teleco-churned-analysis

April 22, 2025

[12]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[16]:
```python
df = pd.read_csv("Customer Churn.csv")
df.head()
```

[16]:
```
   customerID  gender  SeniorCitizen Partner Dependents  tenure PhoneService  \
0  7590-VHVEG  Female              0     Yes         No       1           No
1  5575-GNVDE    Male              0      No         No      34          Yes
2  3668-QPYBK    Male              0      No         No       2          Yes
3  7795-CFOCW    Male              0      No         No      45           No
4  9237-HQITU  Female              0      No         No       2          Yes

      MultipleLines InternetService OnlineSecurity  … DeviceProtection  \
0  No phone service             DSL             No  …               No
1                No             DSL            Yes  …              Yes
2                No             DSL            Yes  …               No
3  No phone service             DSL            Yes  …              Yes
4                No     Fiber optic             No  …               No

  TechSupport StreamingTV StreamingMovies        Contract PaperlessBilling  \
0          No          No              No  Month-to-month              Yes
1          No          No              No        One year               No
2          No          No              No  Month-to-month              Yes
3         Yes          No              No        One year               No
4          No          No              No  Month-to-month              Yes

               PaymentMethod MonthlyCharges  TotalCharges Churn
0           Electronic check          29.85         29.85    No
1               Mailed check          56.95        1889.5    No
2               Mailed check          53.85        108.15   Yes
3  Bank transfer (automatic)          42.30       1840.75    No
4           Electronic check          70.70        151.65   Yes

[5 rows x 21 columns]
```

[22]:
```
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   float64
 20  Churn             7043 non-null   object
dtypes: float64(2), int64(2), object(17)
memory usage: 1.1+ MB
```

Replaceing blanks with 0 as tenure is 0 and no total charges are recorded. Data type is also converted to float

[20]:
```
df['TotalCharges'] = df['TotalCharges'].replace(" ","0")
df['TotalCharges'] = df['TotalCharges'].astype("float")
```

[34]:
```
df.isnull().sum().sum()
```

[34]:
```
0
```

[36]:
```
df.describe()
```

[36]:
```
       SeniorCitizen       tenure  MonthlyCharges  TotalCharges
count    7043.000000  7043.000000     7043.000000   7043.000000
mean        0.162147    32.371149       64.761692   2279.734304
std         0.368612    24.559481       30.090047   2266.794470
min         0.000000     0.000000       18.250000      0.000000
```

```
25%          0.000000     9.000000       35.500000    398.550000
50%          0.000000    29.000000       70.350000   1394.550000
75%          0.000000    55.000000       89.850000   3786.600000
max          1.000000    72.000000      118.750000   8684.800000
```

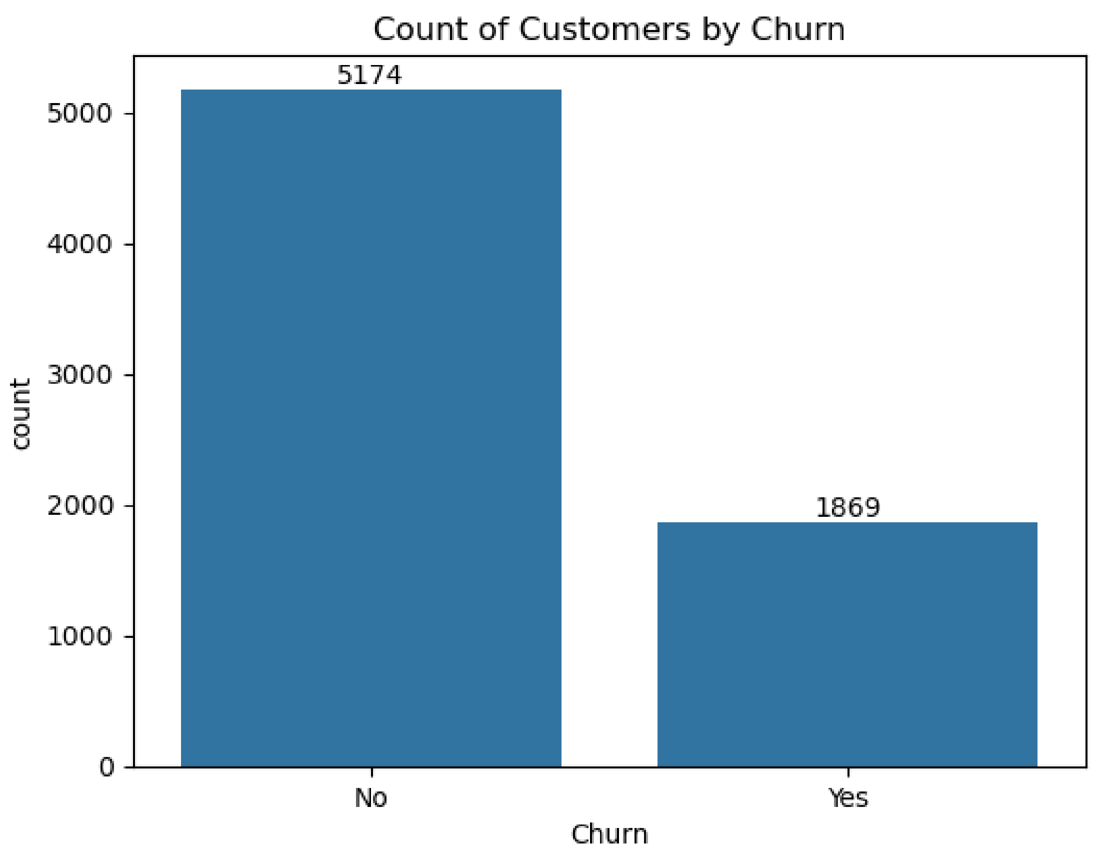
[42]:
```python
df['customerID'].duplicated().sum()
```

[42]:
```
0
```

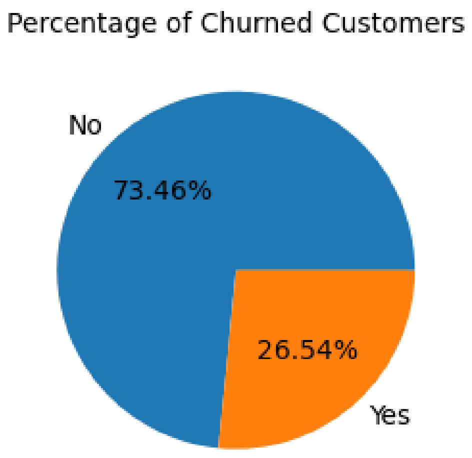
[44]:
```python
def conv(value):
    if value == 1:
        return "yes"
    else:
        return "no"

df['SeniorCitizen'] = df['SeniorCitizen'].apply(conv)
```

0.0.1 Converted 0 and 1 values of senior citizen to yes/no to make it easier to understand

[71]:
```python
ax = sns.countplot(x = 'Churn', data = df)

ax.bar_label(ax.containers[0])
plt.title("Count of Customers by Churn")
plt.show()
```



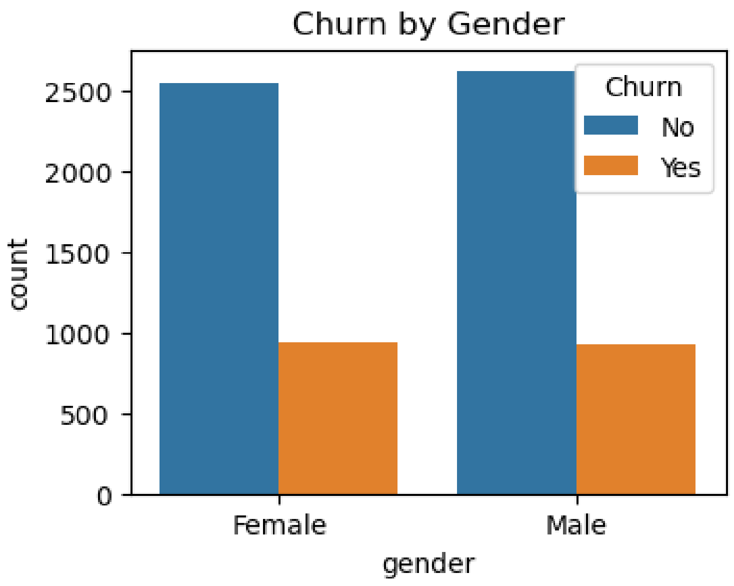
[76]:
```
plt.figure(figsize = (3,4))
gb = df.groupby("Churn").agg({'Churn':"count"})
plt.pie(gb['Churn'], labels = gb.index , autopct = "%1.2f%%")
plt.title("Percentage of Churned Customers", fontsize = 10)
plt.show()
```



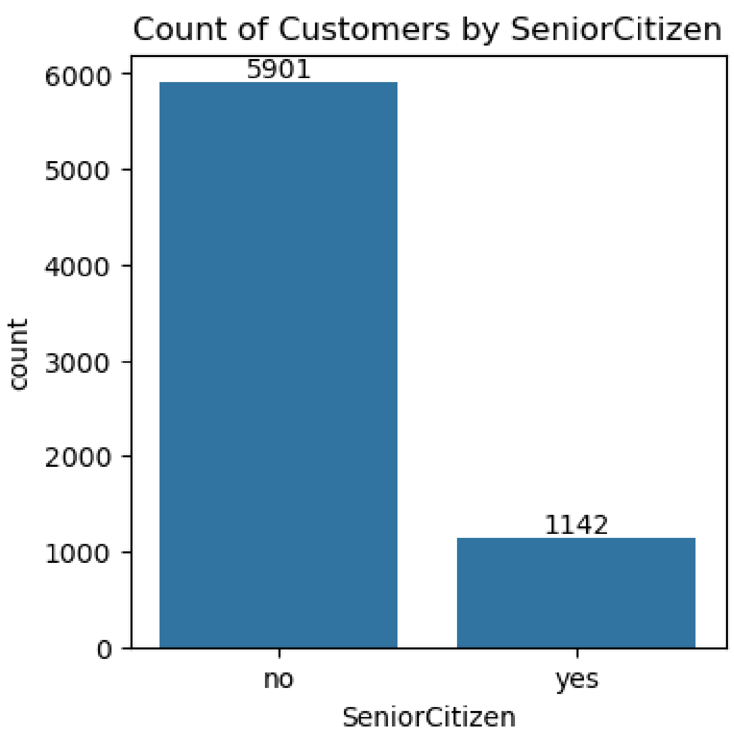
0.0.2 From the given pie chart we can conclude that 26.54% of our customers have churned out

Now let's explore the reason behind it

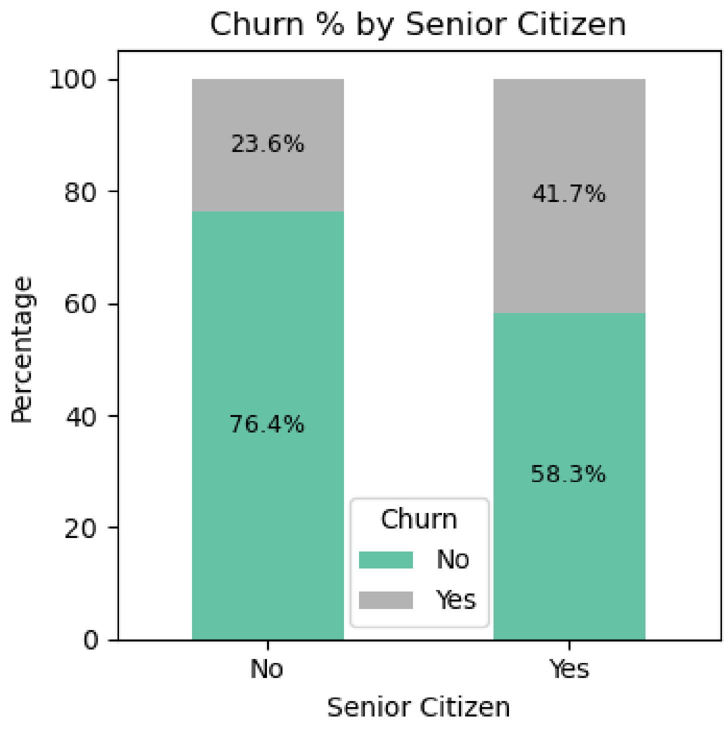
[105]:
```
plt.figure(figsize = (4,3))
sns.countplot(x = "gender", data = df, hue = "Churn")
plt.title("Churn by Gender")
plt.show()
```



[121]:
```
plt.figure(figsize = (4,4))
ax = sns.countplot(x = "SeniorCitizen", data = df)
ax.bar_label(ax.containers[0])
plt.title("Count of Customers by SeniorCitizen")
plt.show()
```

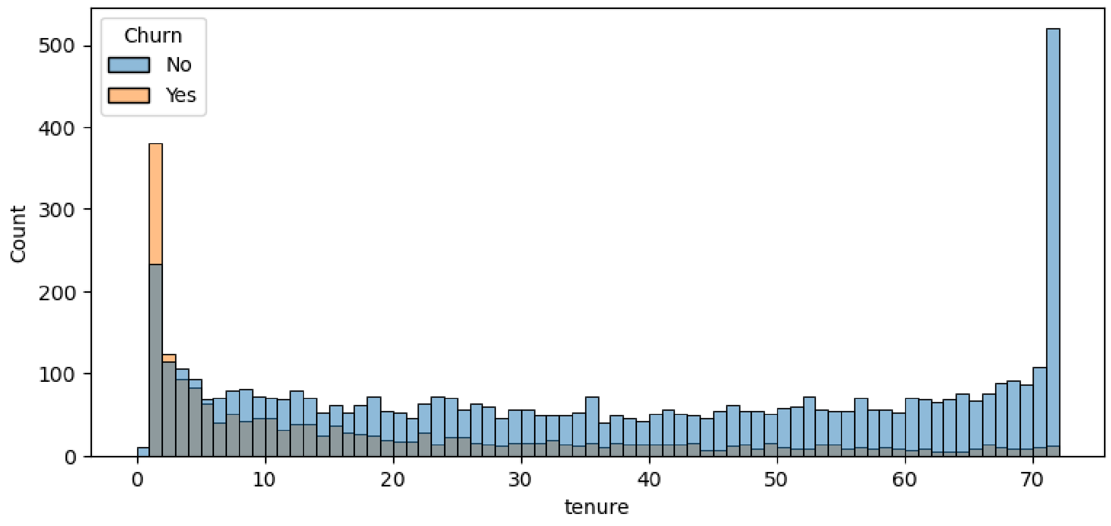


[134]:
```python
ct = pd.crosstab(df['SeniorCitizen'], df['Churn'])

# Step 2: Convert to percent of total (row-wise)
ct_percent = ct.div(ct.sum(axis=1), axis=0) * 100

# Step 3: Plot stacked bar chart
ax = ct_percent.plot(kind='bar', stacked=True, figsize=(4, 4), colormap='Set2')

# Step 4: Add percentage labels
for i, row in enumerate(ct_percent.values):
    cum_sum = 0
    for j, val in enumerate(row):
        if val > 0:
            ax.text(i, cum_sum + val/2, f'{val:.1f}%', ha='center',␣
 ↪va='center', fontsize=9)
            cum_sum += val

# Step 5: Beautify
plt.title('Churn % by Senior Citizen')
plt.xlabel('Senior Citizen')
plt.ylabel('Percentage')
plt.xticks(ticks=[0, 1], labels=['No', 'Yes'], rotation=0)
plt.legend(title='Churn')
plt.tight_layout()
plt.show()
```



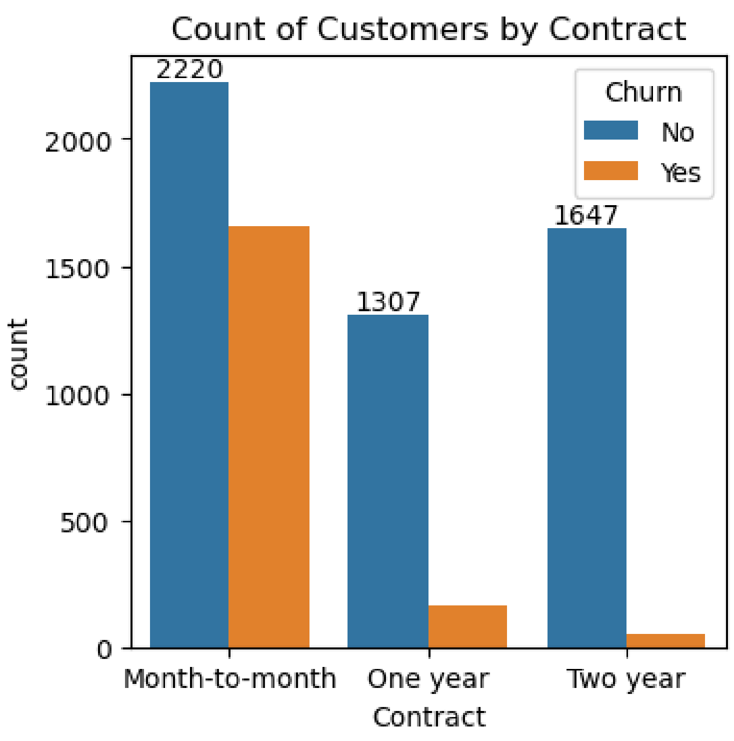
0.0.3 Comparitively a greater percentage of people in senior cititzen category have churned.

[140]:
```
plt.figure(figsize = (9,4))
sns.histplot(x = "tenure", data = df, bins = 72, hue = "Churn")
plt.show()
```



0.0.4 People who have used our services for a long time have stayed and peope who have used out services for one or two months have churned

[149]:
```
plt.figure(figsize = (4,4))
ax = sns.countplot(x = "Contract", data = df, hue = "Churn")
ax.bar_label(ax.containers[0])
plt.title("Count of Customers by Contract")
plt.show()
```



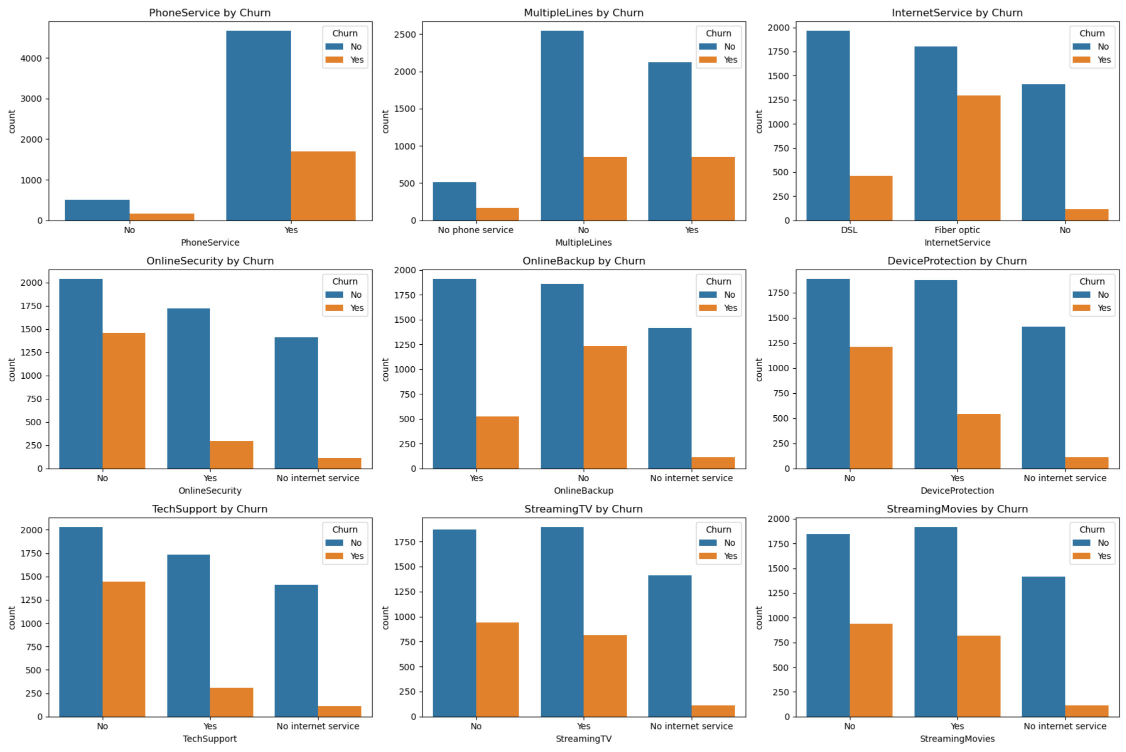
0.0.5 People who have month to month contract are likely to churn than from those who have 1 or 2 years of contract

[152]:
```
df.columns.values
```

[152]:
```
array(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
       'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract',
       'PaperlessBilling', 'PaymentMethod', 'MonthlyCharges',
       'TotalCharges', 'Churn'], dtype=object)
```

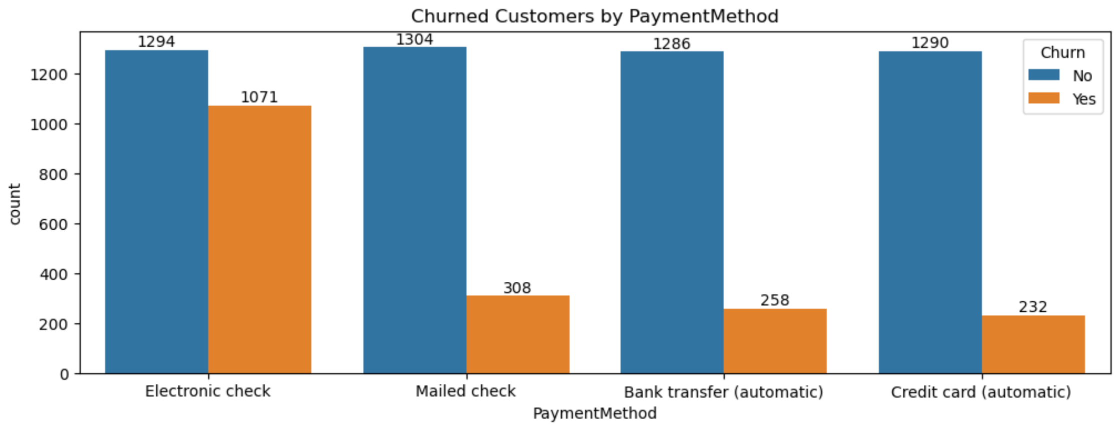
[170]:
```
# List of columns to plot
cols = ['PhoneService', 'MultipleLines', 'InternetService',
        'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
        'TechSupport', 'StreamingTV', 'StreamingMovies']

# Number of subplots
n_cols = 3
n_rows = (len(cols) + n_cols - 1) // n_cols

# Set up the figure
fig, axes = plt.subplots(n_rows, n_cols, figsize=(18, 12))
```

```python
axes = axes.flatten()  # Flatten to easily iterate

# Plot each countplot
for i, col in enumerate(cols):
    sns.countplot(x=col, data=df, hue='Churn', ax=axes[i])
    axes[i].set_title(f'{col} by Churn')
    axes[i].tick_params(axis='x', rotation=0)

# Turn off any unused subplots
for j in range(i+1, len(axes)):
    fig.delaxes(axes[j])

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



0.0.6 The majority of customers who do not churn tend to have services like PhoneService, INternetService (particularly DSL), and OnlineSecurity enabled. For services like OnlineBackup, TechSupport, and StreamingTV, churn rates are noticeably higher when these services are not used or are unavailable.

```
[168]: plt.figure(figsize = (12,4))
ax = sns.countplot(x = "PaymentMethod", data = df, hue = "Churn")
ax.bar_label(ax.containers[0])
ax.bar_label(ax.containers[1])
plt.title("Churned Customers by PaymentMethod")
plt.show()
```



0.0.7 Customer is likely to churn when he is using electronic check as a payment method.