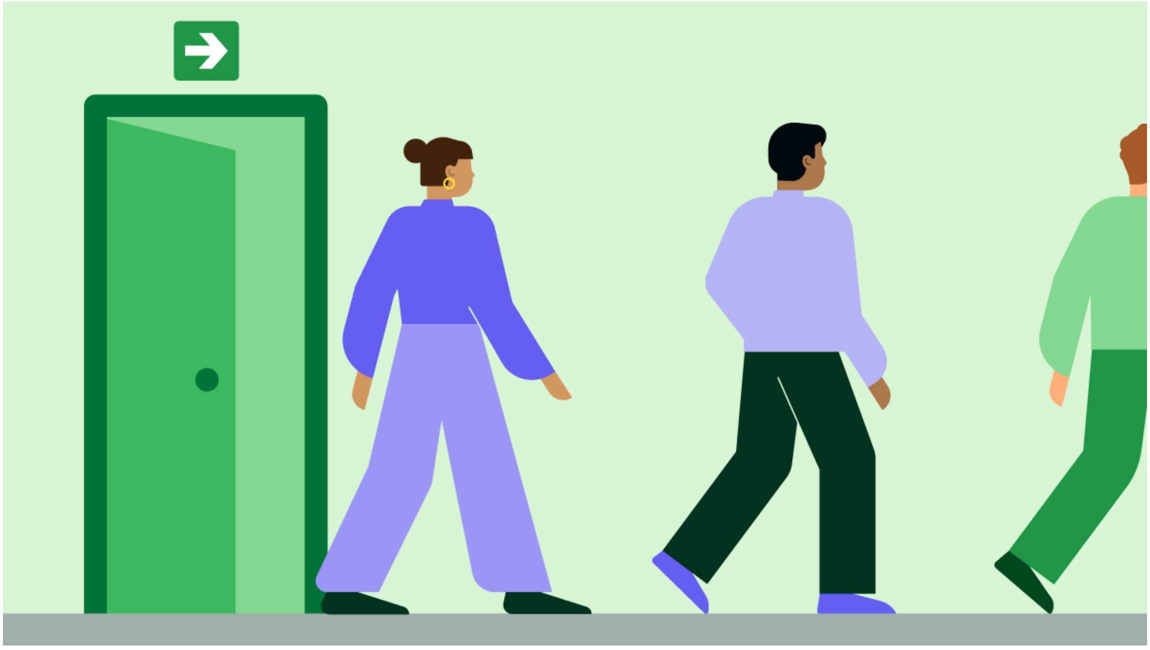


Business Understanding

- In the highly competitive telecommunications industry, customer churn (where customers stop using a company's services) presents a serious threat to profitability and long-term sustainability. Industry churn rates typically range from 15% to 25%, making it one of the most critical performance metrics to monitor and reduce.
- For SyriaTel, churn is more than just a number, it represents lost revenue, higher customer acquisition costs, and weakened market position. Studies show that acquiring a new customer is 5 to 20 times more expensive than retaining an existing one depending on the industry, thus reinforcing why customer retention is a strategic priority.
- SyriaTel offers a wide range of services including mobile calls, messaging, internet, and data bundles. Despite its reputation for customer service and social responsibility, the company continues to experience significant churn due to competitive pricing, service dissatisfaction, and customer disengagement. Left unaddressed, this churn could erode SyriaTel's market share and damage brand loyalty.

Objectives

- The primary objective of this project is to build a predictive model that can identify customers who are most likely to churn. By accurately predicting churn risk, SyriaTel can proactively implement targeted retention strategies to reduce churn and improve customer satisfaction.
- This project aims to:
    - Identify factors that contribute most to customer churn.

- Classify customers as likely to churn ("True") or stay ("False").

- Enable actionable insights to guide SyriaTel's marketing, sales, and support teams in preventing churn.

- Improve customer retention, thus reducing revenue loss and supporting long-term profitability.

Stakeholders

- The key stakeholders for this project include:
  - Marketing Team: Interested in identifying at-risk customers for targeted retention campaigns.
  - Customer Service Team: Needs to understand how support quality and call volume relate to churn and implement new escalation protocols based on churn risk.
  - Finace Team: Monitors revenue Impact from Customer loss and use this insights to forecast revenue and allocate budgets to retention.
  - Executive Team: Concerned with overall business performance and customer retention strategies.

Data Understanding

a) Dataset Overview

- The dataset provided contains customer-level usage and service information from SyriaTel, aimed at identifying patterns that lead to customer churn.
- The dataset has 3333 rows which represents customers and 21 column which captures features that influence their decision to stay (Not churned) or leave(Churned) the service.

Variable Description

There are 20 features (independent variables) and 1 target variable (churn). Below is a breakdown of each variable:

- Customer Identity:

  1. state - This shows the state where the customer resides and it can be help identify geographic patterns in churn.
  2. area code - Associated with the customers Phone number.
  3. phone number - Customer's phone number (serves as an Unique identifier)
- Tenure - how long a customer has been with the company.

  1. account length- Duration of customer's relationship with SyriaTel.
- Service Plan - what services the customer is subscribed to

  1. international plan- Indicates whether the customer has an international calling plan (`yes` / `no`)

2. voice mail plan - Indicates whether the customer has subscribed to voice mail service plan (`yes` / `no`)

- Usage Behaviour - Measures how actively customers use the services.

    - Daytime Usage

        1. total day calls - Total number of calls made during the day
        2. total day minutes - Total number of minutes the customer has spent on calls during the day

        - Evening Usage
            1. total eve minutes - Total minutes of calls made in the evening
            2. total eve calls - Total number of evening calls

```
  - Night Usage
       1. total night minutes- The total number of minutes the
customer has spent on calls during the night.
       2. total night calls - The total number of calls the
customer has made during the night

  - International Usage
       1. total intl minutes -  The total number of minutes the
customer has spent on international calls.
       2. total intl calls  -  The total number of international
calls the customer has made.

  - Voice Mail
       1. number vmail messages - Number of voice mail messages the
customer has received.
```

- Charges (Financial Impact) - Does billing higher charges lead to dissatisfaction.

    1. total night charge - The total charges incurred by the customer for nighttime calls.
    2. total intl charge - The total charges incurred by the customer for international calls.
    3. total eve charge - Total charges Incurred by the customer for evening call
    4. total day charge - Total charge incurred by the customer for daytime calls

- Customer Support Interaction

    1. customer service calls - The number of times the customer has called customer service.

- Target Variable

    1. churn - Whether the customer has churned (`True` = churned, `False` = active)

Libraries

In [125…

```
# Creates interactive charts and maps
!pip install plotly

! pip install -U scikit-learn imbalanced-learn
```

```
Requirement already satisfied: plotly in c:\users\a808865\python\lib\site-packages
(5.9.0)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\a808865\python\lib\site
-packages (from plotly) (8.2.2)
Requirement already satisfied: scikit-learn in c:\users\a808865\python\lib\site-pa
ckages (1.6.1)
Collecting scikit-learn
  Obtaining dependency information for scikit-learn from https://files.pythonhoste
d.org/packages/b2/3b/47b5eaee01ef2b5a80ba3f7f6ecf79587cb458690857d4777bfd77371c6f/
scikit_learn-1.7.1-cp311-cp311-win_amd64.whl.metadata
  Using cached scikit_learn-1.7.1-cp311-cp311-win_amd64.whl.metadata (11 kB)
Requirement already satisfied: imbalanced-learn in c:\users\a808865\python\lib\sit
e-packages (0.13.0)
Requirement already satisfied: numpy>=1.22.0 in c:\users\a808865\python\lib\site-p
ackages (from scikit-learn) (1.24.3)
Requirement already satisfied: scipy>=1.8.0 in c:\users\a808865\python\lib\site-pa
ckages (from scikit-learn) (1.11.1)
Requirement already satisfied: joblib>=1.2.0 in c:\users\a808865\python\lib\site-p
ackages (from scikit-learn) (1.2.0)
Requirement already satisfied: threadpoolctl>=3.1.0 in c:\users\a808865\python\lib
\site-packages (from scikit-learn) (3.6.0)
Requirement already satisfied: sklearn-compat<1,>=0.1 in c:\users\a808865\python\l
ib\site-packages (from imbalanced-learn) (0.1.3)
```

In [126…

```
# Essential for data manipulation and analysis
import pandas as pd

# For data visualization
import matplotlib.pyplot as plt

# For statistical data visualization
import seaborn as sns

# creates interactive visualizations
import plotly.express as px

# Scales features to ensure they contribute equally to the model
from sklearn.preprocessing import StandardScaler

#
from sklearn.linear_model import LogisticRegression


from sklearn.tree import DecisionTreeClassifier

from sklearn.model_selection import train_test_split, cross_val_score, GridSearchC\

from sklearn.preprocessing import OneHotEncoder, LabelEncoder, StandardScaler

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

from scipy import stats


import statsmodels.api as sm

from sklearn.compose import ColumnTransformer


from imblearn.over_sampling import SMOTE
```

Loading Datasets

In [127…

```
# Reading the CSV file & displaying the first 5 rows
df = pd.read_csv('Data\Customer_churn.csv')
df.head()
```

Out[127]:

	state	area code	phone number	account length	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total day charge	...	tc c chai
0	AK	408	341-9764	36	no	yes	30	146.3	128	24.87	...	13
1	AK	408	366-4467	104	no	no	0	278.4	106	47.33	...	6
2	AK	408	336-5406	78	no	no	0	225.1	67	38.27	...	16
3	AK	408	396-2335	110	no	no	0	100.1	90	17.02	...	19
4	AK	408	383-9255	127	no	no	0	182.3	124	30.99	...	14

5 rows × 22 columns

b) Data Processing

In this section, I prepare the data for exploratory data analysis (EDA) and modeling. The following checks was performed:

- Check the overall structiure of the dataset & SUmmary statistics
- Checking for missing values
- Irrelevant Columns - Removing columns that do not contribute to the analysis (Phone number)
- Checking for Data types and converting to correct format if needed
- Duplicates - Identifying and removing any duplicate entries.
- Checking for Outliers

In [128…

```
df.shape
```

Out[128]:

```
(3333, 22)
```

Duplicates

- There is no duplicates

In [129…

```
df.duplicated().sum()
```

Out[129]:

```
0
```

Structure of data

- Number of observation
- counts of columns
- Data type conversions

In [130…

```
# Data Types & Null Values
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3333 entries, 0 to 3332
Data columns (total 22 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   state                   3333 non-null   object 
 1   area code               3333 non-null   int64  
 2   phone number            3333 non-null   object 
 3   account length          3333 non-null   int64  
 4   international plan      3333 non-null   object 
 5   voice mail plan         3333 non-null   object 
 6   number vmail messages   3333 non-null   int64  
 7   total day minutes       3333 non-null   float64
 8   total day calls         3333 non-null   int64  
 9   total day charge        3333 non-null   float64
 10  total eve minutes       3333 non-null   float64
 11  total eve calls         3333 non-null   int64  
 12  total eve charge        3333 non-null   float64
 13  total night minutes     3333 non-null   float64
 14  total night calls       3333 non-null   int64  
 15  total night charge      3333 non-null   float64
 16  total intl minutes      3333 non-null   float64
 17  total intl calls        3333 non-null   int64  
 18  total intl charge       3333 non-null   float64
 19  Total Revenu            3333 non-null   float64
 20  customer service calls  3333 non-null   int64  
 21  churn                   3333 non-null   bool   
dtypes: bool(1), float64(9), int64(8), object(4)
memory usage: 550.2+ KB
```

- Data has no missing value
- Categorical Variable:
  - state
  - area code
  - international plan
  - voicemail plan
- Numerical Variable:
  - account length
  - number vmail messages
  - total day minutes
  - total day calls
  - total day charge
  - total eve minutes
  - total eve calls
  - total eve charge
  - total night minutes

- total night calls
- total night charge
- total intl minutes
- total intl charge
- customer service calls

In [131…

```
# Changing Data Types of Area Code and Phone Number to String
df['area code'] = df['area code'].astype(str)

df['phone number'] = df['phone number'].astype(str)
```

Statistical Summary

In [132…

```
# Summary Statistics
df.describe().round(2)
```

Out[132]:

	account length	number vmail messages	total day minutes	total day calls	total day charge	total eve minutes	total eve calls	total eve charge	total night minutes	tota night calls
count	3333.00	3333.00	3333.00	3333.00	3333.00	3333.00	3333.00	3333.00	3333.00	3333.0C
mean	101.06	8.10	179.78	100.44	30.56	200.98	100.11	17.08	200.87	100.11
std	39.82	13.69	54.47	20.07	9.26	50.71	19.92	4.31	50.57	19.57
min	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	23.20	33.0C
25%	74.00	0.00	143.70	87.00	24.43	166.60	87.00	14.16	167.00	87.0C
50%	101.00	0.00	179.40	101.00	30.50	201.40	100.00	17.12	201.20	100.0C
75%	127.00	20.00	216.40	114.00	36.79	235.30	114.00	20.00	235.30	113.0C
max	243.00	51.00	350.80	165.00	59.64	363.70	170.00	30.91	395.00	175.0C

- Key Observation from summary Statistics:
  - The average customer has been with SyriaTel for 101 days
  - Most customers do not use the voicemail feature, as shown by a median of 0 messages despite a mean of 8.1, indicating a strong skew that may help identify disengaged users.
  - Daytime Usage shows the highest variability, with an average talk time of 179.8 minutes and an average charge of $30.56.
  - Evening Usage shows that the customers average 200.98 minutes with less variability and a lower average charge ($17.08).
  - Nighttime usage is Similar to evening: around 200.87 minutes and $9.04 charge, with lower variability.
  - International Usage is generally low, averaging 10.24 minutes and $2.76 in charges, with a 20-minute maximum, indicating that only a subset of users rely on this feature.
  - While most customers contact customer service 1–2 times, those with up to 9 calls reflects dissatisfaction.

Unique values

In [133…

```
# Unique Outputs
df.nunique()
```

Out[133]:

```
state                      50
area code                   3
phone number             3333
account length            212
international plan          2
voice mail plan             2
number vmail messages      46
total day minutes        1667
total day calls           119
total day charge         1667
total eve minutes        1611
total eve calls           123
total eve charge         1440
total night minutes      1591
total night calls         120
total night charge        933
total intl minutes        162
total intl calls           21
total intl charge         162
Total Revenu             2227
customer service calls     10
churn                       2
dtype: int64
```

In [134…

```
# Displaying unique values
for column in df.columns:
    print(f"{column}:")
    print(f" - Unique Values: {df[column].unique()}...")  # Limiting to the first 5
    print("\n")
```

```
state:
 - Unique Values: ['AK' 'AL' 'AR' 'AZ' 'CA' 'CO' 'CT' 'WA' 'DE' 'FL' 'GA' 'HI' 'I
A' 'ID'
 'IL' 'IN' 'KS' 'KY' 'LA' 'MA' 'MD' 'ME' 'MI' 'MN' 'MO' 'MS' 'MT' 'NC'
 'ND' 'NE' 'NH' 'NJ' 'NM' 'NV' 'NY' 'OH' 'OK' 'OR' 'PA' 'RI' 'SC' 'SD'
 'TN' 'TX' 'UT' 'VA' 'VT' 'WI' 'WV' 'WY']...


area code:
 - Unique Values: ['408' '415' '510']...


phone number:
 - Unique Values: ['341-9764' '366-4467' '336-5406' ... '366-1084' '354-2762' '381
-2413']...


account length:
 - Unique Values: [ 36 104  78 110 127  50 141  96  59   1  86  55  52 121 136 126
48 117
  41 111 130 108 132 115 120  76  97 138 146  74  61 100 177 156 173 101
  51 103  71  58  99  98  85  60  47 109  83 102  93 200  87  16  92  82
 172 125  25  95 107  91 106  73  49 131 144  77  90 134 122  67  72  69
 197  70   8 137 119 149 148  88  19 182 181  13 118  32 167  68  89 112
 163 116  79 153  94  33  80 179  54   5 145 185 129 113  34 135  57  63
  12 140  62  84 157  43 124 192 158  75  66 151  37 105 159  23  81 154
 139  22  65 128  30 152 162 199  31   3 212 123  64  56  45 160 147 155
 150  40 184  24  11 169 180 217 114 170 142   2 183  53 224 171 133 161
 143 166 189  18  44  27 202 194 190  35 168  42  46 174 193  21   7 205
 186 164  38  39   9  17 176 165  29 204 175  28 178 225  20 196  15   6
  10  26 210 232 188 209 221 201 208 195 243   4 191 215]...


international plan:
 - Unique Values: ['no' 'yes']...


voice mail plan:
 - Unique Values: ['yes' 'no']...


number vmail messages:
 - Unique Values: [30  0 29 39 34 33 37 22 24 31 15 36 12 35 28 25 38 32 20 21 27
23 14 16
 19 49 42 41 45 26 40 18 47 17 11  8 51 13 43 46  9 48 44 10 50  4]...


total day minutes:
 - Unique Values: [146.3 278.4 225.1 ... 296.  210.1 223.8]...


total day calls:
 - Unique Values: [128 106  67  90 124 107  74  91  96  85 101 104  95  94 115  66
58 129
  93 105  97  71  55 109  88 102  86 114 120 127 118 117  65  78  60 121
  83  62  84 113  87  81  89  80 119 131 112 133 110 103 138 135 116  98
 100 139  63 123 134  56 140 143 145  73  79  99  82 141 111  68  69 125
  52  70 122  77 163 130 152 108  76 126  92  57  42 150 132  53  75  61
 137  64  54  49 136 147  47  59  72  44 151 157 144 158 165 142 146  51
 156  48 160  35  36  30 148  45 149   0  40]...


total day charge:
 - Unique Values: [24.87 47.33 38.27 ... 50.32 35.72 38.05]...
```

```
total eve minutes:
 - Unique Values: [162.5  81.  199.2 ... 151.  226.4 244.8]...


total eve calls:
 - Unique Values: [ 80 113 127  93 110 118 104  75 123  79  88 117  83  66  99  84
108 114
 103  94 109 136  98  81  65 147  82  96  69  73  76  95  89  91 134  53
 129 107 130 124  87 111 112  74 105  85 144 138 101  70 116 131 100  64
 120 106 115  92  59  68 102 121  90  63 128 132 133  58  71  72  97  12
 126 150 159  42  86 142  77 119 146  61 145  67 125 143 137  78  62  44
  50 122  60  52  36 168 140 148 135  55 155  54 157  57  45  56  48 141
 151 139 152  51 149 154  43 156  49  46 170 153   0 164  37]...


total eve charge:
 - Unique Values: [13.81  6.89 16.93 ... 12.84 19.24 20.81]...


total night minutes:
 - Unique Values: [129.3 163.2 175.5 ... 252.  139.1 223.8]...


total night calls:
 - Unique Values: [109 137 102  57 116  99  79  55 106 104 125 107  91  80  82  76
89 118
  74  93 124  95  94 114  96  90 100  87  71 128  63  98  81 117 113  42
  84 110 123 108  97 120  86  73  59 152 101  66  78 105  92 130  75  85
  53 115  83 138 129  67 103 112  65  62  77 111  88 121 126  72 122 132
 139 144  70  58  69 131 135 119  64 140  60 133  68 142 127  61 158  56
  38 153 146 134 141 136  54 155  52 164 148 145 151  49  44 143  46 157
 150 147 154 166  33 156 175  50  48 149  36  51]...


total night charge:
 - Unique Values: [ 5.82  7.34  7.9   9.2   8.28  9.12  6.77 10.26  6.24 10.37 11.
12 11.09
  6.04  5.06  4.58  6.16  6.48  9.18 11.79  7.15  6.41  7.66 10.15 10.35
  5.42  1.04 10.76  9.27 12.11 10.04  7.32 13.66  7.47 11.8  11.68  9.32
  8.64 11.85 11.54  8.52  9.64  4.59  9.45  8.66 11.1   7.    8.6   8.93
 10.25  8.88  9.1   4.86  5.11  7.14  7.54 11.82  7.99  7.8   8.54  4.23
 12.94  7.58  9.38  9.49  9.52  8.17  9.19  7.03 10.33  8.49  6.29  9.28
 11.44  6.23  9.17  8.12 10.5   6.67 11.31  6.02 10.65  9.62  7.18  8.22
  7.42  4.42  7.83  9.97 11.98  7.98  5.76  8.65  8.58  9.02  6.32 11.01
  8.47  6.44  7.71  8.2   8.59  9.71  6.7   9.98  6.93 11.48  7.21  8.57
  7.4  10.32  9.85  6.08  9.67  6.88  7.56  9.29  9.36  4.34  9.03  7.01
  9.7  16.55  9.    6.21 11.57  9.5  10.45 12.23 11.39 10.82 10.68  8.96
  7.61  9.39  9.25  9.63  6.62 10.85  9.9  12.5   7.29  7.73  8.15  5.27
  8.23 10.11  5.57 10.62  8.35  6.8  10.99  7.63 14.02 11.06  5.92 13.01
  5.49  5.36 10.6   7.69  7.84 12.95  9.47  7.92 13.02 14.81 12.61  8.55
 11.36  4.77 11.11 11.07  9.43  9.42 13.26  9.72  9.56 10.41  7.88  8.18
  5.75  9.51  3.48  5.74 12.46 10.06 11.02  8.09  4.24  7.51  9.61  5.51
  7.43 11.46  7.62 13.41 10.24 13.05  4.04 11.18  9.66 12.06  5.66  6.89
  4.03  8.46  7.87 12.56  8.74  3.59 11.19  9.34  6.87  7.22 10.61  9.23
  6.84  8.01 11.55  7.52 10.43  4.61  7.11  7.48  5.71  6.61  7.72  8.08
  4.73 14.45 11.24 11.45  7.06  8.1  10.42 15.56  9.95  3.2  11.03 10.46
 10.73  9.83  7.91  4.38  8.24 10.86  9.89  6.06  5.81  8.05  3.78  7.08
  4.25  6.27  6.91  4.68  6.86  7.13  6.34  9.87  5.05 11.43 13.87  6.07
 10.39  8.75 10.34  8.86  7.53 10.29 10.88  9.77  8.34  8.73 10.78  8.37
 13.25  6.98 10.16  8.51  5.8   8.9   8.68  5.15 10.64  6.73 13.42  7.46
  8.42  7.7   6.12 10.13 10.22  8.45 12.62 12.08  3.25 13.21 12.07 10.72
  8.78  9.14 10.63  6.37 13.47 14.46 10.92 10.08  2.86 12.16 13.63  8.76
```

```
 14.65 11.78  6.28 10.77 11.87 11.5  13.75 12.59  6.79  9.59  9.46  6.54
  7.07  7.86  9.4  11.74  3.61  6.35 10.74 11.51 14.06  6.71  9.58  6.97
  6.26  8.79 11.17  6.75  4.29  9.74 10.09  9.16 10.81 12.15 11.76  8.63
  8.11 11.21 12.36  8.27 11.83  9.31  4.93  9.82  6.5  12.13 12.42 13.59
  4.9  11.29  6.83 10.53  8.83 11.62  9.05  9.11 10.9  11.15 10.23 12.89
  7.3   7.78 10.96 12.24 12.66  9.48  9.21  9.22  6.76  7.39  8.67  9.73
  8.5   6.13  9.04 11.9  11.28 10.95  6.81 14.09  8.71  8.87 11.22  6.72
  5.03  5.41 10.52  6.45  7.79  6.56  9.57 11.7   9.96  5.    7.76 11.94
 11.34  9.79 11.16 10.18  5.17  5.21  5.2   4.54  7.64  9.3  11.59 10.8
  8.41  8.8   3.29  5.86 10.94 11.65  8.77 11.13  8.03 10.83  5.97 12.88
  5.4  10.58  7.19  9.08 11.41 12.1   8.72  8.4   7.33  9.09  6.6  16.42
  6.42 10.02 10.2   7.93  7.89  9.94  8.21  7.45 11.92  9.91  2.96 13.03
  3.44  8.14  8.02  5.14  8.36  8.13 12.14  8.25 11.25  8.99  8.95 14.32
  7.6  12.09  6.69 13.31  9.65  6.82  6.11 10.01 10.71  9.13  6.94 10.44
 10.31  8.04  9.15  9.37 10.87 10.66  7.75  6.96  8.    5.72  8.38  8.44
  1.97  7.1  10.27  8.61  5.73  8.94  9.54 10.17 11.97  8.48  8.43 10.36
  4.02  7.5  12.22 13.7   7.31  7.2  12.38 10.49  8.97  9.41  9.35 10.19
 10.05 11.67 13.13  5.24  5.85 10.93 10.4   7.57 11.33 12.29  6.52  7.65
  7.24  7.55 16.99 11.77  8.98 12.9   6.68  6.59  7.23 11.27  7.59  5.01
  8.7   5.56  7.44  9.86  8.69 12.69  7.96  8.39  3.05  8.29  8.19  7.77
  5.23 10.38  7.95 11.49 12.45  6.49  8.53  6.03  5.32  6.38 12.7   6.78
 14.5   5.65 12.41 11.86  6.3   8.3   4.46 10.51 10.28  6.64  9.99  6.53
 13.18 15.76  9.6  12.75  9.76  5.88 13.14 10.57  7.09  4.51 12.01 10.55
  8.07  7.25  5.99 10.56 12.58 10.07  2.25  5.78  5.84  8.62  9.06  8.31
  9.26 10.21  8.91  5.31 10.47 11.04 14.13  7.35 13.46  4.74 12.83 11.91
 10.59  5.08  4.95 13.48  5.68  9.33 11.4   3.6   5.77 13.2   7.27  5.47
 13.23  8.82  4.67  6.63  6.39  6.47  9.92 15.85  7.67  8.32  5.58  6.85
  3.26 13.84  5.91 13.91 11.32 10.03  5.94 14.03  7.05 12.03 14.97 11.88
 12.93 12.81 11.93  5.5  12.76 11.42 10.48 11.58 12.87  5.83 12.04  6.74
  6.66  6.55  6.9  13.27 11.08  5.44 12.96  6.95  9.55 10.7  14.1   9.93
 11.35  6.22  5.54  5.39  5.28  7.49  4.94  4.97 12.63 11.66 13.6   6.01
  2.85 12.33  9.44 10.79 12.72 12.49  6.58 12.34  8.16 11.96 13.33  8.06
  9.07  5.55  9.8   3.47  8.89  8.81 13.78 11.64  9.53 15.74 11.53 10.12
 10.3  14.25 13.93  5.95 12.52  9.84  5.7  11.3  11.56  6.92  6.65  5.96
 11.    9.24 10.1  14.78 12.8   3.93  5.3   6.    5.9  10.   11.52  5.79
  2.45 14.82 12.32 12.12 11.63  5.13  5.37  7.97  5.12 12.21 11.72 12.19
 12.26 11.37  3.57  5.35  6.15  2.76 12.48 10.54  6.46 11.84  3.94  3.97
 13.3  10.89 17.19 12.35 14.67  7.37 12.73  3.99  6.99  5.63  3.71 10.98
  7.26  7.41  6.09 11.26 14.08  8.33  9.78 13.82 12.    4.83 15.06 12.53
 13.29 11.89  3.67 13.74  4.41 11.14  5.98  4.7   2.59 11.47 13.69 10.14
 13.95 10.84  4.75  3.7  13.22  4.47 12.18 14.43  8.84  9.69 12.85 12.64
 11.75  6.51 12.65  7.36 12.71 13.5  13.98  7.38 11.05 10.69  5.25  4.64
 12.39 11.73 12.86 12.67  8.26  4.72  6.2   7.94 13.17  4.27  4.84 14.56
  7.81 11.61 13.53 11.23 15.86 12.84 11.38  3.86  6.4   2.89  9.88 12.3
  8.85  4.71 14.04  7.74 12.02 12.27  2.13  7.85  4.55  4.28  6.19  5.1
 13.12  2.4  12.4   6.31 15.43  3.32  2.55 10.75 12.91 13.37 13.49 13.8
 11.71  7.82 14.18  6.18 11.69  4.12 13.9  11.81 13.45  9.75  9.81 11.95
 13.16  6.05 12.28  5.29  4.98 14.3   7.16 12.6   7.28  7.68  5.22  5.45
 12.37  4.09  4.3  15.49  5.33  7.17 15.97 12.77  2.43 13.97 10.67  6.43
 12.17 17.77  4.45  3.51  5.02 15.01 11.2  14.54 13.   13.1   3.82 12.68
 12.2  15.71  5.52 14.   16.39  4.1   6.14  4.96  4.44 10.97 14.69  3.41
  5.89 13.58  4.92  7.12 12.74  9.68  2.03  3.18  5.38]...


total intl minutes:
 - Unique Values: [14.5  9.8 14.6 11.1  9.3  8.7 10.7  8.5 10.2  5.3 12.5  8.1 11.
8  7.5
 10.5 11.9 12.2 14.7  7.9  8.6  7.2  9.5  4.1  5.8  6.9 12.1  8.2  6.6
 11.  10.   8.9 11.3 11.2 11.7 13.1 10.9 12.3  4.2  8.3  7.8 14.9 12.4
 11.4 14.   6.8 15.8 13.   8.8 10.1 12.  11.6  8.4 11.5  6.  13.2  7.1
  9.4  8.  14.3  6.7 10.4  9.   3.8 13.7  2.9  4.8  6.3 12.6 14.2  4.5
 13.3 10.6 14.1 17.2  7.7 10.3  7.4  9.1 16.2 15.5  9.6 17.   9.2 14.4
 12.9  9.7  4.7 16.1  6.4 10.8  6.2  9.9 12.7 12.8  4.9 15.9 15.2  4.3
  6.1 13.8  0.   7.  17.8  5.7 13.4 15.   7.3 14.8 18.   3.5  3.6  5.5
```

```
  7.6 16.5  5.6 16.6  5.9 17.3 13.9  5.  13.5 16.4  5.1  6.5 16.9  5.4
  2.5  3.3 17.6 13.6 15.1 15.7  2.2  4.4 18.9 18.3 17.5 15.4 18.2  2.1
  2.7  3.7 15.3 16.7  3.1  4.6 16.   4.  15.6 20.  17.9 17.1  5.2  2.
  3.9 18.4 16.3  2.4  3.4  1.1  2.6  1.3]...


total intl calls:
 - Unique Values: [ 6  5  2  8  3  7  4  1  9 13 12 11 10 19  0 14 15 20 16 18 1
7]...


total intl charge:
 - Unique Values: [3.92 2.65 3.94 3.   2.51 2.35 2.89 2.3  2.75 1.43 3.38 2.19 3.1
9 2.03
 2.84 3.21 3.29 3.97 2.13 2.32 1.94 2.57 1.11 1.57 1.86 3.27 2.21 1.78
 2.97 2.7  2.4  3.05 3.02 3.16 3.54 2.94 3.32 1.13 2.24 2.11 4.02 3.35
 3.08 3.78 1.84 4.27 3.51 2.38 2.73 3.24 3.13 2.27 3.11 1.62 3.56 1.92
 2.54 2.16 3.86 1.81 2.81 2.43 1.03 3.7  0.78 1.3  1.7  3.4  3.83 1.22
 3.59 2.86 3.81 4.64 2.08 2.78 2.   2.46 4.37 4.19 2.59 4.59 2.48 3.89
 3.48 2.62 1.27 4.35 1.73 2.92 1.67 2.67 3.43 3.46 1.32 4.29 4.1  1.16
 1.65 3.73 0.   1.89 4.81 1.54 3.62 4.05 1.97 4.   4.86 0.95 0.97 1.49
 2.05 4.46 1.51 4.48 1.59 4.67 3.75 1.35 3.65 4.43 1.38 1.76 4.56 1.46
 0.68 0.89 4.75 3.67 4.08 4.24 0.59 1.19 5.1  4.94 4.73 4.16 4.91 0.57
 0.73 1.   4.13 4.51 0.84 1.24 4.32 1.08 4.21 5.4  4.83 4.62 1.4  0.54
 1.05 4.97 4.4  0.65 0.92 0.3  0.7  0.35]...


Total Revenu:
 - Unique Values: [48.42 64.21 67.04 ... 83.99 67.91 72.25]...


customer service calls:
 - Unique Values: [0 1 3 2 5 4 7 6 9 8]...


churn:
 - Unique Values: [False  True]...
```

Outlier Detection

- Removing outliers based on the Interquartile Range (IQR) method.

In [135…

```
numerical_variables = ['account length', 'number vmail messages', 'total day minute
    'total day charge', 'total eve minutes', 'total eve calls', 'total eve charge',
    'total night minutes', 'total night calls', 'total night charge',
    'total intl minutes', 'total intl calls', 'total intl charge',
    'Total Revenu', 'customer service calls']

# Calculating IQR for each numerical variable
iqr_bounds = {}

for col in numerical_variables:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR
    iqr_bounds[col] = {'lower': lower, 'upper': upper}


# Removing outliers based on IQR
```

```python
def remove_outliers_iqr(df, columns, measure=1.5):
    for col in columns:
        Q1 = df[col].quantile(0.25)
        Q3 = df[col].quantile(0.75)
        IQR = Q3 - Q1
        lower = Q1 - measure * IQR
        upper = Q3 + measure * IQR
        df = df[(df[col] >= lower) & (df[col] <= upper)]
    return df

# Applying the function to remove outliers
df_clean = remove_outliers_iqr(df, numerical_variables)
df_clean.shape
```

Out[135]:

```
(2783, 22)
```

Exploratory Data Analysis (EDA)

- Churn Rate
- Distributions of Numeric & Categorical features
- Factors that contribute most to customer churn.
- Univariate Analysis
- Bivariate Analysis

Churn Rate

In [136…

```python
# proportion of churned vs. non-churned customers

df['churn'].value_counts(normalize=True)
```

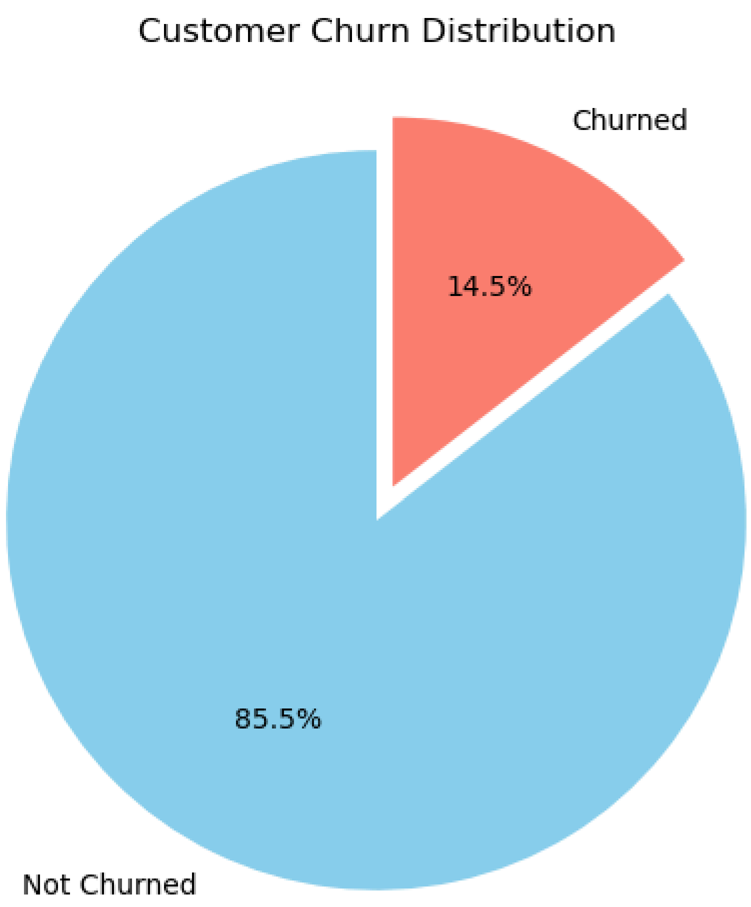
Out[136]:

```
churn
False    0.855086
True     0.144914
Name: proportion, dtype: float64
```

- From the class distribution, we observe that 85.5% of the customers did not churn while only 14.5% churned. This shows a significant class imbalance.
- Using accuracy as the main evaluation metric would not be enough, instead, I will rely on precision, recall, confusion matrix and F1-score to evaluate the model.

In [137…

```python
# Count churn
churn_counts = df['churn'].value_counts()
labels = ['Not Churned', 'Churned']
colors = ['skyblue', 'salmon']

# pie chart
plt.figure(figsize=(6, 6))
plt.pie(churn_counts, labels=labels, colors=colors, autopct='%1.1f%%', startangle=9
plt.title('Customer Churn Distribution')
plt.show()
```

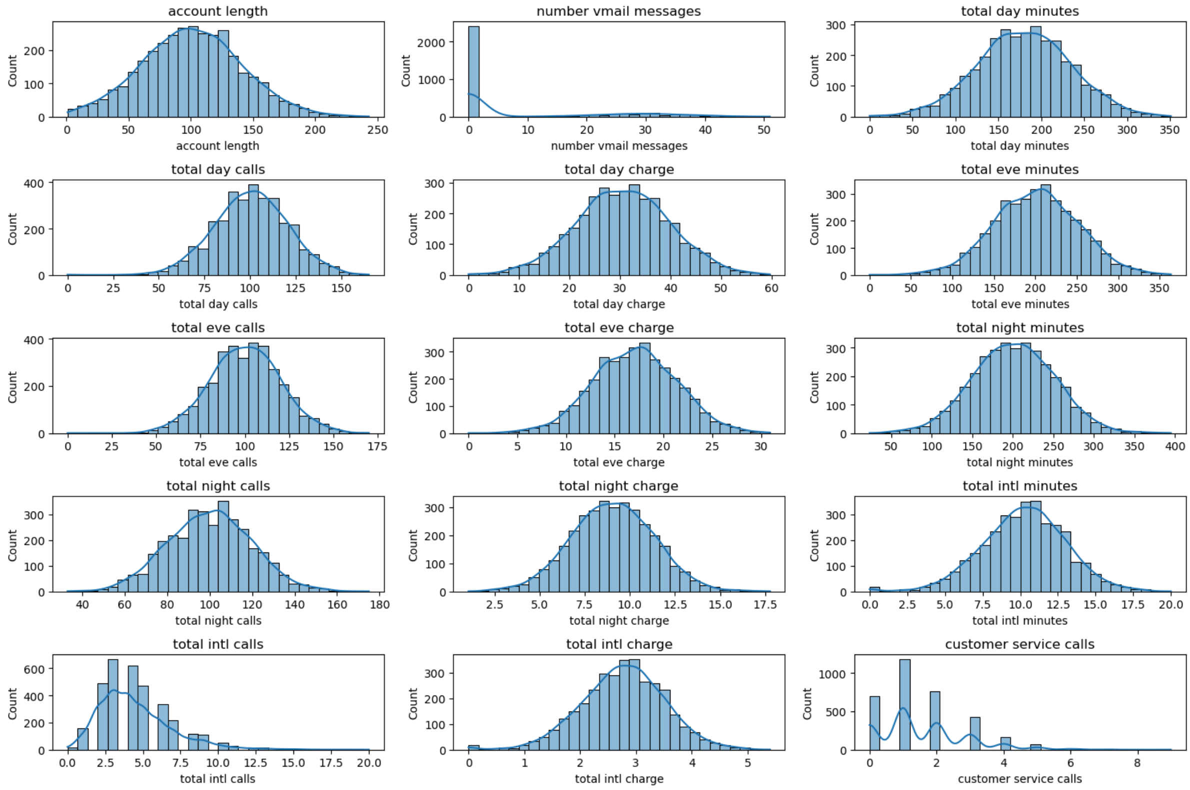


Distributions of features

a) Numeric Features

In [138…

```
# Numerical features
numerical_features = ['account length', 'number vmail messages', 'total day minutes
                      'total day calls', 'total day charge', 'total eve minutes',
                      'total eve calls', 'total eve charge', 'total night minutes',
                      'total night calls', 'total night charge', 'total intl minute
                      'total intl calls', 'total intl charge', 'customer service ca

plt.figure(figsize=(15, 10))
for i, feature in enumerate(numerical_features, 1):
    plt.subplot(5, 3, i)
    sns.histplot(df[feature], kde=True, bins=30)
    plt.title(feature)
plt.tight_layout()
plt.show()
```

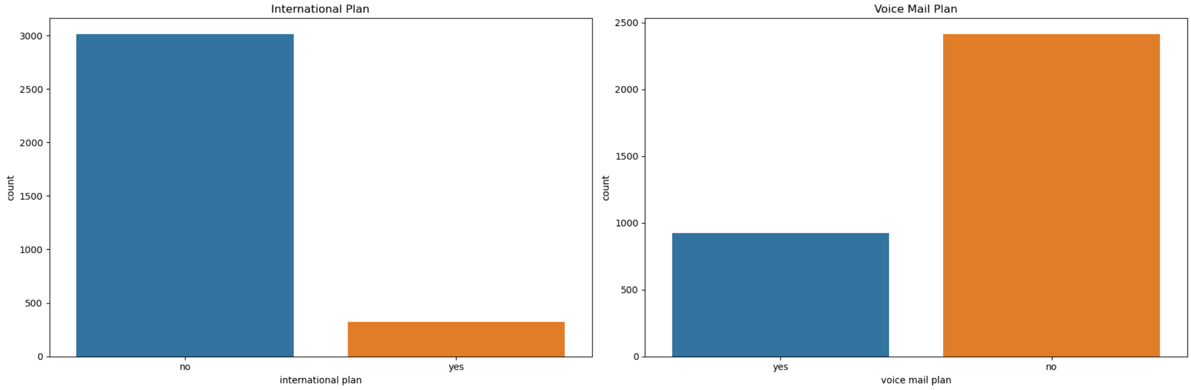


- Observation:
  - Most customers have had accounts for around 100 days, with fewer people being very new or very old customers.
  - Majority of the customers talk moderate amount of minutes during the day, with fewer people talking too little or too much.
  - Customers make between 80 to 120 calls per day. This reflect engagement and a sudden drop could signal dissatisfaction.
  - international customers are a niche but high-value segment because people spend under 10 minutes making these calls
  - Customer service calls is a Key churn indicator because the more they call, the more likely they are thinking of leaving.

b) Categorical Variables

In [139...

```
# Set up the figure and axes for the subplots
fig, axes = plt.subplots(1, 2, figsize=(18, 6))

# Plot for 'international plan'
sns.countplot(x='international plan', data=df, ax=axes[0])
axes[0].set_title('International Plan')

# Plot for 'voice mail plan'
sns.countplot(x='voice mail plan', data=df, ax=axes[1])
axes[1].set_title('Voice Mail Plan')

# Adjust layout to prevent overlap
plt.tight_layout()
plt.show()
```



- Low voicemail or international use signals potential areas to grow or customers at risk of switching to more tailored providers.

Factors that may Contribute to Churning

- To better understand what drives customers to leave, I grouped the variables into thematic areas and explored their impact on churn. These factors include:
  - Tenure (Account Length) - to understand if longer-term customers are more or less likely to churn.
  - Geographical Factors(State) - to detect regional or location-based trends in churn.
  - Service Plan Subscriptions (International Plan & Voice Mail Plan) — Does having specific plans influences churn likelihood
  - Usage Behavior (Call & Minute Usage during all period of time) — Does high or low usage patterns are linked to churn.
  - Financial Impact (Charges): to evaluate if higher billing is associated with customer dissatisfaction and churn.
  - Customer Service Interaction (Number of Customer Service Calls) — Does frequent service contact signals dissatisfaction.

Geographic Churn Analysis

- To support the company's location based retention strategy, I identified which U.S. states have the highest number of churned customers.
- I visualized the churn intensity where darker red states represent higher churn. This helps the company to:
  - Prioritize outreach efforts and marketing campaigns in high-churn states.
  - Design targeted interventions by region.
  - Recognize mid-level churn areas, which may be easier to retain with less effort

In [140…

```
# Mapping state codes to full names for better readability
us_state_abbrev = {
    'AL': 'Alabama', 'AK': 'Alaska', 'AZ': 'Arizona', 'AR': 'Arkansas',
    'CA': 'California', 'CO': 'Colorado', 'CT': 'Connecticut', 'DE': 'Delaware',
    'FL': 'Florida', 'GA': 'Georgia', 'HI': 'Hawaii', 'ID': 'Idaho',
    'IL': 'Illinois', 'IN': 'Indiana', 'IA': 'Iowa', 'KS': 'Kansas',
    'KY': 'Kentucky', 'LA': 'Louisiana', 'ME': 'Maine', 'MD': 'Maryland',
    'MA': 'Massachusetts', 'MI': 'Michigan', 'MN': 'Minnesota', 'MS': 'Mississippi'
    'MO': 'Missouri', 'MT': 'Montana', 'NE': 'Nebraska', 'NV': 'Nevada',
    'NH': 'New Hampshire', 'NJ': 'New Jersey', 'NM': 'New Mexico',
    'NY': 'New York', 'NC': 'North Carolina', 'ND': 'North Dakota',
```

```
    'OH': 'Ohio', 'OK': 'Oklahoma', 'OR': 'Oregon', 'PA': 'Pennsylvania',
    'RI': 'Rhode Island', 'SC': 'South Carolina', 'SD': 'South Dakota',
    'TN': 'Tennessee', 'TX': 'Texas', 'UT': 'Utah', 'VT': 'Vermont',
    'VA': 'Virginia', 'WA': 'Washington', 'WV': 'West Virginia',
    'WI': 'Wisconsin', 'WY': 'Wyoming'
}

# Standardization - Removing any spaces and converting to uppercase
df['state'] = df['state'].str.strip().str.upper()

# Replacing State codes with full names
df['state_full'] = df['state'].map(us_state_abbrev)

# Filtering Churned Customers & Counting by State
churn_by_state = df[df['churn']].groupby('state').size().reset_index(name='churn_co

# Maping Churn Distribution by State
fig = px.choropleth(
    churn_by_state,
    locations='state',                  # state code
    locationmode='USA-states',          # I'm using state codes
    color='churn_count',                # Column to use for color
    scope='usa',                        # Only show the US
    color_continuous_scale='Reds',      # Red scale = more churn
    labels={'churn_count': 'Churns'},
    title='Churn Count by State'
)

fig.show()
```

- Washington, New jersey & Texas have the highest customers who have churned
- Majority of other states reflect a moderate level of churn and shouldn't be overlooked when designing region-specific retention strategies

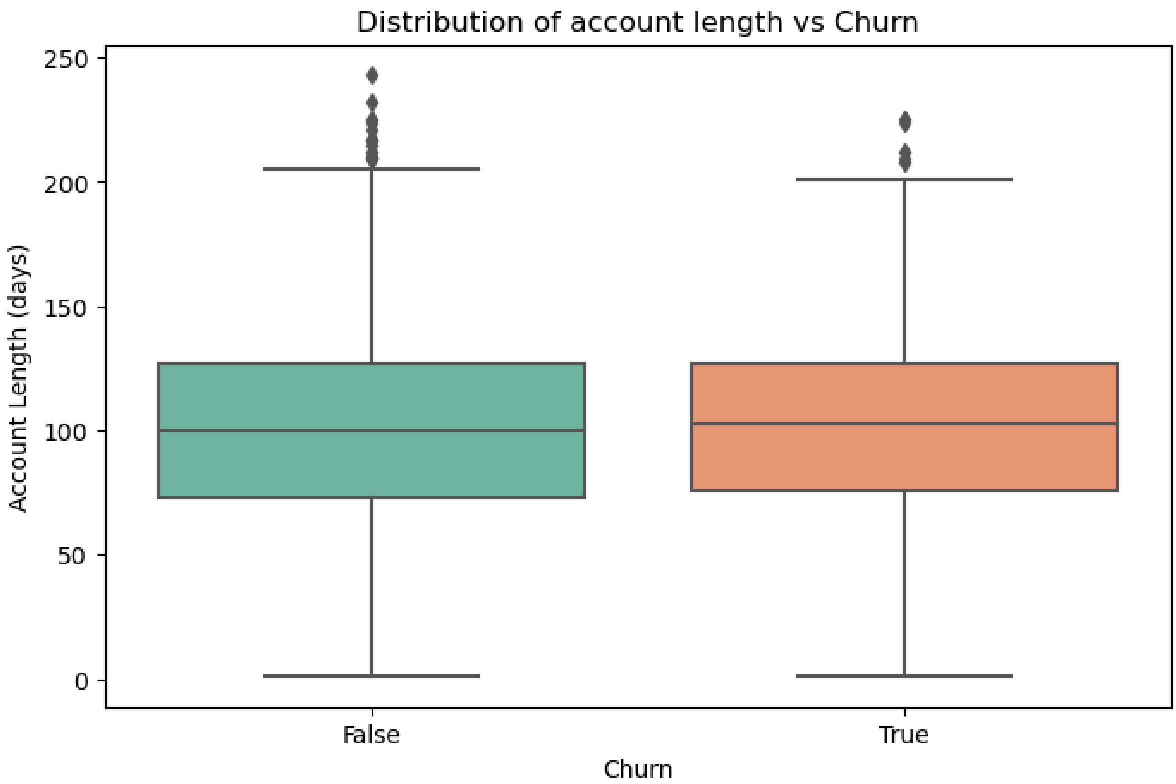
Tenure Analysis

- To understand how customer longevity influences churning, I analyzed how long each customer has been with the company (in days).

- *Are newer customers more likely to churn or do long-term customers tend to stay?*

In [141…

```
# Visualizing Box Plot Account Length vs Churn

plt.figure(figsize=(8, 5))
sns.boxplot(x='churn', y='account length', data=df, palette='Set2')
plt.title('Distribution of account length vs Churn')
plt.xlabel('Churn')
plt.ylabel('Account Length (days)')
plt.show()
```

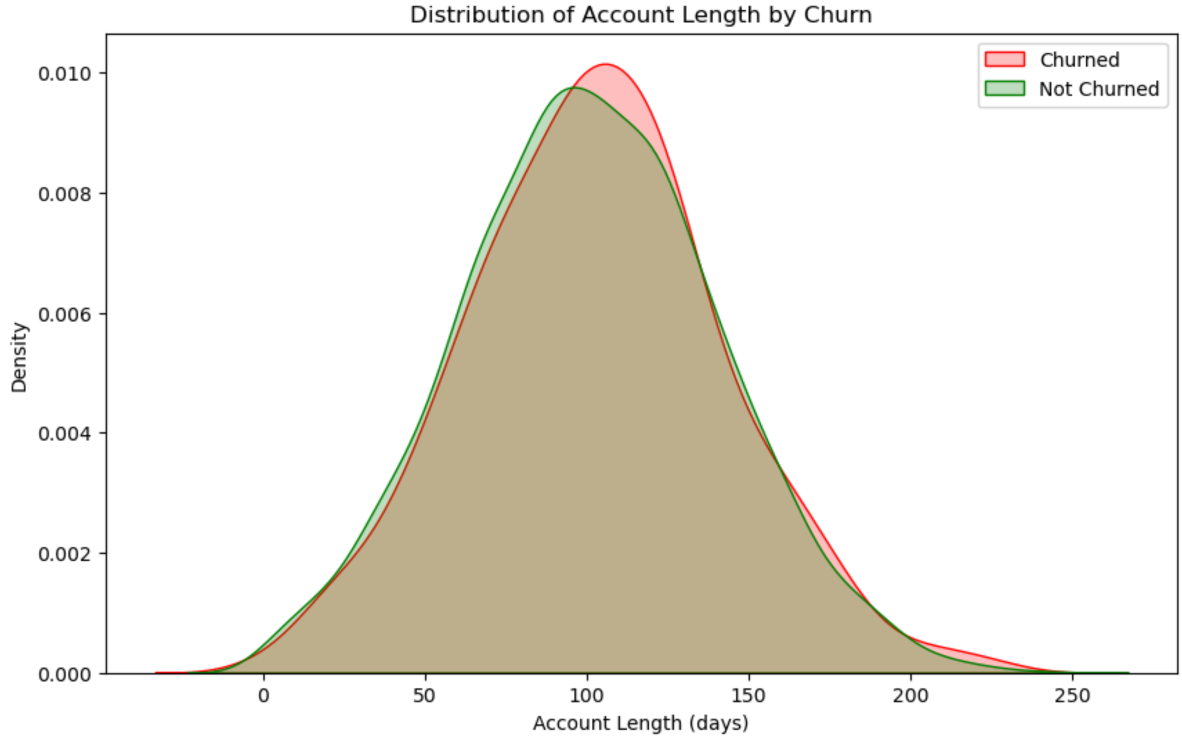


- Observation:
  - The spread and median of tenure for churned and non-churned customers look very similar.
  - Both groups have a median around 100 days.
  - Churned customers do not appear to have dramatically shorter or longer tenures compared to loyal ones.
- Conclusion:
  - Account length alone is not a strong predictor of churn, since the difference between groups is small.

In [142…

```
# Visualizing Distribution of Account Length by Churn using Kernel Density Estimati
plt.figure(figsize=(10, 6))
sns.kdeplot(df[df['churn'] == True]['account length'], label='Churned', shade=True,
sns.kdeplot(df[df['churn'] == False]['account length'], label='Not Churned', shade=
plt.title('Distribution of Account Length by Churn')
plt.xlabel('Account Length (days)')
plt.legend()
plt.show()
```

```
C:\Users\A808865\AppData\Local\Temp\ipykernel_23076\407809180.py:3: FutureWarning:


`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.


C:\Users\A808865\AppData\Local\Temp\ipykernel_23076\407809180.py:4: FutureWarning:


`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.
```



- Observation:
  - Both groups have a very similar shape, peaking around 90–100 days.
  - The red curve (churned) is slightly shifted right, meaning churned customers may have stayed slightly longer on average.
  - The difference is very minimal (~2 days) as there's no major shift or skew.

To confirm that the churned customers have stayed slightly longer on average, I grouped customers by Tenure to see their Churn Rate

In [143…

```
# Defining tenure groups
df['tenure_group'] = pd.cut(df['account length'], bins=[0, 50, 100, 150, 200, 250],

# Calculating churn rate per group and format as percentage
churn_by_group = df.groupby('tenure_group')['churn'].mean()* 100
churn_by_group = churn_by_group.round(2)  # round to 2 decimal places

print(churn_by_group)
```

```
tenure_group
Very New Customers     12.90
New Customers          14.36
Mid Customers          14.85
Long Customers         14.55
Very Long Customers    23.08
Name: churn, dtype: float64
```

- Observation:
  - Customers in the “Very Long” group (the most loyal) have the highest churn rate while those in the other groups have churn rates between 12–15%, relatively flat.
- Conclusion:
  - Customers with the longest tenure show the highest churn rate, highlighting that even loyal users are at risk and should be actively re-engaged.

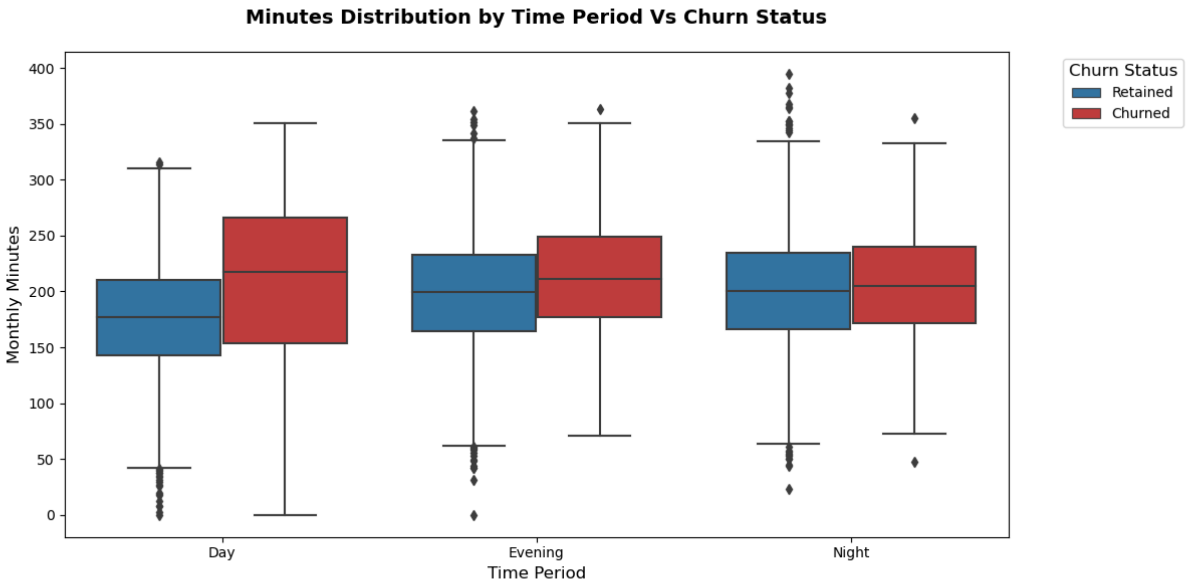
Usage Behavior Analysis

- To understand whether customer activity levels influence churn, I analyzed usage patterns across different time periods and services to see if low or high engagement is linked to a higher likelihood of churn.

In [144…

```python
usage = df.melt(
    id_vars=['churn'],
    value_vars=['total day minutes', 'total eve minutes', 'total night minutes'],
    var_name='time_period',
    value_name='minutes'
)

# 2. Create visualization with matched colors
plt.figure(figsize=(12, 6))
ax = sns.boxplot(
    data=usage,
    x='time_period',
    y='minutes',
    hue='churn',
    palette={False: '#1f77b4', True: '#d62728'},  # Retained: blue, Churned: red
    order=['total day minutes', 'total eve minutes', 'total night minutes']
)

# Correct legend with matched colors
handles, _ = ax.get_legend_handles_labels()
plt.legend(
    handles, ['Retained', 'Churned'],
    title='Churn Status',
    title_fontsize=12,
    bbox_to_anchor=(1.05, 1),
    loc='upper left'
)

# Formatting
plt.title('Minutes Distribution by Time Period Vs Churn Status', fontsize=14, pad=2
plt.xlabel('Time Period', fontsize=12)
plt.ylabel('Monthly Minutes', fontsize=12)
plt.xticks([0, 1, 2], ['Day', 'Evening', 'Night'])
plt.tight_layout()
```

```
plt.show()
```



In [145…

```
usage_columns = [
    'total day calls', 'total day minutes',
    'total eve calls', 'total eve minutes',
    'total night calls', 'total night minutes',
    'total intl calls', 'total intl minutes']

# Grouping by churn and calculating mean usage
usage_summary = df.groupby('churn')[usage_columns].mean().round(2)
usage_summary
```

Out[145]:

churn	total day calls	total day minutes	total eve calls	total eve minutes	total night calls	total night minutes	total intl calls	total intl minutes
False	100.28	175.18	100.04	199.04	100.06	200.13	4.53	10.16
True	101.34	206.91	100.56	212.41	100.40	205.23	4.16	10.70

- Customers who churn tend to use more minutes across all time periods, especially during the day. This may suggest other factors i.e overuse, dissatisfaction with costs, or unmet expectations.
- The number of calls (day, evening, night, international) remains nearly the same between both groups.

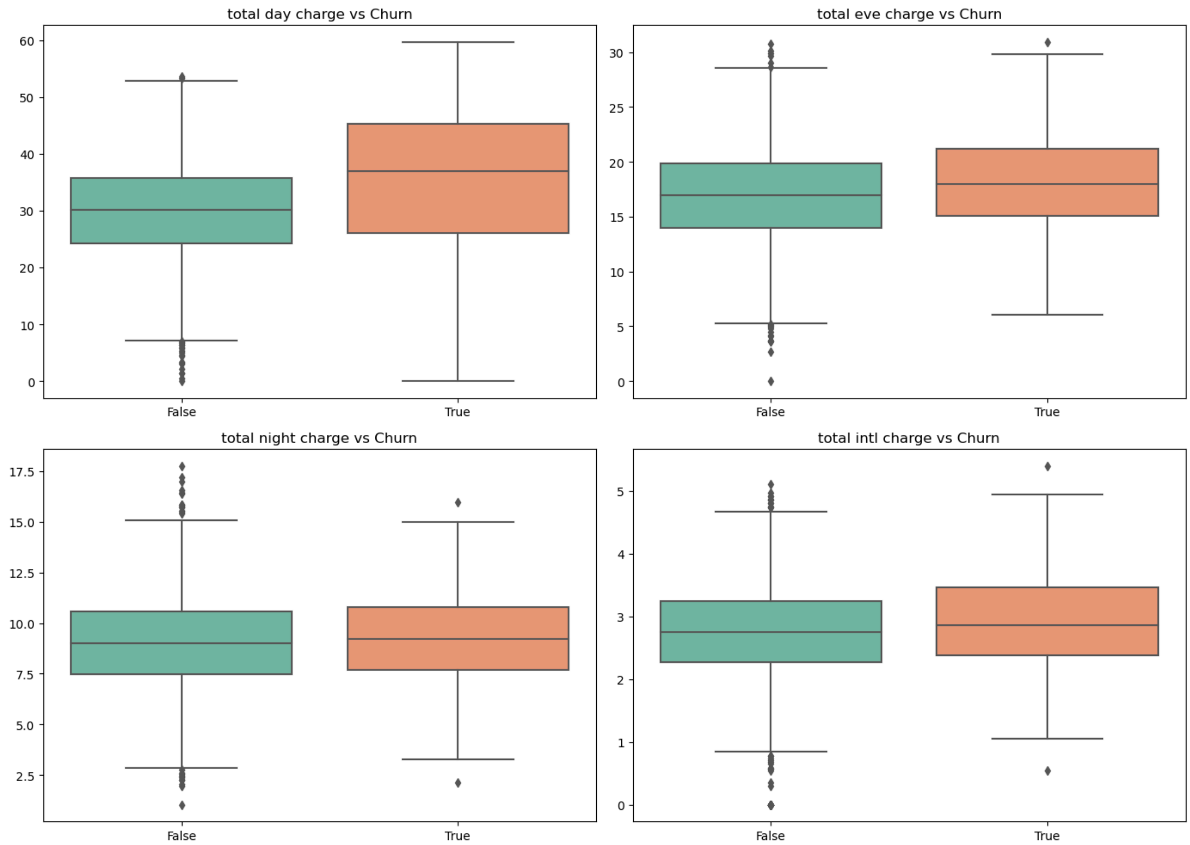
Financial Impact.

- To assess whether high charges contribute to customer churn, I analyzed total charges across different time periods (day, evening, night, and international).

In [146…

```
charge_columns = [
    'total day charge',
    'total eve charge',
    'total night charge',
    'total intl charge']
```

```python
plt.figure(figsize=(14, 10))

for i, col in enumerate(charge_columns, 1):
    plt.subplot(2, 2, i)
    sns.boxplot(x='churn', y=col, data=df, palette='Set2')
    plt.title(f'{col} vs Churn')
    plt.xlabel('')
    plt.ylabel('')

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



In [147…

```python
# Mean Comparison
df.groupby('churn')[charge_columns].mean()
```

Out[147]:

churn	total day charge	total eve charge	total night charge	total intl charge
False	29.780421	16.918909	9.006074	2.743404
True	35.175921	18.054969	9.235528	2.889545

- On average, churned customers paid higher charges across all time periods espicially daytime:
  - Day Charge: 35.18 (churned) vs. 29.78 (not churned)
  - Evening Charge: 18.05 vs. 16.92
  - Night Charge: 9.24 vs. 9.01
  - International Charge: 2.89 vs. 2.74

Customer Support Interaction

To assess whether customer dissatisfaction drives churn, I examined the number of customer service calls made, as repeated contact with support may indicate unresolved issues or frustration.

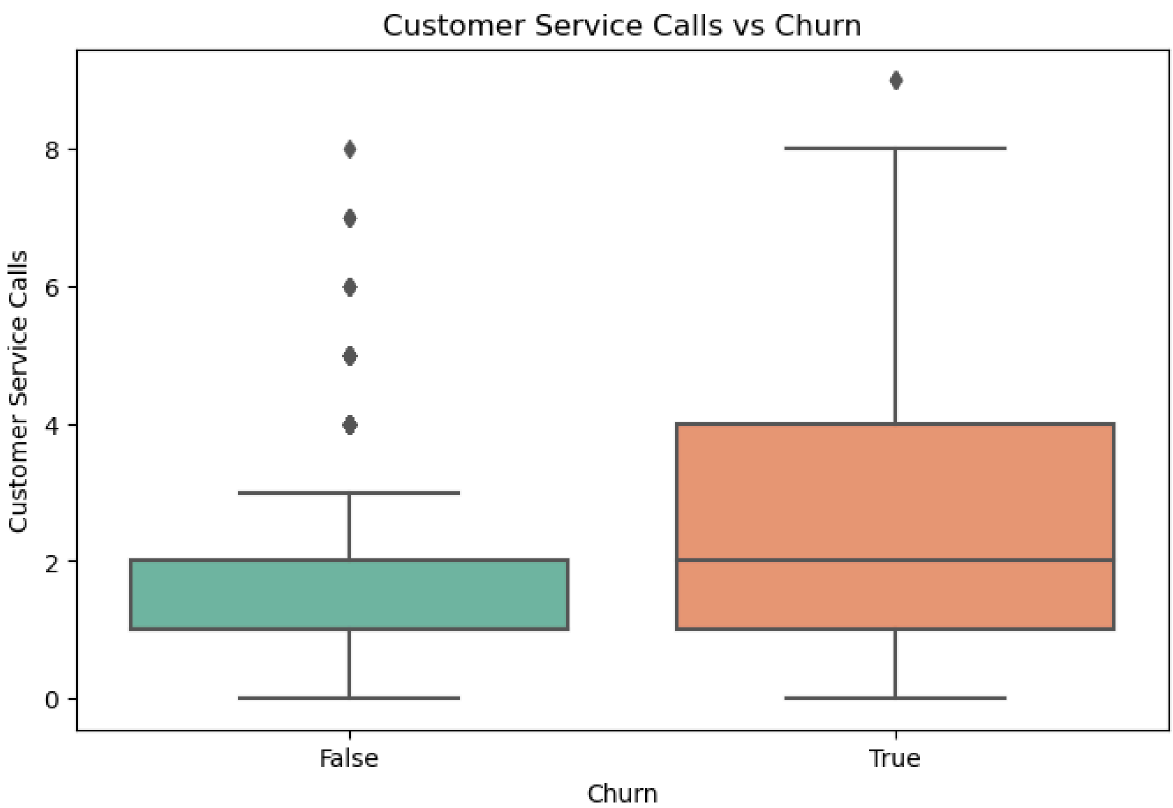
In [148…

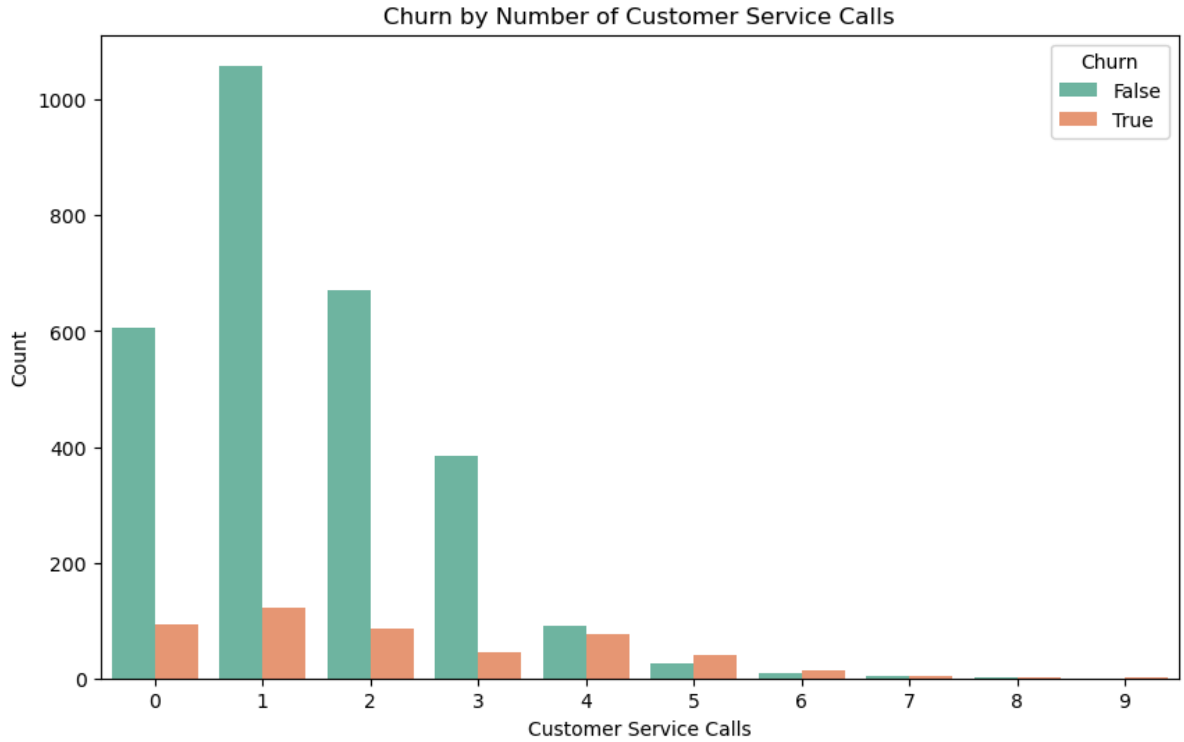
```
# Average customer service calls for churned vs non-churned
mean_calls = df.groupby('churn')['customer service calls'].mean()
print("Average Customer Service Calls by Churn:")
print(mean_calls)
```

```
Average Customer Service Calls by Churn:
churn
False    1.449825
True     2.229814
Name: customer service calls, dtype: float64
```

In [149…

```
# Visualizing Customer Service Calls vs Churn
# Boxplot
plt.figure(figsize=(8,5))
sns.boxplot(x='churn', y='customer service calls', data=df, palette='Set2')
plt.title('Customer Service Calls vs Churn')
plt.xlabel('Churn')
plt.ylabel('Customer Service Calls')
plt.show()

# Countplot
plt.figure(figsize=(10,6))
sns.countplot(x='customer service calls', hue='churn', data=df, palette='Set2')
plt.title('Churn by Number of Customer Service Calls')
plt.xlabel('Customer Service Calls')
plt.ylabel('Count')
plt.legend(title='Churn')
plt.show()
```





- Churned customers made significantly more customer service calls on average (2.23) compared to those who stayed (1.45).
- The churn rate increases sharply with the number of service calls:
  `- Churn decreases with lower support calls, especially for long-tenure customers`
  `- Once customers hit 5 or more service calls, churn shoots up (above 50%+) indicating rising dissatisfaction`

Service Plan

To understand the impact of service subscriptions on churn, I examined whether having an international plan or a voice mail plan made customers more or less likely to churn.

In [150…

```
# Crosstab for International Plan
intl_plan_churn = pd.crosstab(df['international plan'], df['churn'], normalize='inc
print("Churn Rate by International Plan:")
print(intl_plan_churn)

# Crosstab for Voice Mail Plan
vmail_plan_churn = pd.crosstab(df['voice mail plan'], df['churn'], normalize='index
print("\n Churn Rate by Voice Mail Plan:")
vmail_plan_churn
```

```
Churn Rate by International Plan:
churn                  False     True
international plan
no                  0.885050  0.114950
yes                 0.575851  0.424149

 Churn Rate by Voice Mail Plan:
```

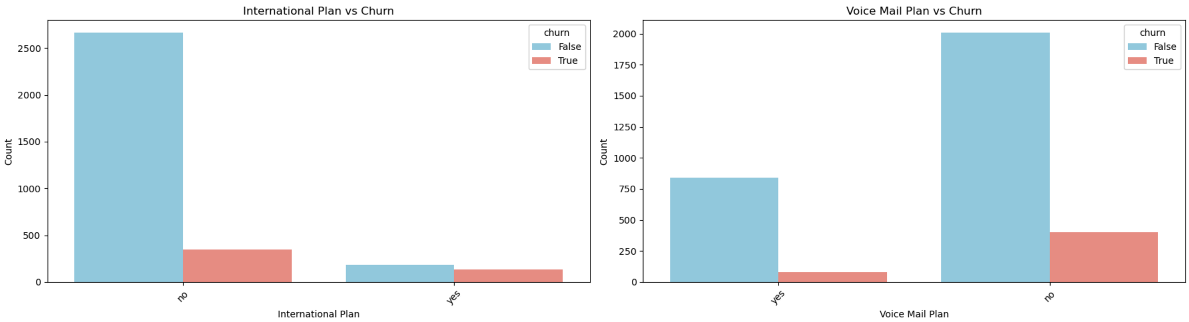
Out[150]:

churn	False	True
voice mail plan		
no	0.832849	0.167151
yes	0.913232	0.086768

- Customers with an international plan churn much more than those without it.
- Customers with a voice mail plan churn less than those without it.

In [151…

```
# Custom plot colors
custom_palette = {True: 'salmon', False: 'skyblue'}

categorical_vars = ['international plan', 'voice mail plan']

fig, axes = plt.subplots(1, 2, figsize=(18, 5))

for i, col in enumerate(categorical_vars):

    sns.countplot(x=col, hue='churn', data=df, ax=axes[i], palette=custom_palette)

    axes[i].set_title(f'{col.title()} vs Churn')

    axes[i].set_ylabel('Count')

    axes[i].set_xlabel(col.title())

    axes[i].tick_params(axis='x', rotation=45)

plt.tight_layout()

plt.show()
```



- High customer service calls + international plan + short tenure is the strongest combined risk factor for churn.
- These customers face issues early, Pay more for international usage and have poor support experiences
- The companz Should Prioritize retention by setting up early-warning systems to flag these combinations and enhance onboarding and issue resolution

Financial Impact

- I compared the total revenue from churned vs. non-churned customers, and shows what percentage of overall revenue each group contributes.

In [152…

```
# charge columns
charges = ['total day charge', 'total eve charge', 'total night charge', 'total int

# Total revenue per customer
df['total_revenue'] = df[charges].sum(axis=1)

# Group by churn status and calculating revenue
revenue_by_churn = df.groupby('churn')['total_revenue'].sum().reset_index()

# Percentage contribution
total_revenue_all = df['total_revenue'].sum()
revenue_by_churn['% of Total Revenue'] = round(
    revenue_by_churn['total_revenue'] / total_revenue_all * 100, 2
)


revenue_by_churn.columns = ['Churned', 'Total Revenue', '% of Total Revenue']
revenue_by_churn
```

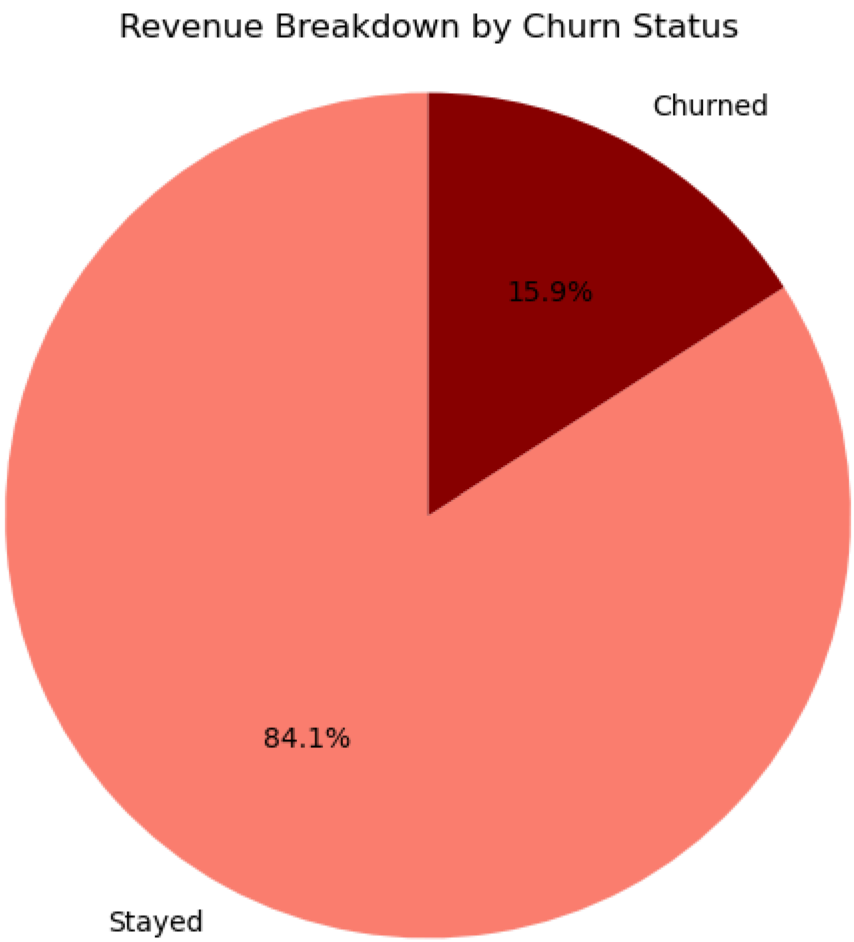
Out[152]:

	Churned	Total Revenue	% of Total Revenue
0	False	166579.10	84.07
1	True	31566.93	15.93

- 15% of the revenue is lost due to churned customers

In [153…

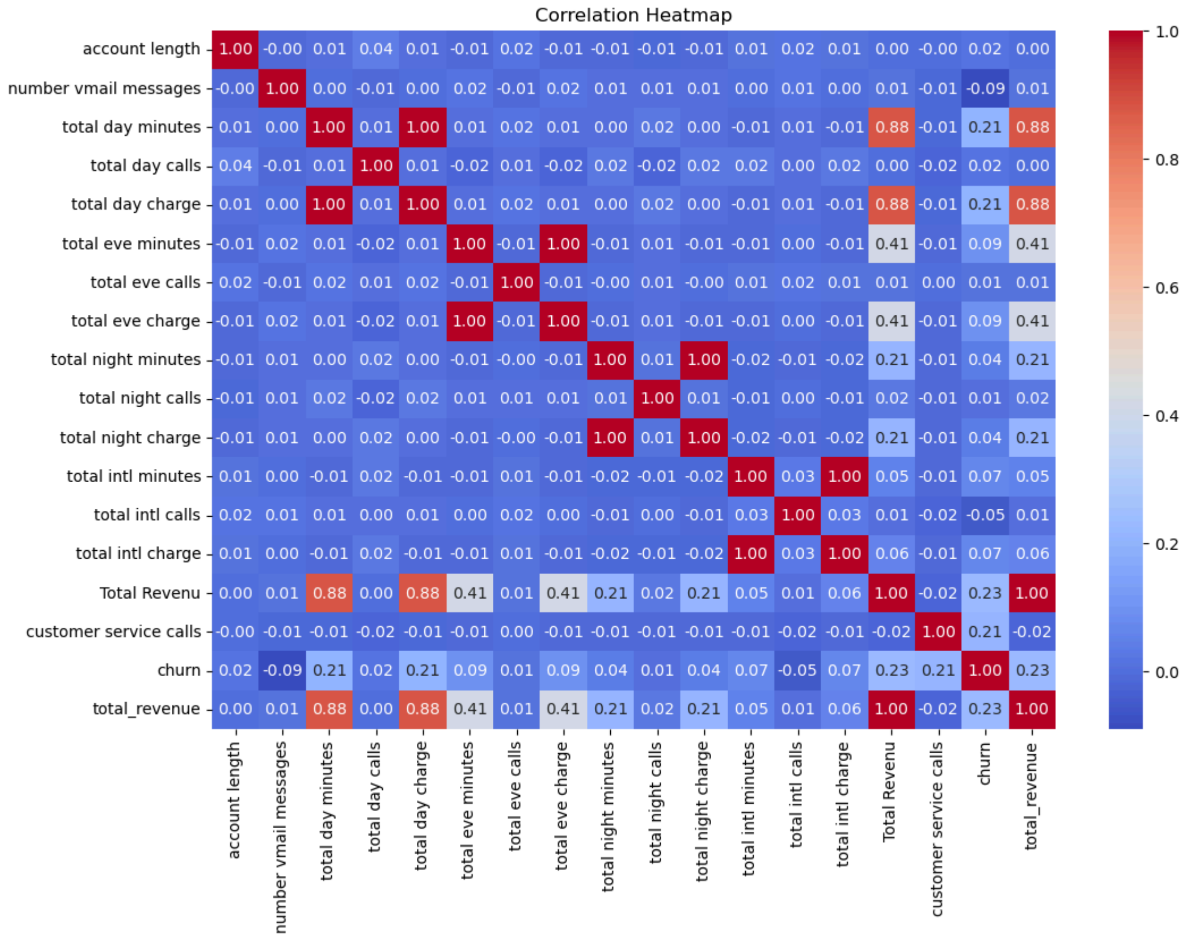
```
# Labels and values
labels = revenue_by_churn['Churned'].map({False: 'Stayed', True: 'Churned'})
sizes = revenue_by_churn['Total Revenue']


if labels.iloc[0] == 'Stayed':
    colors = ['salmon', 'darkred']
else:
    colors = ['darkred', 'salmon']

# Plot pie chart
plt.figure(figsize=(6, 6))
plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', startangle=90)
plt.title('Revenue Breakdown by Churn Status')
plt.axis('equal')
plt.show()
```



Correlation

In [154…

```
# correlation
correlation_matrix = df.corr(numeric_only=True)

# heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Heatmap")
plt.show()
```



- Positive Correlations: Features like customer service calls, total day minutes, and total day charge show a positive correlation with churn, indicating that higher usage in these areas is associated with a higher likelihood of customer churn.
- Negative Correlations: Features such as number vmail messages and total intl calls show a negative correlation with churn, suggesting that higher usage in these areas might reduce the likelihood of churn.
- Weak or Negligible Correlations: Several features like total night calls and account length have little to no correlation with churn, indicating minimal influence on the likelihood of a customer churning.

In [155…

```
## Dropping Features with High Correlation to pair (98% or more)

#Dropping the highly correlated features
to_drop = df.drop(columns=['total day charge', 'total eve charge', 'total night cha

print("Reduced df shape:", df.shape)
```

```
Reduced df shape: (3333, 21)
```

Observation & Conclusion

Observation:

- Churn rates vary significantly across states, with certain regions (e.g., Washington, Texas) showing notably higher churn levels.

- Churn appears more likely among customers at both ends of the tenure spectrum (very new and long term users) with churned users having a slightly higher average account length, indicating early drop-offs and late disengagement.
- Customers subscribed to the international plan are approximately 4 times more likely to churn, suggesting potential dissatisfaction with pricing or perceived value. In contrast, those with the voice mail plan are more likely to stay, indicating its possible role in enhancing user satisfaction.
- Although differences in call usage are subtle, churned customers tend to use more day and international minutes, which may signal either heavy reliance or cost-related concerns.
- Churned users incur higher average charges, especially during daytime and international calls, reinforcing the idea that high-spending customers may feel less value for money.
- There is a strong positive relationship between customer service interactions and churn. Customers who make 4 or more calls to support are at particularly high risk, possibly due to unresolved complaints or poor service experiences.

Recomendation:

- Run localized campaigns in high-churn states (Washington, Texas) with deeper nvestigation into region specific issues like service quality, network coverage or billing concerns.
- Develop onboarding programs for new customers and loyalty program for long term customers to reduce early drop-offs and late disengagement.
- Proactively monitor customers with 3+ surpport calls and prioritize them for resolution. Train customer service team to resolve issues on the first contact to prevent frustration.
- Reassess the value proposition of the international plan. This could involve improving call quality, reducing costs, or bundling with other perks to increase satisfaction.
- Consider making the voice mail plan a default offering or promote it more actively, given its positive association with retention.
- Introduce spending caps or usage notifications for customers who pay more especially during daytime & International calls to help manage expectations and reduce bill shock as these users are more likely to churn.

In [156…

```
df.columns
```

Out[156]:

```
Index(['state', 'area code', 'phone number', 'account length',
       'international plan', 'voice mail plan', 'number vmail messages',
       'total day minutes', 'total day calls', 'total eve minutes',
       'total eve calls', 'total night minutes', 'total night calls',
       'total intl minutes', 'total intl calls', 'Total Revenu',
       'customer service calls', 'churn', 'state_full', 'tenure_group',
       'total_revenue'],
      dtype='object')
```

Modeling

Data processing

Dropping Columns

- Removed the columns for day, evening, night, and international charges due to perfect correlation with their respective minutes columns. Since charges are directly derived from minutes (minutes × rate), they do not provide additional information.
- Dropped computed fields like total_revenue, as they are aggregates of existing variables.
- Excluded irrelevant columns such as phone_number and state, which do not contribute meaningful insights for churn analysis.

In [157…

```
df.drop(columns=['state', 'area code', 'phone number', 'Unnamed: 21', 'Total Revenu
df.head()
```

Out[157]:

	account length	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total eve minutes	total eve calls	total night minutes	total night calls	tot i minut
0	36	no	yes	30	146.3	128	162.5	80	129.3	109	1
1	104	no	no	0	278.4	106	81.0	113	163.2	137	
2	78	no	no	0	225.1	67	199.2	127	175.5	102	1
3	110	no	no	0	100.1	90	233.3	93	204.4	57	1
4	127	no	no	0	182.3	124	169.9	110	184.0	116	

Bianary Encoding

Transforming categorical features into dummy variables as 0 and 1 to be able to use them in classification models.

In [158…

```
categorical_columns = ['international plan', 'voice mail plan', 'churn']

# Mapping for binary categoricals
df['international plan'] = df['international plan'].map({'yes': 1, 'no': 0})
df['voice mail plan'] = df['voice mail plan'].map({'yes': 1, 'no': 0})
df['churn'] = df['churn'].astype(int)  # Convert bool to 0/1


df.head()
```

Out[158]:

	account length	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total eve minutes	total eve calls	total night minutes	total night calls	tot i minut
0	36	0	1	30	146.3	128	162.5	80	129.3	109	14
1	104	0	0	0	278.4	106	81.0	113	163.2	137	9
2	78	0	0	0	225.1	67	199.2	127	175.5	102	14
3	110	0	0	0	100.1	90	233.3	93	204.4	57	11
4	127	0	0	0	182.3	124	169.9	110	184.0	116	9

Defining features & target variable

In [159…

```
# churn is the target variable
X = df.drop('churn', axis=1)  ## Features / predictors
y = df['churn']               # Target variable
```

Logistic Regression

Predict whether a customer will churn (leave) using Logistic Regression.

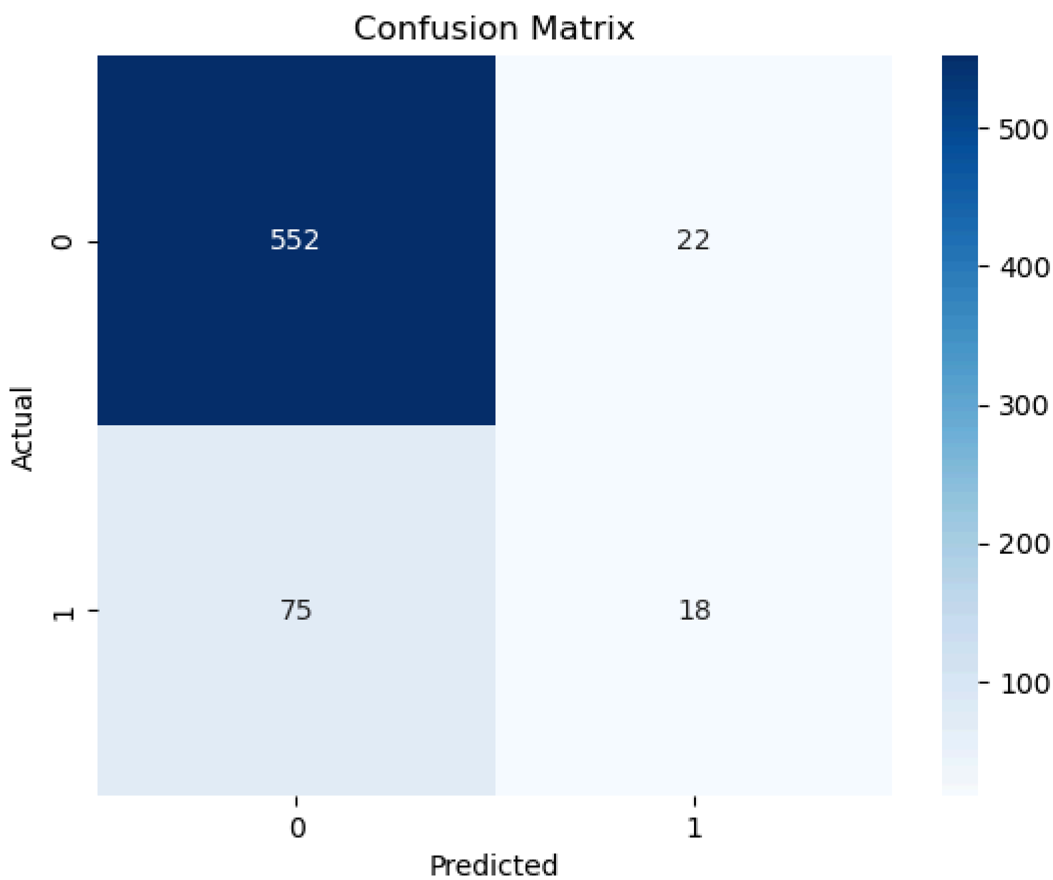
- Spliting the data into training and testing sets.
- Scaling the features for better model performance.
- Training logistic regression model.
- Making predictions on the test data.
- Evaluating the model using:
  - Accuracy
  - Confusion matrix
  - Precision, Recall, F1-score
- Visualizing the confusion matrix using a heatmap.

In [160…

```
# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta

# Confirm the split
print(f"Training set size: {X_train.shape}")
print(f"Testing set size: {X_test.shape}")


# Scaling the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Logistic Regression Model
model = LogisticRegression

model = LogisticRegression(random_state=42)
model.fit(X_train_scaled, y_train)
# Predictions
y_pred = model.predict(X_test_scaled)
```

```python
# Model Evaluation
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")

# Confusion Matrix
print(confusion_matrix(y_test, y_pred))

# Precision, Recall, F1-score, Support
print(classification_report(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()

print(y.value_counts())
```

```
Training set size: (2666, 13)
Testing set size: (667, 13)
Accuracy: 0.85
[[552  22]
 [ 75  18]]
              precision    recall  f1-score   support

           0       0.88      0.96      0.92       574
           1       0.45      0.19      0.27        93

    accuracy                           0.85       667
   macro avg       0.67      0.58      0.59       667
weighted avg       0.82      0.85      0.83       667
```



```
churn
0    2850
1     483
Name: count, dtype: int64
```

- Confusion Matrix:
  - True Negative = 552 : Correctly predicted customers who didn't churn.
  - False Positive= 22 :Predicted churn but actually they didn't churn.
  - False Negative = 75 :Predicted they wouldn’t churn, but they actually churned.
  - True Positive = 18 :Correctly predicted customers who did churn.

-Accuracy: The model correctly predicted 85% of all cases. However, this can be misleading in imbalanced datasets

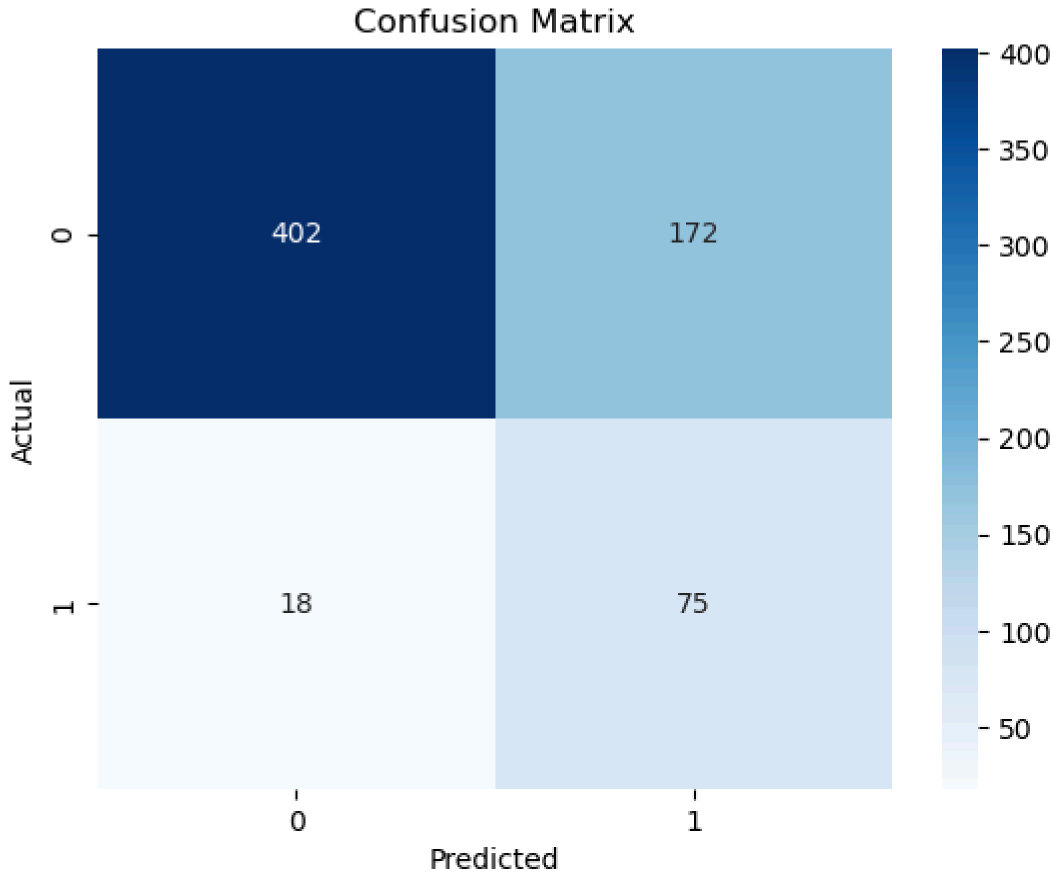
- Precision (Churn = 1): Of all predicted churners, only 45% were actually churners.
- Recall (Churn = 1): Of all actual churners, the model only caught 19%.
- F1 Score (Churn = 1): Balance between precision and recall. Only 0.27, which is low.

Logistic + Smote to candle class imbalance

In [161…

```
# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta

# Apply SMOTE to only the training data - This handles class imbalance
smote = SMOTE(random_state=42)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)

# Scaling the resampled training data and test data
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train_resampled)
X_test_scaled = scaler.transform(X_test)

# Training the model on resampled and scaled data
model_1 = LogisticRegression(random_state=42)
model_1.fit(X_train_scaled, y_train_resampled)

# Predictions
y_pred = model_1.predict(X_test_scaled)

# Evaluation
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")

print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

```
Accuracy: 0.72
[[402 172]
 [ 18  75]]
              precision    recall  f1-score   support

           0       0.96      0.70      0.81       574
           1       0.30      0.81      0.44        93

    accuracy                           0.72       667
   macro avg       0.63      0.75      0.63       667
weighted avg       0.87      0.72      0.76       667
```



- Confusion Matrix:
  - 402 True Negatives: Correctly predicted non-churners
  - 75 True Positives: Correctly predicted churners
  - 18 False Negatives: Churners predicted as non-churners → (the ones I missed - Type II Error) *(Missed churners - predict the customer would stay, but they actually left.)*
  - 172 False Positives: Non-churners predicted as churners → false alarms / Type 1 Error *(predicted the customer will churn, but they actually stayed.)*
- The model is correctly identifying 81% of churners(Recall = 0.81). However, the precision is low (0.30), meaning that many customers flagged as churners are actually staying (many false positives /Type I errors). This may lead to wasting retention efforts on customers who weren't at risk.

Random Forest

In [162…

```
# Scale resampled training data and test data
scaler = StandardScaler()
X_train_resampled_scaled = scaler.fit_transform(X_train_resampled)  # Use resamplec
X_test_scaled = scaler.transform(X_test)  # Test data stays the same

# Initialize Random Forest model
rf_model = RandomForestClassifier(random_state=42)

# Train on resampled and scaled data
rf_model.fit(X_train_resampled_scaled, y_train_resampled)

# Predict on test set
y_pred_rf = rf_model.predict(X_test_scaled)

# Evaluate
print(f"Accuracy: {accuracy_score(y_test, y_pred_rf):.2f}")
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred_rf))
print("Classification Report:")
print(classification_report(y_test, y_pred_rf))
```

```
Accuracy: 0.93
Confusion Matrix:
[[543  31]
 [ 17  76]]
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.95      0.96       574
           1       0.71      0.82      0.76        93

    accuracy                           0.93       667
   macro avg       0.84      0.88      0.86       667
weighted avg       0.93      0.93      0.93       667
```

- The model correctly predicted 93% of the test cases.
- Confusion Matrix:
  - 543 True Negatives: The model correctly predicted 543 customers did not churn.
  - 76 True Positives: The model correctly predicted 76 customers did churn.
  - 31 False Positives: 31 customers were predicted to churn but did not.
  - 17 False Negatives : 17 customers who actually churned were missed by the model.
- Not Churn
  - Precision: 0.97 → Of all predicted not churn, 97% were correct.
  - Recall: 0.95 → Of all actual not churn, 95% were caught.
  - F1-score: 0.96 → Strong overall performance.
- Churn
  - Precision: 0.71 → Of all predicted churns, 71% were correct.
  - Recall: 0.82 → The model captured 82% of actual churners.
  - F1-score: 0.76 → Reasonable performance, but weaker than for class 0.
- The model is very good at identifying non-churners. It does reasonably well in identifying churners (better recall than precision), which is important in churn prediction

because missing a churner can cost money.

Model Comparisons

- class 1 = churn

Model	Confusion Matrix	Accuracy	Recall (Class 1)	F1-Score (Class 1)	Support (Class 1)
1. Logistic (No SMOTE)	`[[552, 22], [75, 18]]`	0.85	0.19	0.27	93
2. Logistic + SMOTE	`[[402, 172], [18, 75]]`	0.72	0.81	0.44	93
3.**Random Forest + SMOTE**	**`[[543, 31], [17, 76]]`**	**0.93**	**0.82**	**0.76**	**93**

Model Interpretation

- Logistic no resampling: has high accuracy, but very poor recall and F1 for the minority class. It misses many potential churners, making it less suitable for identifying at risk customers.
- Logistic + SMOTE improves recall for churners a lot, but loses accuracy by catching churners, but misclassifies many non-churners.
- Random Forest + SMOTE gives the best of both worlds:
  - Highest overall accuracy
  - High recall and F1-score for churners
  - Balanced performance for both classes

Model Selection

- I selected Random Forest model combined with SMOTE resampling as the final model for deployment as it achieves:
  - Highest accuracy (93%)
  - Strong recall (82%) for identifying churners
  - Best F1-score (76%) which balances precision and recall
- Class imbalance is addressed using SMOTE, improving the model's ability to detect minority cases.
- Random Forest captures complex relationships in customer behavior better than linear models like logistic regression.

In [ ]: