📌 Extracão

```
# Importação das bibliotecas
import pandas as pd
import numpy as np
```

```
# Leitura o arquivo
df = pd.read_json('/content/TelecomX_Data.json')
```

```
# Visualização do DataFrame
df.head()
```

Mostrar saída oculta

Próximas etapas: Gerar código com df | New interactive sheet

🔧 Transformação

```
# Copiando as 2 primeiras colunas no DataFrame df
df_aux = df[['customerID', 'Churn']].copy()
df_aux.head()
```

Mostrar saída oculta

Próximas etapas: Gerar código com df_aux | New interactive sheet

```
# Normalizando os dados das colunas aninhadas
df_customer = pd.json_normalize(df['customer'])
df_phone = pd.json_normalize(df['phone'])
df_internet = pd.json_normalize(df['internet'])
df_account = pd.json_normalize(df['account'])
```

```
# Concatenando os DataFrames df_aux, df_customer, df_phone, df_internet, df_account
df_normalizado = pd.concat([df_aux, df_customer, df_phone, df_internet, df_account], axis=1)
df_normalizado.head()
```

Mostrar saída oculta

```
# verificando valores duplicados
df_normalizado.duplicated().unique()
```

```
array([False])
```

```
# Verificando valores nulos no DF
df_normalizado.isnull().sum()
```

Mostrar saída oculta

```
# Verificando valores vazios não nulos em Churn
df_normalizado['Churn'].value_counts()
```

Mostrar saída oculta

```
# Temos 224 valores vazios não nulos no Churn, iremos substituí-los e excluí-los.
df_normalizado['Churn'] = df_normalizado['Churn'].replace('', np.nan)
```

```
# Verificando a transformação dos valores em NaN
df_normalizado['Churn'].unique()
```

```
# Retirando os valores nulos
df_normalizado.dropna(inplace=True)
```

```
# Verificando as informações do DF
df_normalizado.info()
```

Mostrar saída oculta

```
'''
```

```
Transformação da coluna Charges.Total de 'object' para 'float64' Ocorrerá um
erro, pois estamos em uma coluna que contém valores vazios.
'''
df_normalizado['Charges.Total'] = df_normalizado['Charges.Total'].astype(np.float64)
```

Mostrar saída oculta

Próximas etapas: Explicar o erro

```
# Utilizaremos o método pd.to_numeric para converter dados vazios em tipo 'float'
df_normalizado['Charges.Total'] = pd.to_numeric(df_normalizado['Charges.Total'], errors='coerce')
```

```
# Confirmando se temos dados vazios e não vazios
df_normalizado['Charges.Total'].isna().unique()
```

```
array([False,  True])
```

```
'''
Analisando a coluna 'Charges.Total', que contém os valores NaN, vemos que associada
a coluna 'tenure', que são os meses de contrato, vemos que para estes 11 clientes,
o contrato foi de, no máximo, 1 mês.
'''
df_nan = df_normalizado[df_normalizado['Charges.Total'].isna()][['Charges.Total', 'tenure']]
df_nan
```

Mostrar saída oculta

Próximas etapas: Gerar código com df_nan New interactive sheet

```
# Como as 11 ocorrências representam apenas 0,15% dos dados, optamos por excluí-los.
df_normalizado.dropna(inplace=True)
```

```
# Padronizando os nomes das colunas que estão na nomenclatura Pascal Case para Snake Case
df_normalizado.columns = df_normalizado.columns.str.replace(r'([a-z])([A-Z])', r'\1_\2', regex=True)
```

```
# Padronizando as duas últimas colunas que possuem um separador ponto(.)
df_normalizado.columns = df_normalizado.columns.str.replace(r'([a-z]).([A-Z])', r'\1_\2', regex=True)
```

```
# Padronizando os rótulos das colunas em letras minúsculas
df_normalizado.columns = df_normalizado.columns.str.lower()
```

```
# Criando uma lista para auxiliar na padronização dos dados
lista = df_normalizado.select_dtypes(include=['object']).columns.tolist()
```

```
# Deixaremos apenas os dados da coluna 'customerID' em maiúsculas (dados originais)
lista.remove('customer_id')
```

```
# Padronizando os dados do tipo string em letras minúsculas
df_normalizado[lista] = df_normalizado[lista].apply(lambda x: x.str.lower())
```

```
'''
Definindo uma lista para remover da lista inicial. O intuito é refinar os dados
para serem utilizados na análise, alterando as opções 'yes' e 'no' para 1 e 0
respectivamente.
'''
remover_colunas = ['gender', 'contract', 'payment_method']

lista = [item for item in lista if item not in remover_colunas]
```

```
'''
Criando uma segunda lista que contém as opções "yes", "no" e "no internet service".
Posteriormente utilizaremos esta segunda lista para padronizar os dados em "no = 0"
"yes = 1" e "no internet service = 2".
'''
lista_2 = lista[5:12]
```

```
'''
Removeremos as colunas que possuem as 3 opções "yes, no e no internet service"
da lista inicial
'''
lista = [item for item in lista if item not in lista_2]
```

```
'''
Criar um novo DataFrame para alterar os dados "yes", "no", "no internet service"
e preservar o original
'''
df_final = df_normalizado.copy()
```

```
'''
Padronizando os valores "yes" e "no" em que: o 0(zero) indica "no" e o
número 1(um) indica "yes".
'''
for col in lista:
    df_final[col] = df_normalizado[col].replace({'no': 0, 'yes': 1})
```

Mostrar saída oculta

```
'''
Padronizando os valores "yes", "no" e "no internet service' em que:
o 0(zero) indica "no", o número 1(um) indica "yes" e 2(dois) indica "no internet service.
'''
for col in lista_2:
    df_final[col] = df_normalizado[col].replace({'no': 0, 'yes': 1, 'no internet service': 2})
```

Mostrar saída oculta

```
# Criando uma nova coluna que nos mostrará os gastos diários do cliente
df_final['contas_diarias'] = round((df_final['charges_monthly'] / 30), 2)
```

```
# Criando uma nova coluna que nos mostrará os gastos diários do cliente
df_normalizado['contas_diarias'] = round((df_normalizado['charges_monthly'] / 30), 2)
```

```
df_final.info()
```

Mostrar saída oculta

```
# Salvando nosso DataFrame normalizado em format .csv
df_final.to_csv('telecom_x_data.csv', index=False)
```

📊 Carga e análise

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

```
df_normalizado.head()
```

Mostrar saída oculta

```
df_numericas = df_normalizado.select_dtypes(include=['int64', 'float64'])
df_numericas.head()
```

Mostrar saída oculta

Próximas etapas: Gerar código com `df_numericas` New interactive sheet

```
# Visualizando as estatísticas descritivas
df_numericas.describe()
```

Mostrar saída oculta

```
# Adicionando as cores como variáveis do projeto
VERMELHO_1, AZUL_1, CINZA_1 = "#e23155", "#203f75", "#ebebeb"
```

```
# Definindo uma função para salvar os gráficos
def salvar_figura(fig, nome_arquivo, dpi=300):
    fig.tight_layout()
    fig.savefig(nome_arquivo, dpi=dpi, bbox_inches="tight")
```

```
# Visualizando graficamente a evasão dos clientes (Churn)
fig = plt.figure(figsize=(8, 4))
ax = plt.gca() # Get the current axes
churn_counts = df_normalizado['churn'].value_counts()
plt.bar(churn_counts.index, churn_counts.values, color=[AZUL_1, VERMELHO_1])
```

```
for i, count in enumerate(churn_counts.values):
    plt.text(i, count, str(count), ha='center', va='bottom', fontsize=12)

plt.title('Análise da Evasão de Clientes (Geral)', fontsize=18)
plt.xticks(ha='right', fontsize=12)
plt.xlabel('Evasão de Clientes', fontsize= 12)
plt.yticks([])

ax.spines[['top', 'right', 'left', 'bottom']].set_visible(False)
salvar_figura(fig, 'churn_total.png')

plt.show()
```

Mostrar saída oculta

```
fig = plt.figure(figsize=(4, 4))
plt.pie(df_normalizado['churn'].value_counts(), labels=df_normalizado['churn'].value_counts().index, autopct='%1.2f%%', colo
plt.title('Análise de Evasão de Clientes')
plt.axis('equal')
salvar_figura(fig, 'churn_total_pie.png')
plt.show()
```

Mostrar saída oculta

Contagem de evasão por variáveis categóricas

Idade (Sênior Citizen)

```
churn_por_idade = pd.crosstab(df_final['senior_citizen'], df_final['churn'])
#display(churn_por_idade)

churn_idosos = churn_por_idade.apply(lambda x: x / x.sum(), axis=1)
#display(churn_idosos)

fig, ax = plt.subplots(figsize=(10, 6)) # Use plt.subplots to create figure and axes
churn_por_idade.plot(kind='bar', ax=ax, color=[AZUL_1, VERMELHO_1]) # Plot on 'ax', remove figsize from here
plt.title('Relação entre Idosos e Churn', fontsize= 18)
plt.xlabel('Idosos (0: Não, 1: Sim)')
plt.ylabel('Número de Clientes')
plt.xticks(rotation=0)
plt.legend(title='Churn', labels=['Não', 'Sim'])
plt.tight_layout()
for container in ax.containers:
    ax.bar_label(container, fmt='%d', label_type='edge')
ax.spines[['right', 'top']].set_visible(False)
salvar_figura(fig, 'churn_por_idade.png') # Save before showing
plt.show()
```

Mostrar saída oculta

Gênero (Gender)

```
churn_por_genero = pd.crosstab(df_final['gender'], df_final['churn'])
#display(churn_por_genero)

churn_genero = churn_por_genero.apply(lambda x: x / x.sum(), axis=1)
#display(churn_genero)

fig, ax = plt.subplots(figsize=(10, 6)) # Use plt.subplots to create figure and axes
churn_por_genero.plot(kind='bar', ax=ax, color=[AZUL_1, VERMELHO_1]) # Plot on 'ax', remove figsize from here
plt.title('Relação entre Gênero e Churn', fontsize=18)
plt.xlabel('Gênero')
plt.ylabel('Número de Clientes')
plt.xticks(rotation=0)
plt.legend(title='Churn', labels=['Não', 'Sim'])
plt.tight_layout()
for container in ax.containers:
    ax.bar_label(container, fmt='%d', label_type='edge')
ax.spines[['top', 'right']].set_visible(False)
salvar_figura(fig, 'churn_por_genero.png') # Save before showing
plt.show()
```

Mostrar saída oculta

Por Tipo de Contrato (Contract)

```
churn_por_contrato = pd.crosstab(df_final['contract'], df_final['churn'])
#display(churn_por_contrato)

churn_contrato = churn_por_contrato.apply(lambda x: x / x.sum(), axis=1)
#display(churn_contrato)

fig, ax = plt.subplots(figsize=(10, 6)) # Use plt.subplots to create figure and axes
churn_por_contrato.plot(kind='bar', ax=ax, color=[AZUL_1, VERMELHO_1]) # Plot on 'ax', remove figsize from here
plt.title('Relação entre Tipo de Contrato e Churn', fontsize=18)
plt.xlabel('Tipo de Contrato')
plt.ylabel('Número de Clientes')
plt.xticks(rotation=0, ha='right')
plt.legend(title='Churn', labels=['Não', 'Sim'])
plt.tight_layout()
for container in ax.containers:
    ax.bar_label(container, fmt='%d', label_type='edge')
ax.spines[['top', 'right']].set_visible(False)
salvar_figura(fig, 'churn_por_contrato.png') # Save before showing
plt.show()
```

Mostrar saída oculta

Por Método de Pagamento (Payment Method)

```
churn_metodo_pagamento = pd.crosstab(df_final['payment_method'], df_final['churn'])
#display(churn_metodo_pagamento)

churn_pagamento = churn_metodo_pagamento.apply(lambda x: x / x.sum(), axis=1)
#display(churn_pagamento)

fig, ax = plt.subplots(figsize=(10, 6)) # Use plt.subplots to create figure and axes
churn_metodo_pagamento.plot(kind='bar', ax=ax, color=[AZUL_1, VERMELHO_1]) # Plot on 'ax', remove figsize from here
plt.title('Métodos de Pagamento e Churn', fontsize= 18)
plt.xlabel('Métodos de Pagamento')
plt.ylabel('Número de Clientes')
plt.xticks(rotation = 0, ha='center')
plt.legend(title='Churn', labels=['Não', 'Sim'])
plt.tight_layout()
for container in ax.containers:
    ax.bar_label(container, fmt='%d', label_type='edge')
ax.spines[['top', 'right']].set_visible(False)
salvar_figura(fig, 'churn_por_metodo_pagamento.png') # Save before showing
plt.show()
```

Mostrar saída oculta

Tipo de Cobrança (Paperless Biling)

```
churn_tipo_cobranca = pd.crosstab(df_final['paperless_billing'], df_final['churn'])
#display(churn_tipo_cobranca)

churn_cobranca = churn_tipo_cobranca.apply(lambda x: x / x.sum(), axis=1)
#display(churn_cobranca)

fig, ax = plt.subplots(figsize=(10, 6)) # Use plt.subplots to create figure and axes
churn_tipo_cobranca.plot(kind='bar', ax=ax, color=[AZUL_1, VERMELHO_1]) # Plot on 'ax', remove figsize from here
plt.title('Relação entre Cobrança Sem Papel e Churn', fontsize= 18)
plt.xlabel('Cobrança Sem Papel')
plt.ylabel('Número de Clientes')
plt.xticks(rotation=0)
plt.legend(title='Churn', labels=['Não', 'Sim'])
plt.tight_layout()
for container in ax.containers:
    ax.bar_label(container, fmt='%d', label_type='edge')
ax.spines[['top', 'right']].set_visible(False)
salvar_figura(fig, 'churn_tipo_cobranca.png') # Save before showing
plt.show()
```

Mostrar saída oculta

Streaming TV

```
churn_streaming_tv = pd.crosstab(df_final['streaming_tv'], df_final['churn'])
#display(churn_streaming_tv)

churn_streaming = churn_streaming_tv.apply(lambda x: x / x.sum(), axis=1)
```

```
#display(churn_streaming)

fig, ax = plt.subplots(figsize=(10, 6)) # Use plt.subplots to create figure and axes
churn_streaming_tv.plot(kind='bar', ax=ax, color=[AZUL_1, VERMELHO_1]) # Plot on 'ax', remove figsize from here
plt.title('Relação entre Streaming TV e Churn', fontsize= 18)
plt.xlabel('Streaming TV')
plt.ylabel('Número de Clientes')
plt.xticks(rotation=0)
plt.legend(title='Churn', labels=['Não', 'Sim'])
plt.tight_layout()
for container in ax.containers:
    ax.bar_label(container, fmt='%d', label_type='edge')
ax.spines[['top', 'right']].set_visible(False)
salvar_figura(fig, 'churn_streaming_tv.png') # Save before showing
plt.show()
```

Mostrar saída oculta

Streaming de Filmes (Streaming Movies)

```
churn_streaming_Filmes = pd.crosstab(df_final['streaming_movies'], df_final['churn'])
#display(churn_streaming_Filmes)

churn_streaming = churn_streaming_Filmes.apply(lambda x: x / x.sum(), axis=1)
#display(churn_streaming)

fig, ax = plt.subplots(figsize=(10, 6)) # Use plt.subplots to create figure and axes
churn_streaming_Filmes.plot(kind='bar', ax=ax, color=[AZUL_1, VERMELHO_1]) # Plot on 'ax', remove figsize from here
plt.title('Relação entre Streaming Filmes e Churn', fontsize= 18)
plt.xlabel('Streaming Filmes')
plt.ylabel('Número de Clientes')
plt.xticks(rotation=0)
plt.legend(title='Churn', labels=['Não', 'Sim'])
plt.tight_layout()
for container in ax.containers:
    ax.bar_label(container, fmt='%d', label_type='edge')
ax.spines[['top', 'right']].set_visible(False)
salvar_figura(fig, 'churn_streaming_Filmes.png') # Save before showing
plt.show()
```

Mostrar saída oculta

Suporte Técnico (Tech Support)

```
churn_suporte_tecnico = pd.crosstab(df_final['tech_support'], df_final['churn'])
#display(churn_suporte_tecnico)

churn_suport = churn_suporte_tecnico.apply(lambda x: x / x.sum(), axis=1)
#display(churn_suport)

fig, ax = plt.subplots(figsize=(10, 6)) # Use plt.subplots to create figure and axes
churn_suporte_tecnico.plot(kind='bar', ax=ax, color=[AZUL_1, VERMELHO_1]) # Plot on 'ax', remove figsize from here
plt.title('Relação entre Suporte Técnico e Churn', fontsize= 18)
plt.xlabel('Suporte Técnico')
plt.ylabel('Número de Clientes')
plt.xticks(rotation=0)
plt.legend(title='Churn', labels=['Não', 'Sim'])
plt.tight_layout()
for container in ax.containers:
    ax.bar_label(container, fmt='%d', label_type='edge')
ax.spines[['top', 'right']].set_visible(False)
salvar_figura(fig, 'churn_suporte_tecnico.png') # Save before showing
plt.show()
```

Mostrar saída oculta

Serviços de Internet (Internet Services)

```
churn_servico_internet = pd.crosstab(df_final['internet_service'], df_final['churn'])
#display(churn_servico_internet)

churn_internet = churn_servico_internet.apply(lambda x: x / x.sum(), axis=1)
#display(churn_internet)

fig, ax = plt.subplots(figsize=(10, 6)) # Use plt.subplots to create figure and axes
churn_servico_internet.plot(kind='bar', ax=ax, color=[AZUL_1, VERMELHO_1]) # Plot on 'ax', remove figsize from here
plt.title('Serviço de Internet vs Churn', ha= 'center', fontsize= 18)
plt.xlabel('Serviço de Internet')
```

```
plt.ylabel('Número de Clientes')
plt.xticks(rotation=0)
plt.legend(title='Churn', labels=['Não', 'Sim'])
plt.tight_layout()
for container in ax.containers:
    ax.bar_label(container, fmt='%d', label_type='edge')
ax.spines[['top', 'right']].set_visible(False)
salvar_figura(fig, 'churn_servico_internet.png') # Save before showing
plt.show()
```

Mostrar saída oculta

Serviços Telefônicos (Phone Services)

```
churn_servico_telefonico = pd.crosstab(df_final['phone_service'], df_final['churn'])
#display(churn_servico_telefonico)

churn_total = churn_servico_telefonico.apply(lambda x: x / x.sum(), axis=1)
#display(churn_total)

fig, ax = plt.subplots(figsize=(10, 6)) # Use plt.subplots to create figure and axes
churn_servico_telefonico.plot(kind='bar', ax=ax, color=[AZUL_1, VERMELHO_1]) # Plot on 'ax', remove figsize from here
plt.title('Serviço Telefônico e Churn', fontsize= 18)
plt.xlabel('Serviço Telefônico')
plt.ylabel('Número de Clientes')
plt.xticks(rotation=0)
plt.legend(title='Churn', labels=['Não', 'Sim'])
plt.tight_layout()
for container in ax.containers:
    ax.bar_label(container, fmt='%d', label_type='edge')
ax.spines[['top', 'right']].set_visible(False)
salvar_figura(fig, 'churn_servico_telefonico.png')
plt.show()
```

Mostrar saída oculta

Múltiplas Linhas (Multiple Lines)

```
churn_linhas_multiplas = pd.crosstab(df_final['multiple_lines'], df_final['churn'])
#display(churn_linhas_multiplas)

churn_multiplas = churn_linhas_multiplas.apply(lambda x: x / x.sum(), axis=1)
#display(churn_multiplas)

fig, ax = plt.subplots(figsize=(10, 6)) # Use plt.subplots to create figure and axes
churn_linhas_multiplas.plot(kind='bar', ax=ax, color=[AZUL_1, VERMELHO_1]) # Plot on 'ax', remove figsize from here
plt.title('Relação entre Múltiplas Linhas Telefônicas e Churn', fontsize= 18)
plt.xlabel('Múltiplas Linhas Telefônicas')
plt.ylabel('Número de Clientes')
plt.xticks(rotation=0)
plt.legend(title='Churn', labels=['Não', 'Sim'])
plt.tight_layout()
for container in ax.containers:
    ax.bar_label(container, fmt='%d', label_type='edge')
ax.spines[['top', 'right']].set_visible(False)
salvar_figura(fig, 'churn_linhas_multiplas.png')
plt.show()
```

Mostrar saída oculta

Contagem de evasão por variáveis numéricas

Tempo de Contrato (Tenure)

```
fig = plt.figure(figsize=(10, 6))
sns.boxplot(x='churn', y='tenure', data=df_final, hue='churn', palette=[AZUL_1, VERMELHO_1], legend=False)
plt.title('Distribuição do Tempo de Contrato por Churn', fontsize=16)
plt.xlabel('Churn', fontsize=12)
plt.ylabel('Contrato (meses)', fontsize=12)
plt.xticks(ticks=[0, 1], labels=['Não', 'Sim'], rotation=0)
plt.tight_layout()
salvar_figura(fig, 'churn_tempo_contrato.png')
plt.show()
```

Mostrar saída oculta

Total Gasto (Charges Total)

```
fig = plt.figure(figsize=(10, 6))
sns.boxplot(x='churn', y='charges_total', data=df_final, hue='churn', palette=[AZUL_1, VERMELHO_1], legend=False)
plt.title('Distribuição de Total Gasto por Churn', fontsize=16)
plt.xlabel('Churn', fontsize=12)
plt.ylabel('Total Gasto', fontsize=12)
plt.xticks(ticks=[0, 1], labels=['Não', 'Sim'], rotation=0)
plt.tight_layout()
salvar_figura(fig, 'churn_total_gasto.png')
plt.show()
```

Mostrar saída oculta

Contas Diárias

```
fig = plt.figure(figsize=(10, 6))
sns.boxplot(x='churn', y='contas_diarias', data=df_final, hue='churn', palette=[AZUL_1, VERMELHO_1], legend=False)
plt.title('Distribuição das Contas Diárias por Churn', fontsize=16)
plt.xlabel('Churn', fontsize=12)
plt.ylabel('Contas Diárias', fontsize=12)
plt.xticks(ticks=[0, 1], labels=['Não', 'Sim'], rotation=0)
plt.tight_layout()
salvar_figura(fig, 'churn_contas_diarias.png')
plt.show()
```

Mostrar saída oculta

Extra

```
# Calcular a correlação entre 'churn' e as outras variáveis numéricas
correlacoes = df_final.corr(numeric_only=True)['churn'].sort_values(ascending=False)

correlacoes
```

Mostrar saída oculta