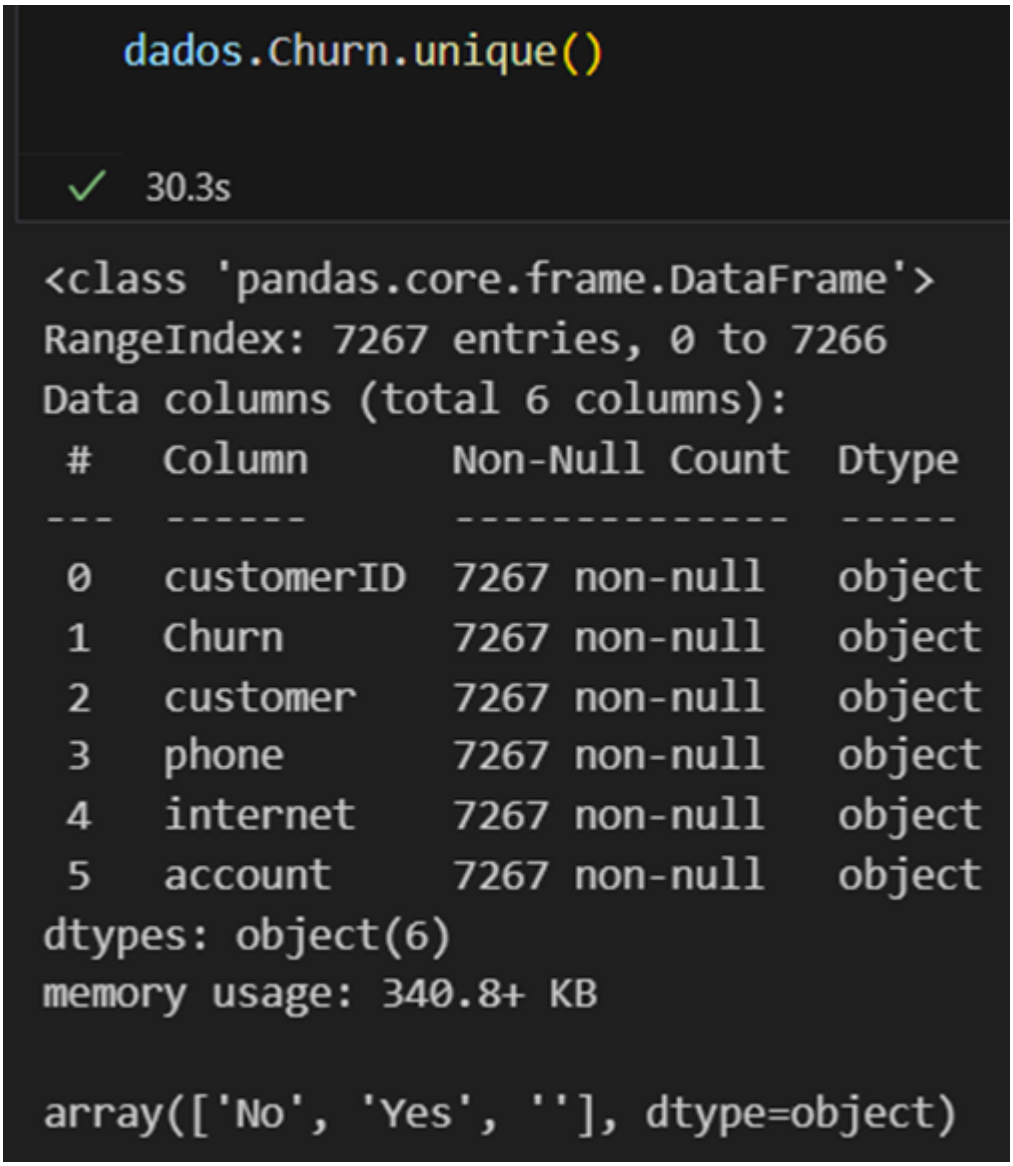
As colunas e seus subitens mais relevantes para análise são:

['Churn', 'customer','phone','internet','account']

Tratamento: Limpeza e transformação de dados

Verificando Inconsistências nos Dados:

```
dados.Churn.unique()
```



Observamos a ocorrência de campo vazio ‘’ em Churn. Realizando a seleção “dados.query('"" in Churn')” foi observado 224 ocorrências de campo vazio em Churn. Esses dados serão excluídos das análises por não serem válidos. Porém antes serão salvos (dados_churn_vazio = dados.query('"" in Churn'), para uma possível

verificação (suposta) seja feita visando entender por que esses 224 casos estão com esse campo vazio.

```
dados.query('"" in Churn')
```

✓ 0.0s

	customerID	Churn	customer	phone	
30	0047-ZHDTW		{'gender': 'Female', 'SeniorCitizen': 0, 'Part...	{'PhoneService': 'Yes', 'MultipleLines': 'Yes'}	{'InternetService': 'Fib 'Onl
75	0120-YZLQA		{'gender': 'Male', 'SeniorCitizen': 0, 'Partne...	{'PhoneService': 'Yes', 'MultipleLines': 'No'}	{'InternetService': 'No', 'Online
96	0154-QYHJU		{'gender': 'Male', 'SeniorCitizen': 0, 'Partne...	{'PhoneService': 'Yes', 'MultipleLines': 'No'}	{'InternetServi 'OnlineSe
98	0162-RZGMZ		{'gender': 'Female', 'SeniorCitizen': 1, 'Part...	{'PhoneService': 'Yes', 'MultipleLines': 'No'}	{'InternetServi 'OnlineSe
175	0274-VVQOQ		{'gender': 'Male', 'SeniorCitizen': 1, 'Partne...	{'PhoneService': 'Yes', 'MultipleLines': 'Yes'}	{'InternetService': 'Fib 'Onl
...	...	...	...	...	
7158	9840-GSRFX		{'gender': 'Female', 'SeniorCitizen': 0, 'Part...	{'PhoneService': 'Yes', 'MultipleLines': 'Yes'}	{'InternetServi 'OnlineSe
7180	9872-RZQQB		{'gender': 'Female', 'SeniorCitizen': 0, 'Part...	{'PhoneService': 'No', 'MultipleLines': 'No ph...	{'InternetServi 'OnlineSe
7211	9920-GNDMB		{'gender': 'Male', 'SeniorCitizen': 0, 'Partne...	{'PhoneService': 'Yes', 'MultipleLines': 'Yes'}	{'InternetService': 'Fib 'Onl
7239	9955-RVWSC		{'gender': 'Female', 'SeniorCitizen': 0, 'Part...	{'PhoneService': 'Yes', 'MultipleLines': 'No'}	{'InternetService': 'No', 'Online
7247	9966-VYRTZ		{'gender': 'Female', 'SeniorCitizen': 0, 'Part...	{'PhoneService': 'Yes', 'MultipleLines': 'No'}	{'InternetService': 'No', 'Online

224 rows × 6 columns

Dados possuí 7267 linhas. Subtraindo 224 linhas, temos 7043. Obtemos o dataframe atualizado com 7043 linhas:

```
dados = dados.query('"" not in Churn')
dados
```

✓ 0.0s

	customerID	Churn	customer	
0	0002-ORFBO	No	{'gender': 'Female', 'SeniorCitizen': 0, 'Part...	{'PhoneServi
1	0003-MKNFE	No	{'gender': 'Male', 'SeniorCitizen': 0, 'Partne...	{'PhoneServi
2	0004-TLHLJ	Yes	{'gender': 'Male', 'SeniorCitizen': 0, 'Partne...	{'PhoneServi
3	0011-IGKFF	Yes	{'gender': 'Male', 'SeniorCitizen': 1, 'Partne...	{'PhoneServi
4	0013-EXCHZ	Yes	{'gender': 'Female', 'SeniorCitizen': 1, 'Part...	{'PhoneServi
...	...	...	...	
7262	9987-LUTYD	No	{'gender': 'Female', 'SeniorCitizen': 0, 'Part...	{'PhoneServi
7263	9992-RRAMN	Yes	{'gender': 'Male', 'SeniorCitizen': 0, 'Partne...	{'PhoneServi
7264	9992-UJOEL	No	{'gender': 'Male', 'SeniorCitizen': 0, 'Partne...	{'PhoneServi
7265	9993-LHIEB	No	{'gender': 'Male', 'SeniorCitizen': 0, 'Partne...	{'PhoneServi
7266	9995-HOTOH	No	{'gender': 'Male', 'SeniorCitizen': 0, 'Partne...	{'PhoneServi

7043 rows × 6 columns

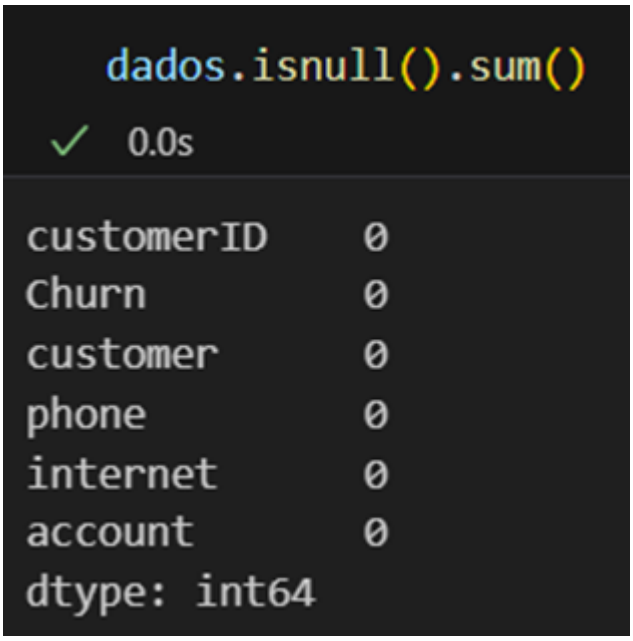
Testamos novamente e a gora sim obtemos apenas No ou Yes no campo Churn, o que é esperado:

```
dados.Churn.unique()
```

✓ 0.0s

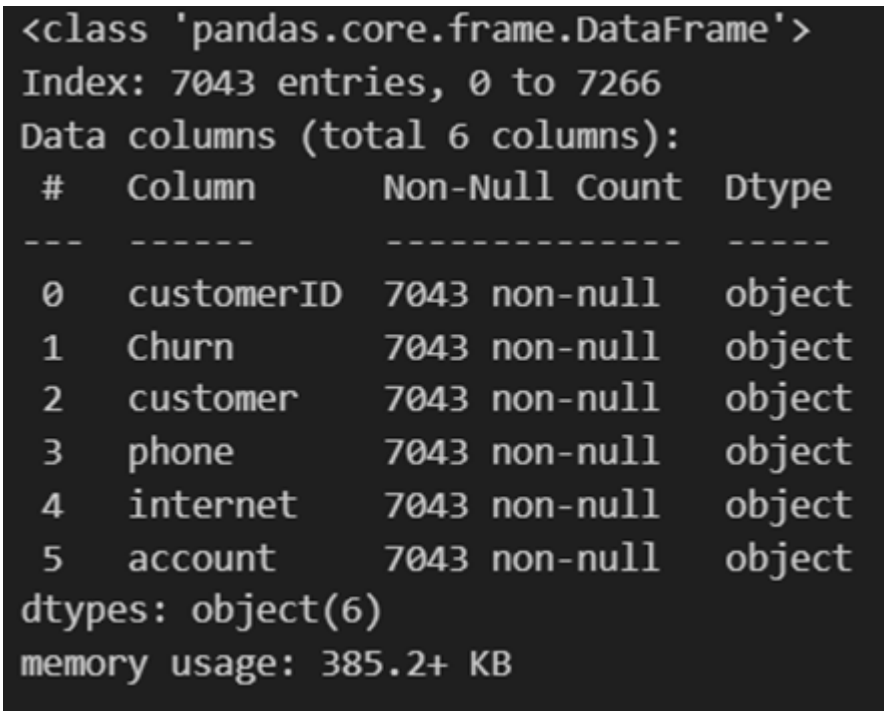
```
array(['No', 'Yes'], dtype=object)
```

```
dados.isnull().sum()
```



Verificando se o campo customerID possui valores duplicados:

```
len(pd.unique(dados["customerID"]))
```



Separando dados em colunas e criando coluna de "Contas Diárias"

```
dados["Monthly"] = dados["account"].apply(lambda x: x["Charges"]["Monthly"])
dados["Contas_Diarias"] = dados["Monthly"] / 30
```

Fazemos o mesmo com as outras colunas:

```
#explodindo cada coluna em um dataframe
df_customer = pd.json_normalize(dados['customer'])
df_phone = pd.json_normalize(dados['phone'])
df_internet = pd.json_normalize(dados['internet'])
df_account = pd.json_normalize(dados['account'])

# Concatenar tudo no DataFrame principal

dados = pd.concat([dados.drop(columns=["customer", "phone", "internet",
"account"]), df_customer, df_phone, df_internet, df_account], axis=1)

dados.columns
```

Obtemos uma tabela com as seguintes colunas:

```
Index(['customerID', 'Churn', 'Monthly', 'Contas_Diarias', 'gender',
       'SeniorCitizen', 'Partner', 'Dependents', 'tenure', 'PhoneService',
       'MultipleLines', 'InternetService', 'OnlineSecurity', 'OnlineBackup',
       'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies',
       'Contract', 'PaperlessBilling', 'PaymentMethod', 'Charges.Monthly',
       'Charges.Total'],
      dtype='object')
```

Clientes evadidos: São salvos no seguinte dataframe:

```
# clientes que saíram (Churn = "Yes")
dados_churn = dados[dados["Churn"] == "Yes"].copy()
```

Clientes não evadidos: São salvos no seguinte dataframe:

```
# clientes que ficaram (Churn = "No")
dados_sem_churn = dados[dados["Churn"] == "No"].copy()
```

Análise Exploratória de Dados

Média, mediana, desvio padrão

Total dos casos

```
dados.describe()
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	Monthly	Contas_Diarias	SeniorCitizen	tenure	Charges.Monthly
count	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000
mean	64.761692	2.158723	0.162147	32.371149	64.761692
std	30.090047	1.003002	0.368612	24.559481	30.090047
min	18.250000	0.608333	0.000000	0.000000	18.250000
25%	35.500000	1.183333	0.000000	9.000000	35.500000
50%	70.350000	2.345000	0.000000	29.000000	70.350000
75%	89.850000	2.995000	0.000000	55.000000	89.850000
max	118.750000	3.958333	1.000000	72.000000	118.750000

Obtendo valores totais e percentuais:

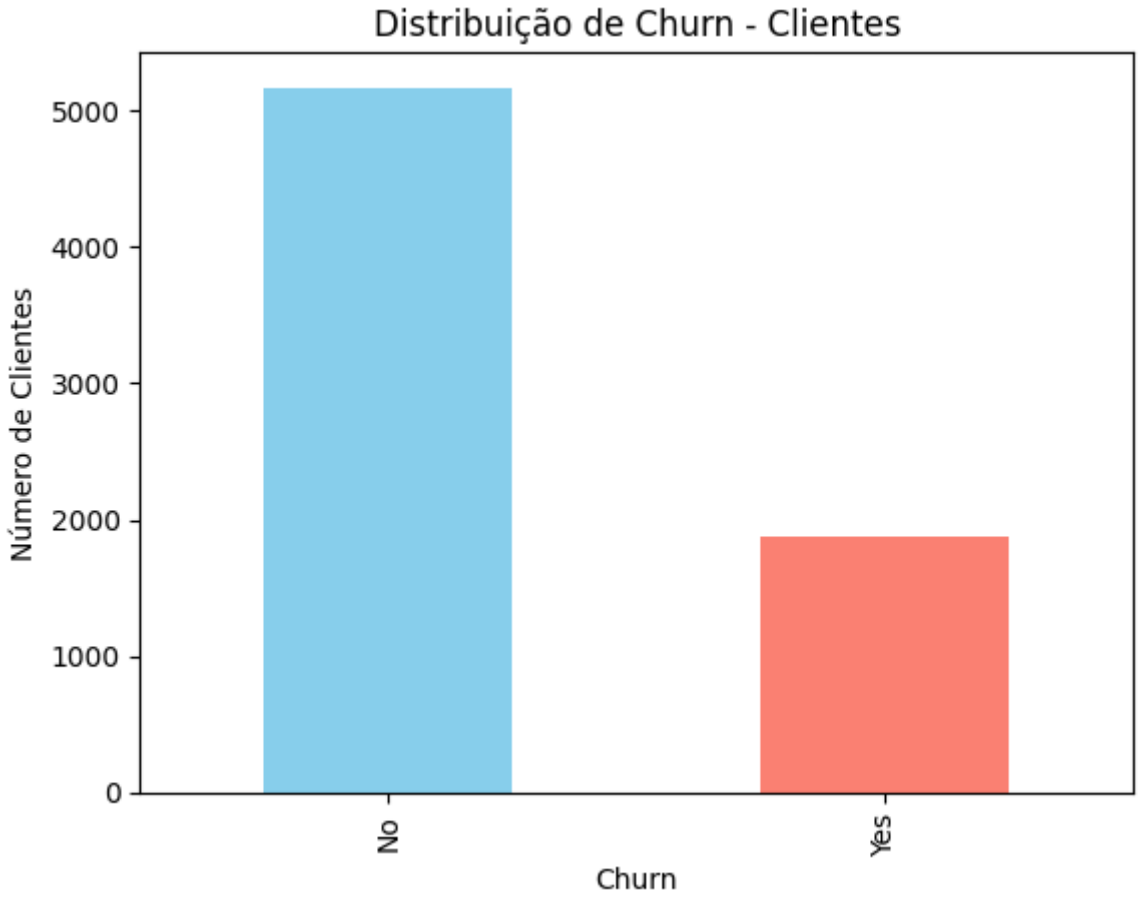
```
print(dados["Churn"].value_counts())
print(dados["Churn"].value_counts(normalize=True) * 100)
```

```
Churn
No     5174
Yes    1869
Name: count, dtype: int64
Churn
No     73.463013
Yes    26.536987
Name: proportion, dtype: float64
```

Total com churn: 1869
Total sem churn: 5174

Obtemos um gráfico que melhor exibe essa porcentagem:

```
import matplotlib.pyplot as plt

dados["Churn"].value_counts().plot(kind="bar", color=["skyblue", "salmon"])
plt.title("Distribuição de Churn - Clientes")
plt.xlabel("Churn")
plt.ylabel("Número de Clientes")
plt.show()
```



Casos onde houve churn

```
# clientes que saíram (Churn = "Yes")
dados_churn = dados[dados["Churn"] == "Yes"].copy()
dados_churn.describe()
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	Monthly	Contas_Diarias	SeniorCitizen	tenure	Charges.Monthly
count	1869.000000	1869.000000	1821.000000	1821.000000	1821.000000
mean	74.441332	2.481378	0.155958	31.833059	64.821170
std	24.666053	0.822202	0.362915	24.656511	30.518745
min	18.850000	0.628333	0.000000	0.000000	18.700000
25%	56.150000	1.871667	0.000000	8.000000	33.750000
50%	79.650000	2.655000	0.000000	27.000000	70.450000
75%	94.200000	3.140000	0.000000	55.000000	90.600000
max	118.350000	3.945000	1.000000	72.000000	118.600000

Casos onde não houve churn

```
# clientes que ficaram (Churn = "No")
dados_sem_churn = dados[dados["Churn"] == "No"].copy()
dados_sem_churn.describe()
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

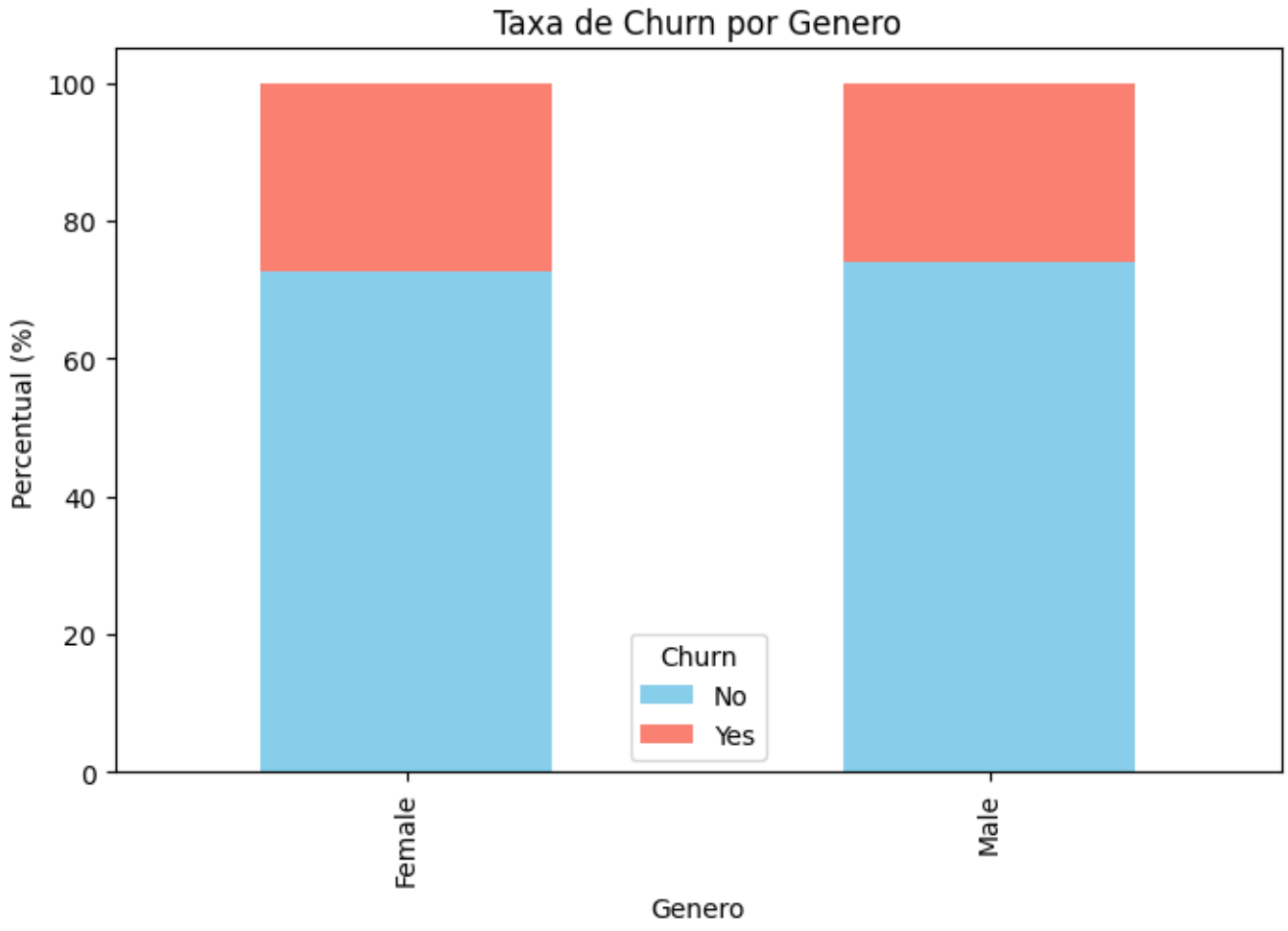
	Monthly	Contas_Diarias	SeniorCitizen	tenure	Charges.Monthly
count	5174.000000	5174.000000	5009.000000	5009.000000	5009.000000
mean	61.265124	2.042171	0.164105	32.687163	64.837832
std	31.092648	1.036422	0.370407	24.517650	29.962423
min	18.250000	0.608333	0.000000	0.000000	18.250000
25%	25.100000	0.836667	0.000000	9.000000	35.800000
50%	64.425000	2.147500	0.000000	29.000000	70.350000
75%	88.400000	2.946667	0.000000	56.000000	89.650000

	Monthly	Contas_Diarias	SeniorCitizen	tenure	Charges.Monthly
max	118.750000	3.958333	1.000000	72.000000	118.750000

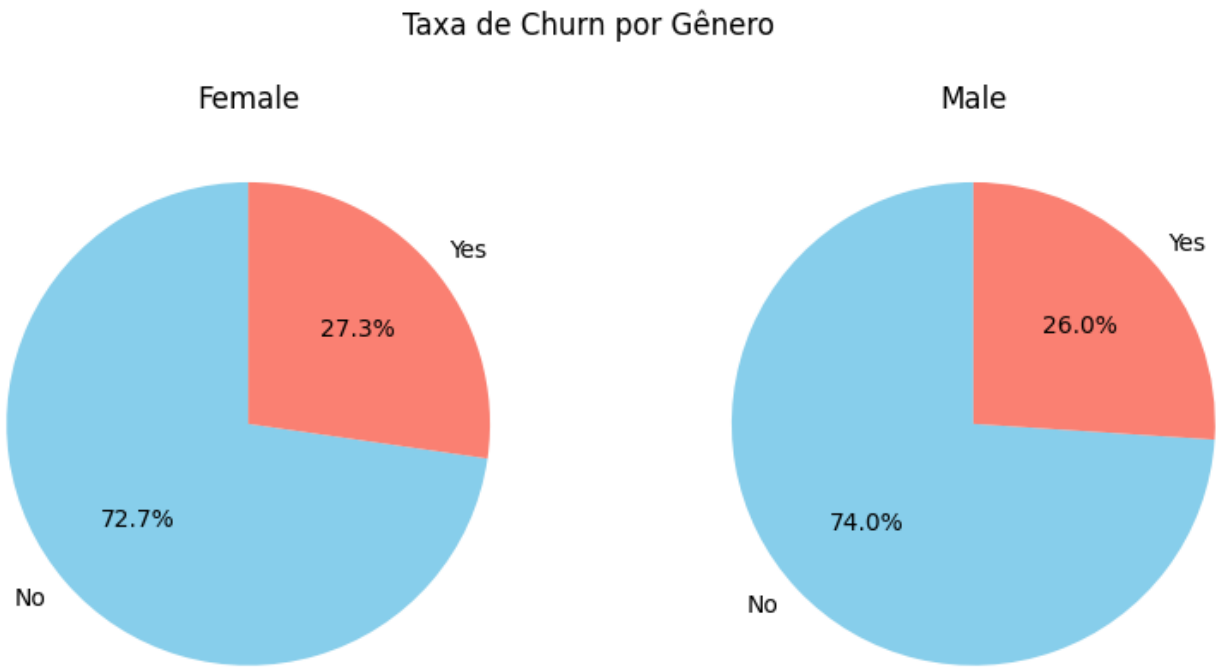
Contagem de Evasão por Variáveis Categóricas

Por gênero

```
taxa_churn_genero = pd.crosstab(dados["gender"], dados2["Churn"],
normalize="index") * 100
taxa_churn_genero


taxa_churn_genero.plot(
    kind="bar", stacked=True, color=["skyblue", "salmon"], figsize=(8,5)
)
plt.title("Taxa de Churn por Genero")
plt.ylabel("Percentual (%)")
plt.xlabel("Genero")
plt.legend(title="Churn")
plt.show()
```



```
fig, axes = plt.subplots(1, len(taxa_churn_genero), figsize=(10,5))

for i, genero in enumerate(taxa_churn_genero.index):
    taxa_churn_genero.loc[genero].plot(
        kind="pie",
        autopct="%.1f%%",
        startangle=90,
        colors=["skyblue", "salmon"],
        ax=axes[i]
    )
    axes[i].set_ylabel("")
    axes[i].set_title(f"{genero}")

plt.suptitle("Taxa de Churn por Gênero")
plt.show()
```



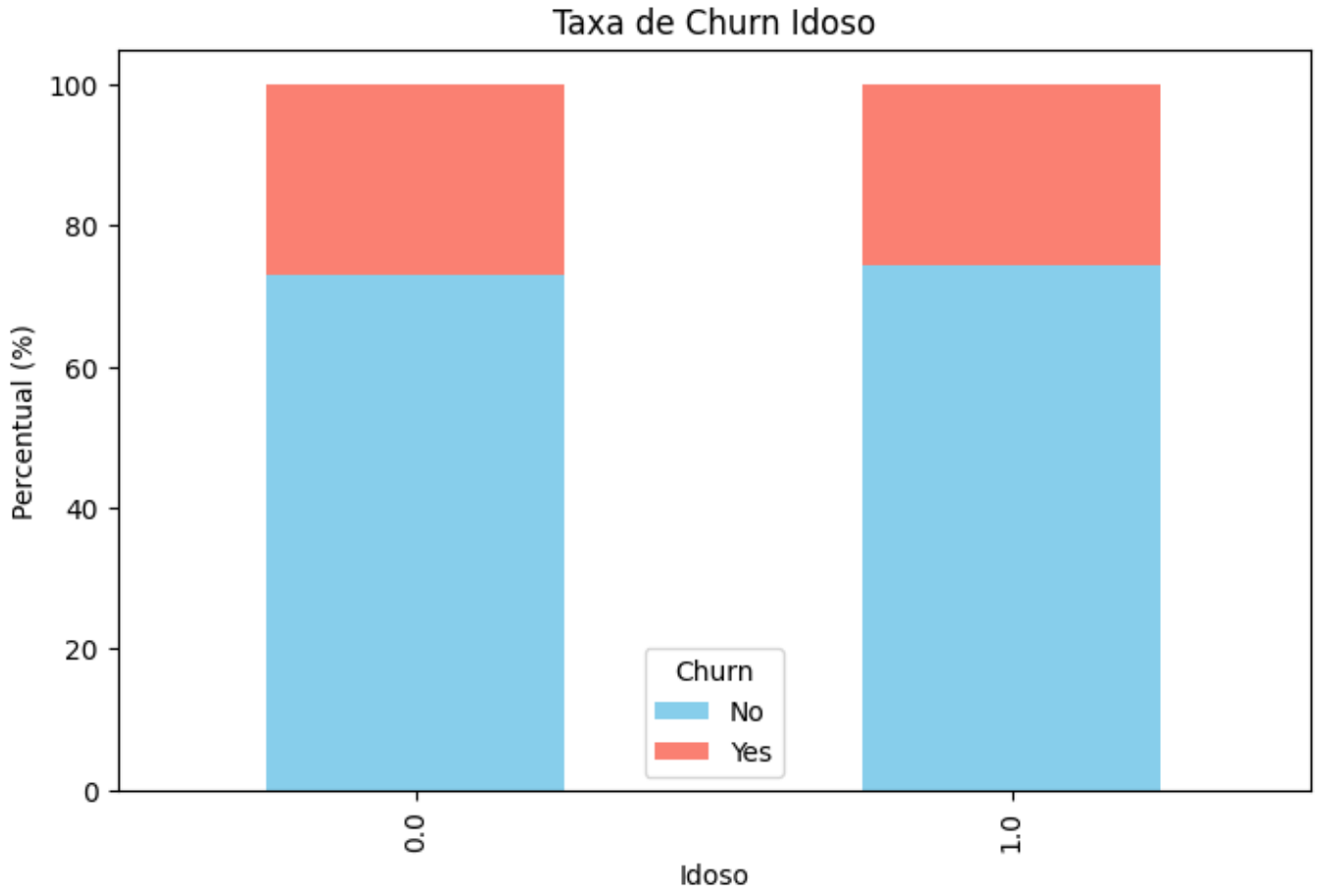

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

Churn	No	Yes
gender		

Churn	No	Yes
gender		
Female	72.708518	27.291482
Male	73.959849	26.040151

Por idade (Idoso)

```
taxa_churn_idoso = pd.crosstab(dados["SeniorCitizen"], dados["Churn"],
normalize="index") * 100
taxa_churn_idoso


taxa_churn_idoso.plot(
    kind="bar", stacked=True, color=["skyblue", "salmon"], figsize=(8,5)
)
plt.title("Taxa de Churn Idoso")
plt.ylabel("Percentual (%)")
plt.xlabel("Idoso")
plt.legend(title="Churn")
plt.show()

taxa_churn_idoso
```

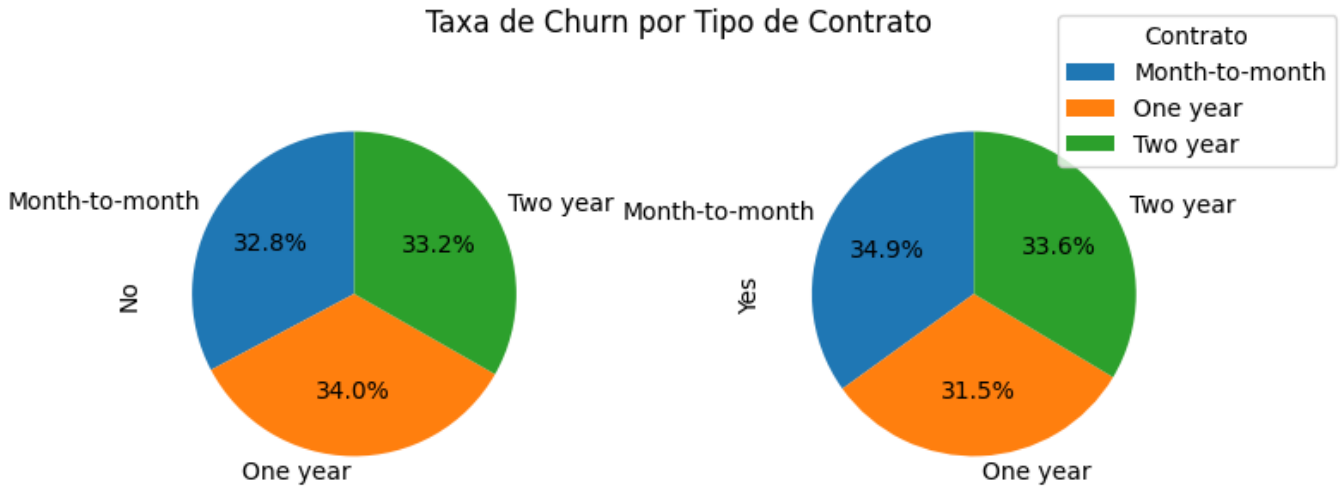


```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

Churn	No	Yes
SeniorCitizen		
0.0	73.148148	26.851852
1.0	74.321881	25.678119

Por contrato

```
taxa_churn_contrato = pd.crosstab(dados["Contract"], dados["Churn"],
normalize="index") * 100
taxa_churn_contrato

taxa_churn_contrato.plot(
    kind="pie", stacked=True, color=["skyblue", "salmon"], figsize=(8,5),
subplots=True, autopct="%.1f%%", startangle=90, legend=False
)
plt.suptitle("Taxa de Churn por Tipo de Contrato", y = 0.8)
#plt.ylabel("Percentual (%)")
#plt.xlabel("Contrato")
plt.legend(title="Contrato", bbox_to_anchor = (1.05, 1), loc = "center")


plt.tight_layout()

plt.show()
```

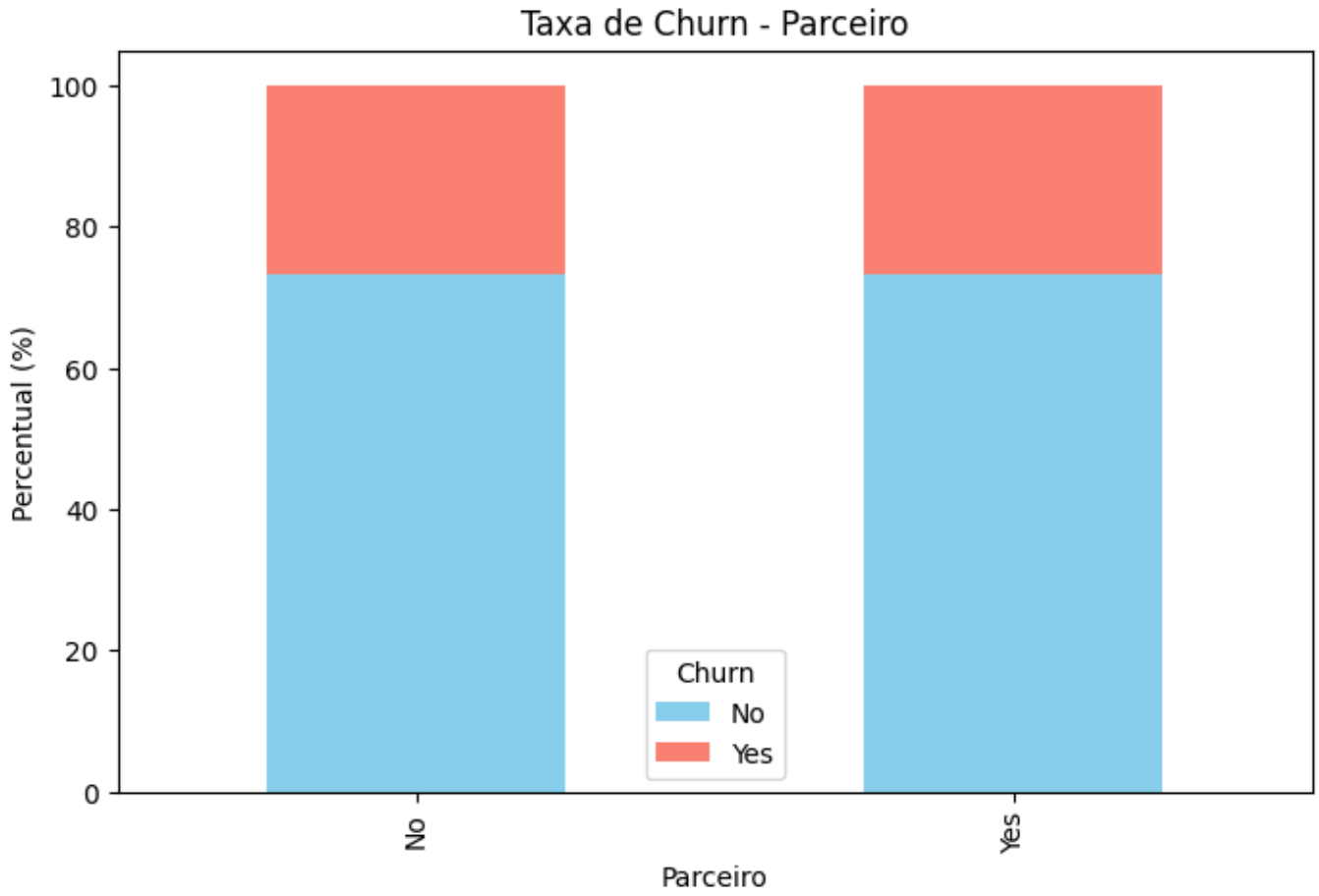



```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

Churn	No	Yes
Contract		
Month-to-month	72.513996	27.486004
One year	75.227432	24.772568
Two year	73.575758	26.424242

Por parceiro

```
taxa_churn_parceiro = pd.crosstab(dados["Partner"], dados["Churn"],
normalize="index") * 100

taxa_churn_parceiro.plot(
    kind="bar", stacked=True, color=["skyblue", "salmon"], figsize=(8,5)
)
plt.title("Taxa de Churn - Parceiro")
plt.ylabel("Percentual (%)")
plt.xlabel("Parceiro")
plt.legend(title="Churn")
plt.show()
```



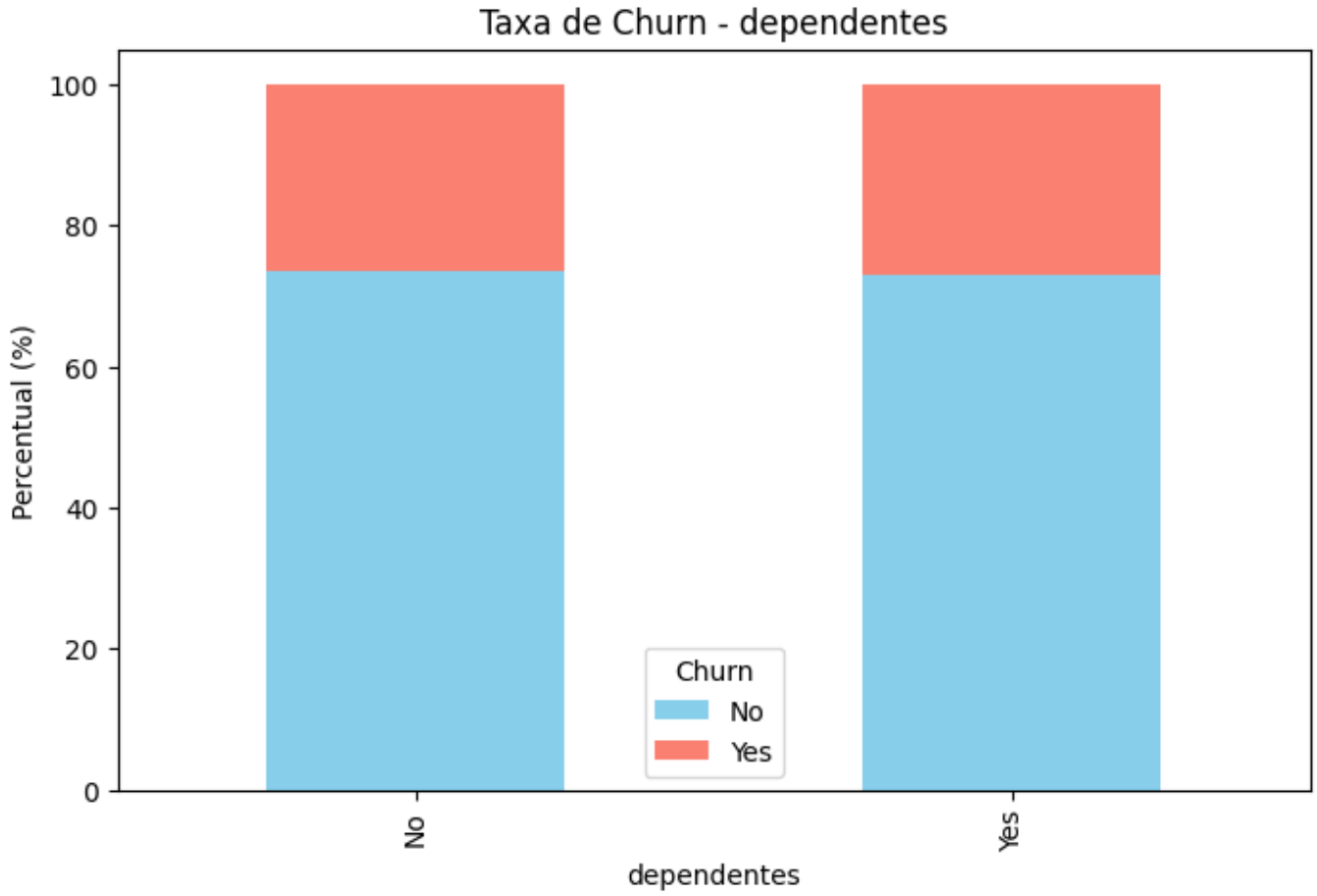

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

Churn	No	Yes
Partner		
No	73.333333	26.666667
Yes	73.343419	26.656581

Por dependentes

```
taxa_churn_dependentes = pd.crosstab(dados["Dependents"], dados["Churn"],
normalize="index") * 100

taxa_churn_dependentes.plot(
    kind="bar", stacked=True, color=["skyblue", "salmon"], figsize=(8,5)
)
```

```
plt.title("Taxa de Churn - dependentes")
plt.ylabel("Percentual (%)")
plt.xlabel("dependentes")
plt.legend(title="Churn")
plt.show()
```

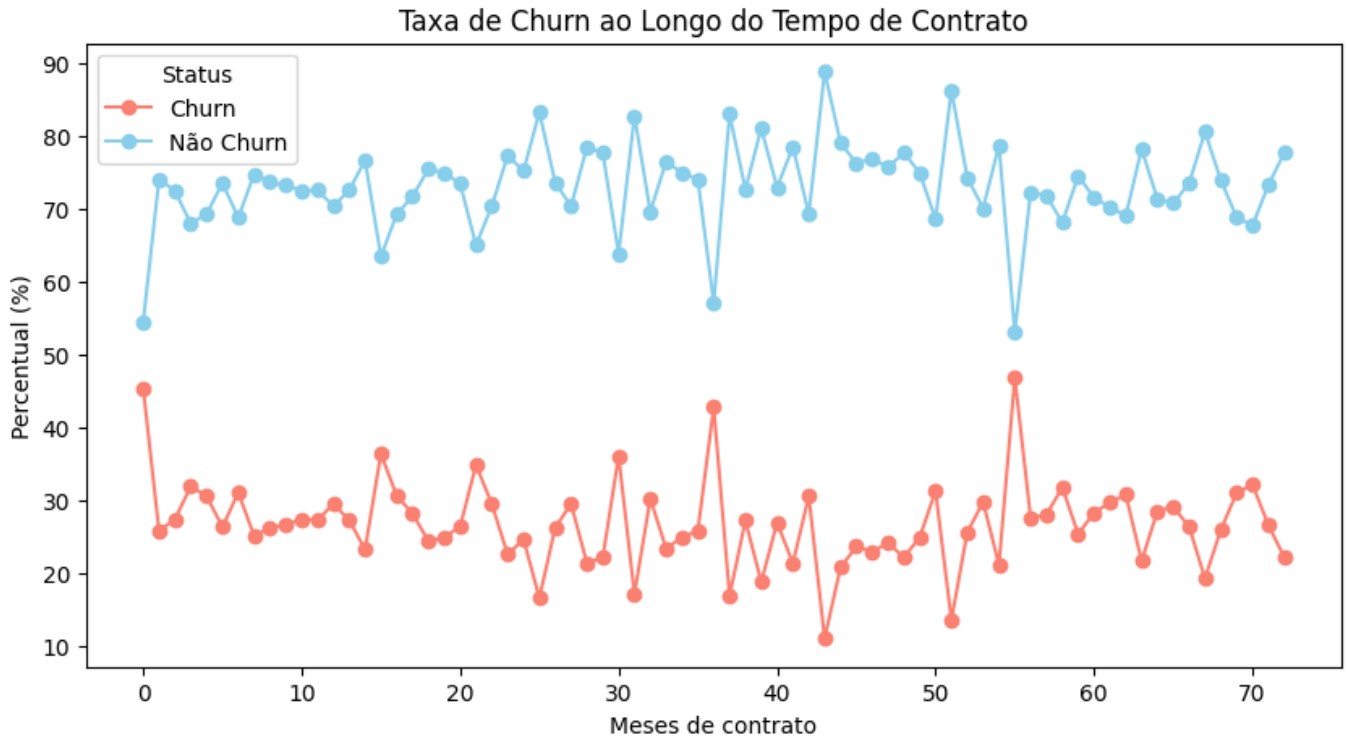



```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

Churn	No	Yes
Dependents		
No	73.544198	26.455802
Yes	72.859922	27.140078

Por meses de contrato

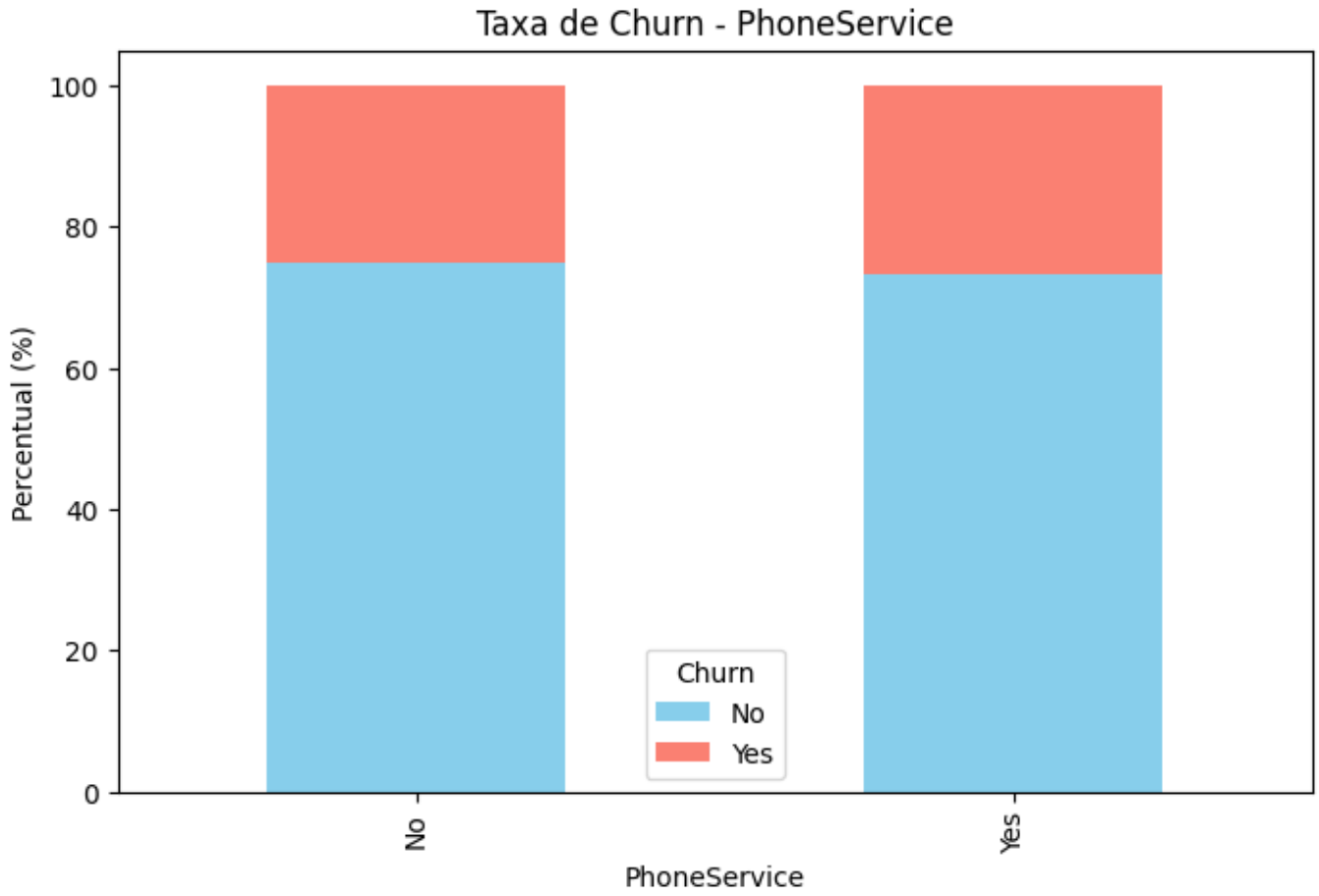
```
taxa_churn_tenure = pd.crosstab(dados["tenure"], dados["Churn"],
normalize="index") * 100

plt.figure(figsize=(10,5))

plt.plot(
    taxa_churn_tenure.index,
    taxa_churn_tenure["Yes"],
    marker="o", color="salmon", label="Churn"
)

plt.plot(
    taxa_churn_tenure.index,
    taxa_churn_tenure["No"],
    marker="o", color="skyblue", label="Não Churn"
)

plt.title("Taxa de Churn ao Longo do Tempo de Contrato")
plt.ylabel("Percentual (%)")
plt.xlabel("Meses de contrato")
plt.legend(title="Status")
plt.show()
```



Serviço de telefone

```
taxa_churn_PhoneService = pd.crosstab(dados["PhoneService"], dados["Churn"],
normalize="index") * 100
```

```
taxa_churn_PhoneService.plot(
    kind="bar", stacked=True, color=["skyblue", "salmon"], figsize=(8,5)
)
plt.title("Taxa de Churn - PhoneService")
plt.ylabel("Percentual (%)")
plt.xlabel("PhoneService")
plt.legend(title="Churn")
plt.show()
```

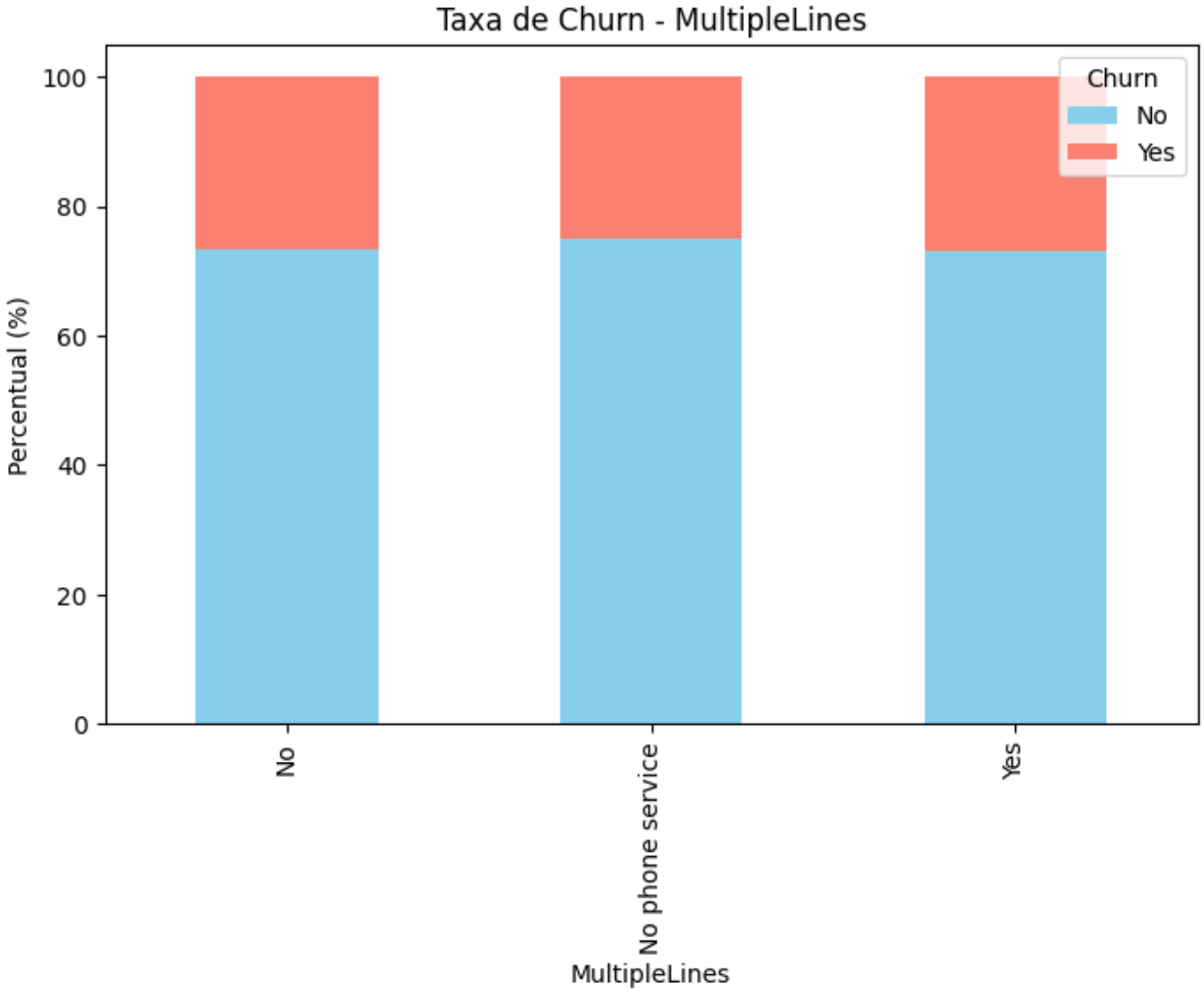



```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

Churn	No	Yes
PhoneService		
No	74.961598	25.038402
Yes	73.167179	26.832821

Multiplas linhas

```
taxa_churn_MultipleLines = pd.crosstab(dados["MultipleLines"], dados["Churn"],
normalize="index") * 100

taxa_churn_MultipleLines.plot(
    kind="bar", stacked=True, color=["skyblue", "salmon"], figsize=(8,5)
)
plt.title("Taxa de Churn - MultipleLines")
plt.ylabel("Percentual (%)")
plt.xlabel("MultipleLines")
plt.legend(title="Churn")
plt.show()
```



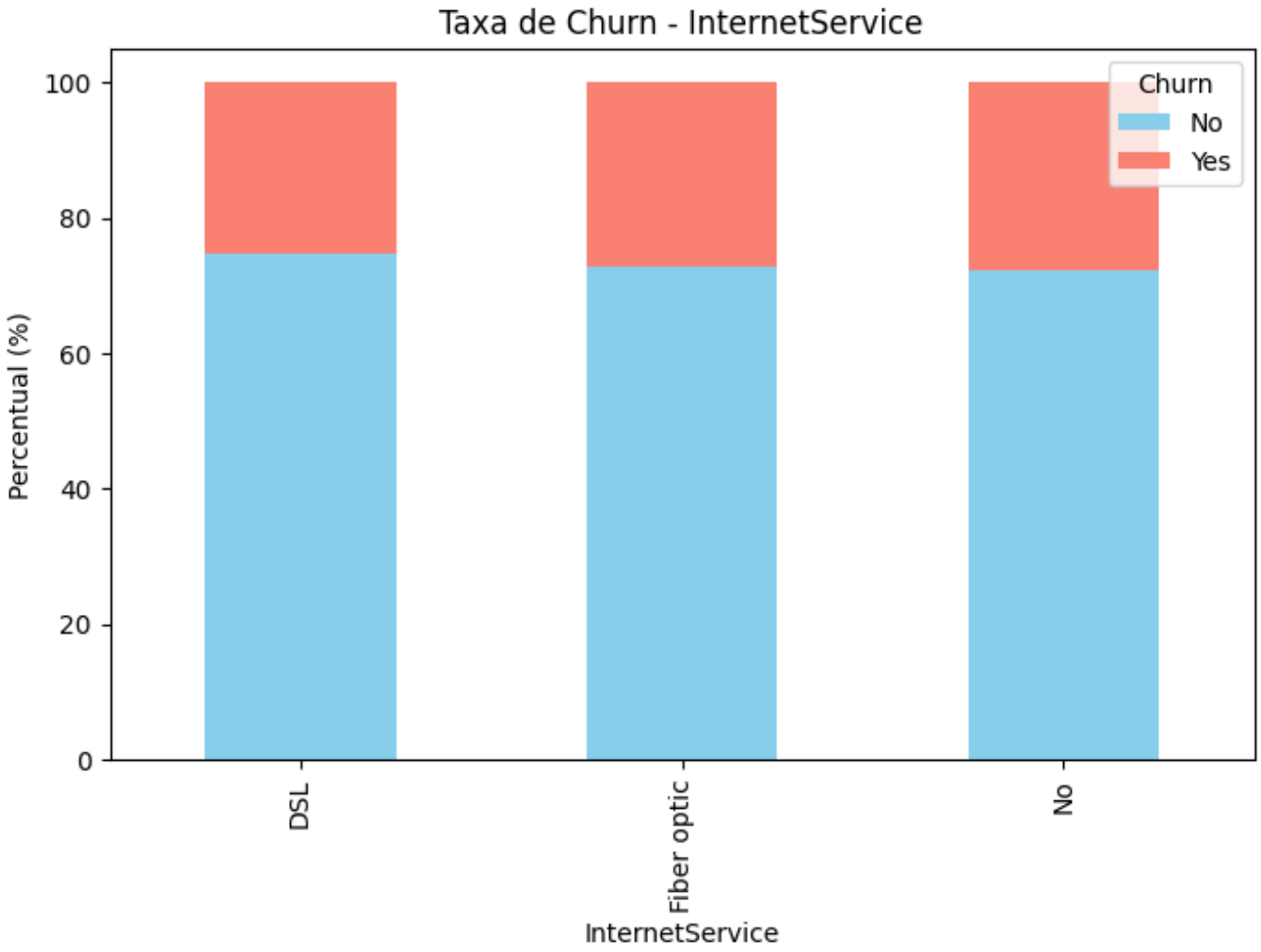

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
```

```
    text-align: right;
}
```

Churn	No	Yes
MultipleLines		
No	73.329283	26.670717
No phone service	74.961598	25.038402
Yes	72.982335	27.017665

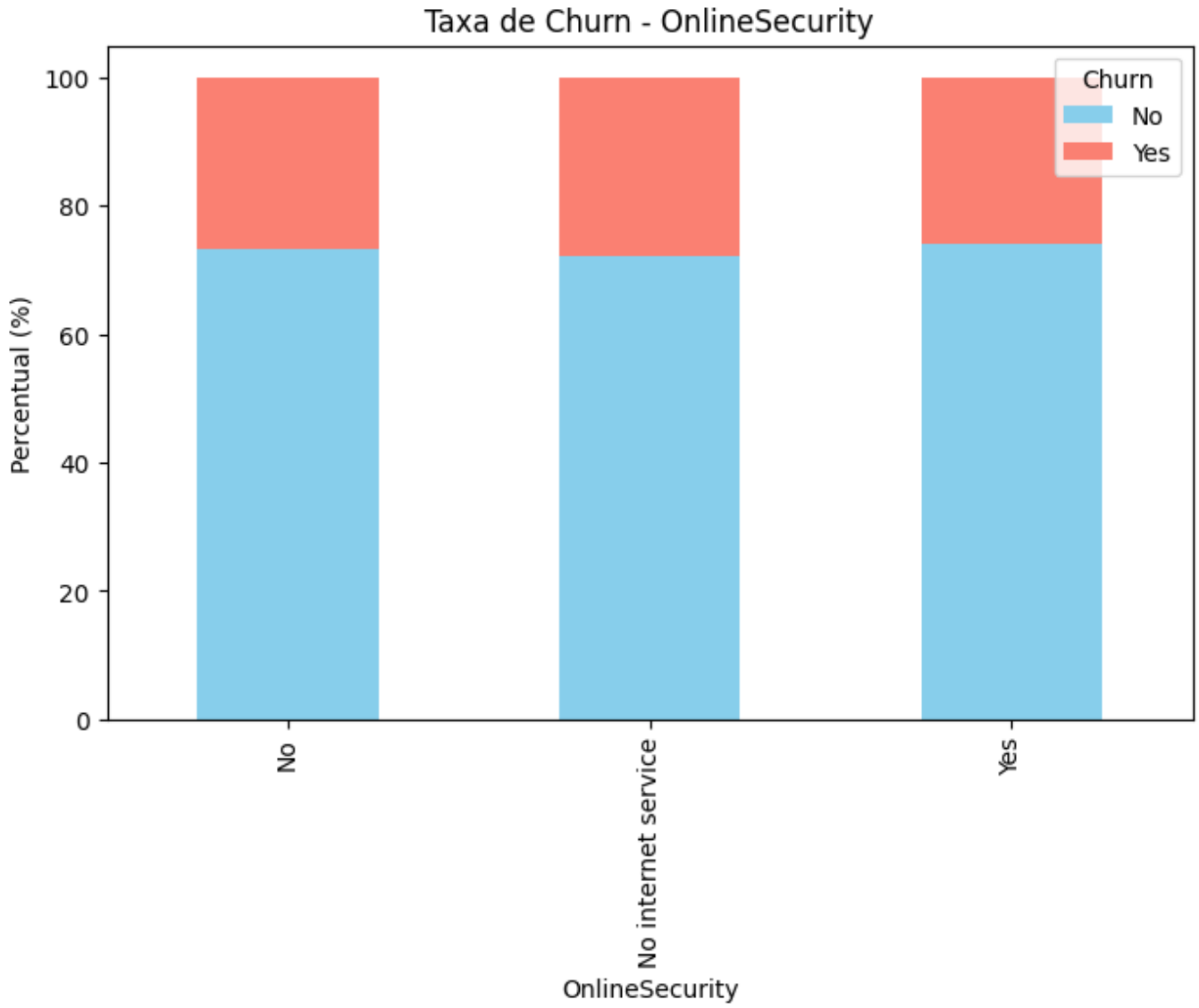
Serviço de internet

```
taxa_churn_InternetService = pd.crosstab(dados["InternetService"], dados["Churn"],
normalize="index") * 100

taxa_churn_InternetService.plot(
    kind="bar", stacked=True, color=["skyblue", "salmon"], figsize=(8,5)
)
plt.title("Taxa de Churn - InternetService")
plt.ylabel("Percentual (%)")
plt.xlabel("InternetService")
plt.legend(title="Churn")
plt.show()
```



Segurança online

```
taxa_churn_OnlineSecurity = pd.crosstab(dados["OnlineSecurity"], dados["Churn"],
normalize="index") * 100

taxa_churn_OnlineSecurity.plot(
    kind="bar", stacked=True, color=["skyblue", "salmon"], figsize=(8,5)
)
plt.title("Taxa de Churn - OnlineSecurity")
plt.ylabel("Percentual (%)")
plt.xlabel("OnlineSecurity")
plt.legend(title="Churn")
plt.show()
```

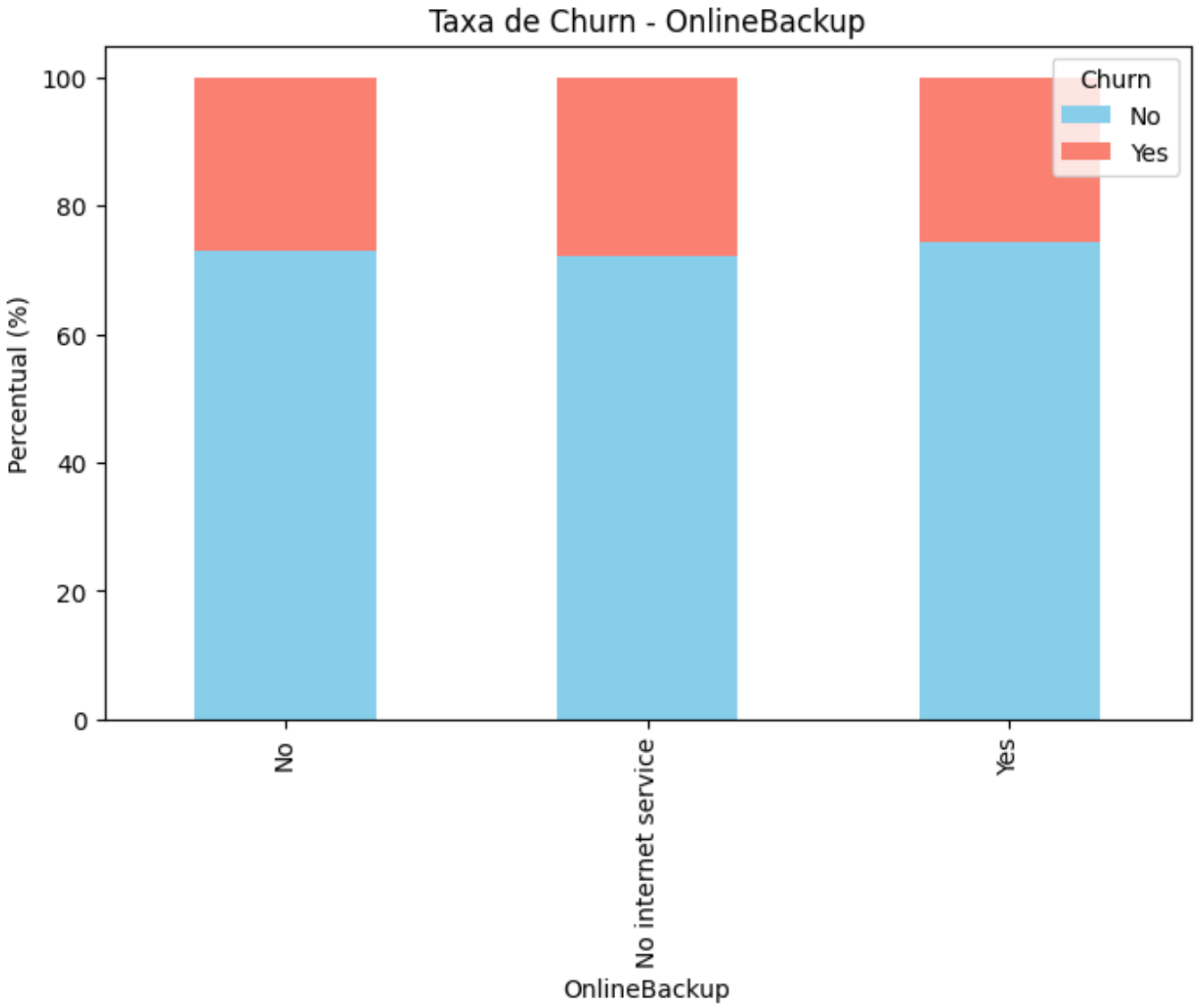



```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

Churn	No	Yes
OnlineSecurity		
No	73.343152	26.656848
No internet service	72.180959	27.819041
Yes	74.206755	25.793245

Backup online

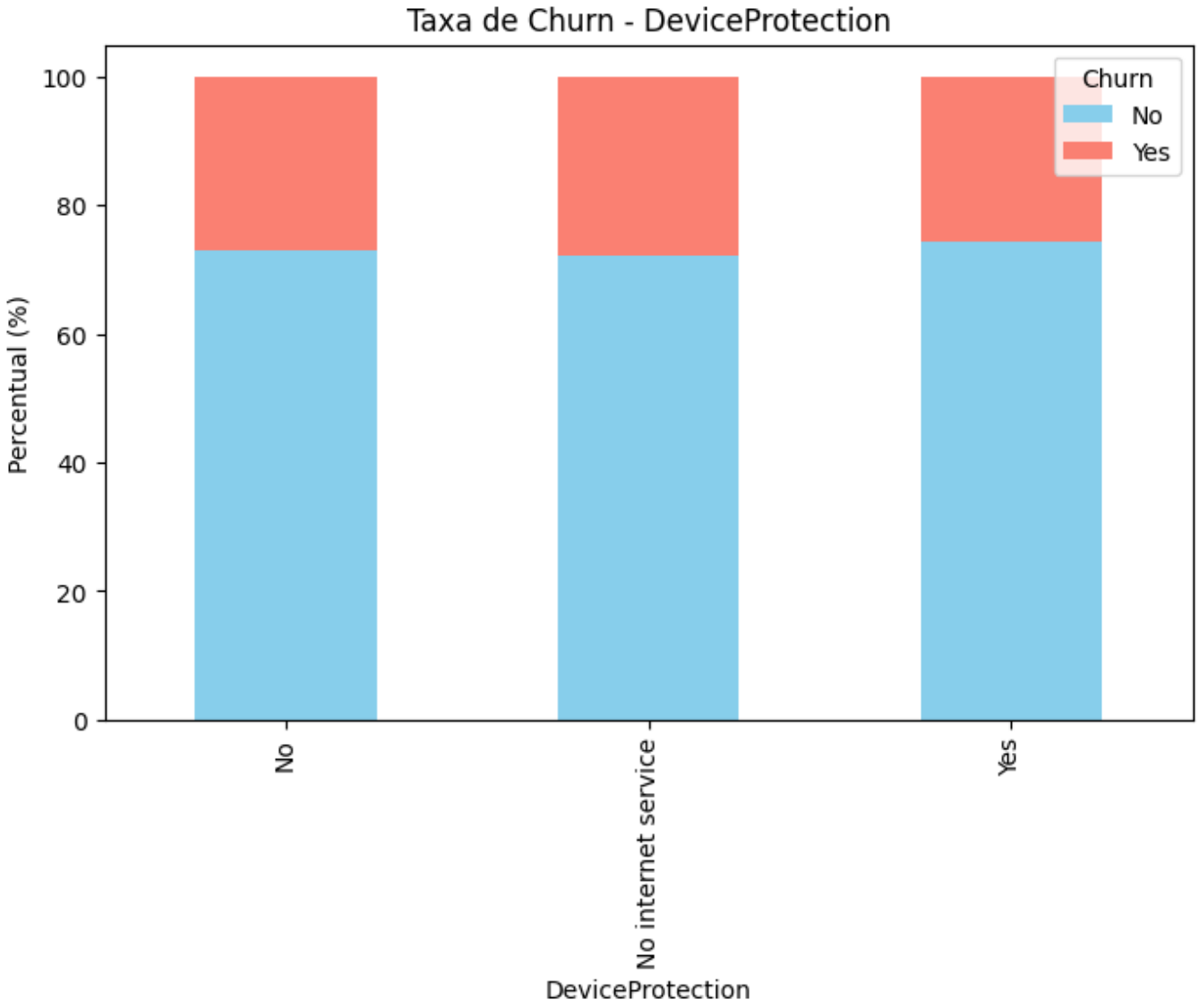
```
taxa_churn_OnlineBackup = pd.crosstab(dados["OnlineBackup"], dados["Churn"],
normalize="index") * 100

taxa_churn_OnlineBackup.plot(
    kind="bar", stacked=True, color=["skyblue", "salmon"], figsize=(8,5)
)
plt.title("Taxa de Churn - OnlineBackup")
plt.ylabel("Percentual (%)")
plt.xlabel("OnlineBackup")
plt.legend(title="Churn")
plt.show()
```



Proteção de dispositivo

```
taxa_churn_DeviceProtection = pd.crosstab(dados["DeviceProtection"],
dados["Churn"], normalize="index") * 100

taxa_churn_OnlineBackup.plot(
    kind="bar", stacked=True, color=["skyblue", "salmon"], figsize=(8,5)
```

```
)
plt.title("Taxa de Churn - DeviceProtection")
plt.ylabel("Percentual (%)")
plt.xlabel("DeviceProtection")
plt.legend(title="Churn")
plt.show()
```



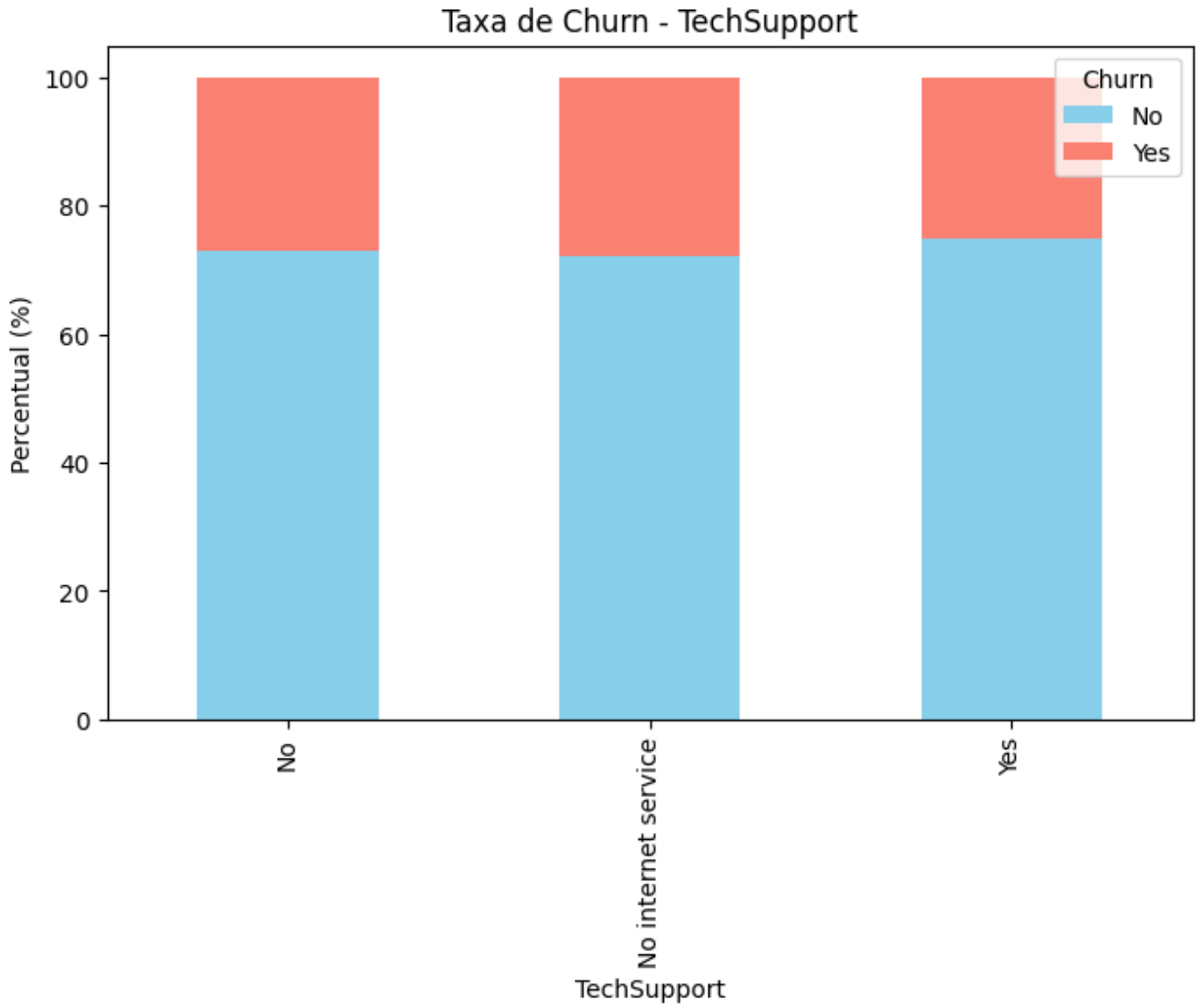

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

Churn	No	Yes
DeviceProtection		

Churn	No	Yes
DeviceProtection		
No	74.032043	25.967957
No internet service	72.180959	27.819041
Yes	73.183170	26.816830

Suporte técnico

```
taxa_churn_TechSupport = pd.crosstab(dados["TechSupport"], dados["Churn"],
normalize="index") * 100

taxa_churn_TechSupport.plot(
    kind="bar", stacked=True, color=["skyblue", "salmon"], figsize=(8,5)
)
plt.title("Taxa de Churn - TechSupport")
plt.ylabel("Percentual (%)")
plt.xlabel("TechSupport")
plt.legend(title="Churn")
plt.show()
```

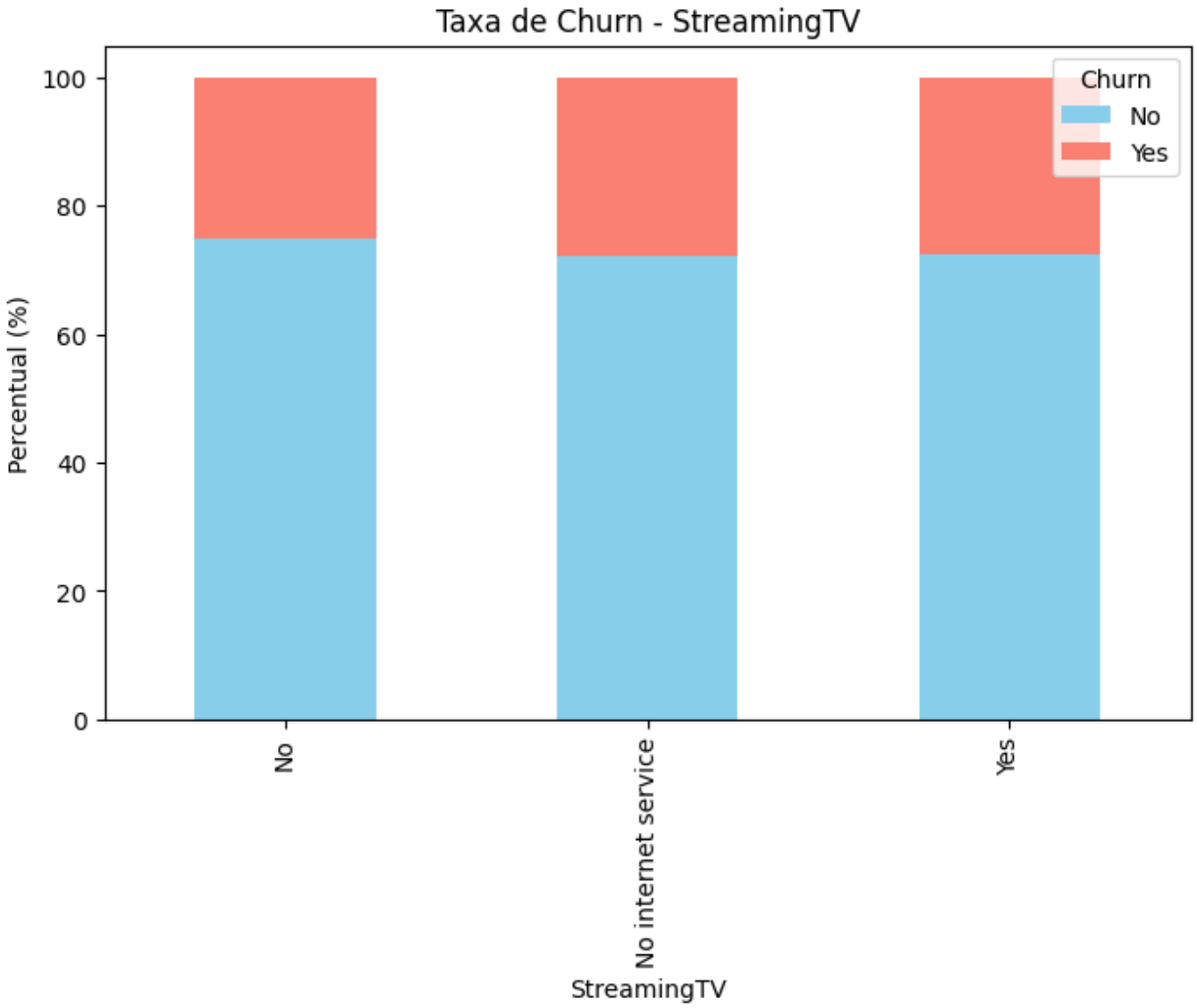



```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

Churn	No	Yes
TechSupport		
No	72.959335	27.040665
No internet service	72.180959	27.819041
Yes	74.848485	25.151515

TV por streaming

```
taxa_churn_StreamingTV = pd.crosstab(dados["StreamingTV"], dados["Churn"],
normalize="index") * 100

taxa_churn_StreamingTV.plot(
    kind="bar", stacked=True, color=["skyblue", "salmon"], figsize=(8,5)
)
plt.title("Taxa de Churn - StreamingTV")
plt.ylabel("Percentual (%)")
plt.xlabel("StreamingTV")
plt.legend(title="Churn")
plt.show()
```

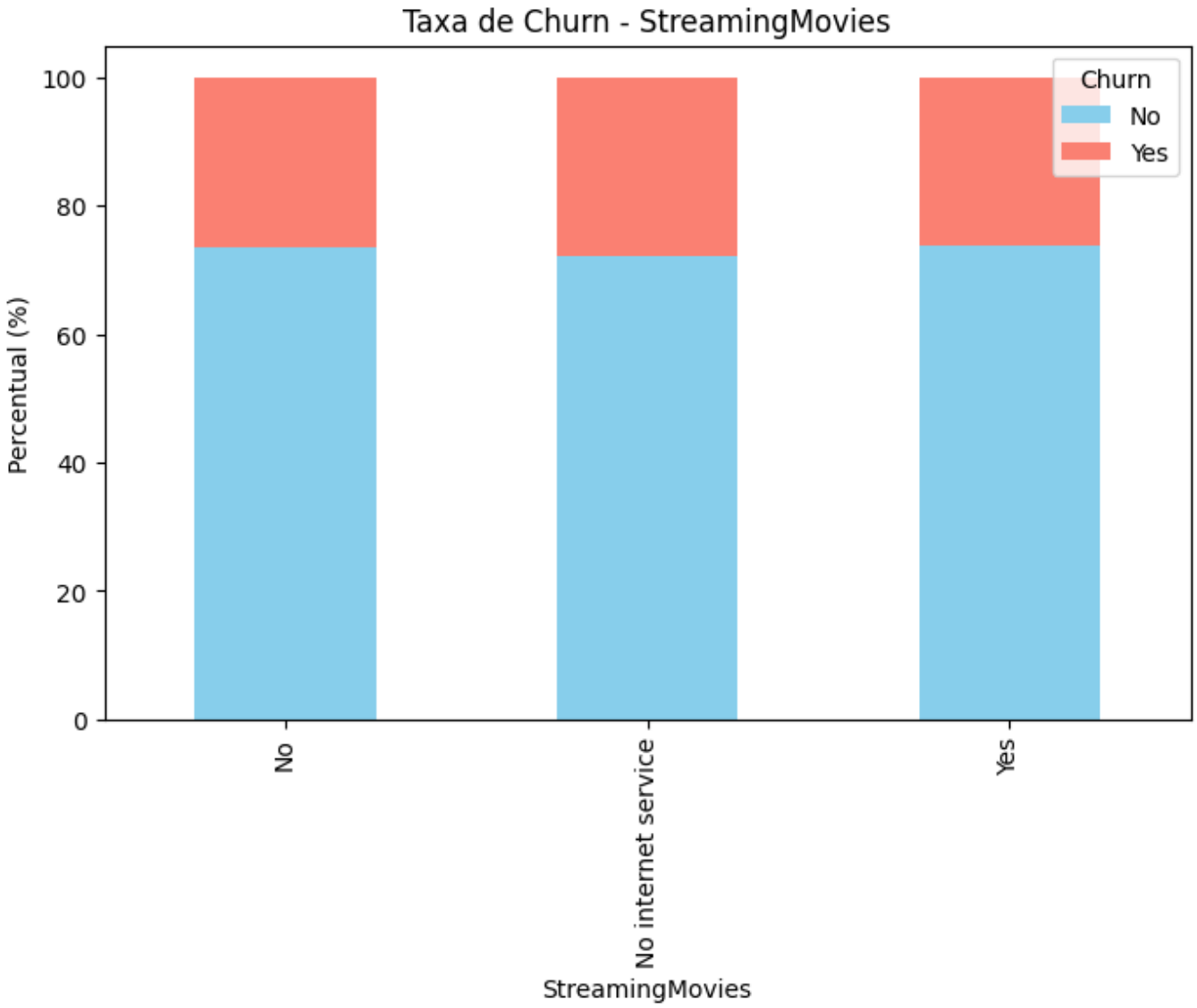



```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

Churn	No	Yes
StreamingTV		
No	74.871229	25.128771
No internet service	72.180959	27.819041
Yes	72.405929	27.594071

Filmes por streaming

```
taxa_churn_StreamingMovies = pd.crosstab(dados["StreamingMovies"], dados["Churn"],
normalize="index") * 100

taxa_churn_StreamingMovies.plot(
    kind="bar", stacked=True, color=["skyblue", "salmon"], figsize=(8,5)
)
plt.title("Taxa de Churn - StreamingMovies")
plt.ylabel("Percentual (%)")
plt.xlabel("StreamingMovies")
plt.legend(title="Churn")
plt.show()
```

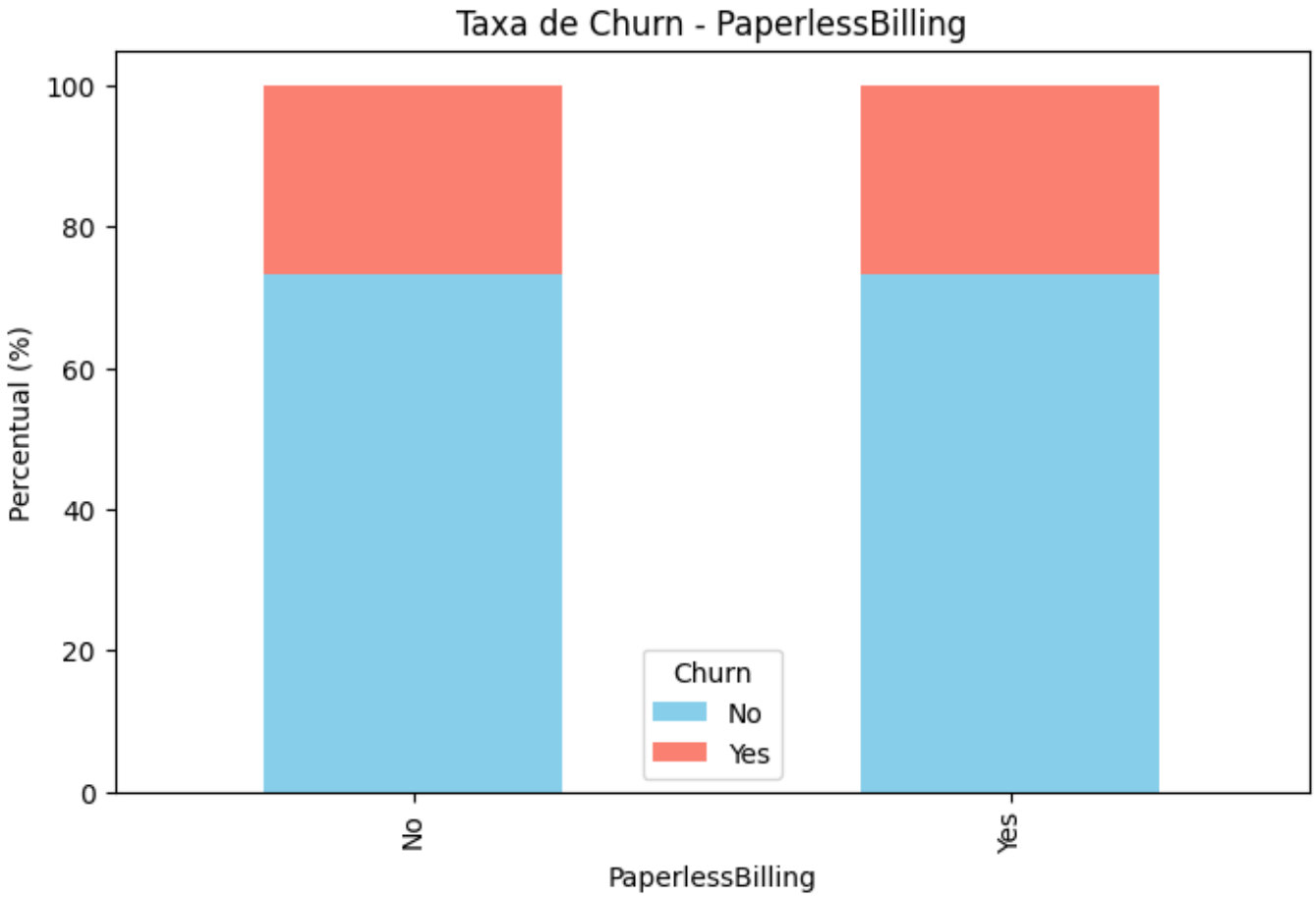



```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

Churn	No	Yes
StreamingMovies		
No	73.489062	26.510938
No internet service	72.180959	27.819041
Yes	73.831071	26.168929

Fatura digital

```
taxa_churn_PaperlessBilling = pd.crosstab(dados["PaperlessBilling"],
dados["Churn"], normalize="index") * 100

taxa_churn_PaperlessBilling.plot(
    kind="bar", stacked=True, color=["skyblue", "salmon"], figsize=(8,5)
)
plt.title("Taxa de Churn - PaperlessBilling")
plt.ylabel("Percentual (%)")
plt.xlabel("PaperlessBilling")
plt.legend(title="Churn")
plt.show()
```



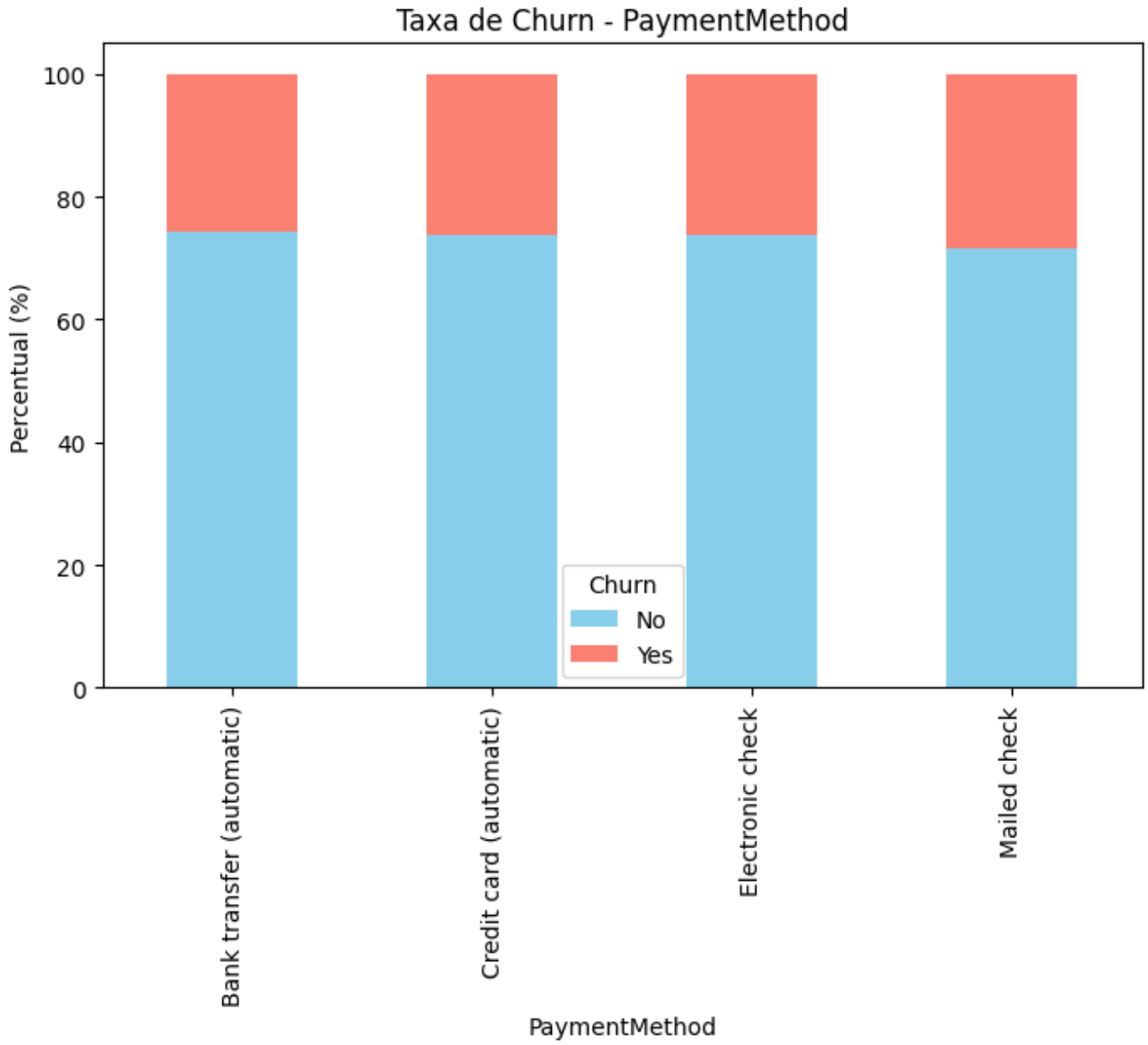

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

Churn	No	Yes
PaperlessBilling		

Churn	No	Yes
PaperlessBilling		
No	73.335732	26.664268
Yes	73.339916	26.660084

Método de pagamento

```
taxa_churn_PaymentMethod = pd.crosstab(dados["PaymentMethod"], dados["Churn"],
normalize="index") * 100

taxa_churn_PaymentMethod.plot(
    kind="bar", stacked=True, color=["skyblue", "salmon"], figsize=(8,5)
)
plt.title("Taxa de Churn - PaymentMethod")
plt.ylabel("Percentual (%)")
plt.xlabel("PaymentMethod")
plt.legend(title="Churn")
plt.show()
```




```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

Churn	No	Yes
PaymentMethod		
Bank transfer (automatic)	74.184963	25.815037
Credit card (automatic)	73.648186	26.351814
Electronic check	73.848827	26.151173

Churn	No	Yes
PaymentMethod		
Mailed check	71.483376	28.516624

Conclussões

Taxa de churn geral: 26,5% dos clientes cancelaram os serviços (1.869 de 7.043 clientes).

Clientes com **churn** têm **gasto médio mensal maior** (R$ 74,44) em comparação aos que permanecem (R$ 61,26).

```
Pode indicar sensibilidade ao preço ou percepção de custo-benefício
insatisfatória.
```

O **churn é maior nos primeiros meses** e tende a cair com a fidelização.

```
Indica que a fase inicial é crítica para retenção.
```

Contratos:

- Month-to-month apresenta churn mais alto (27,5%).
- Contratos anuais e bienais têm churn menor (~25%).

Serviços adicionais:

- Clientes sem serviços de segurança online, backup ou suporte técnico apresentam churn maior.
- Adesão a pacotes adicionais reduz a evasão.

Método de pagamento:

- Clientes com fatura enviada por correio (mailed check) têm maior churn (28,5%).
- Métodos automáticos (débito e cartão) estão associados a menor evasão (~25%).

Perfil etário e demográfico:

```
Não houve diferenças expressivas de churn entre gêneros ou idosos x não idosos.

Presença de parceiros e dependentes também não apresentou impacto significativo.
```