surv_sparrow_churn_prediction

July 9, 2024

[1]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
```

[4]:
```python
df=pd.read_csv('/content/WA_Fn-UseC_-Telco-Customer-Churn.csv')
df
```

[4]:
```
      customerID  gender  SeniorCitizen Partner Dependents  tenure  \
0     7590-VHVEG  Female              0     Yes         No       1
1     5575-GNVDE    Male              0      No         No      34
2     3668-QPYBK    Male              0      No         No       2
3     7795-CFOCW    Male              0      No         No      45
4     9237-HQITU  Female              0      No         No       2
…            …     …            …      …        …      …
7038  6840-RESVB    Male              0     Yes        Yes      24
7039  2234-XADUH  Female              0     Yes        Yes      72
7040  4801-JZAZL  Female              0     Yes        Yes      11
7041  8361-LTMKD    Male              1     Yes         No       4
7042  3186-AJIEK    Male              0      No         No      66

     PhoneService     MultipleLines InternetService OnlineSecurity  …  \
0              No  No phone service             DSL             No  …
1             Yes                No             DSL            Yes  …
2             Yes                No             DSL            Yes  …
3              No  No phone service             DSL            Yes  …
4             Yes                No     Fiber optic             No  …
…            …                …               …              …  …
7038          Yes               Yes             DSL            Yes  …
7039          Yes               Yes     Fiber optic             No  …
7040           No  No phone service             DSL            Yes  …
7041          Yes               Yes     Fiber optic             No  …
7042          Yes                No     Fiber optic            Yes  …

     DeviceProtection TechSupport StreamingTV StreamingMovies        Contract  \
0                No          No          No               No  Month-to-month
1               Yes          No          No               No        One year
2                No          No          No               No  Month-to-month
3               Yes         Yes          No               No        One year
4                No          No          No               No  Month-to-month
…               …         …         …             …             …
7038            Yes         Yes         Yes              Yes        One year
7039            Yes          No         Yes              Yes        One year
7040             No          No          No               No  Month-to-month
7041             No          No          No               No  Month-to-month
7042            Yes         Yes         Yes              Yes        Two year

     PaperlessBilling              PaymentMethod MonthlyCharges  TotalCharges  \
0                 Yes           Electronic check          29.85         29.85
1                  No               Mailed check          56.95        1889.5
2                 Yes               Mailed check          53.85        108.15
3                  No  Bank transfer (automatic)          42.30       1840.75
4                 Yes           Electronic check          70.70        151.65
…               …                         …            …           …
7038              Yes               Mailed check          84.80        1990.5
7039              Yes    Credit card (automatic)         103.20        7362.9
7040              Yes           Electronic check          29.60        346.45
7041              Yes               Mailed check          74.40         306.6
7042              Yes  Bank transfer (automatic)         105.65        6844.5

     Churn
0       No
1       No
2      Yes
3       No
4      Yes
…     …
7038    No
7039    No
7040    No
7041   Yes
7042    No

[7043 rows x 21 columns]
```

[6]:
```
df.columns
```

[6]:
```
Index(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport',
       'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling',
       'PaymentMethod', 'MonthlyCharges', 'TotalCharges', 'Churn'],
      dtype='object')
```

```
[5]: df.dtypes
```

```
[5]: customerID           object
     gender               object
     SeniorCitizen         int64
     Partner              object
     Dependents           object
     tenure                int64
     PhoneService         object
     MultipleLines        object
     InternetService      object
     OnlineSecurity       object
     OnlineBackup         object
     DeviceProtection     object
     TechSupport          object
     StreamingTV          object
     StreamingMovies      object
     Contract             object
     PaperlessBilling     object
     PaymentMethod        object
     MonthlyCharges      float64
     TotalCharges         object
     Churn                object
     dtype: object
```

```
[7]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   customerID        7043 non-null   object
 1   gender            7043 non-null   object
 2   SeniorCitizen     7043 non-null   int64
 3   Partner           7043 non-null   object
 4   Dependents        7043 non-null   object
 5   tenure            7043 non-null   int64
 6   PhoneService      7043 non-null   object
 7   MultipleLines     7043 non-null   object
 8   InternetService   7043 non-null   object
 9   OnlineSecurity    7043 non-null   object
 10  OnlineBackup      7043 non-null   object
 11  DeviceProtection  7043 non-null   object
 12  TechSupport       7043 non-null   object
 13  StreamingTV       7043 non-null   object
 14  StreamingMovies   7043 non-null   object
 15  Contract          7043 non-null   object
 16  PaperlessBilling  7043 non-null   object
 17  PaymentMethod     7043 non-null   object
 18  MonthlyCharges    7043 non-null   float64
 19  TotalCharges      7043 non-null   object
 20  Churn             7043 non-null   object
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB
```

[8]:
```
df.shape
```

[8]:
```
(7043, 21)
```

[12]:
```
df.isnull().sum()
```

[12]:
```
customerID          0
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges        0
Churn               0
dtype: int64
```

[9]:
```
df.describe()
```

[9]:
```
       SeniorCitizen       tenure  MonthlyCharges
count    7043.000000  7043.000000     7043.000000
mean        0.162147    32.371149       64.761692
std         0.368612    24.559481       30.090047
min         0.000000     0.000000       18.250000
```

```
25%          0.000000     9.000000       35.500000
50%          0.000000    29.000000       70.350000
75%          0.000000    55.000000       89.850000
max          1.000000    72.000000      118.750000
```

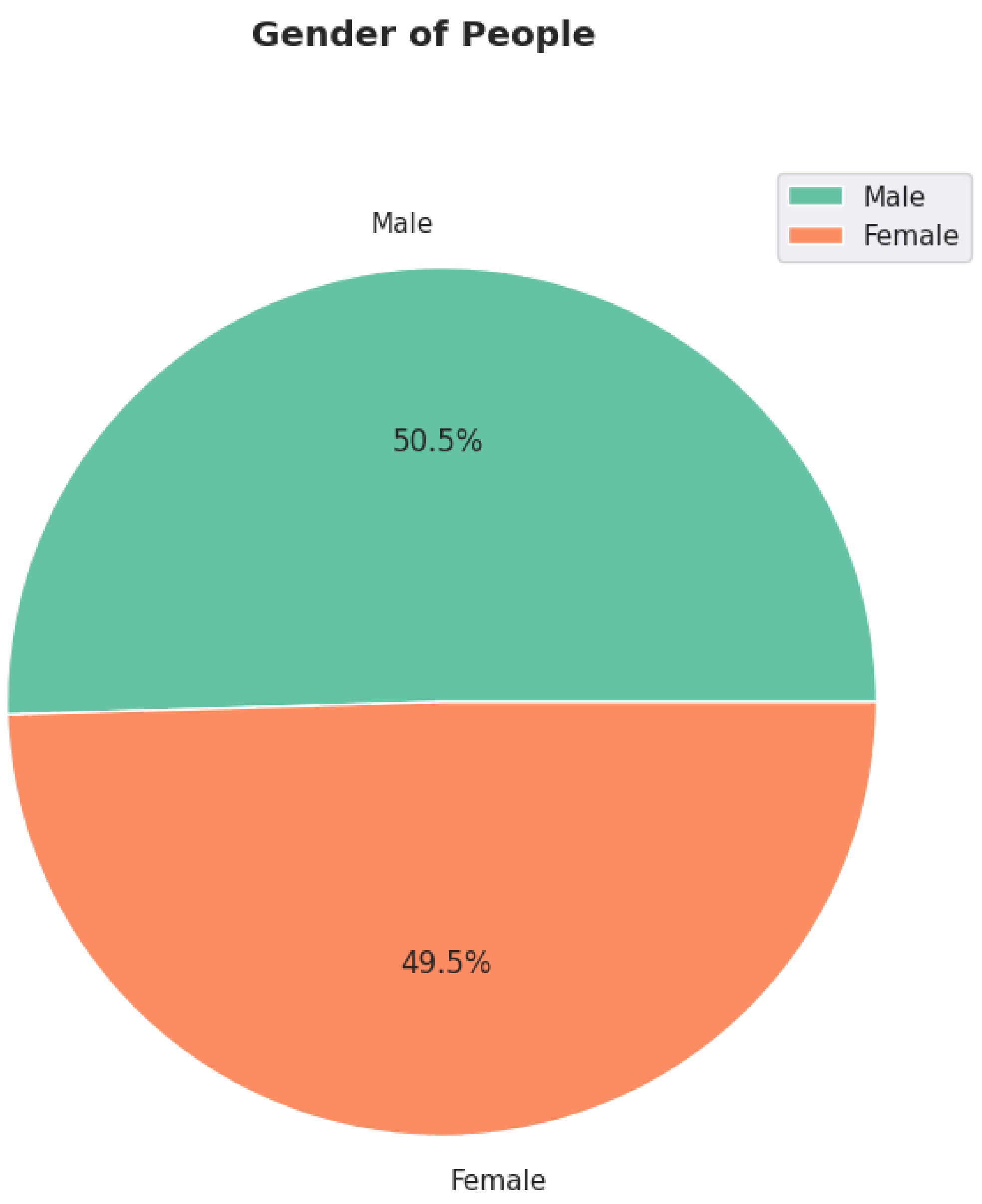
[14]:
```
df.duplicated().sum()
```

[14]:
```
0
```

[16]:
```
df['TotalCharges'] = pd.to_numeric(df['TotalCharges'])
```

CATEGORICAL COLUMNS EXPLORATION

[66]:
```
gender_counts = df['gender'].value_counts()
x = gender_counts.index
y = gender_counts.values

fig, ax = plt.subplots(figsize=(8, 8))

#pie chart
pal = sns.color_palette("Set2", len(gender_counts))
ax.pie(y, labels=x, colors=pal, autopct='%1.1f%%')

#legend and title
plt.legend(bbox_to_anchor=(1, 1))
plt.suptitle('Gender of People', weight='bold')
plt.show()
```

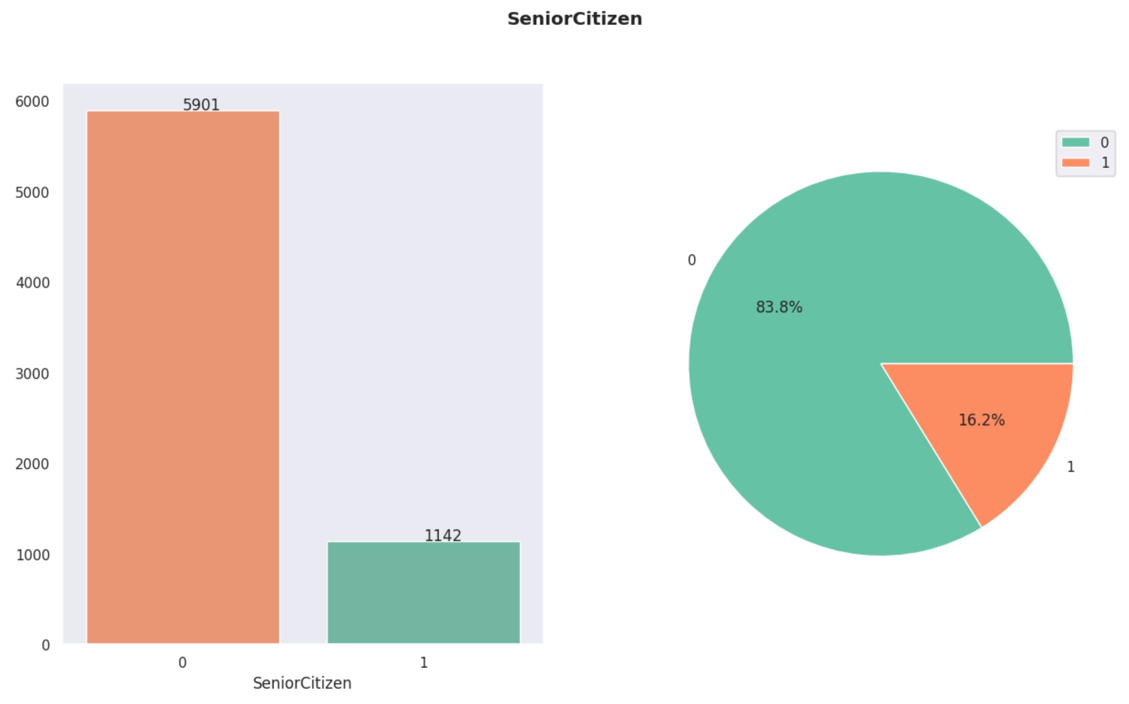


senior citizen

```
[65]: senior_counts = df['SeniorCitizen'].value_counts()
      x = senior_counts.index
      y = senior_counts.values

      fig, ax = plt.subplots(1, 2, figsize=(15, 8))

      #bar plot
```

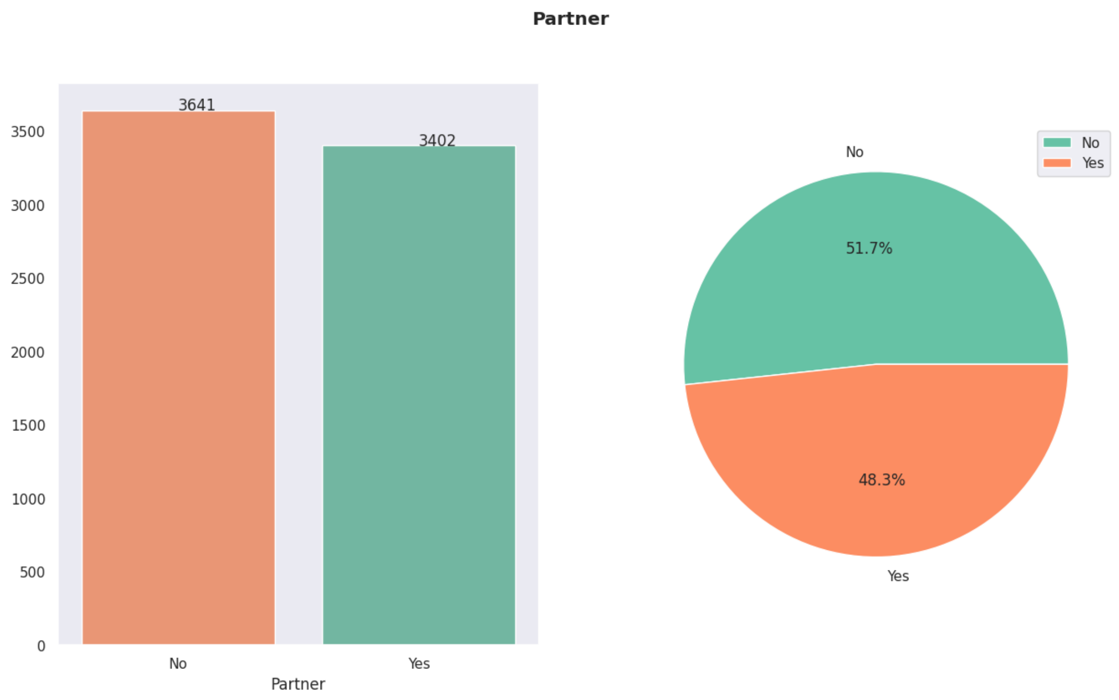
```
sns.set(style="dark", color_codes=True)
pal = sns.color_palette( "Set2",len(senior_counts))
sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0])

for p in ax[0].patches:
    ax[0].annotate(f'{p.get_height():.0f}', (p.get_x() + p.get_width() / 2., p.
 ↪get_height()))

ax[0].set_xlabel('SeniorCitizen')

#pie chart
ax[1].pie(y, labels=x, colors=pal, autopct='%1.1f%%')

# legend and title
plt.legend(bbox_to_anchor=(1, 1))
plt.suptitle('SeniorCitizen', weight='bold')
plt.show()
```

```
<ipython-input-65-5f357ac7da10>:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0])
```



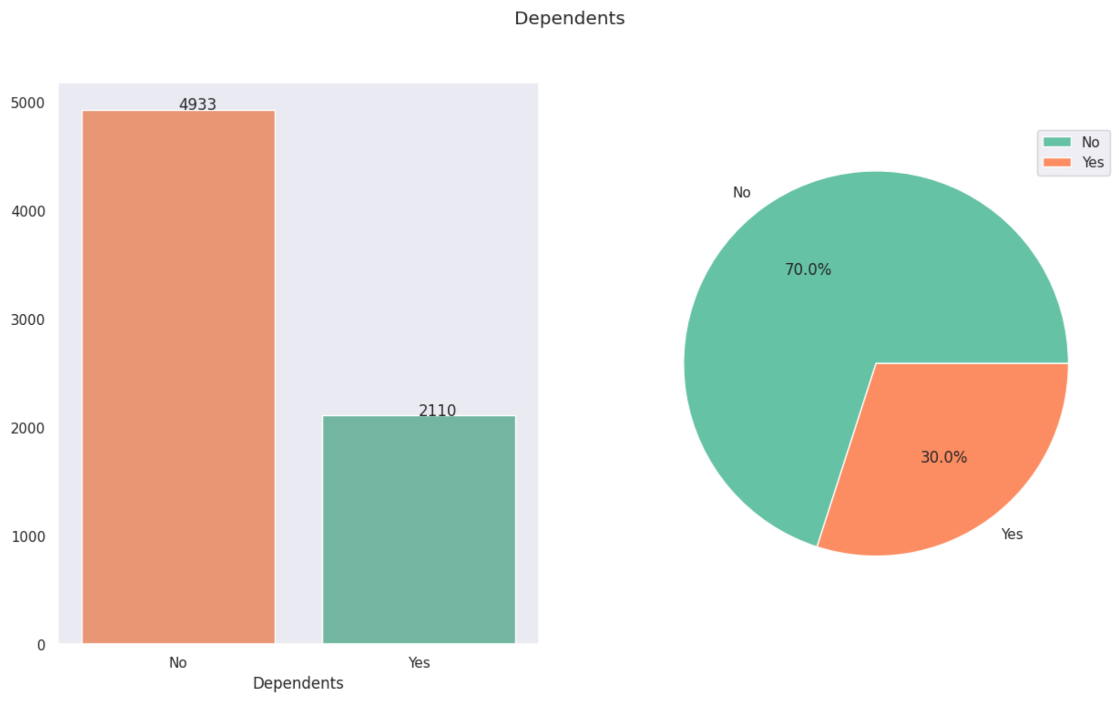
```
[64]: partner_counts = df['Partner'].value_counts()
      x = partner_counts.index
      y = partner_counts.values

      fig, ax = plt.subplots(1, 2, figsize=(15, 8))

      #bar plot
      sns.set(style="dark", color_codes=True)
      pal = sns.color_palette("Set2", len(partner_counts))
      sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0])
      for p in ax[0].patches:
          ax[0].annotate(f'{p.get_height():.0f}', (p.get_x() + p.get_width() / 2., p.
        ↪get_height()))

      ax[0].set_xlabel('Partner')

      #pie chart
      ax[1].pie(y, labels=x, colors=pal, autopct='%1.1f%%')

      #legend and title
      plt.legend(bbox_to_anchor=(1, 1))
      plt.suptitle('Partner', weight='bold')
      plt.show()
```

```
<ipython-input-64-2a6a7b4bb7b4>:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0])
```



```
[63]: dependents_counts = df['Dependents'].value_counts()
x = dependents_counts.index
y = dependents_counts.values

fig, ax = plt.subplots(1, 2, figsize=(15, 8))

#bar plot
sns.set(style="dark", color_codes=True)
pal = sns.color_palette("Set2", len(dependents_counts))
sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0])

for p in ax[0].patches:
    ax[0].annotate(f'{p.get_height():.0f}', (p.get_x() + p.get_width() / 2., p.
  ↪get_height()))

ax[0].set_xlabel('Dependents')

#pie chart
ax[1].pie(y, labels=x, colors=pal, autopct='%1.1f%%')

#legend and title
plt.legend(bbox_to_anchor=(1, 1))
plt.suptitle('Dependents')
plt.show()
```

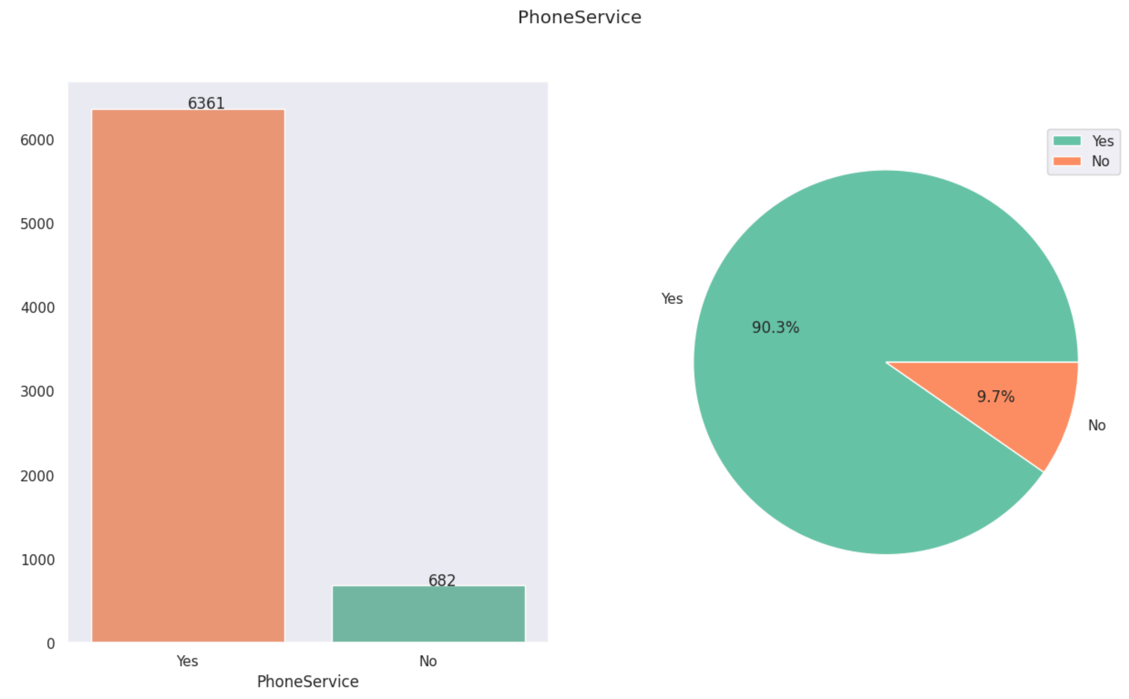
```
<ipython-input-63-909a902d337f>:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0])
```



```
[62]: phone_service_counts = df['PhoneService'].value_counts()
      x = phone_service_counts.index
      y = phone_service_counts.values

      fig, ax = plt.subplots(1, 2, figsize=(15, 8))

      #bar plot
      sns.set(style="dark", color_codes=True)
      pal = sns.color_palette("Set2", len(phone_service_counts))
      sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0])

      for p in ax[0].patches:
          ax[0].annotate(f'{p.get_height():.0f}', (p.get_x() + p.get_width() / 2., p.
        ↪get_height()),)

      ax[0].set_xlabel('PhoneService')
```

```
#pie chart
ax[1].pie(y, labels=x, colors=pal, autopct='%1.1f%%')

#legend and title
plt.legend(bbox_to_anchor=(1, 1))
plt.suptitle('PhoneService')
plt.show()
```

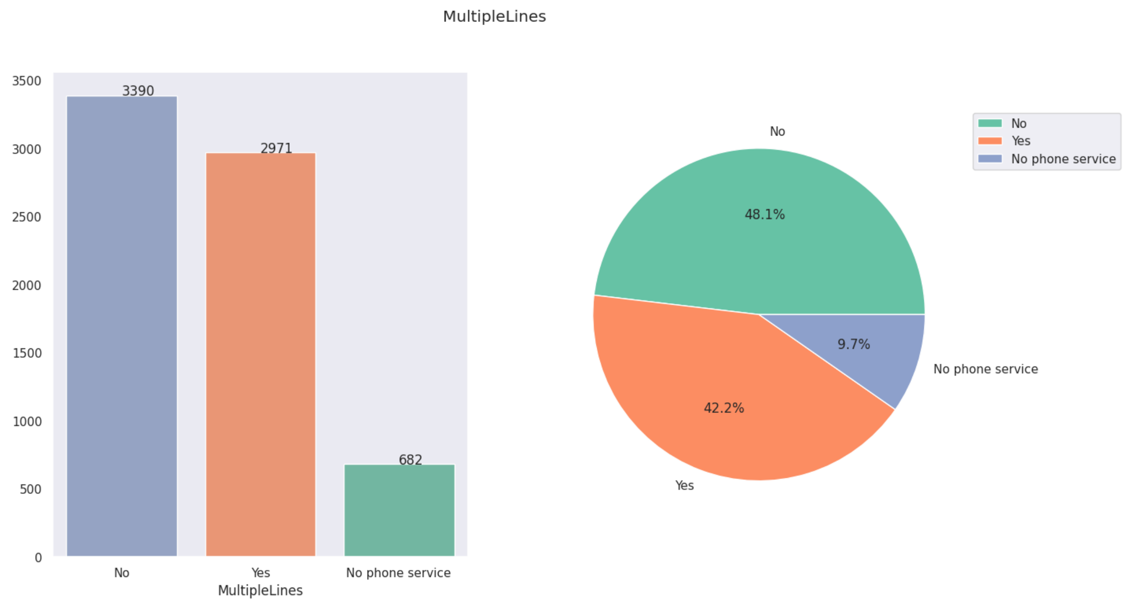
```
<ipython-input-62-55742b8ebd07>:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0])
```



```
[61]: multiple_lines_counts = df['MultipleLines'].value_counts()
x = multiple_lines_counts.index
y = multiple_lines_counts.values

fig, ax = plt.subplots(1, 2, figsize=(15, 8))

#bar plot
sns.set(style="dark", color_codes=True)
pal = sns.color_palette("Set2", len(multiple_lines_counts))
```

```
sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0])

for p in ax[0].patches:
    ax[0].annotate(f'{p.get_height():.0f}', (p.get_x() + p.get_width() / 2., p.
 ↪get_height()))

ax[0].set_xlabel('MultipleLines')

#pie chart
ax[1].pie(y, labels=x, colors=pal, autopct='%1.1f%%')

#legend and title
plt.legend(bbox_to_anchor=(1, 1))
plt.suptitle('MultipleLines')
plt.show()
```

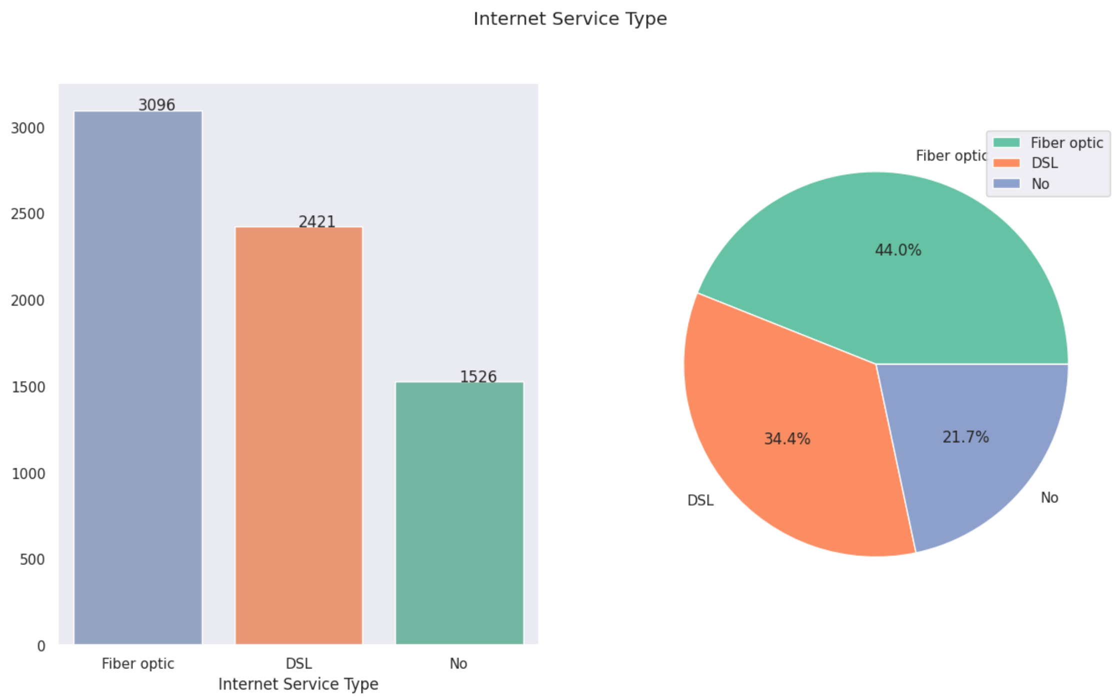
```
<ipython-input-61-dac617941059>:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0])
```



[60]:
```
internet_service_counts = df['InternetService'].value_counts()
x = internet_service_counts.index
y = internet_service_counts.values
```

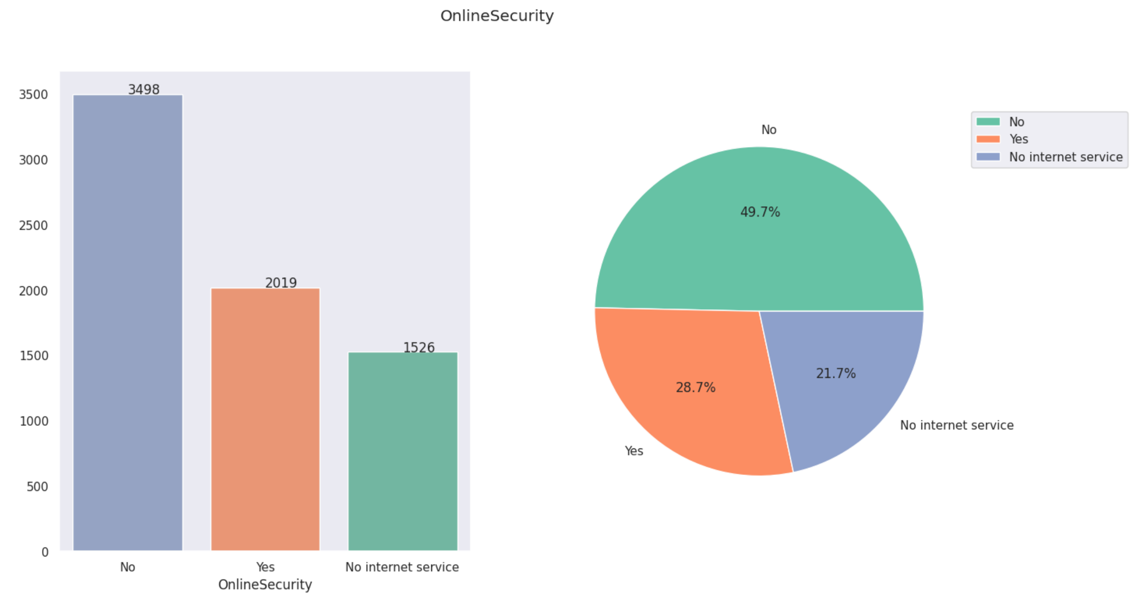
```python
fig, ax = plt.subplots(1, 2, figsize=(15, 8))

#bar plot
sns.set(style="dark", color_codes=True)
pal = sns.color_palette("Set2", len(internet_service_counts))
sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0])

for p in ax[0].patches:
    ax[0].annotate(f'{p.get_height():.0f}', (p.get_x() + p.get_width() / 2., p.
 ↪get_height()))

ax[0].set_xlabel('Internet Service Type')

#pie chart
ax[1].pie(y, labels=x, colors=pal, autopct='%1.1f%%')

#legend and title
plt.legend(bbox_to_anchor=(1, 1))
plt.suptitle('Internet Service Type')
plt.show()
```

```
<ipython-input-60-c247801fd22e>:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0])
```



```
[59]: online_security_counts = df['OnlineSecurity'].value_counts()
      x = online_security_counts.index
      y = online_security_counts.values

      fig, ax = plt.subplots(1, 2, figsize=(15, 8))

      #bar plot
      sns.set(style="dark", color_codes=True)
      pal = sns.color_palette("Set2", len(online_security_counts))
      sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0])

      for p in ax[0].patches:
          ax[0].annotate(f'{p.get_height():.0f}', (p.get_x() + p.get_width() / 2., p.
       ↪get_height()))

      ax[0].set_xlabel('OnlineSecurity')

      #pie chart
      ax[1].pie(y, labels=x, colors=pal, autopct='%1.1f%%')

      #legend and title
      plt.legend(bbox_to_anchor=(1, 1))
      plt.suptitle('OnlineSecurity')
      plt.show()
```

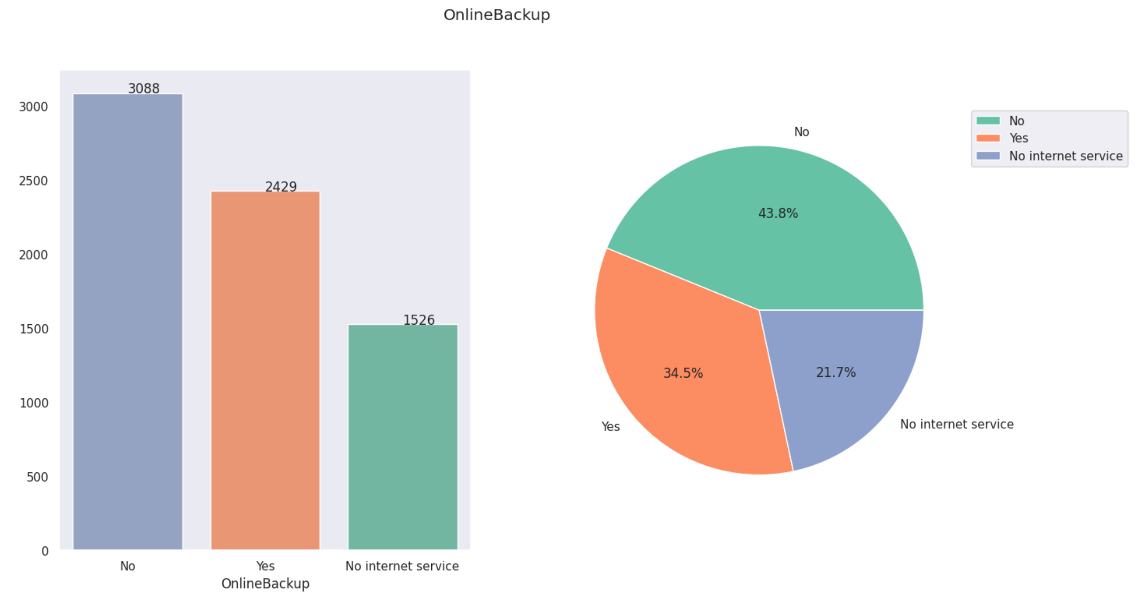
```
<ipython-input-59-c977e446e677>:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0])
```



[58]:
```python
online_backup_counts = df['OnlineBackup'].value_counts()
x = online_backup_counts.index
y = online_backup_counts.values

fig, ax = plt.subplots(1, 2, figsize=(15, 8))



#bar plot
sns.set(style="dark", color_codes=True)
pal = sns.color_palette("Set2", len(online_backup_counts))
sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0])

for p in ax[0].patches:
    ax[0].annotate(f'{p.get_height():.0f}', (p.get_x() + p.get_width() / 2., p.
  ↪get_height()))

ax[0].set_xlabel('OnlineBackup')

#pie chart
ax[1].pie(y, labels=x, colors=pal, autopct='%1.1f%%')
```

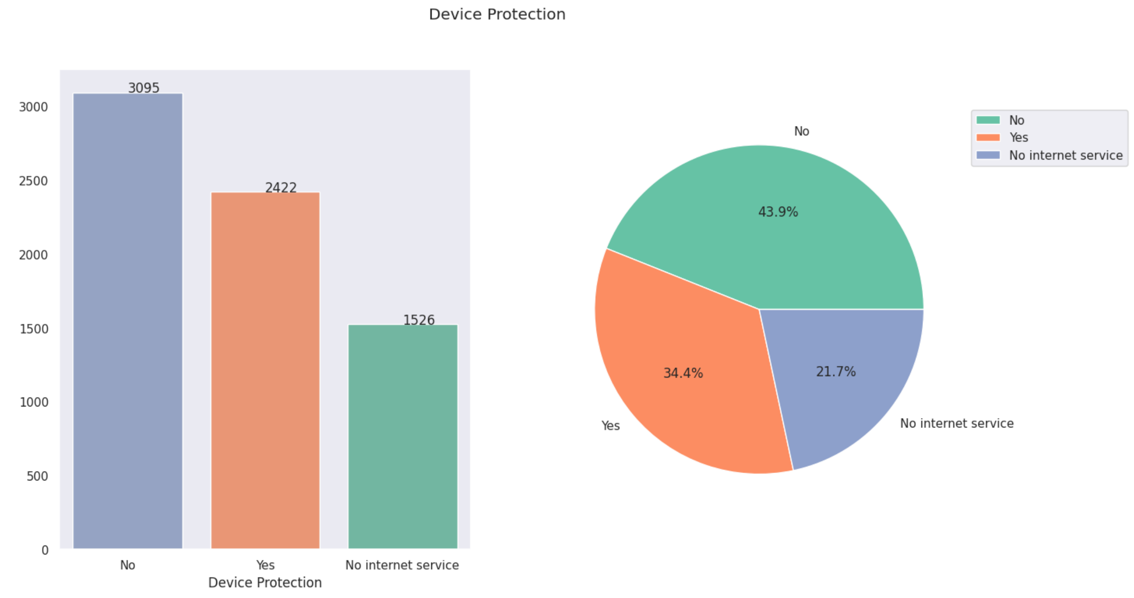
```python
#legend and title
plt.legend(bbox_to_anchor=(1, 1))
plt.suptitle('OnlineBackup')
plt.show()
```

```
<ipython-input-58-466e6472fb68>:11: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0])
```



```python
[77]: device_protection_counts = df['DeviceProtection'].value_counts()
x = device_protection_counts.index
y = device_protection_counts.values

fig, ax = plt.subplots(1, 2, figsize=(15, 8))

#bar plot
sns.set(style="dark", color_codes=True)
pal = sns.color_palette("Set2", len(device_protection_counts))
sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0])

for p in ax[0].patches:
    ax[0].annotate(f'{p.get_height():.0f}', (p.get_x() + p.get_width() / 2., p.
 ↪get_height()))

ax[0].set_xlabel('Device Protection')

#pie chart
ax[1].pie(y, labels=x, colors=pal, autopct='%1.1f%%')

#legend and title
plt.legend(bbox_to_anchor=(1, 1))
plt.suptitle('Device Protection')
plt.show()
```

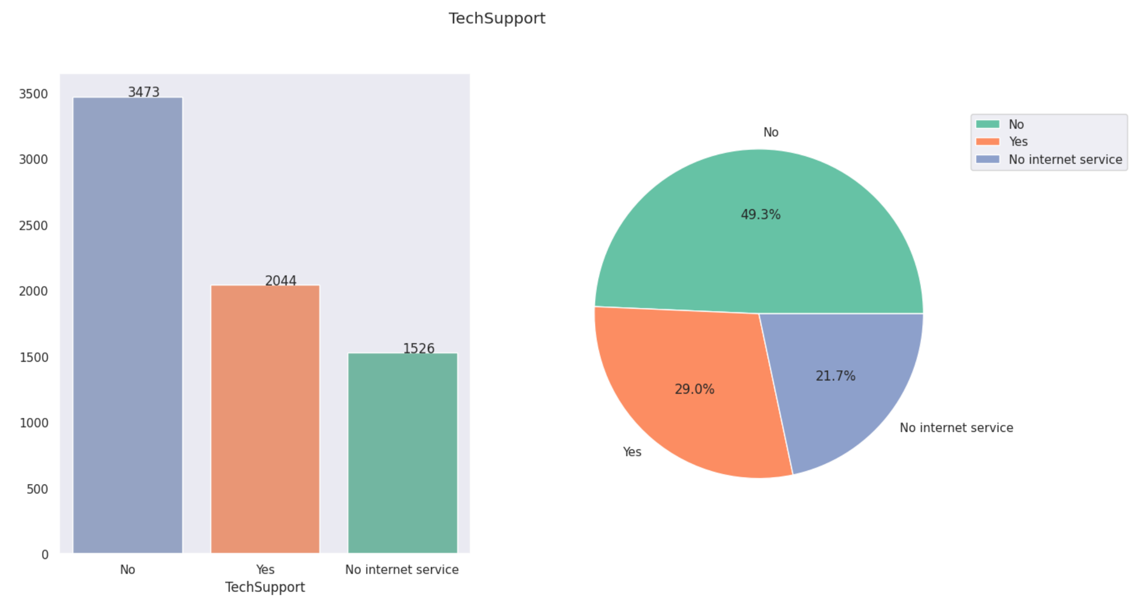
```
<ipython-input-77-045e4563a15b>:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0])
```



```
[69]: tech_support_counts = df['TechSupport'].value_counts()
      x = tech_support_counts.index
      y = tech_support_counts.values

      fig, ax = plt.subplots(1, 2, figsize=(15, 8))
```

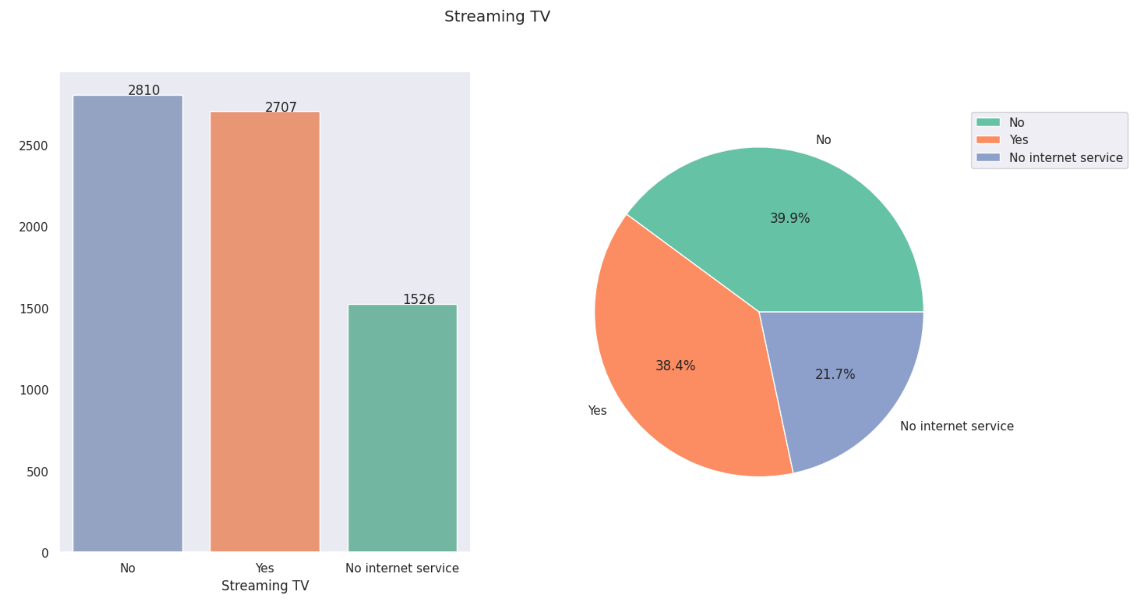
```python
#bar plot
sns.set(style="dark", color_codes=True)
pal = sns.color_palette("Set2", len(tech_support_counts))
sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0])

for p in ax[0].patches:
    ax[0].annotate(f'{p.get_height():.0f}', (p.get_x() + p.get_width() / 2., p.
 ↪get_height()))

ax[0].set_xlabel('TechSupport')

#pie chart
ax[1].pie(y, labels=x, colors=pal, autopct='%1.1f%%')

#legend and title
plt.legend(bbox_to_anchor=(1, 1))
plt.suptitle('TechSupport')
plt.show()
```

```
<ipython-input-69-3ac281d69399>:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0])
```



[72]:
```
streaming_tv_counts = df['StreamingTV'].value_counts()
x = streaming_tv_counts.index
y = streaming_tv_counts.values

fig, ax = plt.subplots(1, 2, figsize=(15, 8))

#bar plot
sns.set(style="dark", color_codes=True)
pal = sns.color_palette("Set2", len(streaming_tv_counts))
sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0])

for p in ax[0].patches:
    ax[0].annotate(f'{p.get_height():.0f}', (p.get_x() + p.get_width() / 2., p.
 ↪get_height()))

ax[0].set_xlabel('Streaming TV')

#pie chart
ax[1].pie(y, labels=x, colors=pal, autopct='%1.1f%%')

#legend and title
plt.legend(bbox_to_anchor=(1, 1))
plt.suptitle('Streaming TV')
plt.show()
```

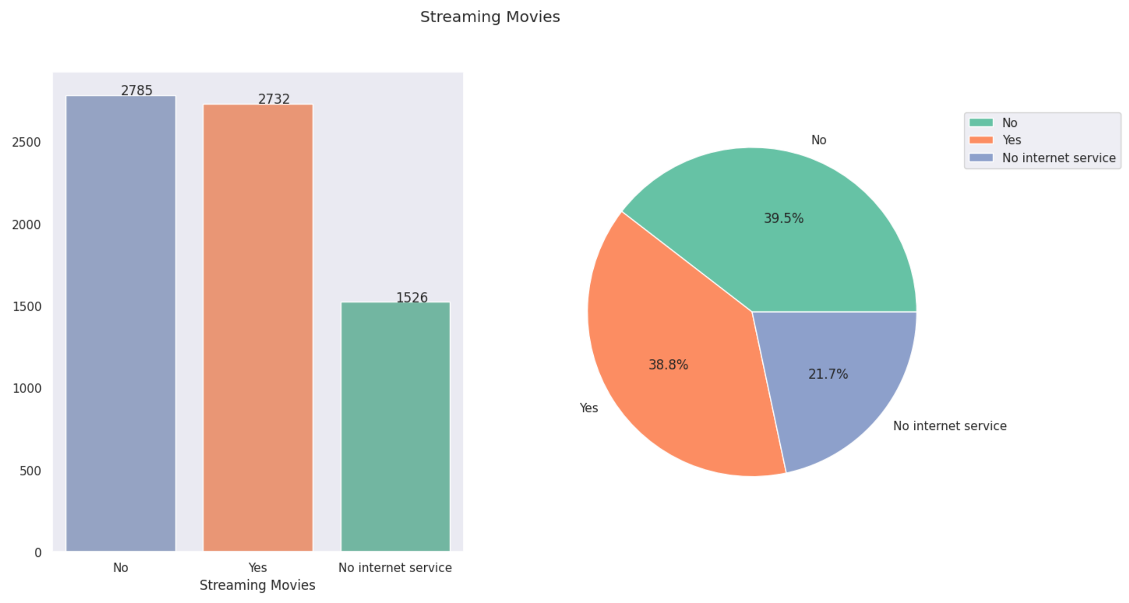
```
<ipython-input-72-7f09920d29f0>:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0])
```



[76]:
```python
streaming_movies_counts = df['StreamingMovies'].value_counts()
x = streaming_movies_counts.index
y = streaming_movies_counts.values

fig, ax = plt.subplots(1, 2, figsize=(15, 8))

#bar plot
sns.set(style="dark", color_codes=True)
pal = sns.color_palette("Set2", len(streaming_movies_counts))
sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0])

for p in ax[0].patches:
    ax[0].annotate(f'{p.get_height():.0f}', (p.get_x() + p.get_width() / 2., p.
  ↪get_height()))

ax[0].set_xlabel('Streaming Movies')

# Pie chart
ax[1].pie(y, labels=x, colors=pal, autopct='%1.1f%%')

#legend and title
plt.legend(bbox_to_anchor=(1, 1))
plt.suptitle('Streaming Movies')
plt.show()
```

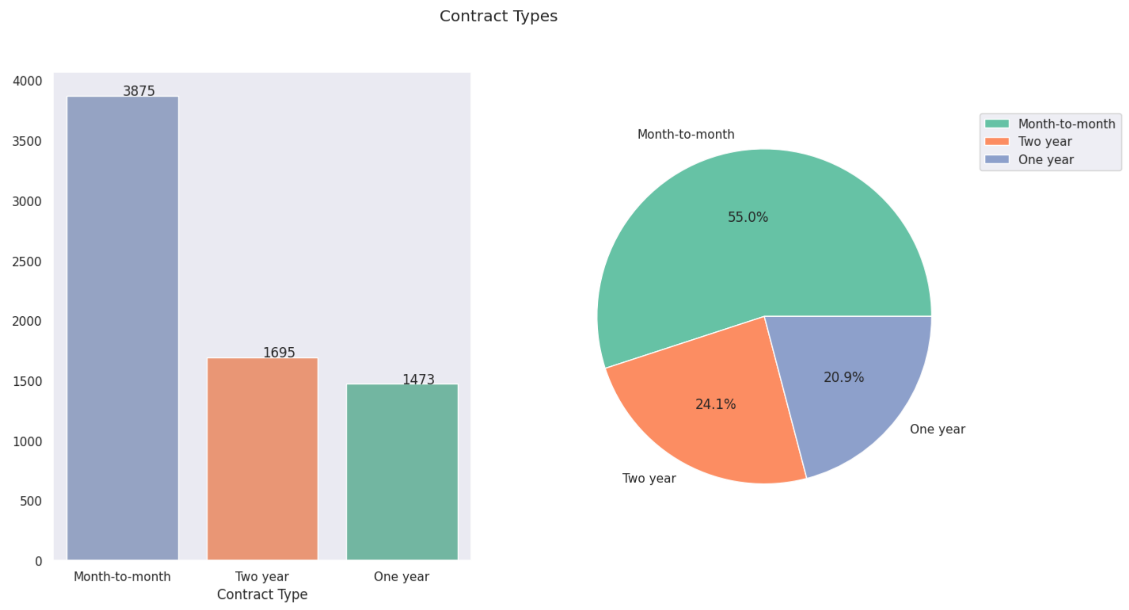
```
<ipython-input-76-d6af87b8fa48>:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
```

```
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0])
```



[73]:
```
contract_counts = df['Contract'].value_counts()
x = contract_counts.index
y = contract_counts.values

fig, ax = plt.subplots(1, 2, figsize=(15, 8))

#bar plot
sns.set(style="dark", color_codes=True)
pal = sns.color_palette("Set2", len(contract_counts))
sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0])

for p in ax[0].patches:
    ax[0].annotate(f'{p.get_height():.0f}', (p.get_x() + p.get_width() / 2., p.
 ↪get_height()))

ax[0].set_xlabel('Contract Type')

#pie chart
ax[1].pie(y, labels=x, colors=pal, autopct='%1.1f%%')

#legend and title
plt.legend(bbox_to_anchor=(1, 1))
plt.suptitle('Contract Types')
plt.show()
```

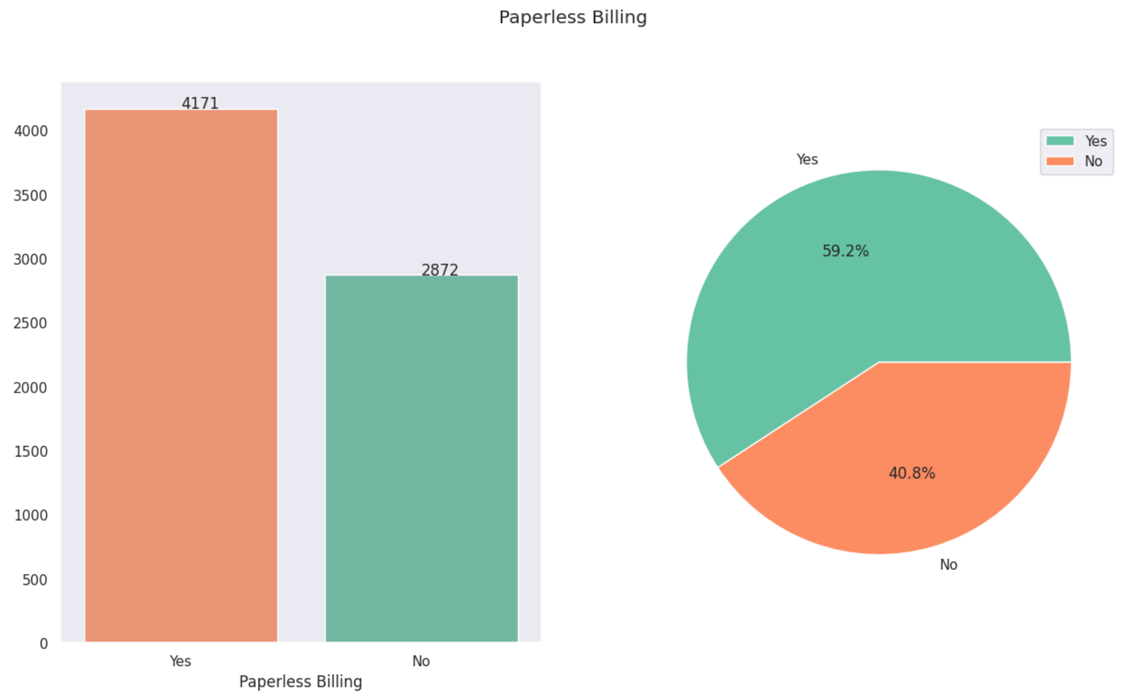
```
<ipython-input-73-78dc6306a91d>:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0])
```



[75]:
```
paperless_billing_counts = df['PaperlessBilling'].value_counts()
x = paperless_billing_counts.index
y = paperless_billing_counts.values

fig, ax = plt.subplots(1, 2, figsize=(15, 8))

#bar plot
sns.set(style="dark", color_codes=True)
pal = sns.color_palette("Set2", len(paperless_billing_counts))
sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0])

for p in ax[0].patches:
    ax[0].annotate(f'{p.get_height():.0f}', (p.get_x() + p.get_width() / 2., p.
 ↪get_height()))

ax[0].set_xlabel('Paperless Billing')
```

```
# Pie chart
ax[1].pie(y, labels=x, colors=pal, autopct='%1.1f%%')

#legend and title
plt.legend(bbox_to_anchor=(1, 1))
plt.suptitle('Paperless Billing')
plt.show()
```

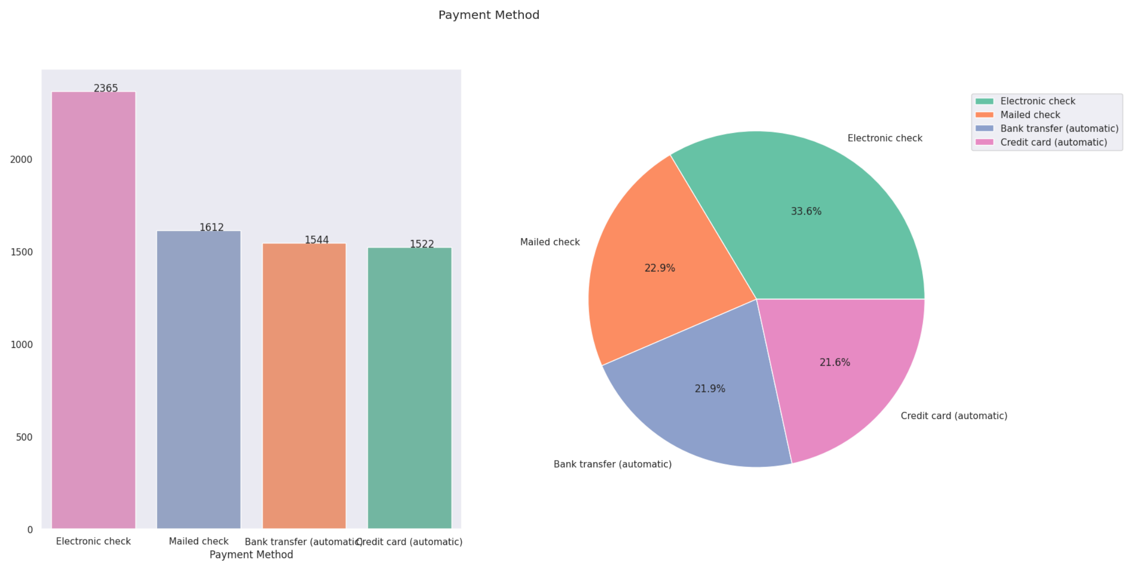
```
<ipython-input-75-0ee52f832f03>:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0])
```



```
[79]: payment_method_counts = df['PaymentMethod'].value_counts()
x = payment_method_counts.index
y = payment_method_counts.values

fig, ax = plt.subplots(1, 2, figsize=(20, 10))

#bar plot
sns.set(style="dark", color_codes=True)
pal = sns.color_palette("Set2", len(payment_method_counts))
```

```
sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0])

for p in ax[0].patches:
    ax[0].annotate(f'{p.get_height():.0f}', (p.get_x() + p.get_width() / 2., p.
↪get_height()))

ax[0].set_xlabel('Payment Method')

#pie chart
ax[1].pie(y, labels=x, colors=pal, autopct='%1.1f%%')

#legend and title
plt.legend(bbox_to_anchor=(1, 1))
plt.suptitle('Payment Method')
plt.show()
```

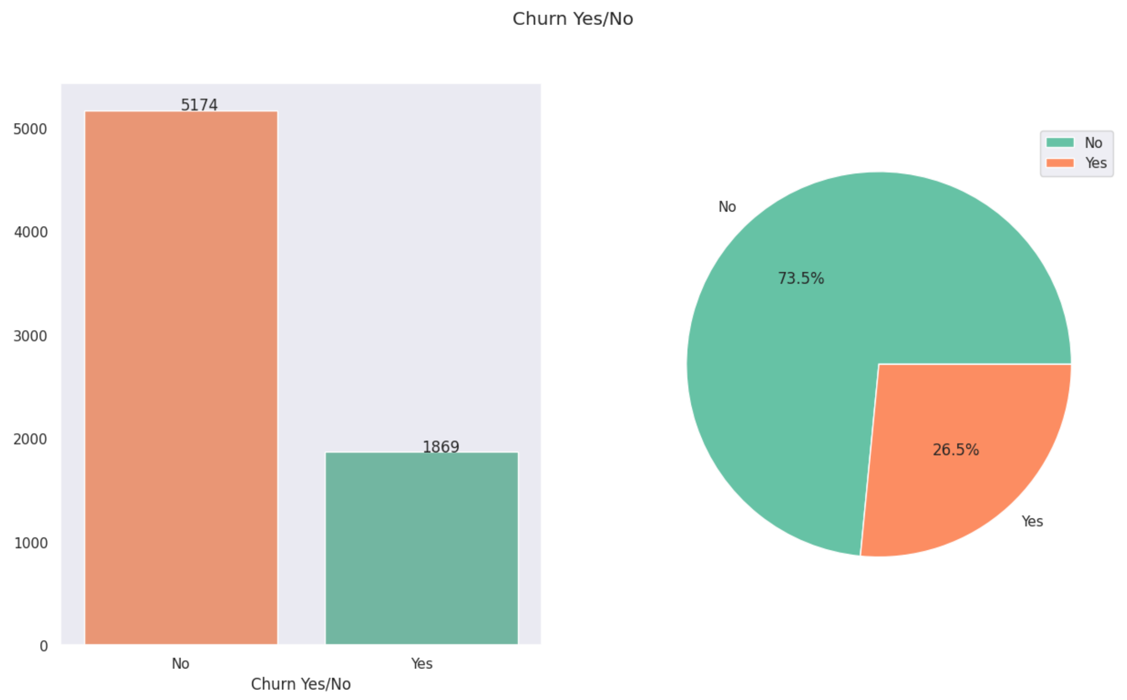
```
<ipython-input-79-67ecd2e2834b>:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0])
```



```
[80]: churn_counts = df['Churn'].value_counts()
      x = churn_counts.index
      y = churn_counts.values
```

```python
fig, ax = plt.subplots(1, 2, figsize=(15, 8))

#bar plot
sns.set(style="dark", color_codes=True)
pal = sns.color_palette("Set2", len(churn_counts))
sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0])

for p in ax[0].patches:
    ax[0].annotate(f'{p.get_height():.0f}', (p.get_x() + p.get_width() / 2., p.
  ↪get_height()))

ax[0].set_xlabel('Churn Yes/No')

#pie chart
ax[1].pie(y, labels=x, colors=pal, autopct='%1.1f%%')

#legend and title
plt.legend(bbox_to_anchor=(1, 1))
plt.suptitle('Churn Yes/No')
plt.show()
```

```
<ipython-input-80-fdaa9f2431d5>:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0])
```

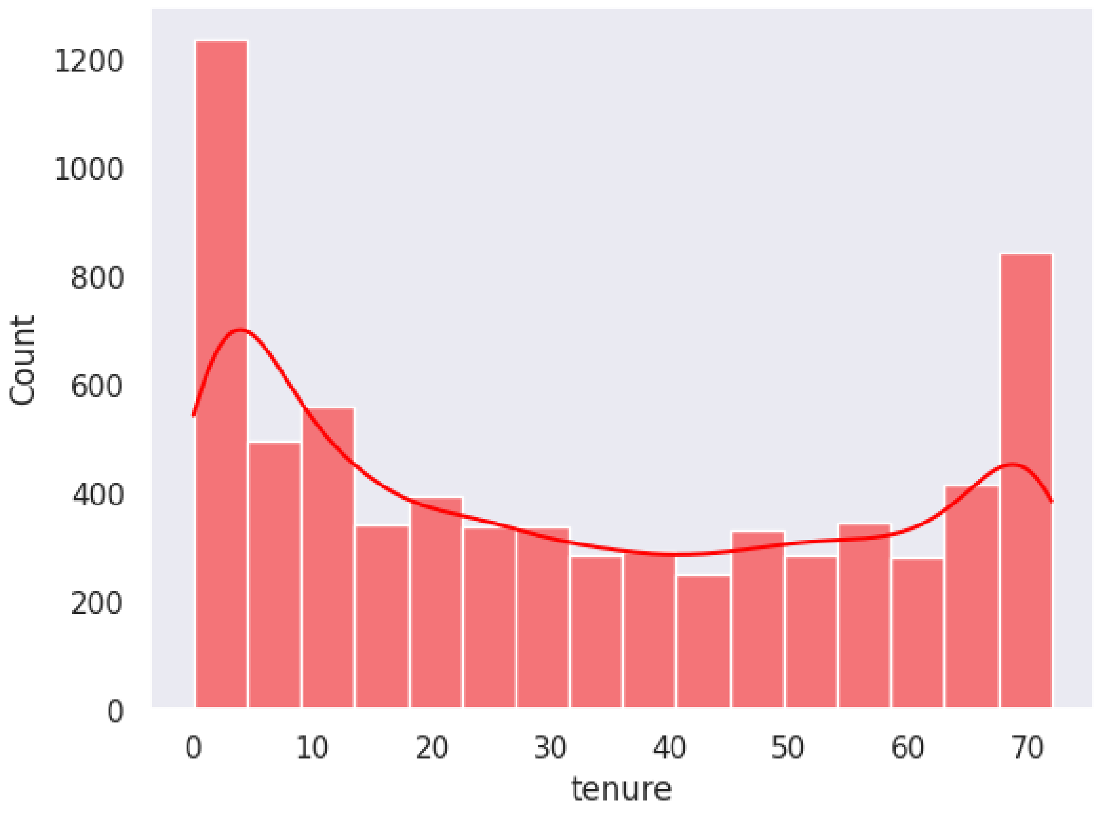


NUMERICAL COLUMNS EXPLORATION

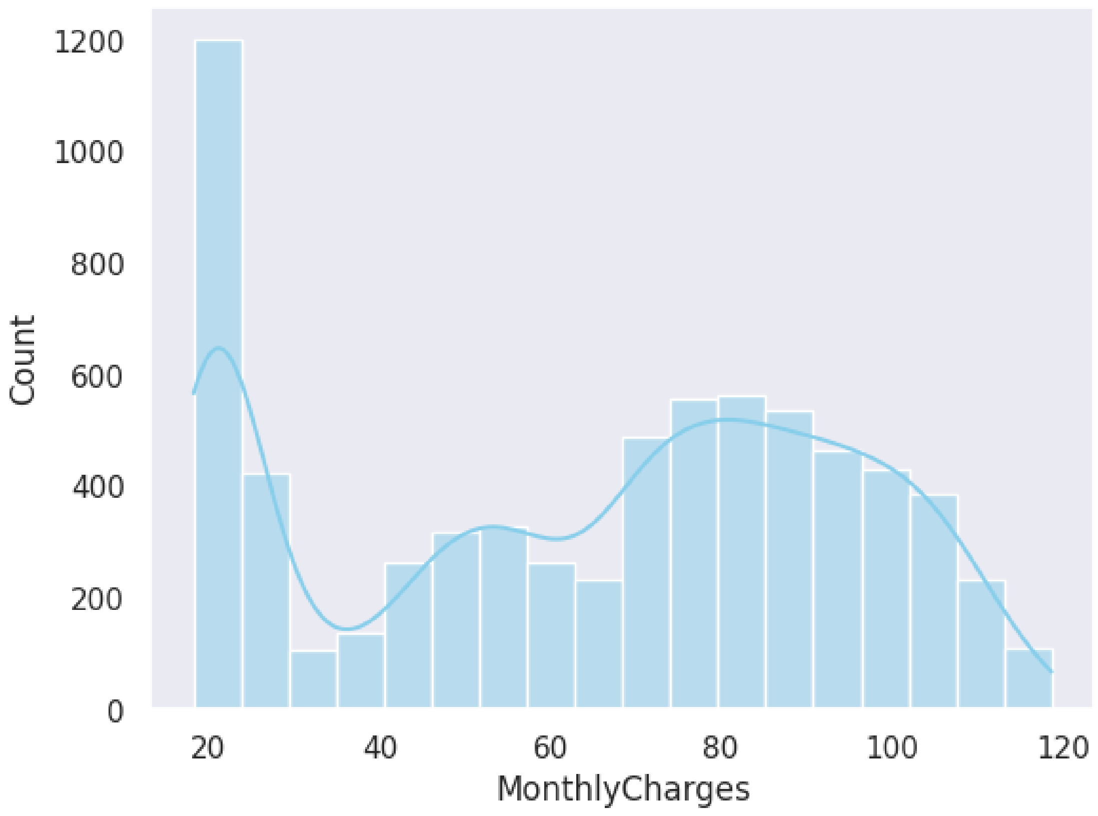
[86]:
```
sns.histplot(x = df['tenure'],kde = True, sns.color_palette='Set2')
plt.show()
```

```
---------------------------------------------------------------------------
SyntaxError                               Traceback (most recent call last)
/usr/local/lib/python3.10/dist-packages/IPython/core/compilerop.py in␣
 ↪ast_parse(self, source, filename, symbol)
     99         Arguments are exactly the same as ast.parse (in the standard␣
 ↪library),
    100         and are passed to the built-in compile function."""
--> 101         return compile(source, filename, symbol, self.flags |␣
 ↪PyCF_ONLY_AST, 1)
    102 
    103     def reset_compiler_flags(self):

SyntaxError: expression cannot contain assignment, perhaps you meant "=="?␣
 ↪(<ipython-input-86-0cc0f3b0e380>, line 1)
```

[89]:
```
sns.histplot(x=df['tenure'], kde=True, color='red')
plt.show()
```



```
[90]: sns.histplot(x = df['MonthlyCharges'],kde = True, color='skyblue')
      plt.show()
```



```
[115]: print(df.columns)
```

```
Index(['Partner', 'Churn'], dtype='object')
```

```
[ ]:
```

```
[11]: fig = px.sunburst(data_frame=df,
                        path=['gender', 'Churn'],
                        color='Churn',
                        title='Gender vs Churn'
                       )

      fig.update_traces(textinfo='label+percent parent')
      fig.update_layout(margin=dict(t=40, l=0, r=0, b=0))
      fig.show()
```

```
[8]: fig = px.sunburst(data_frame=df,
                       path=['SeniorCitizen', 'Churn'],
                       color='Churn',
                       title='SeniorCitizen vs Churn'
                      )
```

```
fig.update_traces(textinfo='label+percent parent')
fig.update_layout(margin=dict(t=40, l=0, r=0, b=0))
fig.show()
```

[10]:
```
fig = px.sunburst(data_frame=df,
                  path=['Partner', 'Churn'],
                  color='Churn',
                  title='Partner vs Churn'
                 )

fig.update_traces(textinfo='label+percent parent')
fig.update_layout(margin=dict(t=40, l=0, r=0, b=0))
fig.show()
```

[12]:
```
fig = px.sunburst(data_frame=df,
                  path=['Dependents', 'Churn'],
                  color='Churn',
                  title='Dependents vs Churn'
                 )

fig.update_traces(textinfo='label+percent parent')
fig.update_layout(margin=dict(t=40, l=0, r=0, b=0))
fig.show()
```

[13]:
```
fig = px.sunburst(data_frame=df,
                  path=['PhoneService', 'Churn'],
                  color='Churn',
                  title='PhoneService vs Churn'
                 )

fig.update_traces(textinfo='label+percent parent')
fig.update_layout(margin=dict(t=40, l=0, r=0, b=0))
fig.show()
```

[18]:
```
fig = px.histogram(data_frame = df,
             x = "MultipleLines",
             color="Churn", title="MultipleLines vs Churn")

fig.show()
```

[19]:
```
fig = px.histogram(data_frame = df,
             x = "InternetService",
             color="Churn", title="InternetService vs Churn")

fig.show()
```

[20]:
```python
fig = px.histogram(data_frame = df,
             x = "OnlineSecurity",
             color="Churn", title="OnlineSecurity vs Churn")

fig.show()
```

[21]:
```python
fig = px.histogram(data_frame = df,
             x = "OnlineBackup",
             color="Churn", title="OnlineBackup vs Churn")

fig.show()
```

[22]:
```python
fig = px.histogram(data_frame = df,
             x = "DeviceProtection",
             color="Churn", title="DeviceProtection vs Churn")

fig.show()
```

[23]:
```python
fig = px.histogram(data_frame = df,
             x = "TechSupport",
             color="Churn", title="TechSupport vs Churn")

fig.show()
```

[24]:
```python
fig = px.histogram(data_frame = df,
             x = "StreamingTV",
             color="Churn", title="StreamingTV vs Churn")

fig.show()
```

[25]:
```python
fig = px.histogram(data_frame = df,
             x = "StreamingMovies",
             color="Churn", title="StreamingMovies vs Churn")

fig.show()
```

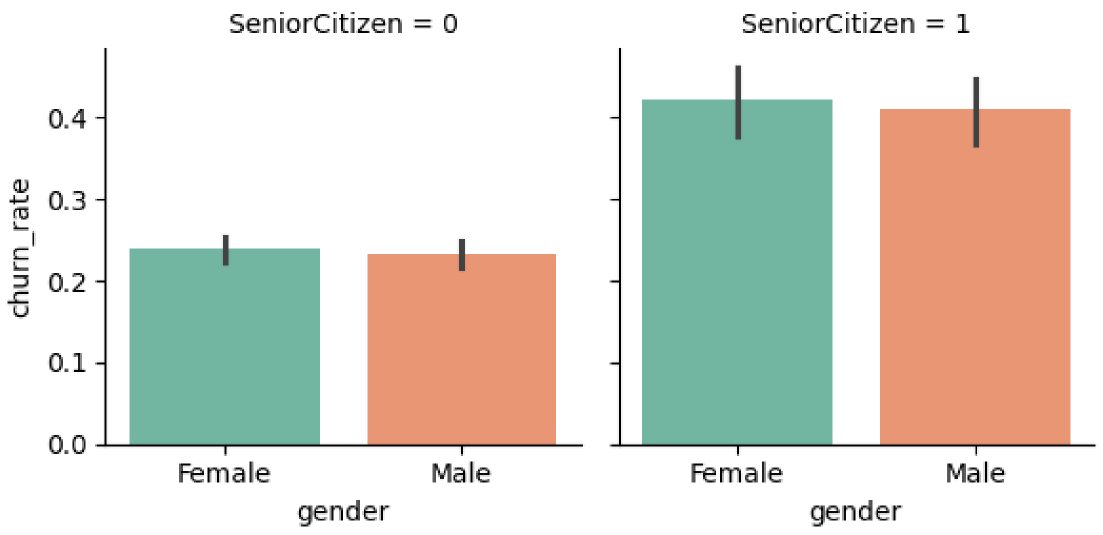
[26]:
```python
fig = px.histogram(data_frame = df,
             x = "Contract",
             color="Churn", title="Contract vs Churn")

fig.show()
```

[27]:
```python
fig = px.histogram(data_frame = df,
             x = "PaperlessBilling",
             color="Churn", title="PaperlessBilling vs Churn")

fig.show()
```

```
[28]: fig = px.histogram(data_frame = df,
                  x = "PaymentMethod",
                  color="Churn", title="PaymentMethod vs Churn")

      fig.show()
```

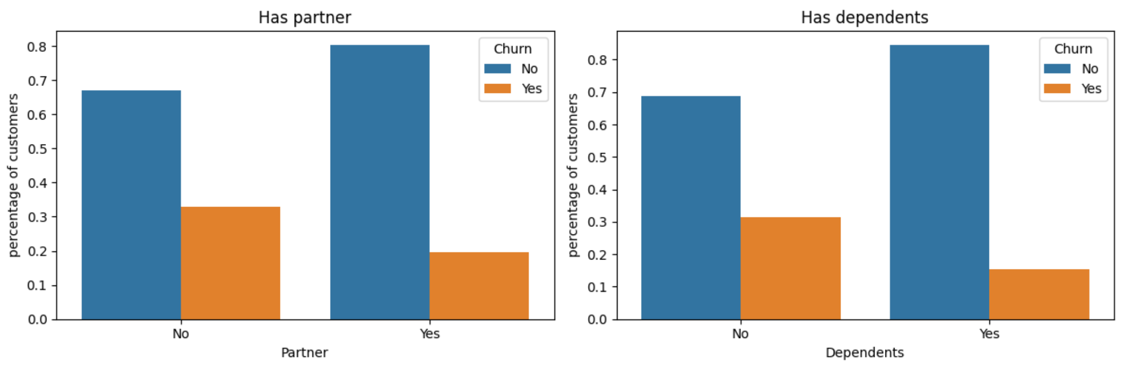
```
[8]: df['churn_rate'] = df['Churn'].replace("No", 0).replace("Yes", 1)
     g = sns.FacetGrid(df, col="SeniorCitizen")
     ax = g.map(sns.barplot, "gender", "churn_rate", palette = "Set2", order=␣
      ↪['Female', 'Male'])
```

```
/usr/local/lib/python3.10/dist-packages/seaborn/axisgrid.py:854: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  func(*plot_args, **plot_kwargs)
/usr/local/lib/python3.10/dist-packages/seaborn/axisgrid.py:854: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  func(*plot_args, **plot_kwargs)
```

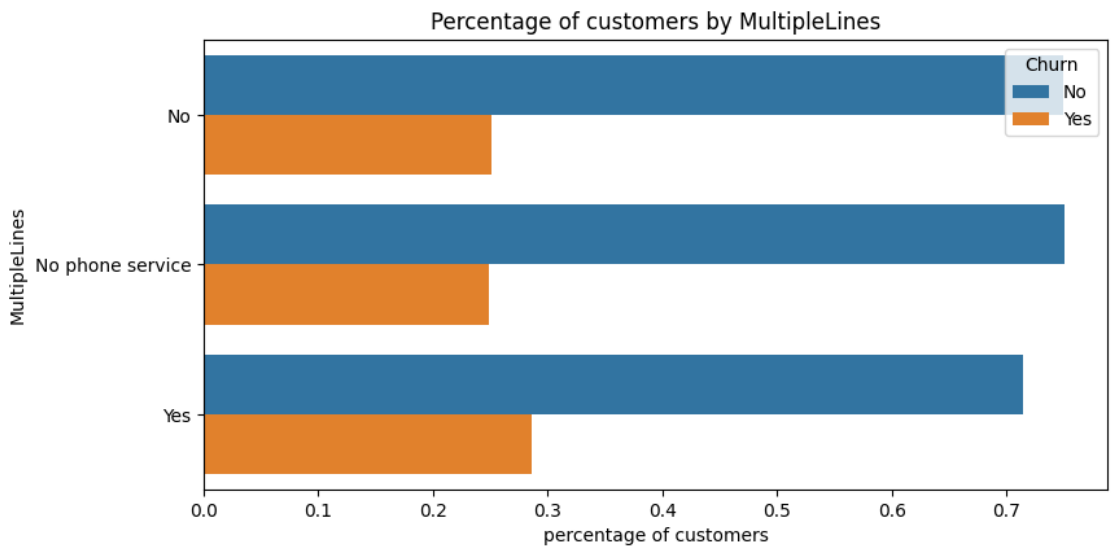


```
[10]: fig, axis = plt.subplots(1, 2, figsize=(12, 4))

      axis_titles = ["Has partner", "Has dependents"]
```

```
columns = ['Partner', 'Dependents']
axis_y = "percentage of customers"

for ax, col, title in zip(axis, columns, axis_titles):

    gp = df.groupby(col)["Churn"].value_counts(normalize=True).rename(axis_y).
 ↪reset_index()

    #plotting
    sns.barplot(x=col, y=axis_y, hue='Churn', data=gp, ax=ax)
    ax.set_title(title)

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



```
[14]: def barplot_percentages(column, orient='v'):

          gp = df.groupby(column)["Churn"].value_counts(normalize=True).
       ↪rename("percentage of customers").reset_index()

          if orient == 'h':
              sns.barplot(y=column, x="percentage of customers", hue='Churn',␣
       ↪data=gp, orient=orient)
          else:
              sns.barplot(x=column, y="percentage of customers", hue='Churn',␣
       ↪data=gp, orient=orient)

          plt.title(f"Percentage of customers by {column}")

      plt.figure(figsize=(9, 4.5))
      barplot_percentages("MultipleLines", orient='h')
      plt.show()
```

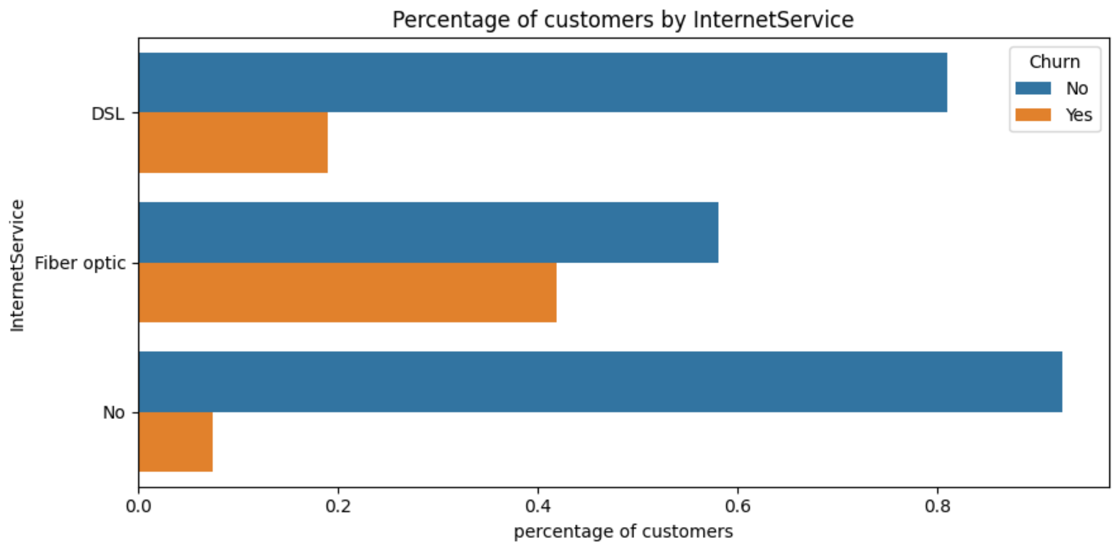


[12]:
```python
def barplot_percentages(column, orient='v'):

    gp = df.groupby(column)["Churn"].value_counts(normalize=True).
 ↪rename("percentage of customers").reset_index()

    if orient == 'h':
        sns.barplot(y=column, x="percentage of customers", hue='Churn',␣
 ↪data=gp, orient=orient)
    else:
        sns.barplot(x=column, y="percentage of customers", hue='Churn',␣
 ↪data=gp, orient=orient)

    plt.title(f"Percentage of customers by {column}")

plt.figure(figsize=(9, 4.5))
barplot_percentages("MultipleLines", orient='h')
plt.show()
```

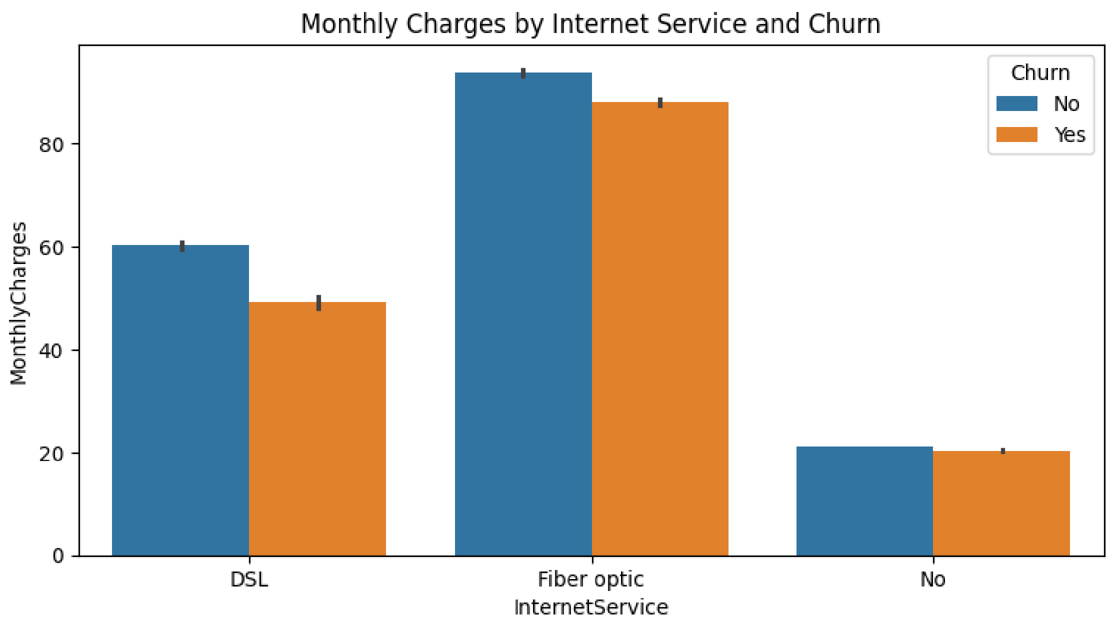
```
<Figure size 900x450 with 0 Axes>
```

[15]:
```python
def barplot_percentages(column, orient='v'):

    gp = df.groupby(column)["Churn"].value_counts(normalize=True).
 ↪rename("percentage of customers").reset_index()

    if orient == 'h':
        sns.barplot(y=column, x="percentage of customers", hue='Churn',␣
 ↪data=gp, orient=orient)
```

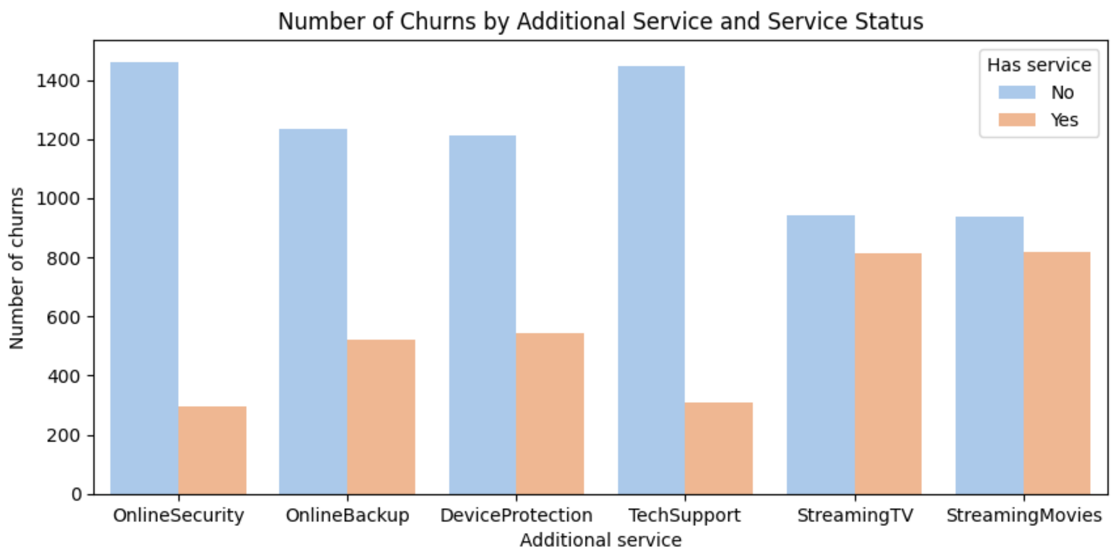
```
    else:
        sns.barplot(x=column, y="percentage of customers", hue='Churn',␣
 ↪data=gp, orient=orient)

    plt.title(f"Percentage of customers by {column}")
    plt.tight_layout()

plt.figure(figsize=(9, 4.5))
barplot_percentages("InternetService", orient="h")
plt.show()
```



[16]:
```
plt.figure(figsize=(9, 4.5))
ax = sns.barplot(x="InternetService", y="MonthlyCharges", hue="Churn", data=df)

plt.title("Monthly Charges by Internet Service and Churn")

plt.show()
```

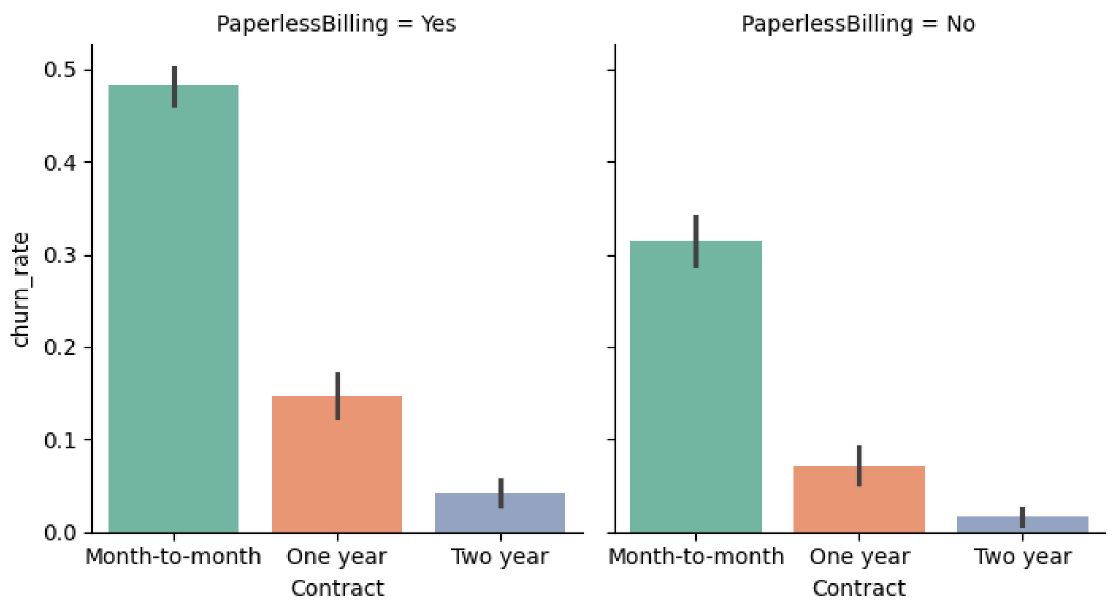


[19]:
```
cols = ["OnlineSecurity", "OnlineBackup", "DeviceProtection", "TechSupport",␣
↪"StreamingTV", "StreamingMovies"]
df1 = df[(df.InternetService != "No") & (df.Churn == "Yes")]
df1 = pd.melt(df1[cols]).rename({'value': 'Has service'}, axis=1)
plt.figure(figsize=(10, 4.5))
ax = sns.countplot(data=df1, x='variable', hue='Has service', hue_order=['No',␣
↪'Yes'], palette="pastel")

ax.set(xlabel='Additional service', ylabel='Number of churns')
ax.set_title('Number of Churns by Additional Service and Service Status')

plt.show()
```

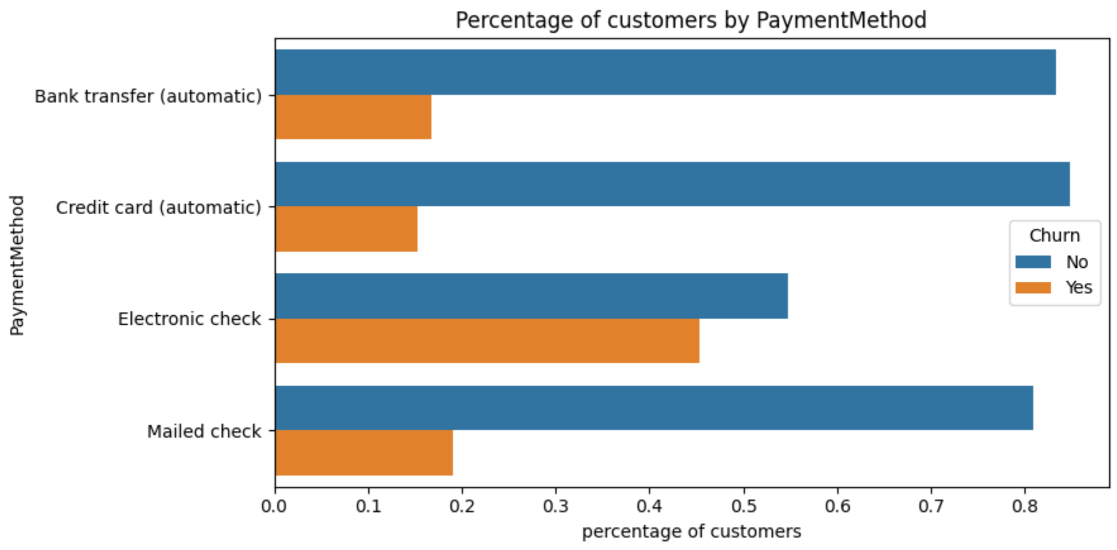


```
[21]: g = sns.FacetGrid(df, col="PaperlessBilling", height=4, aspect=.9)
      ax = g.map(sns.barplot, "Contract", "churn_rate", palette = "Set2", order=␣
        ↪['Month-to-month', 'One year', 'Two year'])
```

```
/usr/local/lib/python3.10/dist-packages/seaborn/axisgrid.py:854: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  func(*plot_args, **plot_kwargs)
/usr/local/lib/python3.10/dist-packages/seaborn/axisgrid.py:854: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  func(*plot_args, **plot_kwargs)
```

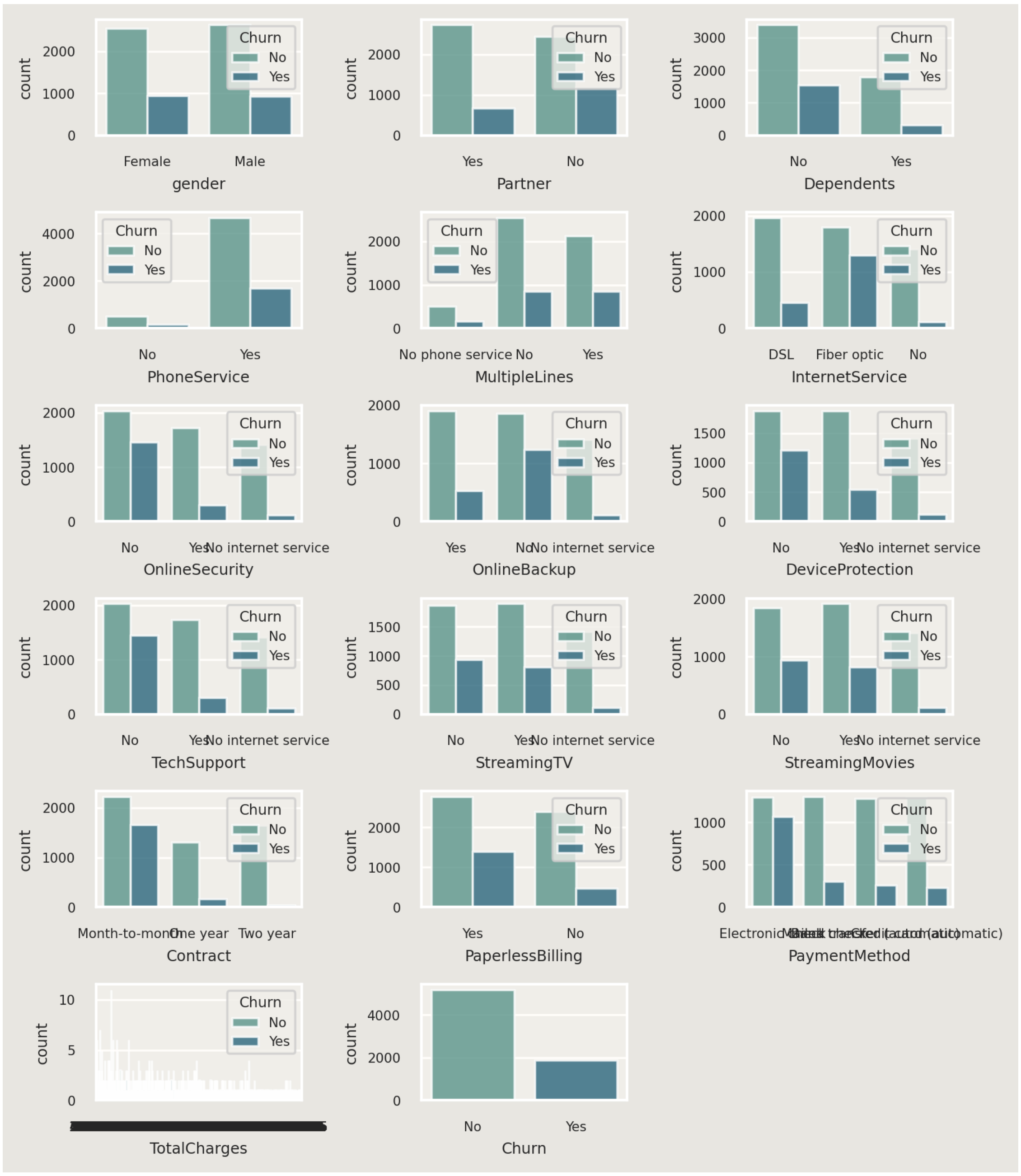


[24]:
```
plt.figure(figsize=(9, 4.5))
barplot_percentages("PaymentMethod", orient='h',)
```



[26]:
```
train_cat_visual_1 = df.select_dtypes(
                        include = ['object', 'category']).columns.tolist()
train_cat_visual_1.remove('customerID')
```

[27]:
```
sns.set_theme(rc = {'figure.dpi': 250, 'axes.labelsize': 7,
                    'axes.facecolor': '#f0eee9', 'grid.color': '#fffdfa',
                    'figure.facecolor': '#e8e6e1'}, font_scale = 0.55)

fig, ax = plt.subplots(6, 3, figsize = (6.5, 7.5))

for indx, (column, axes) in list(enumerate(list(zip(train_cat_visual_1,
                                                    ax.flatten())))):

    sns.countplot(ax = axes, x = df[column], hue = df['Churn'],
                  palette = 'crest', alpha = 0.8)

else:
    [axes.set_visible(False) for axes in ax.flatten()[indx + 1:]]

axes_legend = ax.flatten()

axes_legend[1].legend(title = 'Churn', loc = 'upper right')
axes_legend[2].legend(title = 'Churn', loc = 'upper right')

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



```
[28]: train_num_visual_0 = ['MonthlyCharges', 'tenure','TotalCharges']
```

OUTLIER ANALYSIS

```
[33]: columns_to_check = ['tenure', 'MonthlyCharges']

      def count_outliers(data, col):
          q1 = data[col].quantile(0.25)
          q3 = data[col].quantile(0.75)
```

```
    iqr = q3 - q1
    lower_limit = q1 - 1.5 * iqr
    upper_limit = q3 + 1.5 * iqr

    outliers_below = data[data[col] < lower_limit][col].size
    outliers_above = data[data[col] > upper_limit][col].size
    total_outliers = outliers_below + outliers_above

    if total_outliers == 0:
        print(f"No outliers in {col}")
    else:
        print(f"There are outliers in {col}")
        print(f'Count of outliers: {total_outliers}')

for col in columns_to_check:
    count_outliers(df, col)
```

```
No outliers in tenure
No outliers in MonthlyCharges
```

[34]:
```
df.drop(['customerID','Churn'],axis = 1,inplace = True)
```

[35]:
```
df
```

[35]:
```
      gender  SeniorCitizen Partner Dependents  tenure PhoneService  \
0     Female              0     Yes         No       1           No
1       Male              0      No         No      34          Yes
2       Male              0      No         No       2          Yes
3       Male              0      No         No      45           No
4     Female              0      No         No       2          Yes
…       …             …       …       …      …        …
7038    Male              0     Yes        Yes      24          Yes
7039  Female              0     Yes        Yes      72          Yes
7040  Female              0     Yes        Yes      11           No
7041    Male              1     Yes         No       4          Yes
7042    Male              0      No         No      66          Yes

         MultipleLines InternetService OnlineSecurity OnlineBackup  \
0     No phone service             DSL             No          Yes
1                   No             DSL            Yes           No
2                   No             DSL            Yes          Yes
3     No phone service             DSL            Yes           No
4                   No     Fiber optic             No           No
…                  …              …             …           …
7038               Yes             DSL            Yes           No
7039               Yes     Fiber optic             No          Yes
7040  No phone service             DSL            Yes           No
```

```
7041              Yes     Fiber optic            No            No
7042               No     Fiber optic           Yes            No

     DeviceProtection TechSupport StreamingTV StreamingMovies        Contract  \
0                  No          No          No              No  Month-to-month
1                 Yes          No          No              No        One year
2                  No          No          No              No  Month-to-month
3                 Yes         Yes          No              No        One year
4                  No          No          No              No  Month-to-month
…                 …          …          …             …             …
7038              Yes         Yes         Yes             Yes        One year
7039              Yes          No         Yes             Yes        One year
7040               No          No          No              No  Month-to-month
7041               No          No          No              No  Month-to-month
7042              Yes         Yes         Yes             Yes        Two year

     PaperlessBilling              PaymentMethod  MonthlyCharges TotalCharges  \
0                 Yes           Electronic check           29.85        29.85
1                  No               Mailed check           56.95       1889.5
2                 Yes               Mailed check           53.85       108.15
3                  No  Bank transfer (automatic)           42.30      1840.75
4                 Yes           Electronic check           70.70       151.65
…                 …                         …              …           …
7038              Yes               Mailed check           84.80       1990.5
7039              Yes    Credit card (automatic)          103.20       7362.9
7040              Yes           Electronic check           29.60       346.45
7041              Yes               Mailed check           74.40        306.6
7042              Yes  Bank transfer (automatic)          105.65       6844.5

      churn_rate
0              0
1              0
2              1
3              0
4              1
…            …
7038           0
7039           0
7040           0
7041           1
7042           0

[7043 rows x 20 columns]
```

ON HOT ENCODING

```
[36]: df1=pd.get_dummies(data=df,columns=['gender', 'Partner', 'Dependents',
             'PhoneService', 'MultipleLines', 'InternetService', 'OnlineSecurity',
             'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV',
             'StreamingMovies', 'Contract', 'PaperlessBilling',␣
        ↪'PaymentMethod'],drop_first=True)
      df1
```

```
[36]:       SeniorCitizen  tenure  MonthlyCharges TotalCharges  churn_rate  \
      0                 0       1           29.85        29.85           0
      1                 0      34           56.95       1889.5           0
      2                 0       2           53.85       108.15           1
      3                 0      45           42.30      1840.75           0
      4                 0       2           70.70       151.65           1
      …               …     …             …          …          …
      7038              0      24           84.80       1990.5           0
      7039              0      72          103.20       7362.9           0
      7040              0      11           29.60       346.45           0
      7041              1       4           74.40        306.6           1
      7042              0      66          105.65       6844.5           0

            gender_Male  Partner_Yes  Dependents_Yes  PhoneService_Yes  \
      0           False         True           False             False
      1            True        False           False              True
      2            True        False           False              True
      3            True        False           False             False
      4           False        False           False              True
      …             …          …             …                …
      7038         True         True            True              True
      7039        False         True            True              True
      7040        False         True            True             False
      7041         True         True           False              True
      7042         True        False           False              True

            MultipleLines_No phone service  …  StreamingTV_No internet service  \
      0                               True  …                            False
      1                              False  …                            False
      2                              False  …                            False
      3                               True  …                            False
      4                              False  …                            False
      …                              …  …                          …
      7038                           False  …                            False
      7039                           False  …                            False
      7040                            True  …                            False
      7041                           False  …                            False
      7042                           False  …                            False

            StreamingTV_Yes  StreamingMovies_No internet service  \
      0             False                                    False
      1             False                                    False
      2             False                                    False
      3             False                                    False
      4             False                                    False
      …             …                                      …
      7038           True                                    False
      7039           True                                    False
      7040          False                                    False
      7041          False                                    False
      7042           True                                    False

            StreamingMovies_Yes  Contract_One year  Contract_Two year  \
      0                   False              False              False
      1                   False               True              False
      2                   False              False              False
      3                   False               True              False
      4                   False              False              False
      …                   …                  …                  …
      7038                 True               True              False
      7039                 True               True              False
      7040                False              False              False
      7041                False              False              False
      7042                 True              False               True

            PaperlessBilling_Yes  PaymentMethod_Credit card (automatic)  \
      0                     True                                  False
      1                    False                                  False
      2                     True                                  False
      3                    False                                  False
      4                     True                                  False
      …                    …                                    …
      7038                  True                                  False
      7039                  True                                   True
      7040                  True                                  False
      7041                  True                                  False
      7042                  True                                  False

            PaymentMethod_Electronic check  PaymentMethod_Mailed check
      0                               True                       False
      1                              False                        True
      2                              False                        True
      3                              False                       False
      4                               True                       False
      …                              …                          …
      7038                           False                        True
      7039                           False                       False
```

```
7040                                True                              False
7041                               False                               True
7042                               False                              False

[7043 rows x 31 columns]
```

```
[37]: df1 = df1[['SeniorCitizen', 'tenure', 'MonthlyCharges', 'TotalCharges',
              'gender_Male', 'Partner_Yes', 'Dependents_Yes',
             'PhoneService_Yes', 'MultipleLines_No phone service',
             'MultipleLines_Yes', 'InternetService_Fiber optic',
             'InternetService_No', 'OnlineSecurity_No internet service',
             'OnlineSecurity_Yes', 'OnlineBackup_No internet service',
             'OnlineBackup_Yes', 'DeviceProtection_No internet service',
             'DeviceProtection_Yes', 'TechSupport_No internet service',
             'TechSupport_Yes', 'StreamingTV_No internet service', 'StreamingTV_Yes',
             'StreamingMovies_No internet service', 'StreamingMovies_Yes',
             'Contract_One year', 'Contract_Two year', 'PaperlessBilling_Yes',
             'PaymentMethod_Credit card (automatic)',
             'PaymentMethod_Electronic check', 'PaymentMethod_Mailed␣
 ↪check','churn_rate']]
```

```
[38]: df1
```

```
[38]:       SeniorCitizen  tenure  MonthlyCharges TotalCharges  gender_Male  \
0                 0       1           29.85        29.85        False
1                 0      34           56.95       1889.5         True
2                 0       2           53.85       108.15         True
3                 0      45           42.30      1840.75         True
4                 0       2           70.70       151.65        False
…               …     …             …           …          …
7038              0      24           84.80       1990.5         True
7039              0      72          103.20       7362.9        False
7040              0      11           29.60       346.45        False
7041              1       4           74.40        306.6         True
7042              0      66          105.65       6844.5         True

      Partner_Yes  Dependents_Yes  PhoneService_Yes  \
0            True           False             False
1           False           False              True
2           False           False              True
3           False           False             False
4           False           False              True
…           …             …               …
7038         True            True              True
7039         True            True              True
7040         True            True             False
7041         True           False              True
```

```
7042       False           False             True

      MultipleLines_No phone service  MultipleLines_Yes  …  StreamingTV_Yes  \
0                               True              False  …            False
1                              False              False  …            False
2                              False              False  …            False
3                               True              False  …            False
4                              False              False  …            False
…                                …                 …  …              …
7038                           False               True  …             True
7039                           False               True  …             True
7040                            True              False  …            False
7041                           False               True  …            False
7042                           False              False  …             True

      StreamingMovies_No internet service  StreamingMovies_Yes  \
0                                   False                False
1                                   False                False
2                                   False                False
3                                   False                False
4                                   False                False
…                                     …                  …
7038                                False                 True
7039                                False                 True
7040                                False                False
7041                                False                False
7042                                False                 True

      Contract_One year  Contract_Two year  PaperlessBilling_Yes  \
0                 False              False                  True
1                  True              False                 False
2                 False              False                  True
3                  True              False                 False
4                 False              False                  True
…                   …                …                   …
7038               True              False                  True
7039               True              False                  True
7040              False              False                  True
7041              False              False                  True
7042              False               True                  True

      PaymentMethod_Credit card (automatic)  PaymentMethod_Electronic check  \
0                                     False                            True
1                                     False                           False
2                                     False                           False
3                                     False                           False
4                                     False                            True
```

```
…                                    …                             …
7038                             False                         False
7039                              True                         False
7040                             False                          True
7041                             False                         False
7042                             False                         False

      PaymentMethod_Mailed check  churn_rate
0                          False           0
1                           True           0
2                           True           1
3                          False           0
4                          False           1
…                          …         …
7038                        True           0
7039                       False           0
7040                       False           0
7041                        True           1
7042                       False           0

[7043 rows x 31 columns]
```

imputation - fill missing values in the "TotalCharges" column of DataFrame df1 with the mean of the existing values.

[41]:
```python
import pandas as pd
import numpy as np
from sklearn.impute import SimpleImputer

# Replace empty strings with NaN
df1['TotalCharges'] = df1['TotalCharges'].replace(' ', np.nan)

# Imputation
imputer = SimpleImputer(missing_values=np.nan, strategy="mean")
df1.TotalCharges = imputer.fit_transform(df1["TotalCharges"].values.reshape(-1,␣
↪1))
```

[42]:
```python
#feature scaling
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report,confusion_matrix
scaler = StandardScaler()
scaler.fit(df1.drop(['churn_rate'],axis = 1))
scaled_features = scaler.transform(df1.drop('churn_rate',axis = 1))
```

[43]:
```python
#feature selection

from sklearn.model_selection import train_test_split
```

```
X = scaled_features
Y = df1['churn_rate']
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size = 0.
 ↪3,random_state=44)
```

Prediction Using Logistic Regression

```
# This is formatted as code
```

```
[44]: from sklearn.linear_model import LogisticRegression
logmodel = LogisticRegression()
logmodel.fit(X_train,Y_train)
```

```
[44]: LogisticRegression()
```

```
[45]: predR = logmodel.predict(X_test)
```

```
[47]: print(classification_report(Y_test,predR)) # Use predR instead of pred
print(confusion_matrix(Y_test,predR)) # Use predR instead of pred
```

```
              precision    recall  f1-score   support

           0       0.84      0.90      0.87      1557
           1       0.65      0.53      0.58       556

    accuracy                           0.80      2113
   macro avg       0.74      0.71      0.73      2113
weighted avg       0.79      0.80      0.79      2113

[[1397  160]
 [ 262  294]]
```

```
[48]: logmodel.score(X_test,Y_test)
```

```
[48]: 0.8002839564600095
```

Prediction using Decision Tree

```
[49]: from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier()
dtc.fit(X_train, Y_train)
y_pred = dtc.predict(X_test)
```

```
[50]: confusion_matrix(Y_test, y_pred)
```

```
[50]: array([[1260,  297],
       [ 284,  272]])
```

```
[51]: print(classification_report(Y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.82      0.81      0.81      1557
           1       0.48      0.49      0.48       556

    accuracy                           0.73      2113
   macro avg       0.65      0.65      0.65      2113
weighted avg       0.73      0.73      0.73      2113
```

```
[52]: from sklearn.metrics import accuracy_score
      print(accuracy_score(Y_test, y_pred))
```

```
0.7250354945575012
```