IFPE Campus Paulista

Curso Tecnológico em Análise e Desenvolvimento de Sistemas

Mineração de Dados

Relatório do Projeto da 2ª Unidade

Equipe:

- Eduardo José da Silva
- Adonai Ermínio
- Arthur Pedrosa

1. Compreensão do Negócio

a. Objetivo de Negócio

O objetivo do negócio é criar uma ferramenta com inteligência artificial através de modelos de classificação que sejam capazes de identificar clientes que estão propensos a cancelar o serviço com a empresa de telecomunicações. Com esta previsão, a empresa pode implementar estratégias de retenção de clientes para reduzir a taxa de churn e aumentar a receita.

b. Objetivo de Mineração:

O objetivo da mineração é encontrar dados que estejam relacionados a pessoas que cancelaram os planos de telefonia.

c. Critérios de Sucesso Mineração:

Após realizar uma pesquisa, identificamos benchmarks relevantes para a tarefa que estamos executando, utilizando bases de dados de churn iguais às nossas. Um exemplo significativo foi encontrado no Kaggle de onde pegamos nossa base de dados, no projeto chamado Orange Telecom's Churn Analysis and ML, onde os autores utilizaram modelos populares, como Logistic Regression, KNN, Decision Tree, Random Forest, SVM, Gradient Boosting, XGBoost, LightGBM e CatBoost, para avaliar a acurácia preditiva. Os resultados obtidos por eles incluíram as acurácias:

Logistic Regression: 86.55%

KNN: 88.96%
Decision Tree: 94.72%
Random Forest: 96.48%
SVC: 89.80%
Gradient Boosting: 97.28%
XGBoost: 97.28%

LightGBM: 97.32%
CatBoost: 97.40%

Esses benchmarks servirão como referência para avaliarmos a eficácia do nosso modelo.

2. Compreensão dos Dados

Sobre as informações do Pandas Profiling podemos extrair que:

Iremos utilizar a coluna Churn para saída;

Percebemos que ela está muito desbalanceada, necessitando de algumas tecnicas para fazer o desbalanceamento a posterior;

Temos 2665 dados, mas que podem se tornar em média 700 posteriormente, já que podemos usar o undersampling como técnica para o desbalanceamento de classes;

Não temos dados ausentes e nem dados duplicados.

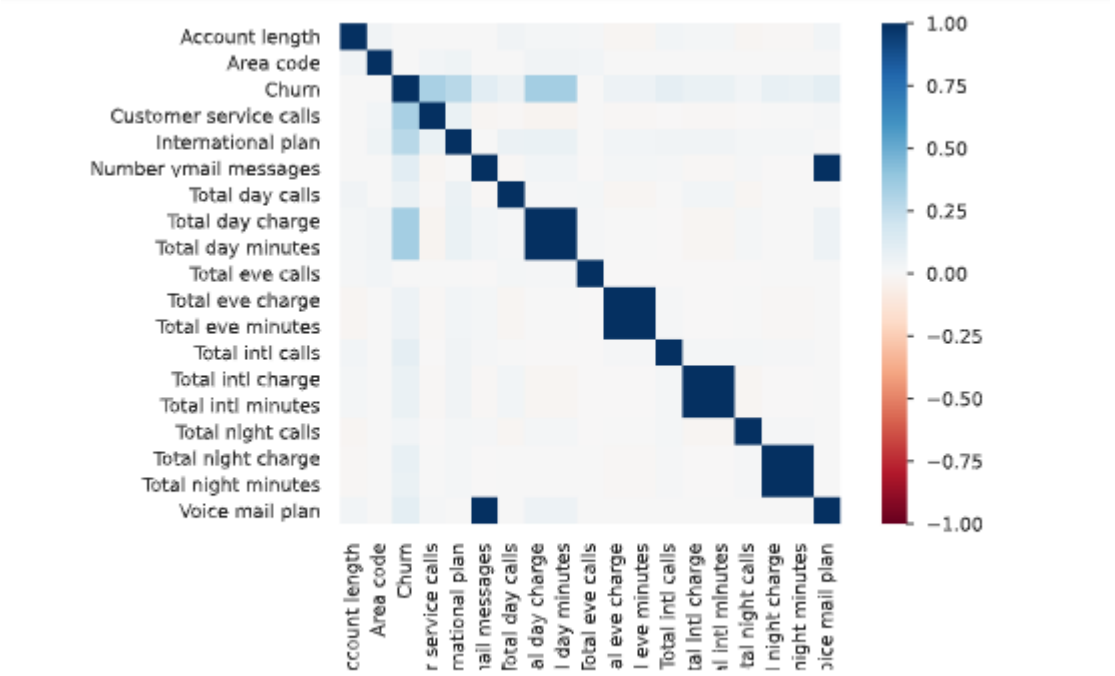
Overview | Alerts 13 | Reproduction

Dataset statistics

Number of variables	20
Number of observations	2666
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	398.5 KiB
Average record size in memory	153.0 B

Variable types

Text	1
Numeric	15
Categorical	1
Boolean	3



Análise características

Iremos analisar mais especificamente as características da nossa base de dados para visualizar melhor quais podem ser mais influentes na detecção de Churn, culminando também em quais características são as mais importantes, por termos uma base de dados aparentemente com um grande número de características.

Interpretação Geral da Matriz de Correlação

Churn:

Customer service calls (0.321) - Essa característica tem uma correlação positiva e moderada com churn, o que indica que um maior número de ligações para o serviço ao cliente podem ter uma maior probabilidade de cancelar o plano.

International plan (0.275) - Essa característica tem uma correlação positiva e moderada com churn, o que sugere que os clientes que tem um plano internacional podem ter uma maior probabilidade de cancelar o plano.

Total day charge (0.343) e Total day minutes (0.344) - Essas duas características têm uma correlação positiva com churn, o que indica que clientes que gastam mais em chamadas diárias podem ter uma probabilidade maior de cancelar o plano.

Number vmail messages (0.108) - Essa característica tem uma correlação mais fraca, mas ainda podem indicar que mais mensagens de voz podem estar relacionadas ao churn.

Voice mail plan (0.096) - Essa característica é ainda mais fraca que a anterior, mas ainda pode indicar que clientes com plano de correio de voz podem ter um pouco mais de probabilidade de cancelar o plano.

Total intl calls (0.093) - Característica com correlação fraca, mas que pode indicar que quanto mais chamadas internacionais um cliente faz, maior pode ser a probabilidade do cliente cancelar o plano.

Total intl charge (0.068) e Total intl minutes (0.068) - As duas características têm baixa correlação com churn, mas podem indicar que o tempo e o custo das chamadas internacionais podem ter impacto na probabilidade do cliente cancelar o plano.

Multicolinearidade entre Características:

A multicolinearidade é uma situação em que duas ou mais variáveis independentes em um modelo de regressão (é igualmente relevante em problemas de classificação) encontram-se altamente correlacionadas. Essa alta correlação pode afetar a qualidade dos resultados do modelo e dificultar a interpretação dos resultados.

Total day charge e Total day minutes (1.000) - Essas duas características tem uma correlação perfeita, o que é esperado já que o total cobrado de ligações durante o dia é uma função do total de minutos.

Total eve charge e Total eve minutes (1.000) - Essas duas características também tem correlação perfeita assim como a anterior, só que para ligações à noite.

Total intl charge e Total intl minutes (1.000) - Essa outras duas características seguem a mesma lógica das outras com correlação perfeita e com cobrança total de acordo com os minutos usados.

Number vmail messages e Voice mail plan (0.998) - Uma correlação quase perfeita entre essas duas características pois para ter mais mensagens de voz só é possível se o cliente tiver um plano de correio de voz.

Outras Características:

Account length (0.000) - Account length tem a correlação de zero com churn, o que indica que o tempo que o cliente tem uma conta não está relacionado com a probabilidade do cliente cancelar o plano.

Area code - A área do código não tem correlação com churn, o que indica que o código da área do cliente não é uma característica relevante para o cancelamento.

Total night charge (0.071) e Total night minutes (0.070) - Essas características tem uma correlação fraca com churn, mas ainda assim são características positivas.

Total eve charge (0.050) e Total eve minutes (0.050) - Outras características que são fracas de correlação com churn, mas ainda são positivas.

Análise aprofundada variáveis maior correlação

Iremos analisar algumas características que têm maior correlação com o churn. A ideia é entender como essas variáveis se comportam e qual pode ser sua influência na decisão de churn dos clientes.

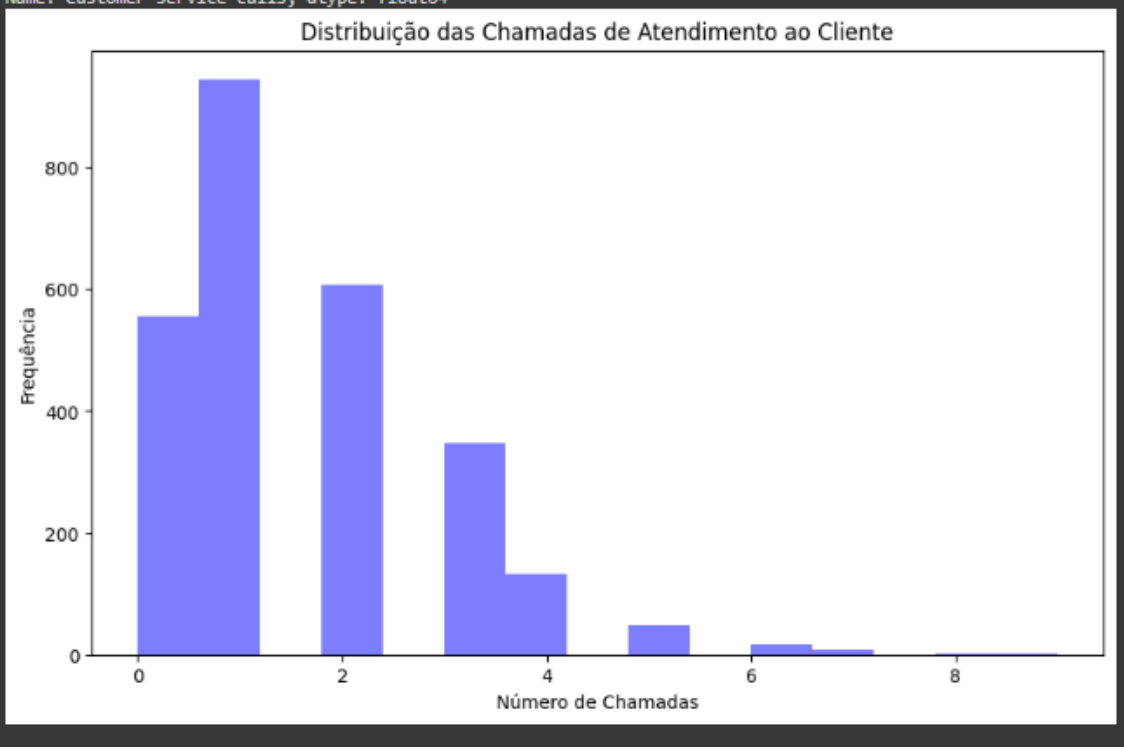
Vamos focar nas variáveis com maior correlação com Churn, como:
Customer service calls
International plan
Total day charge
Voice mail plan
Number vmail messages
Total intl calls
Total intl charge
Total day minutes
Total intl minutes

Customer service calls:

Customer service calls

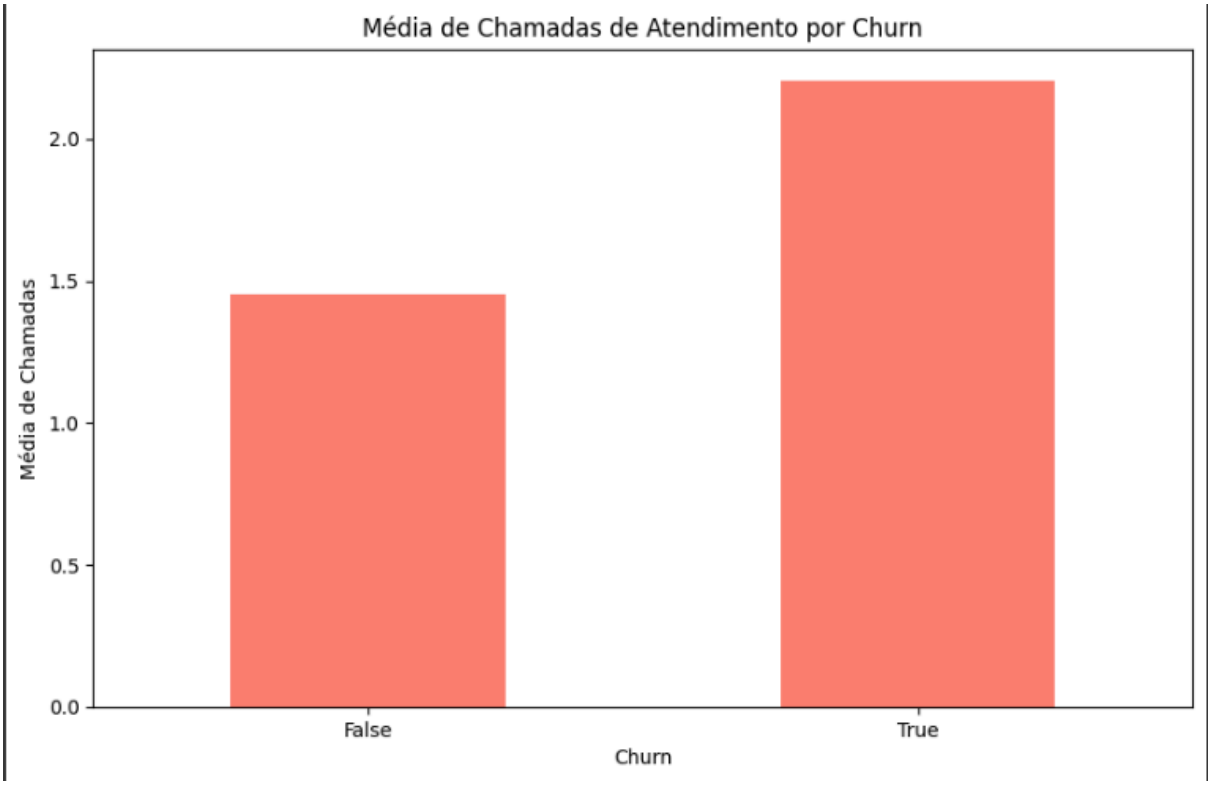
Essa é uma coluna que representa o número de chamadas que um cliente fez para o serviço de atendimento ao cliente. Essas ligações podem indicar uma insatisfação com o serviço, principalmente se essas chamadas tiverem um número alto de repetições.

```
Customer service calls - Descrição Estatística
count    2666.000000
mean        1.562641
std         1.311236
min         0.000000
25%         1.000000
50%         1.000000
75%         2.000000
max         9.000000
Name: Customer service calls, dtype: float64
```



Percebe-se que em média os clientes ligam ao menos uma vez para o serviço de atendimento ao cliente, mas podemos observar que temos valores de ligação igual a zero, o que indica que alguns clientes ligaram mais de uma vez para o atendimento.

O número máximo de chamadas é de 9, o que pode mostrar uma grande insatisfação desse(s) cliente(s), mas precisamos tomar cuidado com esses dados pois eles podem indicar outliers.



Analisando esse gráfico do número de chamadas que um cliente fez para o serviço de atendimento ao cliente, podemos observar que os clientes que deram churn fizeram um maior número de chamadas a esse serviço comparados com os que ligaram menos.

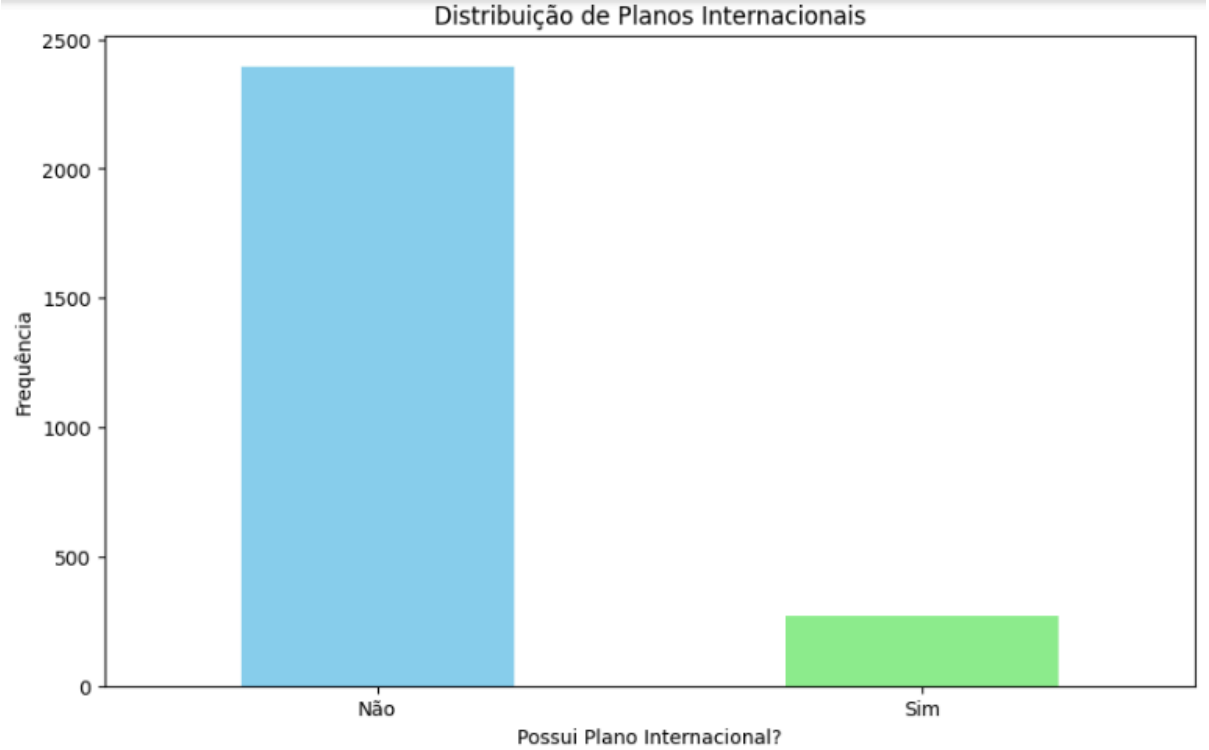
```
data = df['Customer service calls']

Q1 = np.percentile(data, 25)
Q3 = np.percentile(data, 75)

IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

outliers = [x for x in data if x < lower_bound or x > upper_bound]

print(f"Q1: {Q1}, Q3: {Q3}, IQR: {IQR}, Limite Inferior: {lower_bound}, Limite Superior: {upper_bound}")
print(f"Outliers: {outliers}")
```

```
Q1: 1.0, Q3: 2.0, IQR: 1.0, Limite Inferior: -0.5, Limite Superior: 3.5
Outliers: [4, 5, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 5, 4, 5, 4, 4, 4, 4, 4, 4, 7, 4, 4, 4, 4, 4, 5, 4, 4, 4,
```

Os outliers são os valores que estão acima do limite superior de 3.5. A lista de outliers inclui chamadas de serviço ao cliente com frequência de 4 ou mais. A presença de muitos valores de 4 ou mais indica que há uma quantidade significativa de clientes que realizaram um número elevado de chamadas para o serviço ao cliente, o que pode indicar problemas de atendimento ou insatisfação dos clientes com o plano.

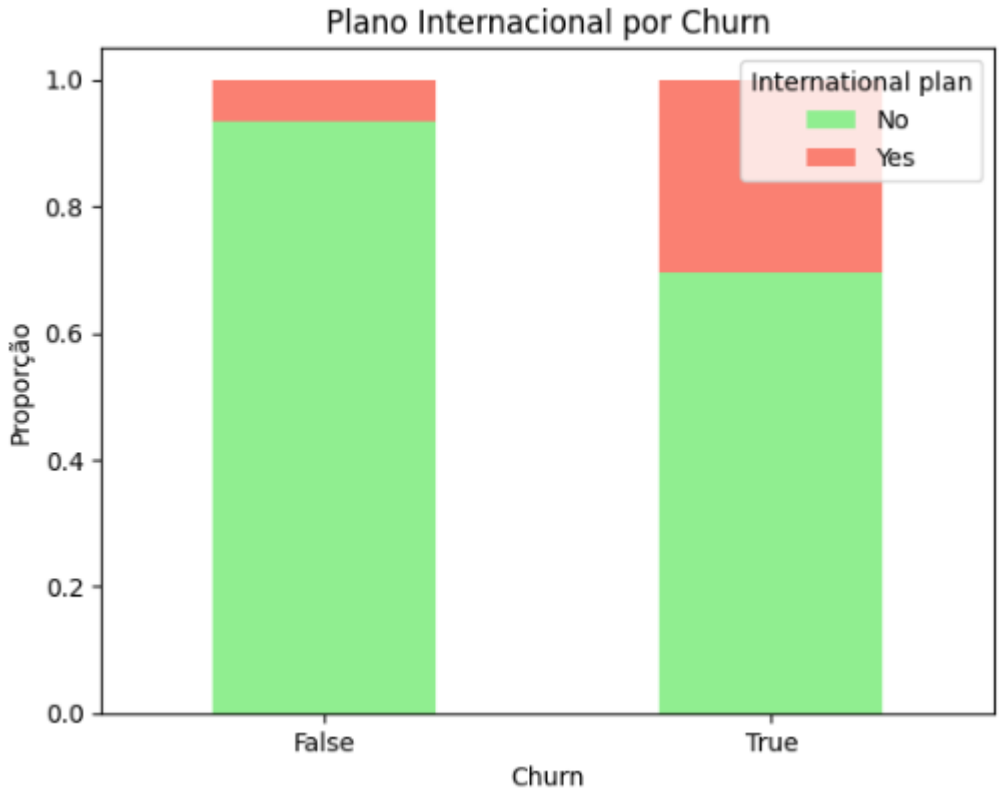
International plan:

Essa coluna indica se um cliente tem ou não um plano internacional.



Podemos observar que a distribuição dos planos internacionais na nossa base de dados indica que a maioria dos clientes não possui um plano internacional, enquanto apenas uma pequena parte opta por adquiri-lo.

O que sugere que o plano internacional oferecido não atende plenamente às expectativas dos clientes.

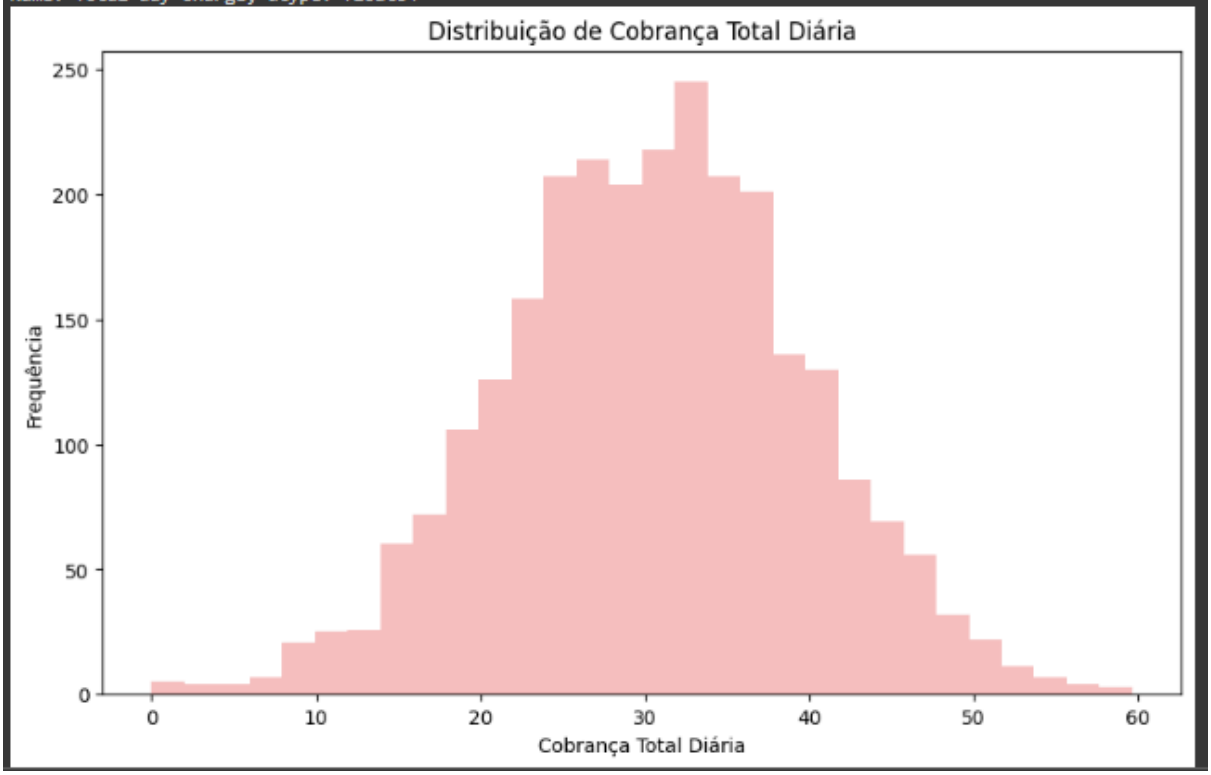


Podemos observar que há uma tendência de Churn entre clientes com planos internacionais, o que reforça que o plano pode não atender às expectativas dos clientes. Mas também temos Churn em clientes que não tem planos internacionais, indicando que outros fatores além do plano internacional podem estar contribuindo para o cancelamento dos serviços.

Total day charge:

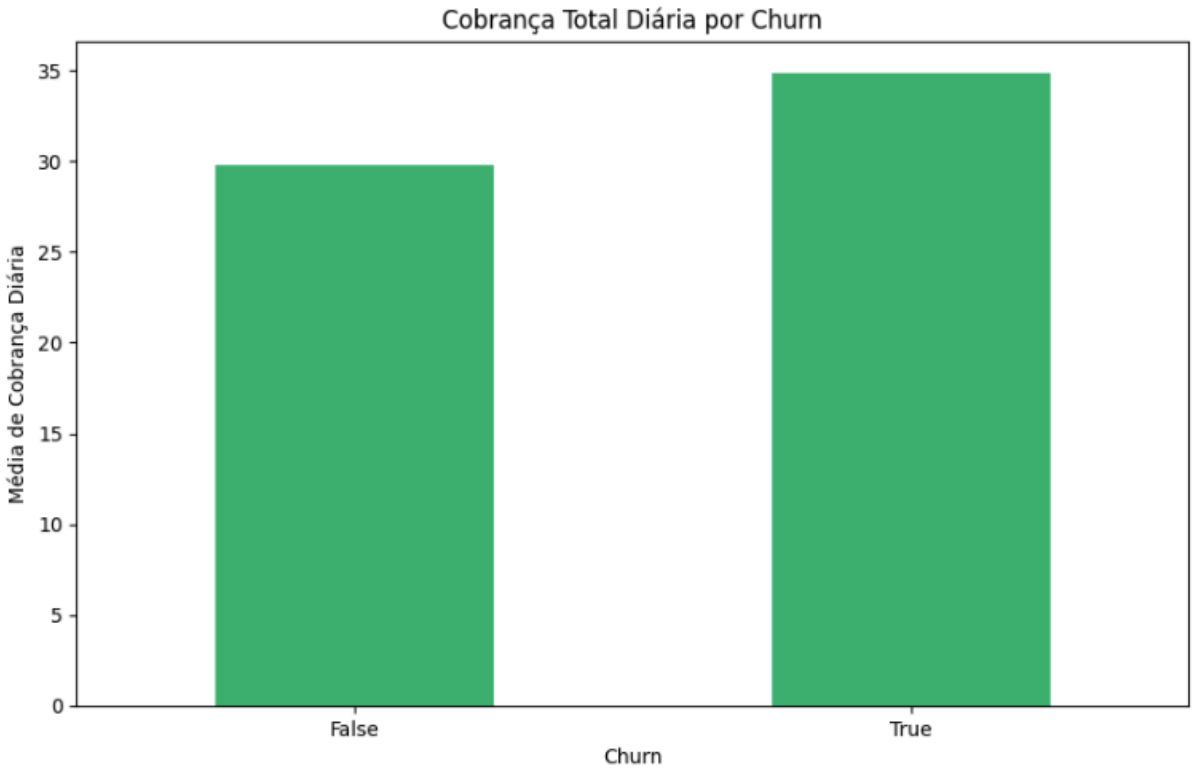
> Essa coluna representa a cobrança total durante o dia.

```
Total day charge - Descrição Estatística
count    2666.000000
mean       30.512404
std         9.215733
min         0.000000
25%        24.380000
50%        30.590000
75%        36.700000
max        59.640000
Name: Total day charge, dtype: float64
```



Podemos analisar que em média a cobrança diária está em torno de 30.51, sugerindo que muitos clientes estão pagando esse valor diariamente. A distribuição desses dados é relativamente simétrica, sem a presença significativa de valores extremos que possam distorcer a média.

Além disso, observa-se uma variação considerável nesses valores, onde alguns clientes podem ter cobranças diárias significativamente mais altas, enquanto outros pagam valores bem menores.



Clientes que realizaram churn apresentam, em média, uma cobrança diária mais alta, o que pode sugerir uma insatisfação com o custo diário. No entanto, a diferença entre a média de cobranças diárias dos clientes que deram churn e daqueles que não deram churn não é tão significativa, indicando que até mesmo clientes que permanecem com o serviço também enfrentam custos diários elevados.

```
data = df['Total day charge']

Q1 = np.percentile(data, 25)
Q3 = np.percentile(data, 75)

IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

outliers = [x for x in data if x < lower_bound or x > upper_bound]

print(f"Q1: {Q1}, Q3: {Q3}, IQR: {IQR}, Limite Inferior: {lower_bound}, Limite Superior: {upper_bound}")
print(f"Outliers: {outliers}")
```

```
Q1: 24.38, Q3: 36.7, IQR: 12.320000000000004, Limite Inferior: 5.899999999999995, Limite Superior: 55.18000000000001
Outliers: [57.36, 59.64, 57.04, 5.25, 5.78, 58.96, 2.13, 0.0, 0.0, 3.32, 56.07, 1.34, 55.78, 4.59, 2.99, 55.47, 58.7, 0.44, 1.33, 3.21, 5.08]
```

Os valores baixos próximos a 0 é o seu limite inferior e valores superiores a 55.18 é seu limite superior.

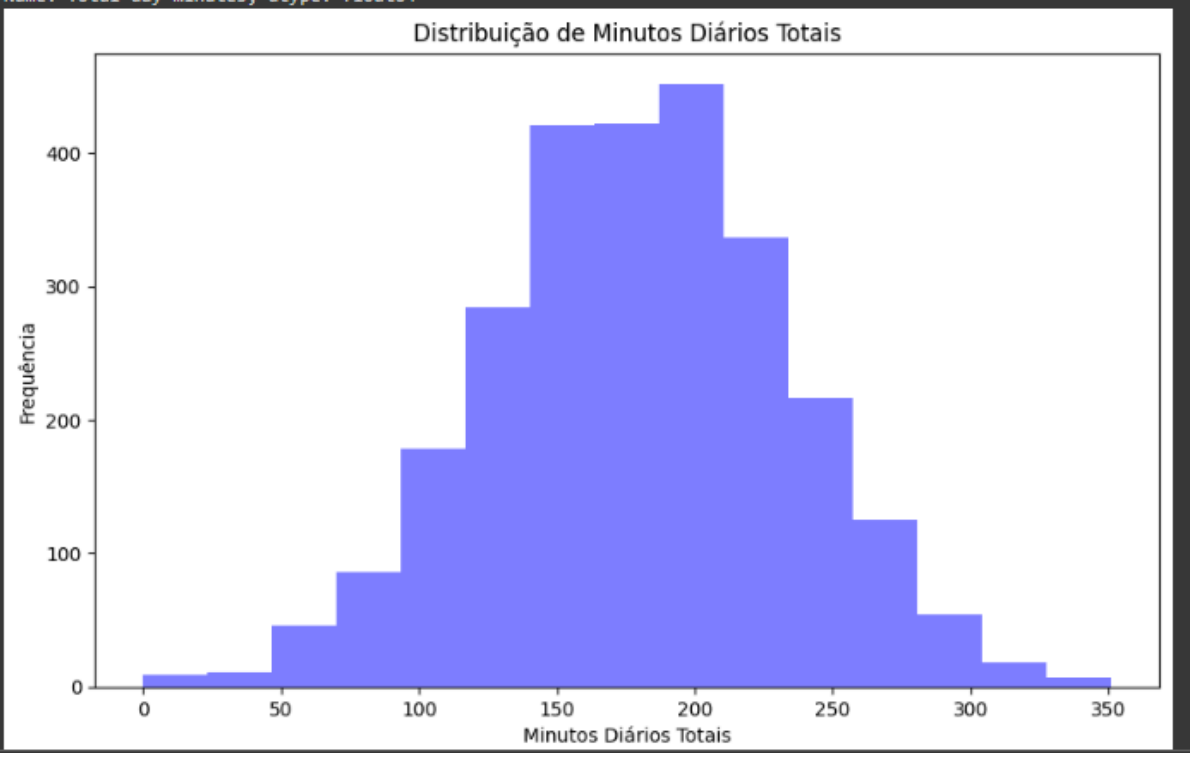
O valor mais alto identificado sendo 59.64 são os clientes que têm cobranças diárias significativamente altas.

Os valores próximos a zero ou de zero indicam clientes com baixas ou nulas cobranças diárias podem indicar falta de engajamento com o serviço.

Total day minutes:

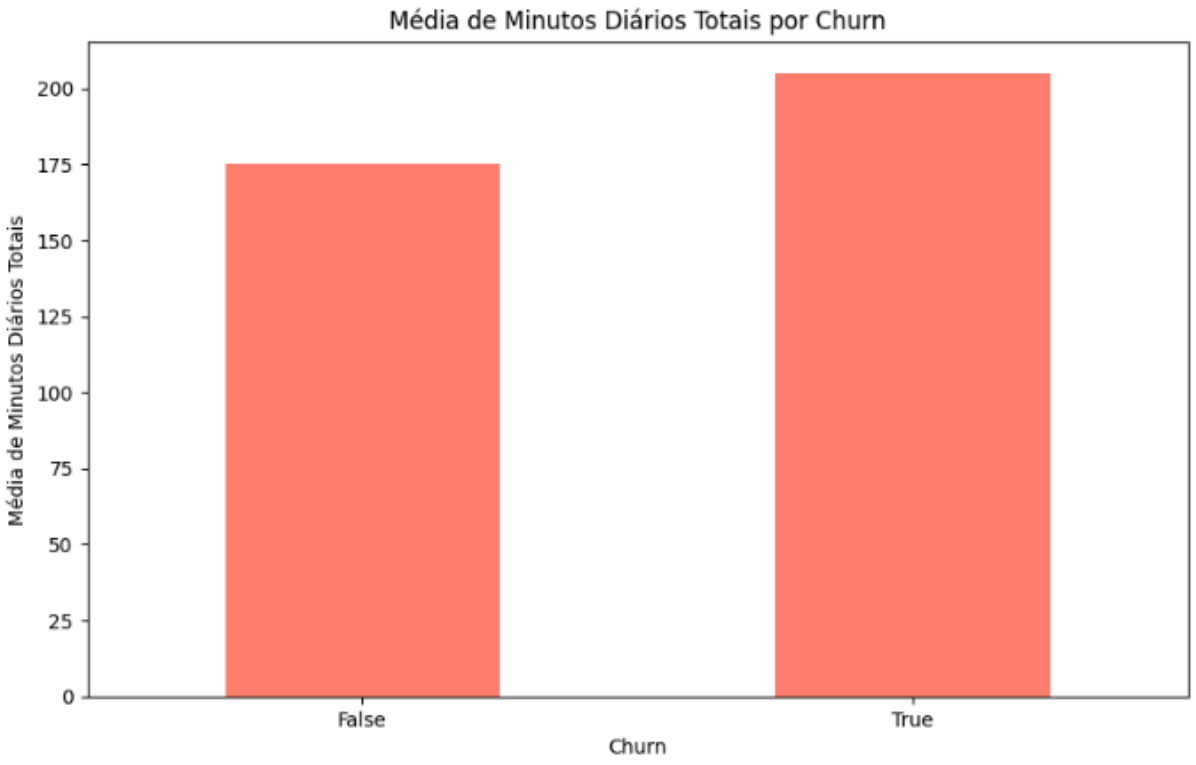
> O número total de minutos usados durante chamadas diurnas. Esta é uma métrica essencial para entender o comportamento de chamada de um cliente durante os horários de pico.

```
Total day minutes - Descrição Estatística
count    2666.00000
mean      179.48162
std        54.21035
min         0.00000
25%       143.40000
50%       179.95000
75%       215.90000
max       350.80000
Name: Total day minutes, dtype: float64
```



Podemos analisar que a mediana e a média estão muito próximas, sugerindo que a distribuição dos Total day minutes é aproximadamente simétrica. Contudo, o desvio padrão relativamente alto e a presença de valores extremos (0 minutos e 350.8 minutos) indicam uma certa variabilidade no comportamento dos clientes.

A presença de valores mínimos de 0 minutos pode indicar clientes que não utilizam o serviço de minutos diários, o que pode ser relevante ao analisar o churn.



Assim como na coluna Total day charge, na coluna Total day minutos os clientes que realizaram churn apresentam, em média, um uso de minutos diário em chamadas mais alto, o que pode sugerir uma insatisfação com o custo diário, pois usando mais, eles obtem mais gastos.

No entanto, a diferença entre a média de uso de minutos diários dos clientes que deram churn e daqueles que não deram churn não é tão significativa, indicando que até mesmo clientes que permanecem com o serviço também tem um número de minutos em chamadas diárias altas.

```
data = df['Total day minutes']

Q1 = np.percentile(data, 25)
Q3 = np.percentile(data, 75)

IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

outliers = [x for x in data if x < lower_bound or x > upper_bound]

print(f"Q1: {Q1}, Q3: {Q3}, IQR: {IQR}, Limite Inferior: {lower_bound}, Limite Superior: {upper_bound}")
print(f"Outliers: {outliers}")
```

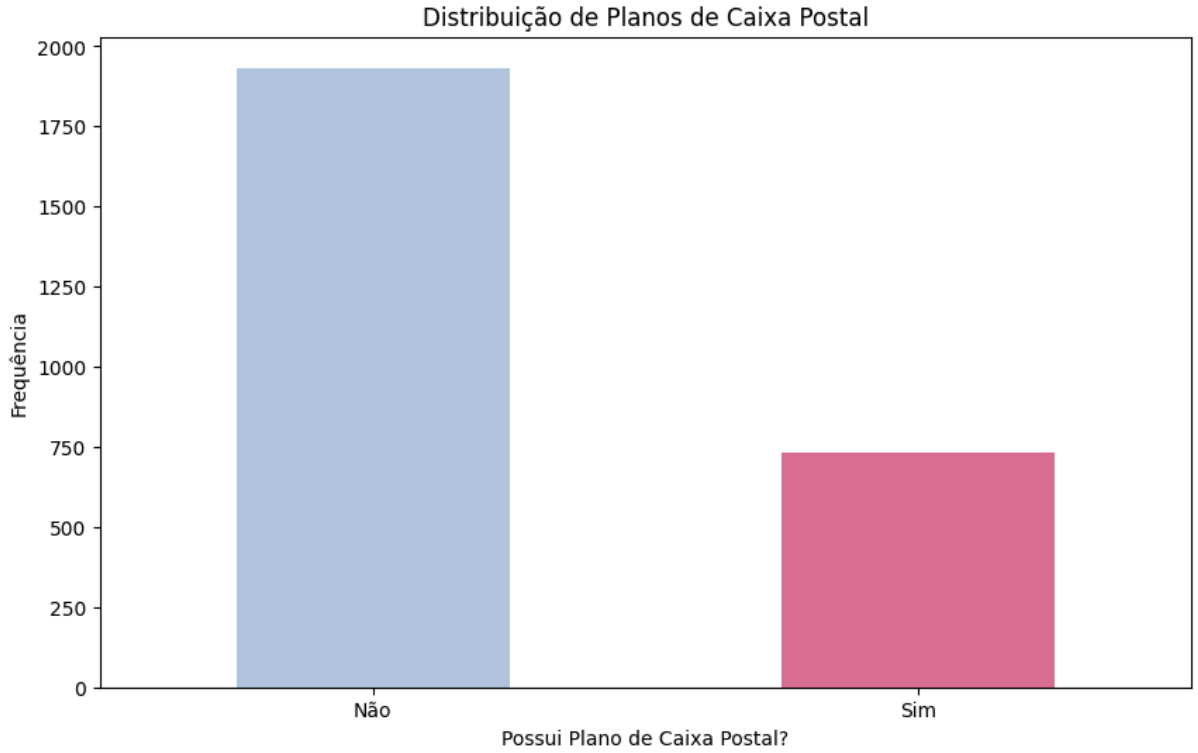
```
Q1: 143.4, Q3: 215.9, IQR: 72.5, Limite Inferior: 34.650000000000006, Limite Superior: 324.65
Outliers: [337.4, 350.8, 335.5, 30.9, 34.0, 346.8, 12.5, 0.0, 0.0, 19.5, 329.8, 7.9, 328.1, 27.0, 17.6, 326.3, 345.3, 2.6, 7.8, 18.9, 29.9]
```

Esses outliers representam clientes que estão utilizando muito mais minutos do que a maioria. Eles podem ser usuários mais exigentes ou que dependem mais do serviço. Esses clientes podem ter necessidades diferentes e possivelmente têm um maior impacto sobre a rede.

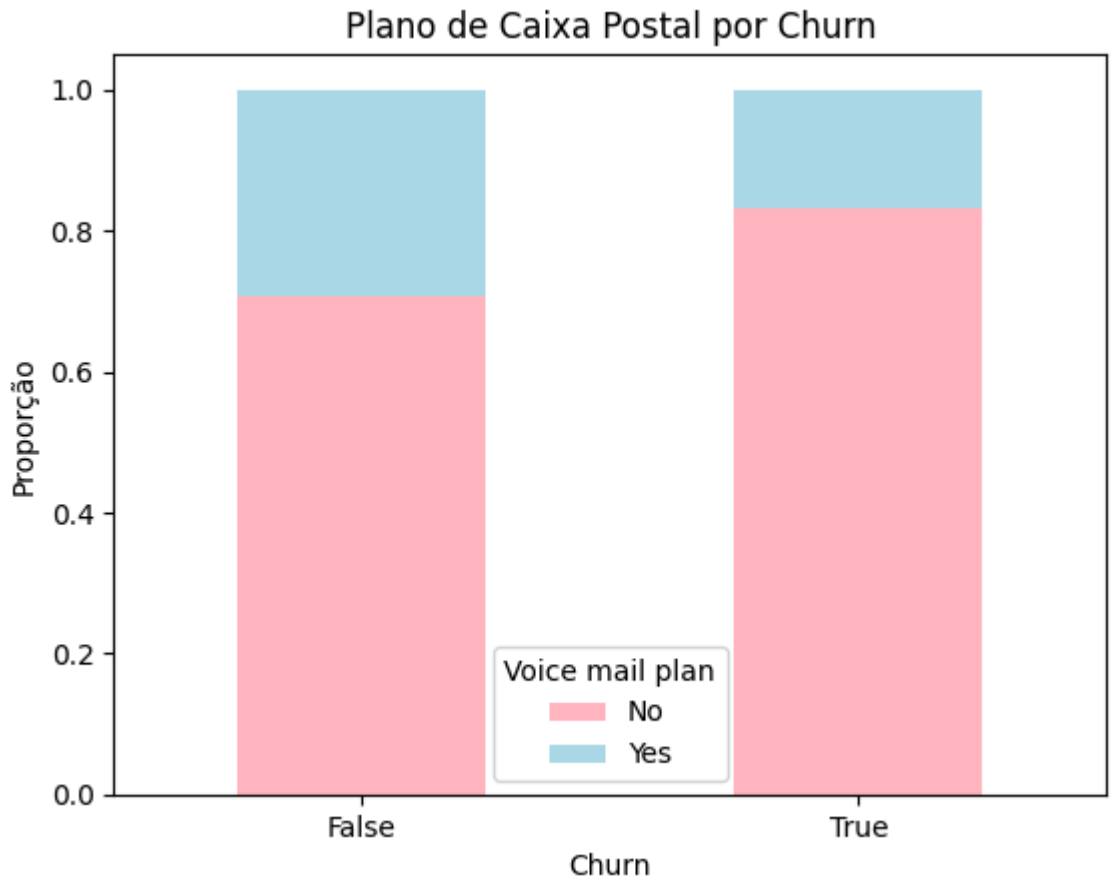
Essa coluna tem uma variação significativa, com alguns clientes utilizando muito mais ou muito menos minutos diários do que a média. Esses extremos podem ser relevantes para entender o comportamento dos clientes, especialmente em relação ao churn.

Voice mail plan:

> Essa coluna indica se um cliente possui um plano de caixa postal.



Podemos observar que a maioria dos clientes da nossa base de dados não possui um plano e caixa postal, o que pode indicar uma falta de interesse ou necessidade por parte dos usuários em relação a esse serviço.



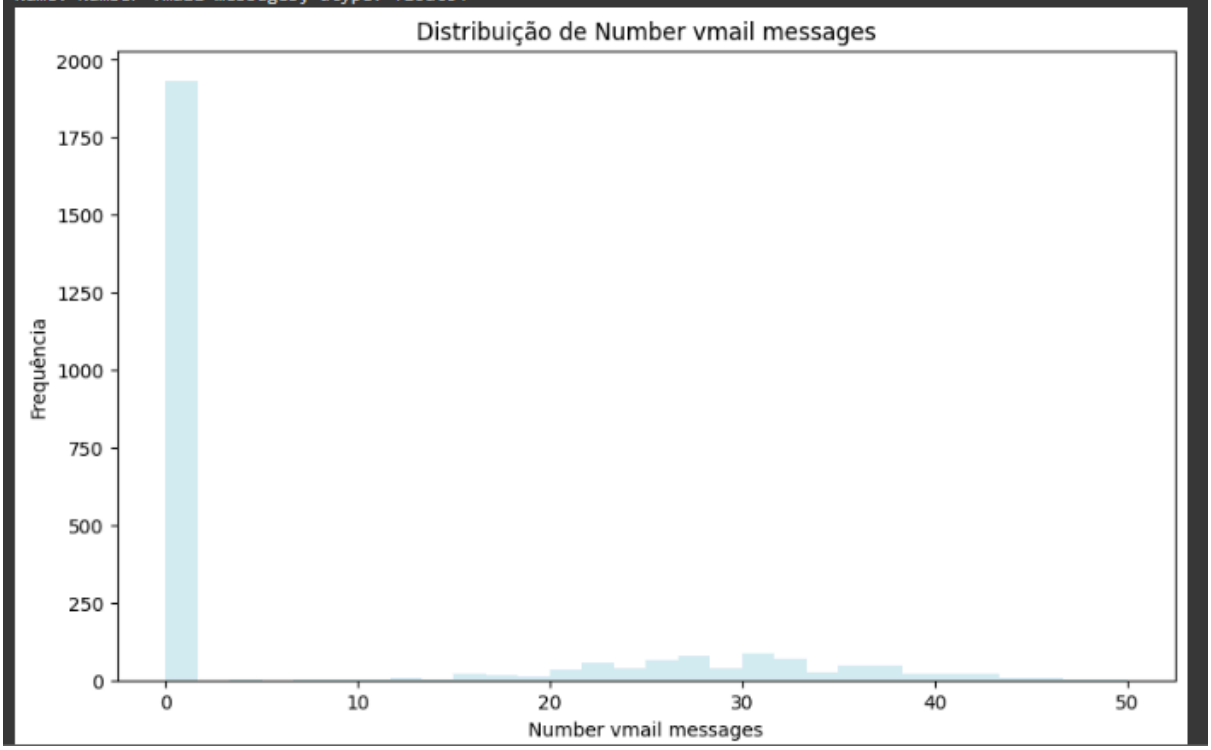
Aqui podemos observar uma correlação entre clientes que possuem um plano de caixa postal e o churn, mas é importante notar que a maioria dos clientes que mantêm o plano de caixa postal não cancelou o serviço.

Isso sugere que, embora ter um plano de caixa postal possa influenciar no churn, não parece ser uma variável com grande impacto isolado.

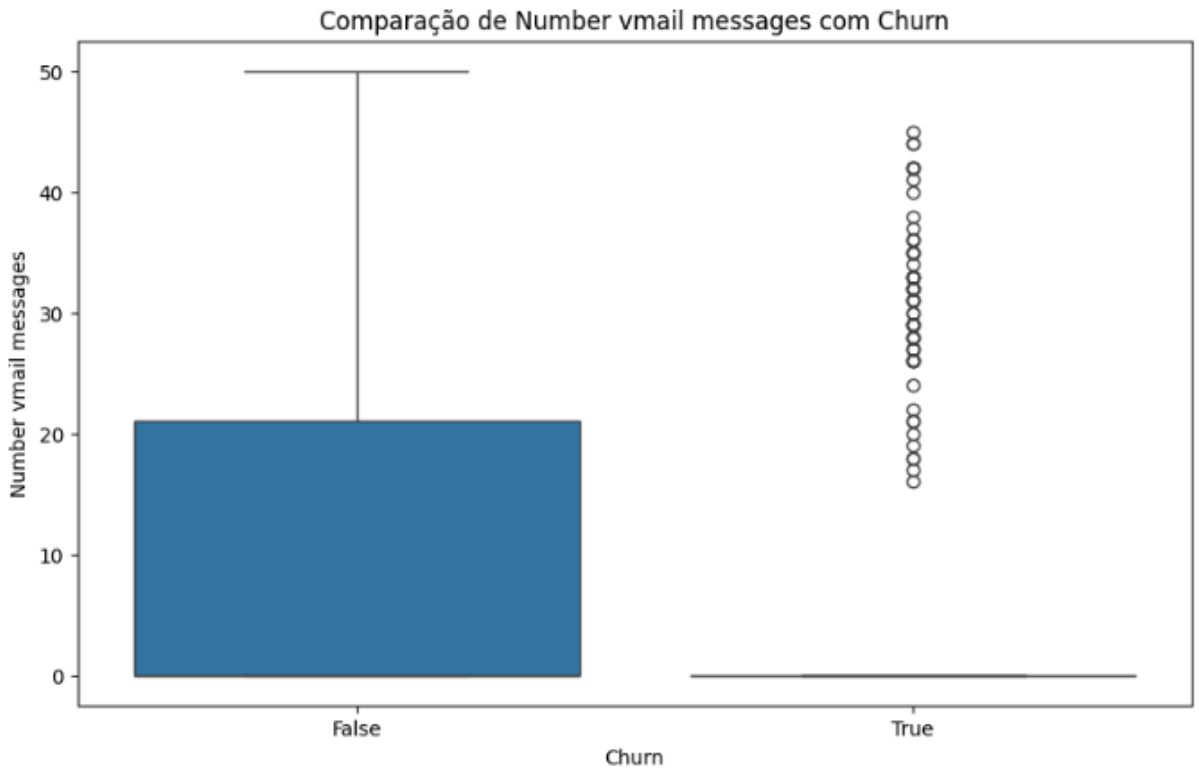
Number vmail messages:

> Esta coluna representa o número de mensagens de voz recebidas pelo cliente. Pode ser útil entender se a quantidade de mensagens de voz tem alguma relação com o churn.

```
Number vmail messages - Descrição Estatística
count    2666.000000
mean        8.021755
std        13.612277
min         0.000000
25%         0.000000
50%         0.000000
75%        19.000000
max        50.000000
Name: Number vmail messages, dtype: float64
```



Podemos analisar que a média de mensagens recebidas pelos clientes é de 8.02, mas a maioria dos clientes não recebeu nenhuma mensagem, o que resulta em uma variância de 13.61. Com um valor máximo de 50 mensagens, essa coluna apresenta possíveis outliers que podem necessitar de tratamento posterior.



Podemos analisar que os clientes que mais receberam mensagens de voz, são em sua maioria clientes que deram churn.

```
data = df['Number vmail messages']

Q1 = np.percentile(data, 25)
Q3 = np.percentile(data, 75)

IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

outliers = [x for x in data if x < lower_bound or x > upper_bound]

print(f"Q1: {Q1}, Q3: {Q3}, IQR: {IQR}, Limite Inferior: {lower_bound}, Limite Superior: {upper_bound}")
print(f"Outliers: {outliers}")
```

```
Q1: 0.0, Q3: 19.0, IQR: 19.0, Limite Inferior: -28.5, Limite Superior: 47.5
Outliers: [50, 50]
```

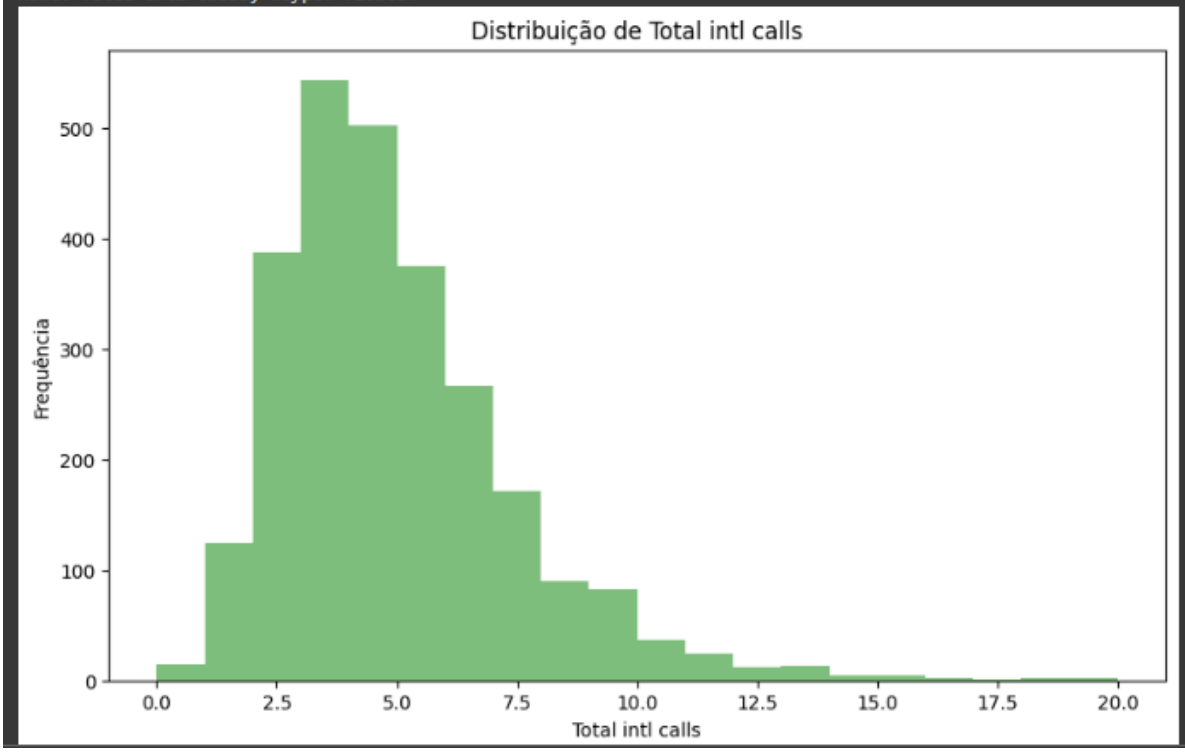
A maioria dos clientes (até o 75º percentil) tem até 19 mensagens de voicemail, o que sugere que o uso do serviço é relativamente baixo na maior parte da base de clientes. Os valores de 50 mensagens de voicemail são considerados outliers, pois estão acima do limite superior de 47.5.

Podemos analisar que essa coluna tem uma distribuição com poucos outliers e a maioria dos clientes utiliza pouco ou nada o serviço de voicemail.

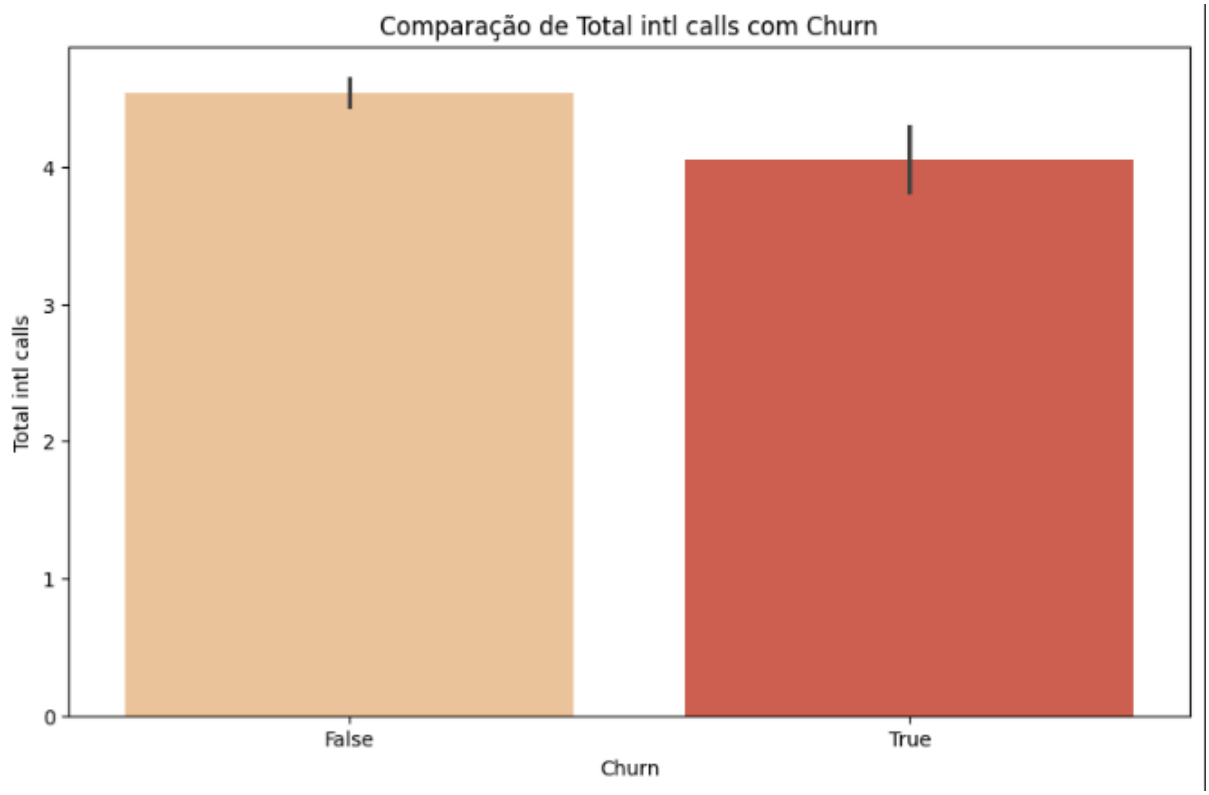
Total intl calls:

> Esta coluna representa o número total de chamadas internacionais realizadas pelo cliente.

```
Total intl calls - Descrição Estatística
count    2666.000000
mean        4.467367
std         2.456195
min         0.000000
25%         3.000000
50%         4.000000
75%         6.000000
max        20.000000
Name: Total intl calls, dtype: float64
```



A análise dos dados mostra que o número médio de ligações internacionais por cliente é de aproximadamente 4 chamadas, com um desvio padrão de 2, indicando uma variabilidade moderada no comportamento dos clientes. O número máximo de chamadas internacionais registrado é de 20, o que sugere a presença de um grupo específico de clientes que fazem uso frequente desse serviço. Esse perfil pode ser associado a clientes que mantêm contatos regulares fora do país, e essa utilização mais intensa pode influenciar suas expectativas e percepções sobre os serviços.



Podemos analisar que o número de chamadas internacionais pode ter uma certa influência no churn, mas não se mostra como um fator determinante. Como podemos visualizar no gráfico, há presença de valores elevados de chamadas internacionais, que poderiam ser considerados outliers, tanto entre os clientes que cancelaram (churn) quanto entre aqueles que não cancelaram. Isso indica que, embora a frequência de ligações internacionais possa impactar a decisão de cancelar o serviço, ela não é isoladamente um preditor confiável para o churn.

```
data = df['Total intl calls']

Q1 = np.percentile(data, 25)
Q3 = np.percentile(data, 75)

IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

outliers = [x for x in data if x < lower_bound or x > upper_bound]

print(f"Q1: {Q1}, Q3: {Q3}, IQR: {IQR}, Limite Inferior: {lower_bound}, Limite Superior: {upper_bound}")
print(f"Outliers: {outliers}")
```

```
Q1: 3.0, Q3: 6.0, IQR: 3.0, Limite Inferior: -1.5, Limite Superior: 10.5
Outliers: [19, 15, 11, 12, 13, 11, 12, 11, 13, 12, 11, 11, 18, 11, 13, 12, 12, 13, 11, 11, 14, 13, 11, 13,
```

Não foram identificados valores abaixo do limite inferior, já que as chamadas internacionais são uma variável discreta (ou seja, o número de chamadas é sempre zero ou positivo).

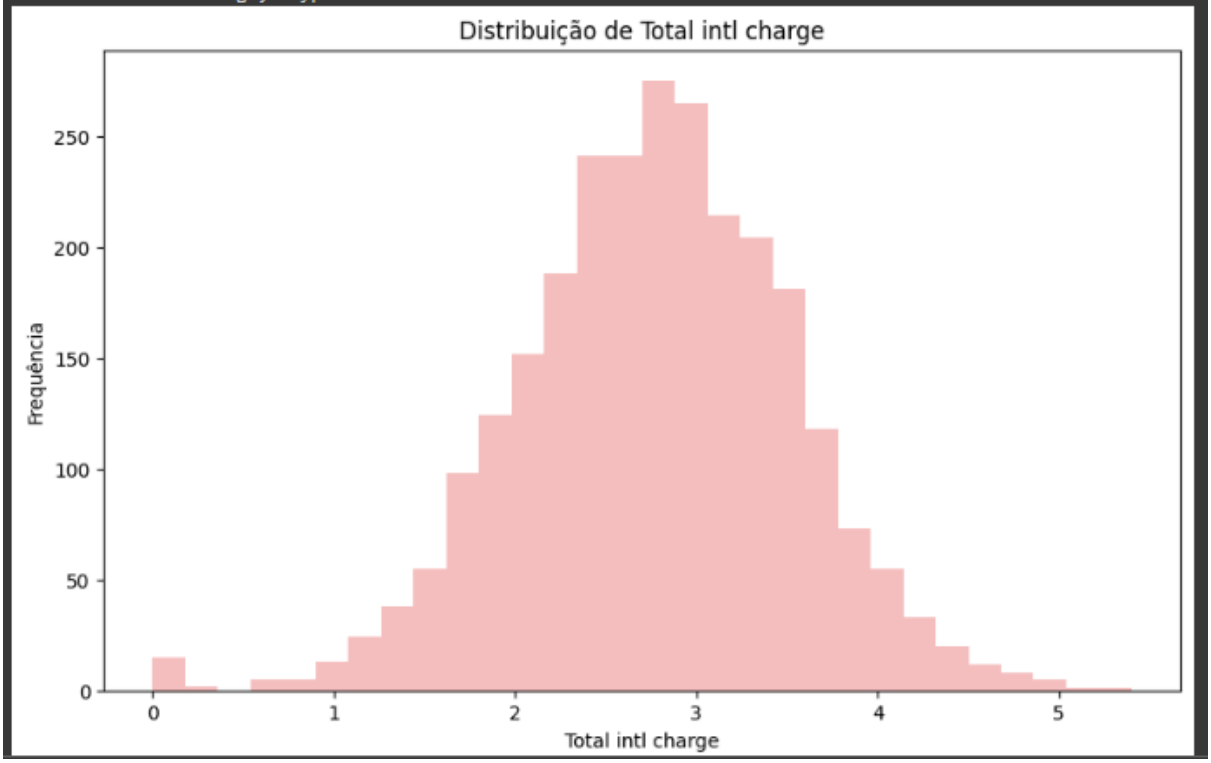
Os valores acima de 10.5 são considerados outliers. Isso sugere que existem clientes que fazem um número relativamente alto de chamadas internacionais. O valor máximo

encontrado foi 20 chamadas, o que é significativamente maior do que a maioria dos dados na distribuição. O que pode indicar clientes com um comportamento específico.

Total intl charge:

> Esta coluna representa o total de cobranças relacionadas às chamadas internacionais.

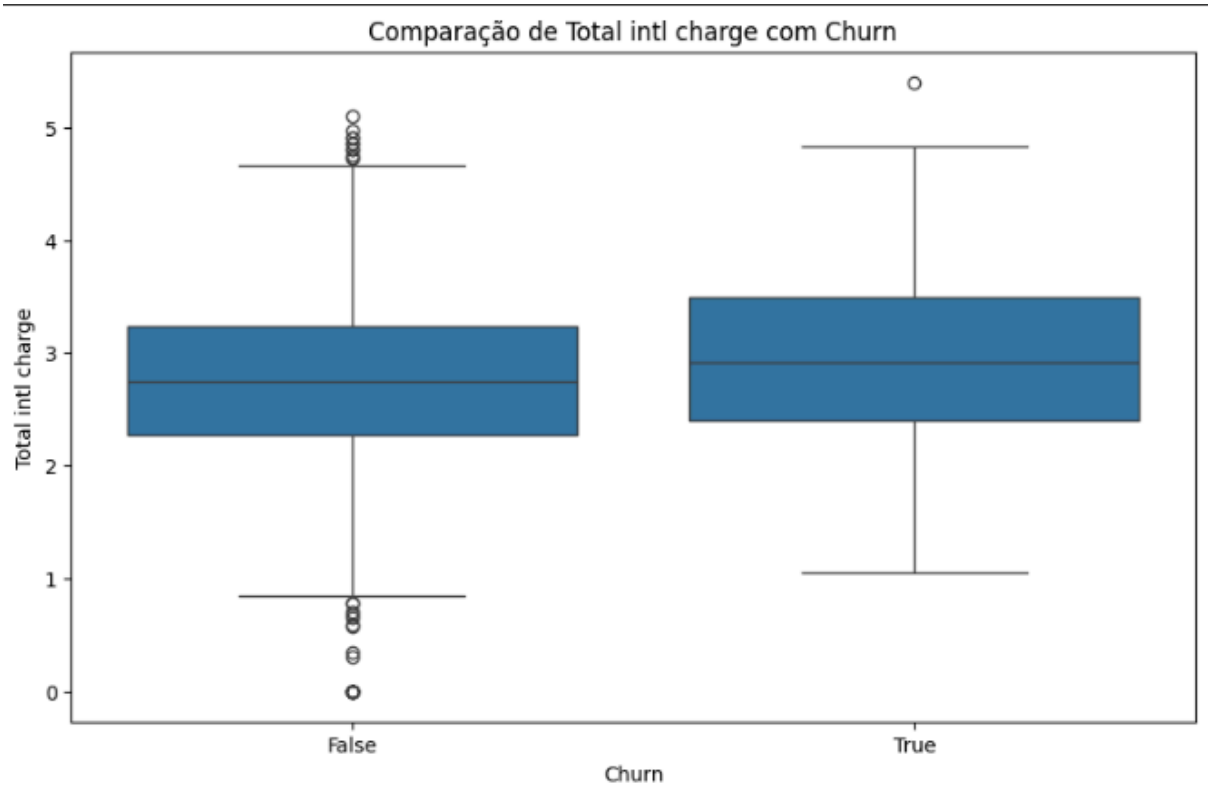
```
Total intl charge - Descrição Estatística
count    2666.000000
mean        2.764490
std         0.752812
min         0.000000
25%         2.300000
50%         2.750000
75%         3.270000
max         5.400000
Name: Total intl charge, dtype: float64
```



A maioria das cobranças internacionais está concentrada entre 2,30 e 3,27, indicando que a maioria dos clientes possui gastos moderados com chamadas internacionais. No entanto, há uma pequena quantidade de clientes que pagam valores significativamente mais altos, chegando até $5,40. A presença de um valor mínimo de 0 sugere que uma proporção de clientes não faz uso de chamadas internacionais.

Com um desvio padrão de 0,752 e um valor máximo de 5,400, podemos observar que, embora exista alguma variação entre os clientes, essa variação não é excessivamente

dispersa. Isso implica que os custos de chamadas internacionais são relativamente consistentes para a maioria dos clientes, com exceção de alguns casos isolados.



Podemos observar que, tanto para clientes que não cancelaram o plano quanto para os que cancelaram, a mediana do total de cobranças de ligações internacionais é bastante próxima. Isso sugere que o gasto médio com chamadas internacionais é similar entre esses dois grupos.

O gráfico reflete que a maioria dos clientes, independentemente de terem cancelado ou não, possui cobranças internacionais dentro de uma faixa similar. No entanto, é importante destacar a presença de outliers, especialmente entre os clientes que não cancelaram o plano. Esses outliers incluem tanto valores muito baixos (próximos a zero) quanto valores elevados, indicando que existem perfis de clientes com comportamentos atípicos em relação ao uso de chamadas internacionais.

```python
data = df['Total intl charge']

Q1 = np.percentile(data, 25)
Q3 = np.percentile(data, 75)

IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

outliers = [x for x in data if x < lower_bound or x > upper_bound]

print(f"Q1: {Q1}, Q3: {Q3}, IQR: {IQR}, Limite Inferior: {lower_bound}, Limite Superior: {upper_bound}")
print(f"Outliers: {outliers}")
```

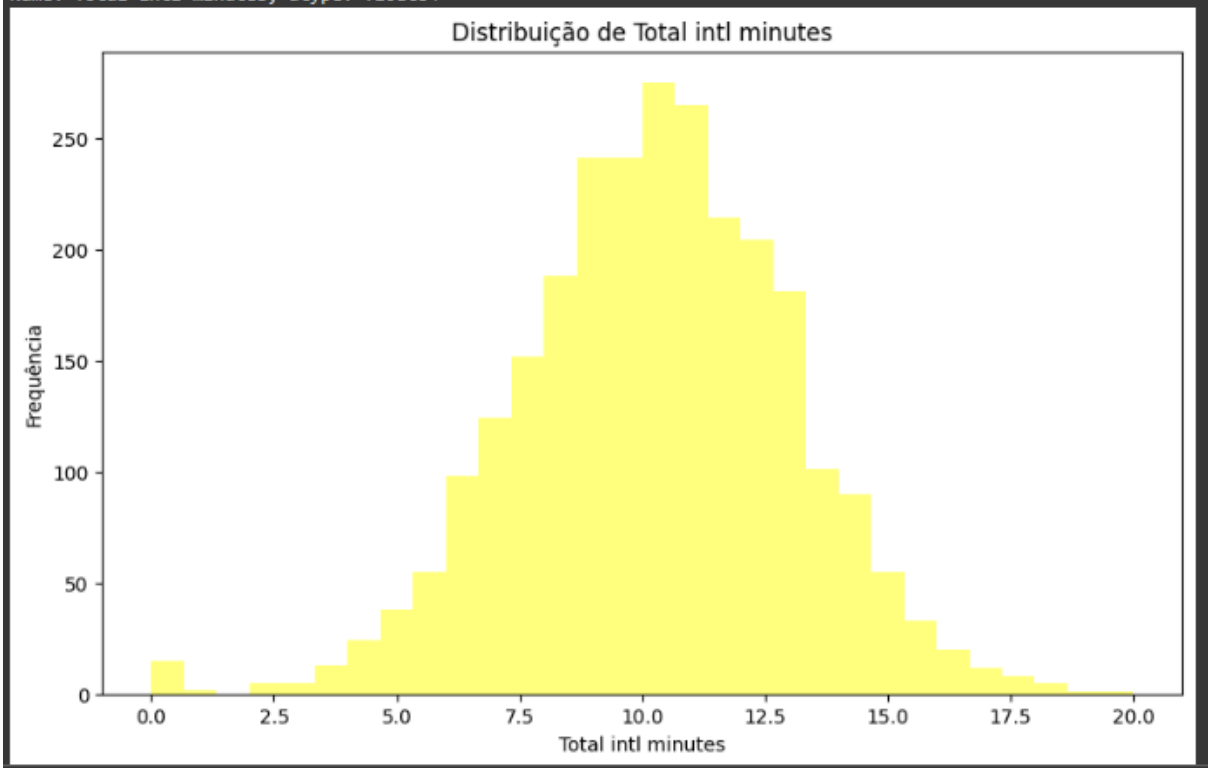
```
Q1: 2.3, Q3: 3.27, IQR: 0.9700000000000002, Limite Inferior: 0.8449999999999995, Limite Superior: 4.7250000000000005
Outliers: [5.4, 0.0, 4.75, 5.1, 0.0, 4.86, 0.0, 4.73, 4.73, 4.91, 0.0, 0.0, 0.35, 0.0, 0.0, 0.0, 0.59, 0.0, 4.83, 4.97
```

Os outliers são os valores que estão abaixo do limite inferior de 0.84 ou acima do limite superior de 4.73. A lista de outliers inclui valores tanto baixos quanto altos, como 0.0, 5.4, 4.75, entre outros. Podemos observar que há uma variabilidade significativa nas cobranças internacionais.

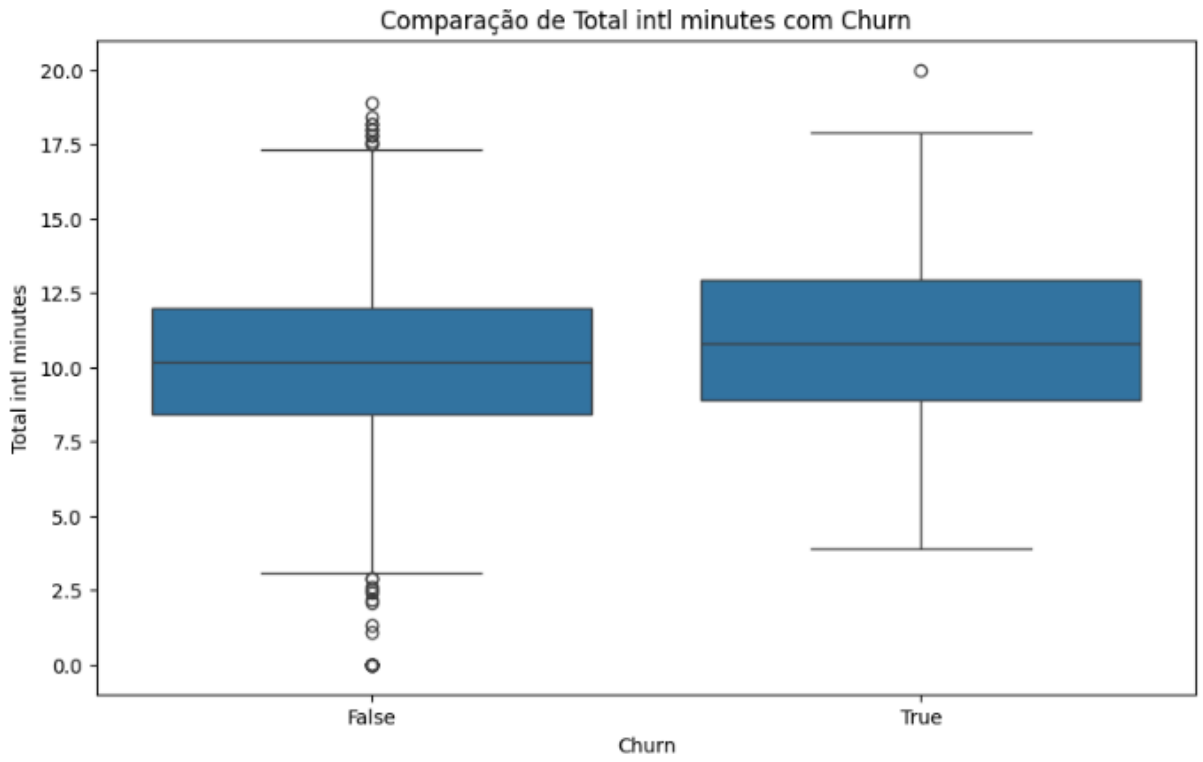
Total intl minutes:

```
Essa coluna mostra o total de minutos utilizados em chamadas internacionais.
```

```
Total intl minutes - Descrição Estatística
count    2666.000000
mean       10.237022
std         2.788349
min         0.000000
25%         8.500000
50%        10.200000
75%        12.100000
max        20.000000
Name: Total intl minutes, dtype: float64
```



Podemos observar que a média do tempo total gasto em chamadas internacionais é de aproximadamente 10.24 minutos. A maioria dos clientes realiza entre 8.5 e 12.1 minutos de chamadas internacionais, indicando uma faixa relativamente concentrada de uso. Além disso, é notável que existe uma pequena parcela de clientes que não faz chamadas internacionais, o que é evidenciado por valores próximos a zero.



Ao comparar o tempo total de chamadas internacionais com o churn, podemos observar que a mediana do tempo gasto é bastante similar para ambos os grupos, os que cancelaram o serviço (Churn = True) e os que não cancelaram (Churn = False). A faixa interquartil também é semelhante, indicando que a maioria dos clientes, independentemente do churn, gasta um tempo similar em chamadas internacionais.

Contudo, é notável a presença de outliers em ambos os grupos. No grupo que não cancelou o serviço, existem clientes que fizeram tanto chamadas internacionais muito curtas (próximas de 0 minutos) quanto chamadas significativamente longas (acima de 17.5 minutos). Já no grupo que cancelou, também há outliers, mas eles estão mais concentrados em chamadas longas.

Essa distribuição sugere que o tempo total de chamadas internacionais pode não ser um fator determinante para o churn, mas ainda assim, alguns clientes com padrões de uso extremos (muito baixo ou muito alto) podem estar mais propensos a cancelar o serviço.

```python
data = df['Total intl minutes']

Q1 = np.percentile(data, 25)
Q3 = np.percentile(data, 75)

IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

outliers = [x for x in data if x < lower_bound or x > upper_bound]

print(f"Q1: {Q1}, Q3: {Q3}, IQR: {IQR}, Limite Inferior: {lower_bound}, Limite Superior: {upper_bound}")
print(f"Outliers: {outliers}")
```

```
Q1: 8.5, Q3: 12.1, IQR: 3.5999999999999996, Limite Inferior: 3.1000000000000005, Limite Superior: 17.5
Outliers: [20.0, 0.0, 17.6, 18.9, 0.0, 18.0, 0.0, 18.2, 0.0, 0.0, 1.3, 0.0, 0.0, 0.0, 2.2, 0.0, 17.9, 18.4,
```

Muitos dos outliers identificados são 0.0, indicando que alguns clientes não utilizam o serviço de chamadas internacionais. O que pode ser analisado como clientes que não tem a necessidade de realizar chamadas internacionais.

A partir das análises do gráfico e dos outliers do último código, podemos ver clientes que fazem uso mais intenso do serviço e que são os mais propensos a churn. Isso pode ser interessante para identificar um segmento de clientes que talvez precise de um plano especial ou de um melhor acompanhamento para evitar churn (cancelamento).

Conclusão
Para cada análise buscamos padrões que podem indicar uma relação entre as características em relação ao churn, que será nossa variável de saída. Essa visualização nos ajudou a entender melhor sobre as características e obter um bom entendimento sobre elas e sobre o comportamento dos clientes

Observação: Não adicionamos todas as análises individuais neste documento, mas todas estão disponíveis no Colab.

3. Preparação dos Dados

Chegamos na parte de preparação dos dados, aqui iremos preparar todas as características, inclusive as que não fizemos uma análise aprofundada. Pois iremos utilizar a maioria delas na modelagem.

State

Iremos optar por retirar a State por sua nula correlação com churn e por ser uma coluna que não é boa para fazer normalização. Pois mesmo observando e analisando todas as possibilidades dela ter alguma interferência em conjunto com outras colunas não conseguimos sentir confiança em manter ela. Achando assim que ela é desnecessária.

```
[ ]  df = df.drop(["State"], axis=1)
```

Account length

Optamos por não remover os outliers dessa coluna, assim como faremos com as outras colunas pois elas podem ser indicadores importantes de padrões de uso que levam ao churn.

Para melhorar o desempenho do algoritmo quando chegarmos na modelagem, iremos normalizar os dados utilizando o MinMaxScaler (caso optar por fazer limpeza colocar o Capped da nova coluna)

```
df['Account length'] = MinMaxScaler().fit_transform(df[['Account length']])
```

```
df['Account length']
```

	Account length
0	0.524793
1	0.438017
2	0.561983
3	0.342975
4	0.305785
...	...
2661	0.322314
2662	0.789256
2663	0.276860
2664	0.111570
2665	0.301653

2666 rows × 1 columns

Area code

Nesse primeiro momento decidimos por manter essa coluna e fazer um tratamento usando One-Hot Encoding.

```
[ ]  # Aplicando One-Hot Encoding na coluna 'AREA CODE'
     df = pd.get_dummies(df, columns=["Area code"], prefix="AreaCode", drop_first=True)

     # Convertendo as colunas geradas para o tipo inteiro (se necessário)
     df[["AreaCode_415", "AreaCode_510"]] = df[["AreaCode_415", "AreaCode_510"]].astype(int)
```

A coluna original "Area code" será substituída por duas novas colunas, "AreaCode_415" e "AreaCode_510" e cada linha da base de dados terá 0 ou 1, indicando qual o código da área. Utilizamos o drop_first=True e o código da área 408 não será representado explicitamente, mas sim implicitamente, onde o valor das outras colunas for igual a 0.

International plan

Aqui iremos normalizar a variável categórica International plan com valores "No" e "Yes" para 0 e 1. Para gantantir que o modelo de aprendizado de máquina possa processar e aprender com essas variáveis de forma eficaz.

```
df['International plan'] = df['International plan'].replace({'No': 0, 'Yes': 1})
```

```
df['International plan']
```

	International plan
0	0
1	0
2	0
3	1
4	1
...	...
2661	0
2662	0
2663	0
2664	0
2665	0

2666 rows × 1 columns

Voice mail plan

Iremos fazer o mesmo com a variável Voice main plan, e iremos normaliza-la para valores 0 e 1. Para garantir que o modelo de aprendizado de máquina possa processar e aprender com essas variáveis de forma eficaz.

```
df['Voice mail plan'] = df['Voice mail plan'].replace({'No': 0, 'Yes': 1})
```

```
df['Voice mail plan']
```

	Voice mail plan
0	1
1	1
2	0
3	0
4	0
...	...
2661	0
2662	1
2663	0
2664	0
2665	1

2666 rows × 1 columns

Number vmail messages

Por essa coluna ter uma correlação significativa com o churn e que a maioria dos clientes que cancelaram o plano utilizam o serviço de voicemail, optamos por não fazer a limpeza dos dados considerados outliers, pois eles podem ser indicadores importantes de padrões de uso que levam ao churn.

Para melhorar o desempenho do algoritmo quando chegarmos na modelagem, iremos normalizar os dados utilizando o MinMaxScaler (caso seja optado por fazer a limpeza de outliers, modificar o nome da coluna)

```
df['Number vmail messages'] = MinMaxScaler().fit_transform(df[['Account length']])
```

```
df['Number vmail messages']
```

	Number vmail messages
0	0.524793
1	0.438017
2	0.561983
3	0.342975
4	0.305785
...	...
2661	0.322314
2662	0.789256
2663	0.276860
2664	0.111570
2665	0.301653

2666 rows × 1 columns

Total day minutes

Por essa coluna ter uma correlação significativa com o churn e que a maioria dos clientes que cancelam o plano utilizam o serviço de voicemail, optamos por não fazer a limpeza dos dados considerados outliers, pois eles podem ser indicadores importantes de padrões de uso que levam ao churn.

Para melhorar o desempenho do algoritmo quando chegarmos na modelagem, iremos normalizar os dados utilizando o MinMaxScaler

```
df['Total day minutes'] = MinMaxScaler().fit_transform(df[['Total day minutes']])
```

```
df['Total day minutes']
```

	Total day minutes
0	0.755701
1	0.460661
2	0.693843
3	0.853478
4	0.475200
...	...
2661	0.383979
2662	0.445268
2663	0.658780
2664	0.515393
2665	0.668187

2666 rows × 1 columns

Total day calls

Optamos por não remover os outliers dessa coluna, assim como faremos com as outras colunas pois elas podem ser indicadores importantes de padrões de uso que levam ao churn.

Para melhorar o desempenho do algoritmo quando chegarmos na modelagem, iremos normalizar os dados utilizando o MinMaxScaler (caso optar por fazer limpeza colocar o Capped da nova coluna)

```
df['Total day calls'] = MinMaxScaler().fit_transform(df[['Total day calls']])
```

```
df['Total day calls']
```

	Total day calls
0	0.68750
1	0.76875
2	0.71250
3	0.44375
4	0.70625
...	...
2661	0.61250
2662	0.48125
2663	0.35625
2664	0.68125
2665	0.70625

2666 rows x 1 columns

Total day charge

Dado nosso objetivo principal de classificar churn e tendo em vista que os clientes onde estão os outliers podem ter uma alta correlação com churn, mantê-los pode ser útil. Eles podem ser indicadores importantes de padrões de uso que levam ao churn.

Para melhorar o desempenho do algoritmo quando chegarmos na modelagem, iremos normalizar os dados utilizando o MinMaxScaler

```
df['Total day charge'] = MinMaxScaler().fit_transform(df[['Total day charge']])
```

```
df['Total day charge']
```

	Total day charge
0	0.755701
1	0.460597
2	0.693830
3	0.853454
4	0.475184
...	...
2661	0.383970
2662	0.445171
2663	0.658786
2664	0.515426
2665	0.668176

2666 rows × 1 columns

Total eve minutes

Como visto na análise exploratória, considerando que tanto os outliers superiores quanto inferiores podem fornecer insights valiosos sobre comportamentos extremos que influenciam o churn, foi decidido não remover esses outliers de imediato.

Mas para melhorar o desempenho do algoritmo quando chegarmos na modelagem, iremos normalizar os dados utilizando o MinMaxScaler

```
df['Total eve minutes'] = MinMaxScaler().fit_transform(df[['Total eve minutes']])
```

```
df['Total eve minutes']
```

	Total eve minutes
0	0.542755
1	0.537531
2	0.333242
3	0.170195
4	0.407754
...	...
2661	0.521584
2662	0.592521
2663	0.421776
2664	0.794061
2665	0.731097

2666 rows × 1 columns

Total eve calls

Aqui também decidimos por não remover os outliers da nossa coluna inicialmente pois eles podem ser importantes para o entedimento do churn. Valores muito altos ou muito baixos podem representar diferentes razões para os clientes cancelarem o plano. O que pode influenciar diretamente a classificação do churn.

Mas para melhorar o desempenho do algoritmo quando chegarmos na modelagem, iremos normalizar os dados utilizando o MinMaxScaler

```
df['Total eve calls'] = MinMaxScaler().fit_transform(df[['Total eve calls']])
```

```
df['Total eve calls']
```

	Total eve calls
0	0.582353
1	0.605882
2	0.647059
3	0.517647
4	0.717647
...	...
2661	0.400000
2662	0.741176
2663	0.323529
2664	0.341176
2665	0.482353

2666 rows × 1 columns

Total eve charge

Optamos por não remover os outliers da nossa coluna inicialmente pois eles podem ser importantes para o entendimento do churn. Valores muito altos ou muito baixos podem representar diferentes razões para os clientes cancelarem o plano. O que pode influenciar diretamente a classificação do churn. Precisamos entender se o comportamento de clientes atípicos pode ter grande influência.

Mas para melhorar o desempenho do algoritmo quando chegarmos na modelagem, iremos normalizar os dados utilizando o MinMaxScaler

```
df['Total eve charge'] = MinMaxScaler().fit_transform(df[['Total eve charge']])
```

```
df['Total eve charge']
```

	Total eve charge
0	0.542866
1	0.537690
2	0.333225
3	0.170171
4	0.407959
...	...
2661	0.521514
2662	0.592688
2663	0.421870
2664	0.794241
2665	0.731155

2666 rows × 1 columns

Total night minutes

Optamos por não remover os 22 outliers da nossa coluna inicialmente pois eles podem ser importantes para o entendimento do churn ou pelo menos não interferirem na nossa modelagem.

Valores muito altos ou muito baixos podem representar diferentes razões para os clientes cancelarem o plano. O que pode influenciar diretamente a classificação do churn. Precisamos entender se o comportamento de clientes atípicos pode ter grande influência.

Mas para melhorar o desempenho do algoritmo quando chegarmos na modelagem, iremos normalizar os dados utilizando o MinMaxScaler

```
df['Total night minutes'] = MinMaxScaler().fit_transform(df[['Total night minutes']])
```

```
df['Total night minutes']
```

	Total night minutes
0	0.572161
1	0.599772
2	0.338457
3	0.436095
4	0.407629
...	...
2661	0.505835
2662	0.670083
2663	0.420154
2664	0.421862
2665	0.562767

2666 rows × 1 columns

Total night calls

Optamos por não remover os outliers da nossa coluna inicialmente pois eles podem ser importantes para o entendimento do churn.

Valores muito altos ou muito baixos podem representar diferentes razões para os clientes cancelarem o plano. O que pode influenciar diretamente a classificação do churn. Precisamos entender se o comportamento de clientes atípicos pode ter grande influencia.

Mas para melhorar o desempenho do algoritmo quando chegarmos na modelagem, iremos normalizar os dados utilizando o MinMaxScaler

```
df['Total night calls'] = MinMaxScaler().fit_transform(df[['Total night calls']])
```

```
df['Total night calls']
```

	Total night calls
0	0.436090
1	0.526316
2	0.533835
3	0.421053
4	0.661654
...	...
2661	0.714286
2662	0.375940
2663	0.676692
2664	0.436090
2665	0.330827

2666 rows × 1 columns

Total night charge

Optamos por manter os outliers da nossa coluna inicialmente pois eles podem ser importantes para o entendimento do churn.

Valores muito altos ou muito baixos podem representar diferentes razões para os clientes cancelarem o plano. O que pode influenciar diretamente a classificação do churn. Esses valores indicam comportamento real dos clientes e pode fornecer compreensões relevantes durante a modelagem.

Mas para melhorar o desempenho do algoritmo quando chegarmos na modelagem, iremos normalizar os dados utilizando o MinMaxScaler

```
df['Total night charge'] = MinMaxScaler().fit_transform(df[['Total night charge']])
```

```
df['Total night charge']
```

	Total night charge
0	0.572152
1	0.600000
2	0.338608
3	0.436076
4	0.407595
...	...
2661	0.505696
2662	0.670253
2663	0.420253
2664	0.422152
2665	0.562658

2666 rows × 1 columns

Total intl minutes

Iremos optar por manter os outliers nesse primeiro momento pois eles representam os comportamentos reais dos clientes, como os que não usam o serviço ou os que têm uso intenso, e podem revelar padrões e ajudar na identificação de churn.

Mas para melhorar o desempenho do algoritmo quando chegarmos na modelagem, iremos normalizar os dados utilizando o MinMaxScaler

```
df['Total intl minutes'] = MinMaxScaler().fit_transform(df[['Total intl minutes']])
```

```
df['Total intl minutes']
```

	Total intl minutes
0	0.500
1	0.685
2	0.610
3	0.330
4	0.505
...	...
2661	0.590
2662	0.495
2663	0.480
2664	0.705
2665	0.685

2666 rows × 1 columns

Total intl calls

Optamos por não remover os outliers nessa coluna nesse primeiro momento porque esses dados podem ser mantidos para dividir os clientes em diferentes segmentos com suas devidas características e observar quais os serviços específicos que cada grupo de clientes necessitam de acordo com o perfil de uso. Esses valores podem indicar comportamento real dos clientes e pode fornecer compreensões relevantes durante a modelagem.

Mas para melhorar o desempenho do algoritmo quando chegarmos na modelagem, iremos normalizar os dados utilizando o MinMaxScaler. A intenção é reduzir a influência de grandes valores e trazer os dados para uma escala comum.

```
df['Total intl calls'] = MinMaxScaler().fit_transform(df[['Total intl calls']])
```

```
df['Total intl calls']
```

	Total intl calls
0	0.15
1	0.15
2	0.25
3	0.35
4	0.15
...	...
2661	0.25
2662	0.30
2663	0.20
2664	0.30
2665	0.20

2666 rows × 1 columns

Total intl charge

Optamos por não remover os outliers nessa coluna inicialmente pois eles podem ser importantes para o entendimento do churn. Esses valores podem indicar comportamento real dos clientes e pode fornecer compreensões relevantes durante a modelagem.

Mas para melhorar o desempenho do algoritmo quando chegarmos na modelagem, iremos normalizar os dados utilizando o MinMaxScaler. A intenção é reduzir a influência de grandes valores e trazer os dados para uma escala comum.

```
df['Total intl charge'] = MinMaxScaler().fit_transform(df[['Total intl charge']])
```

```
df['Total intl charge']
```

	Total intl charge
0	0.500000
1	0.685185
2	0.609259
3	0.329630
4	0.505556
...	...
2661	0.590741
2662	0.494444
2663	0.479630
2664	0.705556
2665	0.685185

2666 rows × 1 columns

Customer service calls

Manter os outliers é uma decisão inicial considerando que eles representam valores que podem indicar comportamento real dos clientes diante do churn e pode fornecer compreensões relevantes durante a modelagem para o cancelamento do plano, já que chamadas frequentes se revelam com maior número de churn.

Mas para melhorar o desempenho do algoritmo quando chegarmos na modelagem, iremos normalizar os dados utilizando o MinMaxScaler

```
df['Customer service calls'] = MinMaxScaler().fit_transform(df[['Customer service calls']])
```

```
df['Customer service calls']
```

	Customer service calls
0	0.111111
1	0.111111
2	0.000000
3	0.222222
4	0.333333
...	...
2661	0.222222
2662	0.222222
2663	0.333333
2664	0.222222
2665	0.000000

2666 rows × 1 columns

Churn

Nesse primeiro momento optamos por não fazer o balanceamento de classes, dado que na observação do pandas profiling observamos esse grande desbalanceamento. Iremos fazer a primeira modelagem sem esse tratamento e verificar sua resposta aos modelos e caso necessário voltaremos a parte de tratamento e faremos o balanceamento.

Mas neste momento iremos transformar os valores de True e False para 1 e 0. Mesmo entendendo que os modelos conseguem lidar com valores booleanos já transformando eles em zeros e uns, deixamos explícito para termos mais controle sobre a codificação.

```
df['Churn'] = df['Churn'].astype(int)
```

```
df['Churn']
```

	Churn
0	0
1	0
2	0
3	0
4	0
...	...
2661	0
2662	0
2663	0
2664	0
2665	0

2666 rows × 1 columns

Nesse segundo momento iremos fazer o balanceamento de classe, dado um grande número de erros na matriz de confusão dos modelos. Sua acurácia pode ter gerado um falso positivo. Por este motivo iremos fazer uma redução da classe majoritária, pegando um número randômico dela e a tornando o mesmo tamanho da classe minoritária.

```
[ ]  # Antes do balanceamento
     print('Distribuição original:', Counter(y_train))
```

```
Distribuição original: Counter({False: 1595, True: 271})
```

```
[ ]  rus = RandomUnderSampler(random_state=42)

     X_res, y_res = rus.fit_resample(X_train, y_train)

     # Após o balanceamento
     print('Distribuição após undersampling:', Counter(y_res))
```

```
Distribuição após undersampling: Counter({0: 271, 1: 271})
```

Inicialmente tínhamos uma desproporção na classe de churn, onde havia muito mais valores na classe 0 (não churn) em comparação com a classe 1 (churn). Então foi utilizada a técnica Undersampling para ajustar essa desproporção da classe, como resultado o número da classe majoritária foi reduzido até ficar com o mesmo valor da minoritária.

Esse balanceamento foi feito depois da divisão dos dados para treinamento e testes.

Caracterização da Base de Dados

Após o processo de preparação dos dados restaram 19 características das 20 iniciais. Decidimos fazer a modelagem com todas as características por elas poderem ter correlação maior quando unidas a mais de uma característica.

A base de dados final era composta por 2666 instâncias. O conjunto de dados foi particionado em 70% para treinamento (1866.2 instâncias) e 30% para teste (799.8 instâncias). Após o balanceamento esse número foi reduzido como visto na última classe (Churn) da preparação dos dados.

Os atributos numéricos foram normalizados para o intervalo [0, 1] para garantir que todas as variáveis tivessem a mesma escala.

4. Modelagem

a. 5 Técnicas Escolhidas e Justificativas

As cinco técnicas escolhidas para a simulação são Regressão Logística, Random Forest, Gradient Boosting, Support Vector e KNN. Cada uma foi selecionada por suas características específicas que se adequam bem ao problema de classificação de churn.

Regressão Logística: A Regressão Logística foi uma escolha por ser uma boa base para problemas de classificação binária. Ela produz uma probabilidade de que uma instância pertença a uma das duas classes, o que é bom para o nosso modelo de churn ou não churn.

Random Forest: O Random Forest foi uma técnica escolhida por ser poderosa para classificação, principalmente em problemas de dados desbalanceados como o churn, por causa da sua capacidade de manipular grandes conjuntos de dados e capturar relações complexas entre os atributos. Além de ser um ensemble, que refere-se a um método que combina várias técnicas de modelagem para melhorar a performance e a robustez do modelo. É uma técnica baseada em conjuntos que utiliza múltiplas árvores de decisão para melhorar a precisão e reduzir o risco de overfitting.

Gradient Boosting: O Gradient Boosting foi escolhido por ele ser adequado para problemas onde precisamos de alta performance preditiva, já que precisamos de precisão

quando se trata de manter clientes, onde perdê-los teria um custo alto. Além de também ser um ensemble.

Support Vector: O Support Vector foi escolhido por ser uma técnica poderosa para classificação e por ser útil em problemas de alta dimensionalidade, onde os dados possuem um número grande de características em comparação com o número de exemplos disponíveis, que é o caso da nossa base de dados. Além de tentar elevar ao máximo a margem entre as classes, o que pode ser vantajoso quando as classes são difíceis de separar, que é o nosso caso de churn.

KNN: O KNN foi escolhido por ser uma técnica boa para classificação, mesmo sendo um modelo simples. Além de ser uma técnica a ser utilizada como critério de avaliação, tendo em vista que não escolhemos muitas das que encontramos no projeto que encontramos.

b. Configurações experimentais utilizadas – quais hiperparâmetros foram testados para cada técnica? Como foi realizado o particionamento dos dados? (ex. 70% para treinamento e 30% para teste). Foram realizadas repetições do experimento?

Hiperparâmetros Testados:

Parâmetro de regularização (`C`): -5, 8, 15

Penalidade (`penalty`): L1, L2

Solvers: liblinear, lbfgs

Particionamento dos Dados:

70% dos dados foram utilizados para treinamento e 30% para teste.

Foi utilizado 5-fold cross-validation para avaliar a robustez dos modelos, especialmente para Random Forest, Gradient Boosting, SVM e KNN

Repetições do Experimento:

Para garantir a estabilidade dos resultados, foram realizadas várias repetições do experimento. Primeiro utilizamos o modelo de Regressão Logística, em seguida o mesmo modelo com ajuste dos hiperparâmetros. Em seguida utilizamos o Random Forest e após ele novamente com Cross-Validation. Depois, com os dados não balanceados, fizemos o treinamento dos modelos sem ajustes ou Cross-Validation.

Após o balanceamento fizemos uma sequência de treinamento dos mesmos modelos sem e com o Cross-Validation, com 5 folds cada, em que o modelo é treinado e testado 5 vezes, cada vez usando uma parte diferente dos dados para teste e as demais para treinamento. Assim garantimos uma avaliação mais confiável.

c. Métricas Computadas

Computamos inicialmente as técnicas de acurácia e matriz de confusão, mas quando fizemos a validação cruzada acrescentamos um relatório de classificação que incluia precisão, recall e f1-score.

Sem balanceamento:

Regressão Logística:

```
Métricas de Treinamento
Acurácia =  0.8606645230439443

Matriz de Confusão
[[1559   36]
 [ 224   47]]
Métricas de Testes
Acurácia =  0.8675

Matriz de Confusão
[[672  11]
 [ 95  22]]
```

Regressão Logística com ajuste de hiperparâmetro:

```
Resultados:

Melhores Parâmetros: {'C': 31.622776601683793, 'penalty': 'l2', 'solver': 'liblinear'}

Melhor Acurácia: 0.8617353156227152
```

Random Forest:

```
Métricas de Treinamento
Acurácia =  1.0

Matriz de Confusão
[[1595    0]
 [   0  271]]
Métricas de Testes
Acurácia =  0.94375

Matriz de Confusão
[[683   0]
 [ 45  72]]
```

Random Forest com Cross-Validation:

```
Acurácia em cada fold (não balanceada):  [0.94652406 0.9383378  0.95710456 0.95174263 0.93565684]
Acurácia média (não balanceada):  0.9458731774454847
Desvio padrão das acurácias (não balanceada):  0.008027122882641914
Relatório de Classificação (não balanceada):
              precision    recall  f1-score   support

           0       0.95      0.99      0.97      1595
           1       0.92      0.68      0.78       271

    accuracy                           0.94      1866
   macro avg       0.93      0.83      0.87      1866
weighted avg       0.94      0.94      0.94      1866
```

Gradient Boosting:

```
Métricas de Treinamento
Acurácia =  0.9769560557341908

Matriz de Confusão
[[1594    1]
 [  42  229]]
Métricas de Testes
Acurácia =  0.94875

Matriz de Confusão
[[680   3]
 [ 38  79]]
```

Support Vector:

```
Métricas de Treinamento
Acurácia =  0.9008574490889604

Matriz de Confusão
[[1589    6]
 [ 179   92]]
Métricas de Testes
Acurácia =  0.88375

Matriz de Confusão
[[680   3]
 [ 90  27]]
```

KNN:

```
Métricas de Treinamento
Acurácia =  0.9067524115755627

Matriz de Confusão
[[1569   26]
 [ 148  123]]
Métricas de Testes
Acurácia =  0.88375

Matriz de Confusão
[[672  11]
 [ 82  35]]
```

Com balanceamento:

Regressão Logística:

```
Métricas de Treinamento
Acurácia =  0.7841328413284133

Matriz de Confusão
[[213  58]
 [ 59 212]]
Métricas de Testes
Acurácia =  0.75625

Matriz de Confusão
[[528 155]
 [ 40  77]]
```

Regressão Logística com ajuste de hiperparâmetro:

```
Resultados:

Melhores Parâmetros: {'C': 0.4393970560760795, 'penalty': 'l2', 'solver': 'lbfgs'}

Melhor Acurácia: 0.773054706082229
```

Random Forest:

```
Métricas de Treinamento
Acurácia =  1.0

Matriz de Confusão
[[271   0]
 [  0 271]]
Métricas de Testes
Acurácia =  0.885

Matriz de Confusão
[[611  72]
 [ 20  97]]
```

Random Forest com Cross-Validation:

```
Acurácia em cada fold (balanceada):  [0.87155963 0.87155963 0.81481481 0.89814815 0.84259259]
Acurácia média (balanceada):  0.8597349643221202
Desvio padrão das acurácias (balanceada):  0.02851981420665028
Relatório de Classificação (balanceada):
              precision    recall  f1-score   support

           0       0.85      0.85      0.85       271
           1       0.85      0.85      0.85       271

    accuracy                           0.85       542
   macro avg       0.85      0.85      0.85       542
weighted avg       0.85      0.85      0.85       542


Matriz de Confusão:
[[231  40]
 [ 40 231]]
```

Gradient Boosting:

```
Métricas de Treinamento
Acurácia =  0.966789667896679

Matriz de Confusão
[[270   1]
 [ 17 254]]
Métricas de Testes
Acurácia =  0.8975

Matriz de Confusão
[[624  59]
 [ 23  94]]
```

Gradient Boosting com Cross-Validation:

```
Acurácia em cada fold (balanceada):  [0.83486239 0.88073394 0.87037037 0.88888889 0.85185185]
Acurácia média (balanceada):  0.8653414882772681
Desvio padrão das acurácias (balanceada):  0.01963488457374813
Relatório de Classificação (balanceada):
              precision    recall  f1-score   support

           0       0.85      0.88      0.87       271
           1       0.88      0.85      0.86       271

    accuracy                           0.86       542
   macro avg       0.86      0.86      0.86       542
weighted avg       0.86      0.86      0.86       542


Matriz de Confusão:
[[239  32]
 [ 42 229]]
```

Support Vector:

```
Métricas de Treinamento
Acurácia =  0.8302583025830258

Matriz de Confusão
[[225  46]
 [ 46 225]]
Métricas de Testes
Acurácia =  0.80625

Matriz de Confusão
[[562 121]
 [ 34  83]]
```

Support Vector com Cross-Validation:

```
Acurácia em cada fold (balanceada):  [0.74311927 0.83486239 0.73148148 0.83333333 0.80555556]
Acurácia média (balanceada):  0.7896704043493035
Desvio padrão das acurácias (balanceada):  0.044168005980514145
Relatório de Classificação (balanceada):
              precision    recall  f1-score   support

           0       0.78      0.81      0.79       271
           1       0.80      0.77      0.79       271

    accuracy                           0.79       542
   macro avg       0.79      0.79      0.79       542
weighted avg       0.79      0.79      0.79       542


Matriz de Confusão:
[[219  52]
 [ 62 209]]
```

KNN:

```
Métricas de Treinamento
Acurácia =  0.8523985239852399

Matriz de Confusão
[[236  35]
 [ 45 226]]
Métricas de Testes
Acurácia =  0.7975

Matriz de Confusão
[[557 126]
 [ 36  81]]
```

KNN com Cross-Validation:

```
Acurácia em cada fold (balanceada):  [0.71559633 0.79816514 0.73148148 0.77777778 0.7962963 ]
Acurácia média (balanceada):  0.7638634046890928
Desvio padrão das acurácias (balanceada):  0.034059978101141024
Relatório de Classificação (balanceada):
              precision    recall  f1-score   support

           0       0.75      0.79      0.77       271
           1       0.78      0.74      0.76       271

    accuracy                           0.76       542
   macro avg       0.76      0.76      0.76       542
weighted avg       0.76      0.76      0.76       542


Matriz de Confusão:
[[213  58]
 [ 70 201]]
```

Optamos por fazer o Cross-Validation em todas depois do balanceamento pois elas se apresentaram muito melhores e queríamos ter certeza da sua capacidade de prever o churn e de não estar sofrendo overfitting.

Em comparação com os benchmarks que fizemos a comparação, não conseguimos alcançar os bons resultados deles, mas ainda assim obtivemos resultados satisfatórios. Para melhorar mais esses resultados teríamos que avaliar ainda mais a fundo.

d. Avaliação e Prescrição do Modelo

Quando equilibramos as classes com o balanceamento, os modelos detectam melhor quando as pessoas estão saindo do plano, especialmente na detecção das classes minoritárias. Embora os modelos balanceados não tenham uma acurácia que supere os não balanceados, os balanceados ainda fazem um trabalho melhor na detecção de clientes que deram churn.

O Random Forest seguido do Gradient Boosting são os modelos mais indicados para serem implantados, devido ao seu equilíbrio robusto entre precisão, recall, e capacidade de generalização.

5. Avaliação

a. Discussão dos Resultados

Objetivos de Negócio e Mineração de Dados:

O objetivo principal do nosso projeto é identificar os clientes que estão em risco de abandonar os serviços de telecomunicação de uma empresa e assim conseguir classificar os clientes que estão mais propensos a darem churn, fazendo promoções ou melhorias no serviço para evitar esse abandono e assim aumentar a lucratividade da empresa.

Enfrentamos problemas na nossa base de dados com o desbalanceamento de classes, onde tínhamos muito mais clientes que não deram churn do que clientes que deram churn. Esse desequilíbrio de classes nos deu resultados com alta acurácia, mas com um poder de previsão da classe minoritária (churn) muito baixo, com muitos erros (falsos negativos).

Os resultados pós-balanceamento foram melhores em relação a classificação de churn na matriz de confusão, em comparação com os resultados não balanceados. Mas a acurácia geral teve uma queda. A métrica de recall nos modelos com Cross-Validation mostraram que o modelo realmente melhorou. E o recall alto para a classe de churn significa que mais clientes em risco foram identificados corretamente.

Os modelos como Random Forest, Gradient Boosting, e Support Vector, após o balanceamento, mostraram um equilíbrio mais adequado entre precisão e recall, especialmente para a classe de churn. O Random Forest se destacou como o modelo com melhor desempenho em termos de precisão e recall equilibrados, sugerindo que ele poderia ser altamente eficaz em identificar clientes que podem cancelar seus serviços.

De acordo com todas as análises feitas, foi possível atingir com uma precisão razoável, (em um conjunto de dados que ainda pode ser mais ajustado em futuras minerações) o objetivo do negócio, através da aprendizagem de máquina que foi capaz de prever, os clientes que deram churn

b. Análise dos Modelos para Implantação

Para considerar a implantação de um modelo para nosso projeto, precisamos avaliar qual deles pode ser integrado e obter bons resultados de performance e de classificação de churn, de forma a não cometer muitos erros para não trazer prejuízos para a empresa.

Embora todos os outros modelos tenham apresentado bons resultados após o balanceamento de classes no que diz respeito a classificação de churn, os melhores resultados obtidos foram do modelo Random Forest e Gradient Boosting em seguida. Indicando que esses são os melhores modelos para serem encaminhados para a implantação. Com as avaliações mais rigorosas com Cross-Validation eles ofereceram um

bom equilíbrio entre precisão e recall, são robustos e consistentes. Eles podem ser aplicados em testes inicialmente, no fluxo de trabalho da empresa, para ver qual dos dois conseguem manter uma consistência na identificação de clientes em risco e contribuindo para a redução de churn.

Ambos os modelos podem ser implementados em pipelines de produção para prever o churn em tempo real. Além disso, o processo de ajuste e balanceamento das classes pode ser refinado com o tempo, especialmente com mais dados disponíveis, o que permitirá a empresa ajustar suas estratégias de retenção de clientes de maneira mais proativa.