SYRIATEL CUSTOMER CHURN PREDICTION

PROJECT OVERVIEW:

In Syria, the telecommunications industry faces a significant challenge in retaining customers amidst increasing competition and evolving consumer preferences. SyriaTelcom, one of the leading telecom service providers in the country, seeks to reduce customer churn by identifying patterns and factors contributing to customer attrition. High customer churn not only results in revenue loss but also undermines the company's reputation and market position.

BUSINESS PROBLEM OBJECTIVE:

SyriaTel, a telecommunications company, aims to proactively address customer churn to retain valuable customers, reduce revenue loss, and enhance overall customer satisfaction and loyalty. To achieve this objective, SyriaTel seeks to develop a predictive model capable of identifying customers at risk of churn. By leveraging historical customer data and predictive analytics, SyriaTel aims to anticipate potential churn events and implement targeted retention strategies to mitigate churn and foster long-term customer relationships.

OBJECTIVE:

The objective of this project is to analyze SyriaTelcom's customer data to understand the factors influencing churn and develop predictive models to forecast customer attrition. By leveraging machine learning algorithms and predictive analytics, the project aims to:

Identify key features and patterns associated with customer churn and non-churn.

Build predictive models to forecast the likelihood of churn for individual subscribers.

Provide actionable insights to SyriaTelcom for implementing targeted retention strategies and reducing customer attrition.

Enhance customer satisfaction and loyalty by addressing the underlying issues driving churn.

Improve SyriaTelcom's market position and competitiveness in the telecommunications industry by fostering long-term customer relationships.

RESEARCH QUESTIONS:

1 .What are the key factors contributing to customer churn ?

2 .How do characteristics, such as location, influence the likelihood of customer churn?

3 .Are there specific contract terms or pricing plans associated with higher churn rates among customers?

4 .Which is the best model to accurately predict churn?

DATA UNDERSTANDING:

The dataset used in this project was obtained from SyriaTelcom's internal database, which contains comprehensive records of customer interactions and telecommunications services(+3000 customers and 20 columns).This makes it highly suitable for addressing the business problem at hand of predicting customer churn for Syria Telcom.Contained in the dataset are:

Extensive customer information:Contains a set of variables that provide insights into customer behavior, preferences, and usage patterns. This includes features such as account length, call details (e.g., duration, charges), service subscriptions (e.g., voice mail plan, international plan), and customer service interactions.

Historical Data: The dataset spans a considerable time period, allowing us to analyze historical trends and patterns in customer churn. By leveraging historical data, we can identify recurring patterns and factors that contribute to customer attrition.

Granular Call Details: Detailed information about call usage, including call duration, charges, and time of day, enables us to explore how different calling patterns may influence customer churn. This granularity allows for a more nuanced understanding of customer behavior.

Target Variable (Churn): The dataset includes a binary target variable indicating whether a customer has churned or not. This allows us to frame the prediction task as a supervised learning problem, where the goal is to accurately classify customers as churners or non-churners based on their characteristics and behavior.

Data Exploration:

Load the dataset.

Understand the structure of the dataset (columns, data types, etc.).

Check for missing values.

Explore the distribution of target variable (churn).

Understand the distribution and characteristics of features.

IMPORT LIBRARIES AND LOAD DATA

In [1]:
```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_

# Set seaborn style
sns.set(style="whitegrid")
```

In [2]:
```
#load the data
df = pd.read_csv('bigml_59c28831336c6604c800002a 2.csv')
# observe the first five entries of the dataset
df.head()
```

Out[2]:

	state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total day charge	...	tot e ca
0	KS	128	415	382-4657	no	yes	25	265.1	110	45.07	...	
1	OH	107	415	371-7191	no	yes	26	161.6	123	27.47	...	1
2	NJ	137	415	358-1921	no	no	0	243.4	114	41.38	...	1
3	OH	84	408	375-9999	yes	no	0	299.4	71	50.90	...	
4	OK	75	415	330-6626	yes	no	0	166.7	113	28.34	...	1.

5 rows × 21 columns

UNDERSTAND THE DATAFRAME STRUCTURE

In [3]:
```
#print names of all the columns in the dataset
df.shape
```

Out[3]: (3333, 21)

In [4]:
```
#check the data types of each column
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3333 entries, 0 to 3332
Data columns (total 21 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   state                   3333 non-null   object 
 1   account length          3333 non-null   int64  
 2   area code               3333 non-null   int64  
 3   phone number            3333 non-null   object 
 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  customer service calls  3333 non-null   int64  
 20  churn                   3333 non-null   bool   
dtypes: bool(1), float64(8), int64(8), object(4)
memory usage: 524.2+ KB
```

Our dataset has 3,333 rows and 21 columns and no null values.We also observe that there is a mix of data types.Our target variable 'churn' is a boolean data type making this a classification problem

- state : Different states of the customers
- account length: number of days a customer's account has been active
- area code : location of the customer
- phone number : customer's phone number
- international plan : whether the customer uses the international plan or not
- voice mail plan : whether the customer has subscribed to vmail plan or not
- number vmail messages : if customer has a vmail plan, how many vmail messages do they get
- total day minutes : total number of call minutes used during the day
- total day calls : total number of calls made during the day
- total day charge : total charge on day calls
- total eve minutes : total number of call minutes used in the evening
- total eve calls : total calls made in the evening
- total eve charge : total charge on evening calls
- total night minutes: Total number of call minutes used at night
- total night calls : Total number of night calls

- total night charge : Total charge on night calls
- total intl minutes : total international minutes used
- total intl calls : total number of international calls made
- total intl charge : total charge on international calls
- customer service calls : number of calls made to customer service
- churn : boolean on whether the customer left or not

In [5]:
```
#check for missing values
df.isnull().sum()
```

Out[5]:
```
state                     0
account length            0
area code                 0
phone number              0
international plan        0
voice mail plan           0
number vmail messages     0
total day minutes         0
total day calls           0
total day charge          0
total eve minutes         0
total eve calls           0
total eve charge          0
total night minutes       0
total night calls         0
total night charge        0
total intl minutes        0
total intl calls          0
total intl charge         0
customer service calls    0
churn                     0
dtype: int64
```

Our dataset has no missing values.

In [6]:
```
df.duplicated().sum()
```

Out[6]:
```
0
```

our dataset has no duplicate row values.

In [7]:
```
#summary of descriptive statistics for numerical columns
df.describe(include=[np.number])
```

Out[7]:

	account length	area code	number vmail messages	total day minutes	total day calls	total day charge	total e minut
count	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.0000
mean	101.064806	437.182418	8.099010	179.775098	100.435644	30.562307	200.9803
std	39.822106	42.371290	13.688365	54.467389	20.069084	9.259435	50.7138
min	1.000000	408.000000	0.000000	0.000000	0.000000	0.000000	0.0000
25%	74.000000	408.000000	0.000000	143.700000	87.000000	24.430000	166.6000
50%	101.000000	415.000000	0.000000	179.400000	101.000000	30.500000	201.4000
75%	127.000000	510.000000	20.000000	216.400000	114.000000	36.790000	235.3000
max	243.000000	510.000000	51.000000	350.800000	165.000000	59.640000	363.7000

DATA PREPARATION.

By making a copy of the dataframe, we can conduct an indepth analysis of the data's properties, distributions, and relationships, gaining valuable insights that will inform subsequent analytical steps. This approach maintains the integrity of the original dataset while enabling us to perform in-depth EDA with confidence and accuracy.

In [8]:
```
#make a copy of original dataframe
df2 = df.copy()
df2.head()
```

Out[8]:

	state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total day charge	...	tot e ca
0	KS	128	415	382-4657	no	yes	25	265.1	110	45.07	...	
1	OH	107	415	371-7191	no	yes	26	161.6	123	27.47	...	1
2	NJ	137	415	358-1921	no	no	0	243.4	114	41.38	...	1
3	OH	84	408	375-9999	yes	no	0	299.4	71	50.90	...	
4	OK	75	415	330-6626	yes	no	0	166.7	113	28.34	...	1.

5 rows × 21 columns

In [9]:
```
# Drop 'phone number' columns
df2.drop(columns=['phone number'],axis=1,inplace=True)
df2.head()
```

Out[9]:

	state	account length	area code	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total day charge	total eve minutes	total eve calls
0	KS	128	415	no	yes	25	265.1	110	45.07	197.4	99
1	OH	107	415	no	yes	26	161.6	123	27.47	195.5	103
2	NJ	137	415	no	no	0	243.4	114	41.38	121.2	110
3	OH	84	408	yes	no	0	299.4	71	50.90	61.9	88
4	OK	75	415	yes	no	0	166.7	113	28.34	148.3	122

From our dataset we will be dropping the 'phone number' column as it is a unique identifier for each customer therefore not relevant for analysis.

EXPLORATORY DATA ANALYSIS (EDA)

In this analysis, the 'churn' feature serves as the dependent variable.The 'churn' variable signifies whether a customer has terminated their contract with SyriaTel. A value of 'True' means a contract termination, while 'False' indicates that the customer has not terminated their contract and maintains an active account.

In [10]:
```
df.nunique()
```

Out[10]:
```
state                       51
account length             212
area code                    3
phone number              3333
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
customer service calls      10
churn                        2
dtype: int64
```

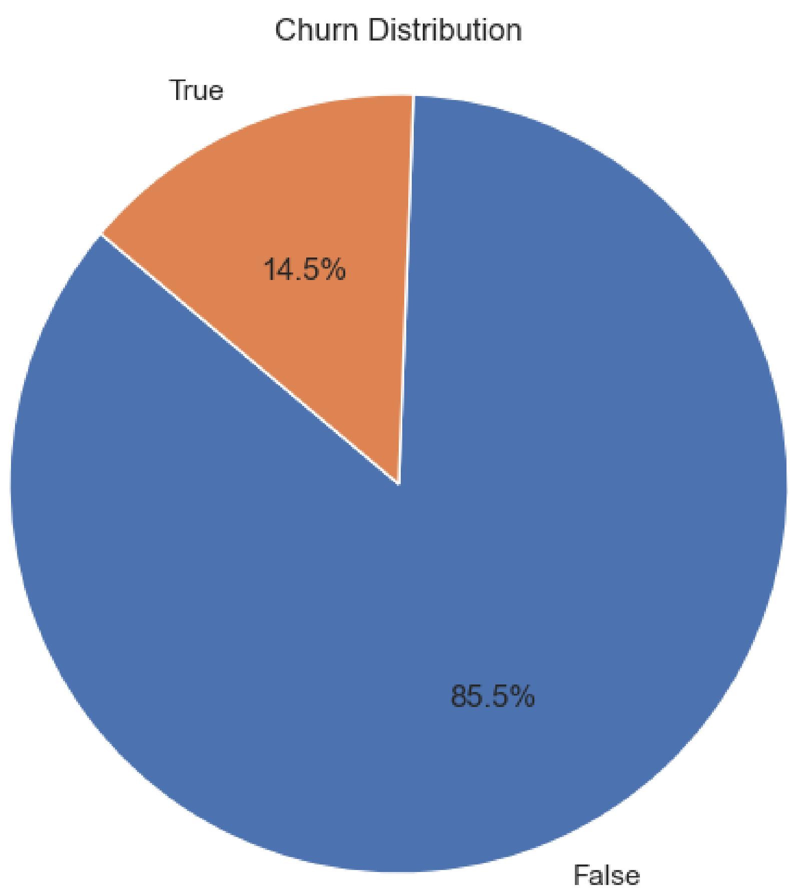
NUMERICAL FEATURES:(account length, number vmail messages, total day minutes, total day calls, total day charge, total eve minutes, total eve calls,total eve charge,total night minutes,total night calls,total night charge,total intl minutes,total intl charge,customer service calls)

CATEGORICAL FEATURES: (state,area code,international plan,voicemail plan)

In [11]:
```
numeric_cols = ['account length','number vmail messages','total day minutes','
                'total eve minutes','total eve calls','total eve charge','tota
                'total night charge','total intl minutes','total intl calls','
categoric_cols = ['state','area code','international plan','voice mail plan']
```

PIE-CHART OF NUMBER OF CHURNED AND NON-CHURNED CUSTOMERS.

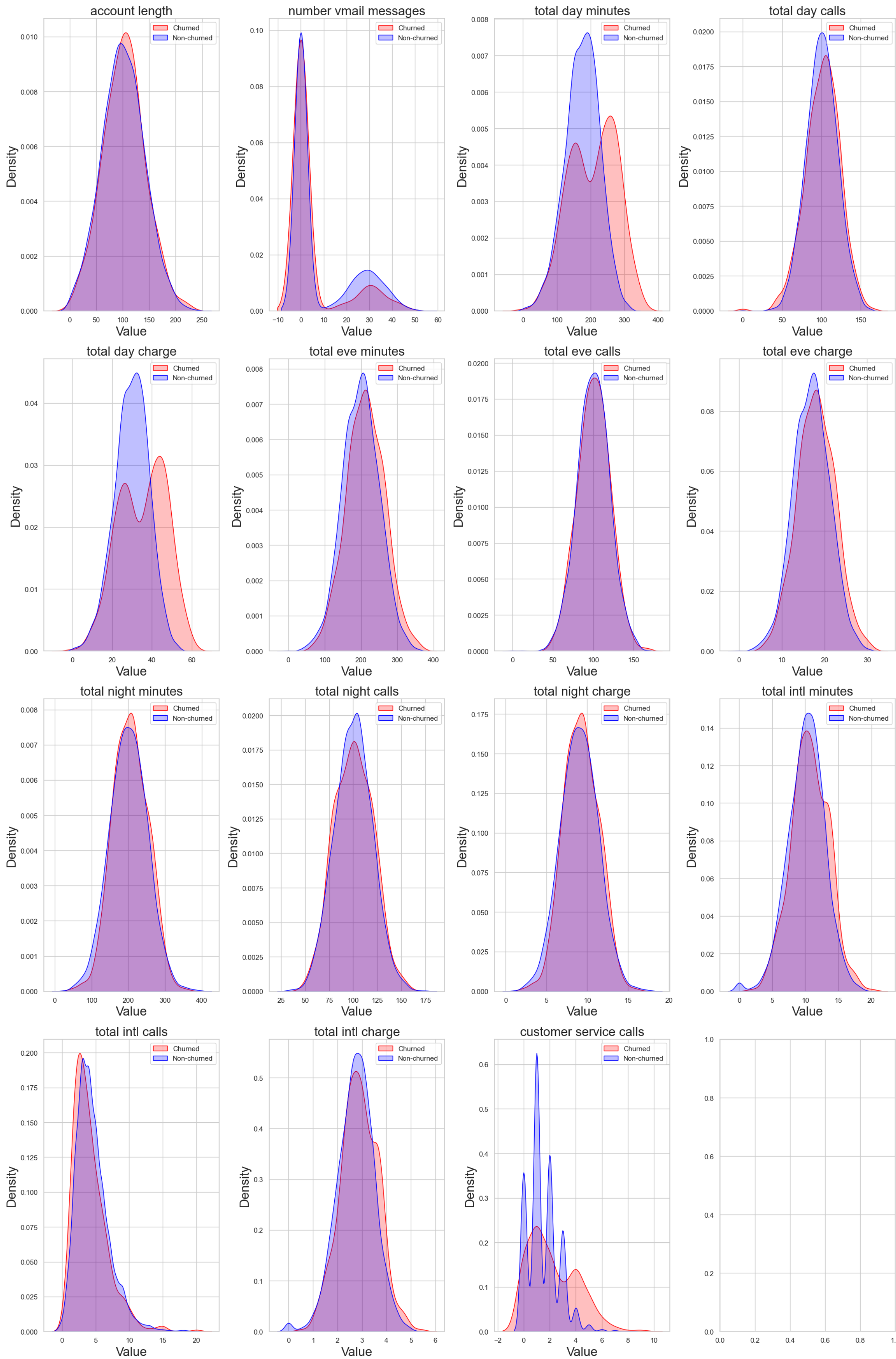
In [12]:
```
# Count the number of churned and non-churned customers
churn_counts = df['churn'].value_counts()

# Plot a pie chart
plt.figure(figsize=(8, 6))
plt.pie(churn_counts, labels=churn_counts.index, autopct='%1.1f%%', startangle
plt.title('Churn Distribution')
plt.axis('equal')
plt.savefig('Churn_Distribution')
plt.show()
```



The above pie chart shows the distribution of churned and non-churned syria tel customers.The distribution is indicated in percentage,with 14.5% "true churn" indicates customers who have ended their subscription. 85.5% "false churn" indicates customers who are still active subscribers.This also shows "non-churn") has a much higher count compared to the other class ("churn"), indicating that the dataset has a class imbalance that may lead to model complications such as model bias.

DISTRIBUTION PLOT OF NUMERICAL VARIABLES

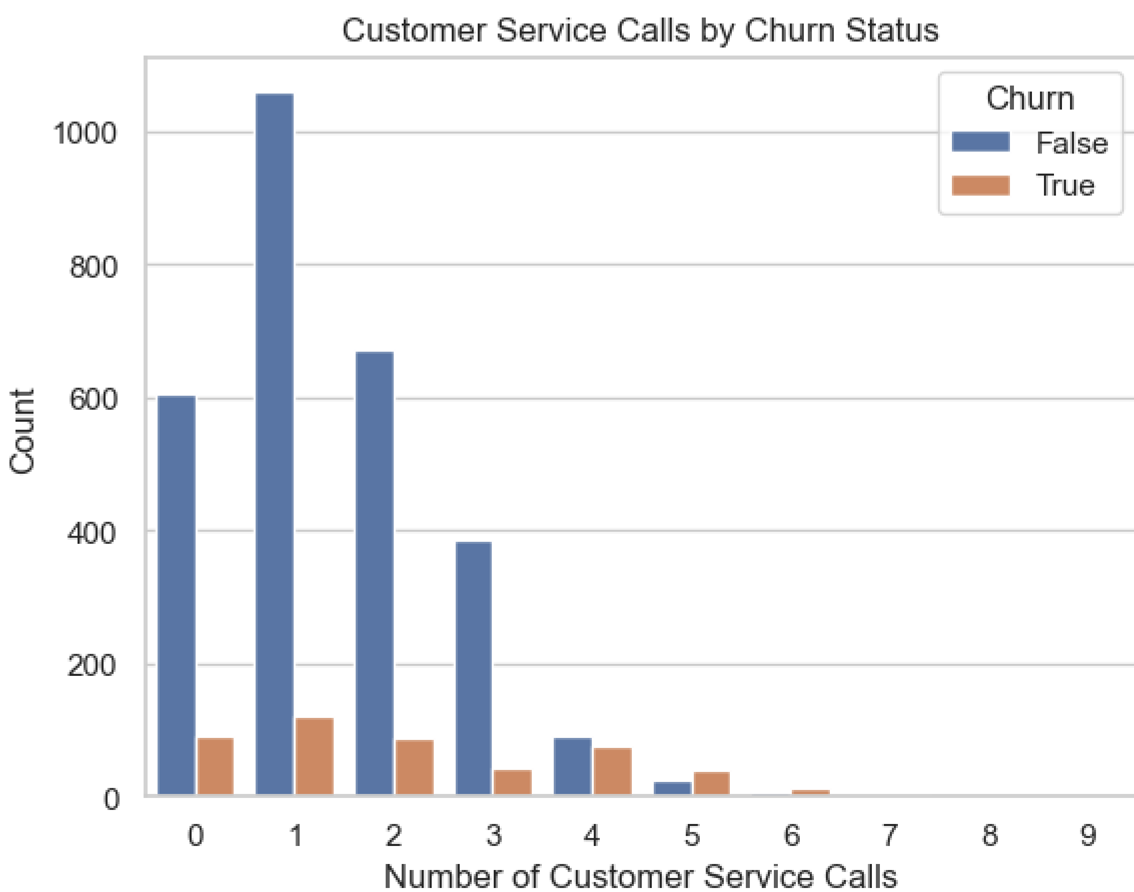
In [13]:
```
# Create subplots for each numerical variable
num_plots = len(numeric_cols)
num_rows = 4
num_cols = 4
fig, axes = plt.subplots(nrows=num_rows, ncols=num_cols, figsize=(20, 30))

# Iterate through numerical variables
for i, var in enumerate(numeric_cols):
    row = i // num_cols
    col = i % num_cols

    # Plot churned customers
    sns.kdeplot(df[df['churn'] == True][var], shade=True, ax=axes[row, col], c

    # Plot non-churned customers
    sns.kdeplot(df[df['churn'] == False][var], shade=True, ax=axes[row, col],

    # Set title, labels, and legend
    axes[row, col].set_title(var, fontsize=20)
    axes[row, col].set_xlabel('Value', fontsize=20)
    axes[row, col].set_ylabel('Density', fontsize=20)
    axes[row, col].legend()

plt.tight_layout()
plt.savefig('numerical_distribution_plot')
plt.show()
```


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 calls
total intl charge
customer service calls
Churned
Non-churned
Density
Value

Above are distribution plots of churned and non-churned customers in the numerical category.We observe that non-churned customers are more than churned customers.we also observe that the distribution is normal while that of total international calls is skewed to the right though still normally distributed.Customers service calls is observed to have a few peaks in it distribution, this could be due the column countaining float and not an interger number.

CUSTOMER SERVICE CALLS

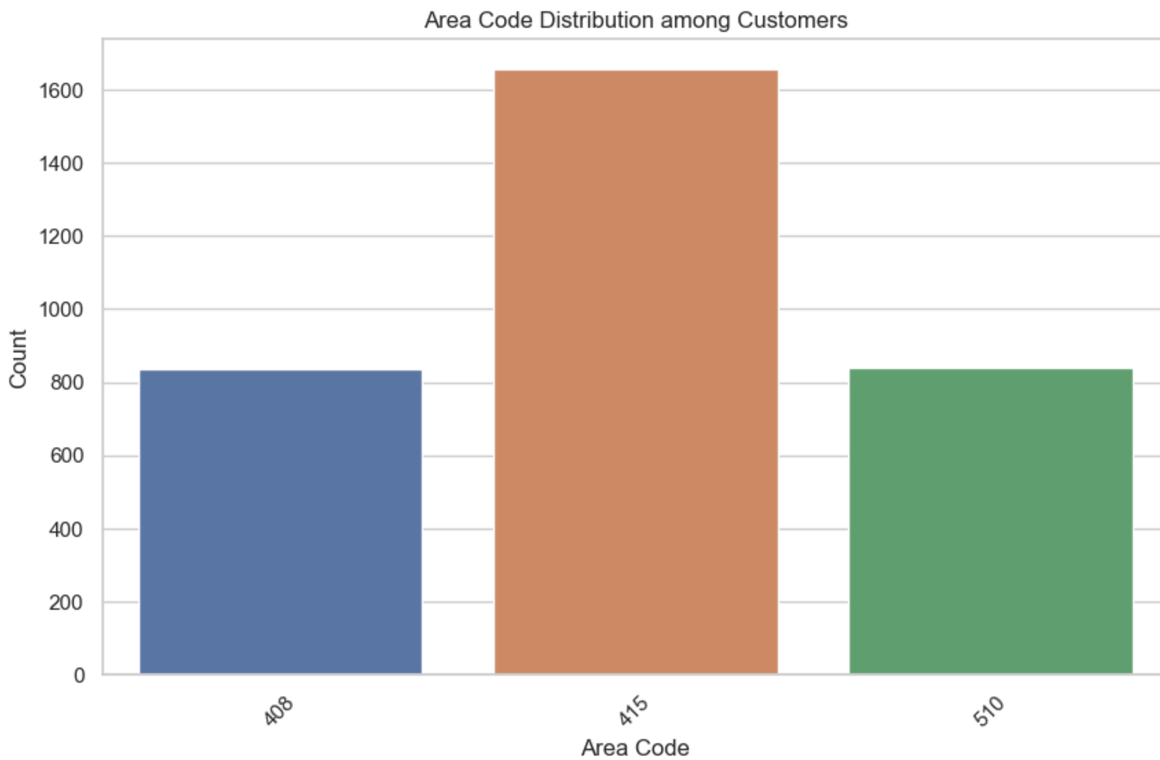
In [14]:
```
sns.countplot(x='customer service calls', data=df2, hue='churn').set(title='Cu
plt.xlabel('Number of Customer Service Calls')
plt.ylabel('Count')
plt.title('Customer Service Calls by Churn Status')
plt.legend(title='Churn', loc='upper right')

# Save the plot
plt.savefig('customer_service_calls_plot')
plt.show()
```



There is an obvious relationship between true churn rate and customer service calls.After 4 calls this customers are a lot more likely to terminate service this could be due to disatisfaction with the overall customer experience and/or unmet expectations.

PLOT DISTRIBUTION OF CUSTOMERS IN THE VARIOUS AREA CODES.

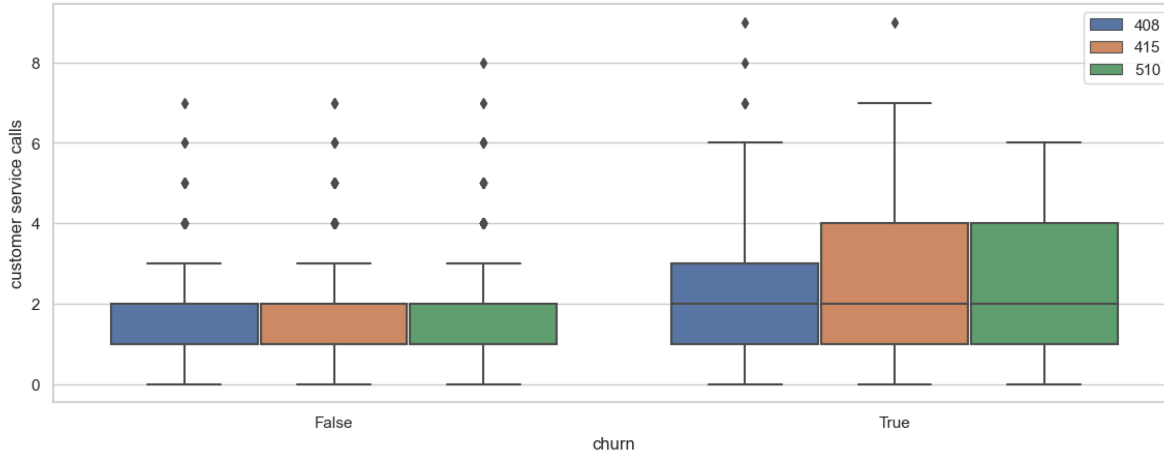
In [15]:
```
#count and plot the number of customers in each area code
plt.figure(figsize=(10, 6))
sns.countplot(data=df2, x='area code')
plt.xlabel('Area Code')
plt.ylabel('Count')
plt.title('Area Code Distribution among Customers')
plt.xticks(rotation=45)
plt.savefig('Area_Code_Distribution_among_Customers')
plt.show()
```



This distribution indicates that the majority of SyriaTel customers, specifically 50% of them, are located in the area with the code 415. The remaining 50% of customers are evenly split between the areas with codes 510 and 408.

DISTRIBUTION OF CHURNED AND NON -CHURNED CUSTOMERS IN THE AREA CODES.

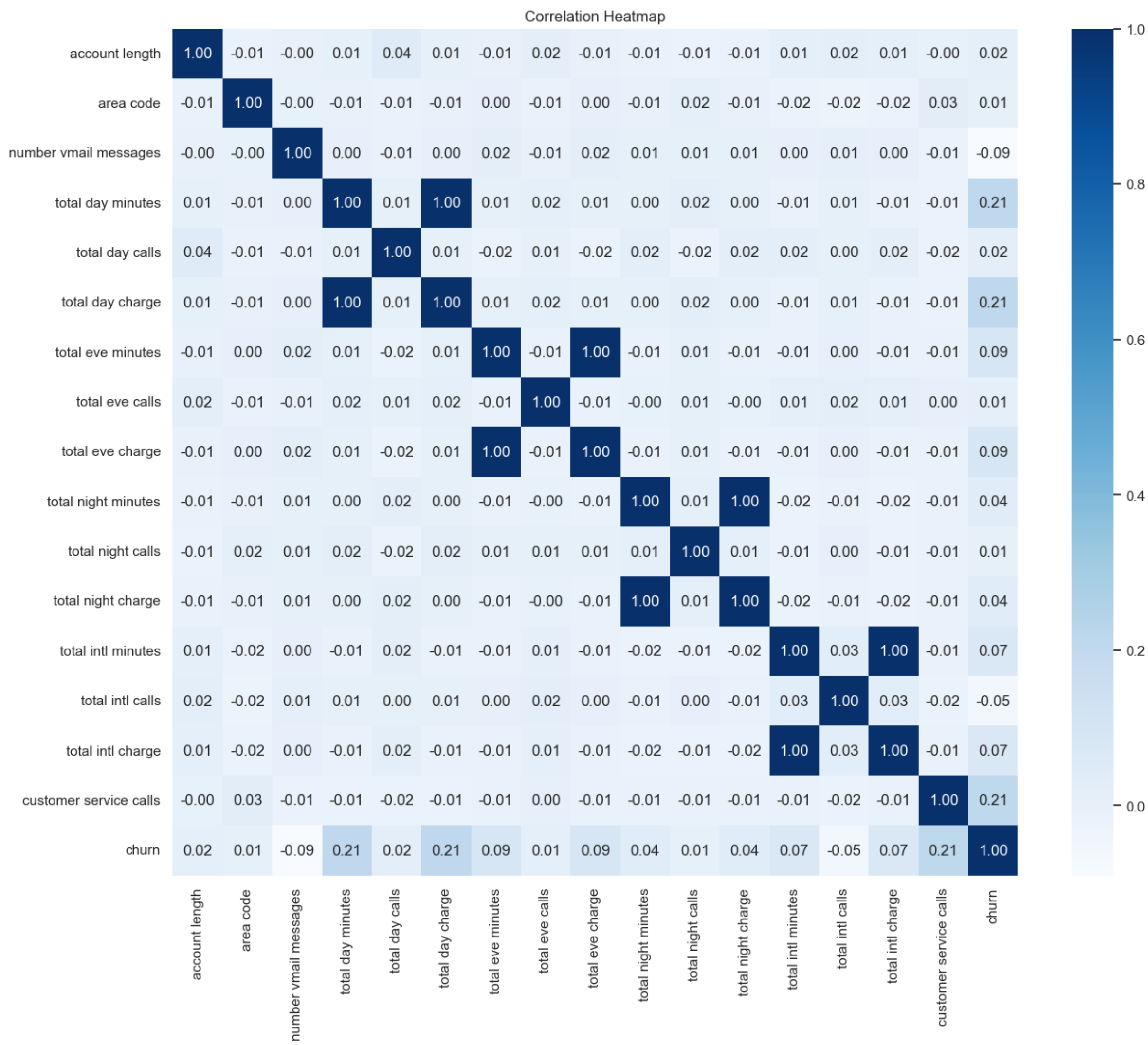
In [16]:
```
plt.figure(figsize=(14,5))
sns.boxplot(data=df2,x='churn',y='customer service calls',hue='area code');
plt.legend(loc='upper right');
```



This plot displays the combined counts of churn and non-churn instances for each area code with area-code 415 having the highest churn count.

NUMERICAL COLUMN CORRELATION PLOT.

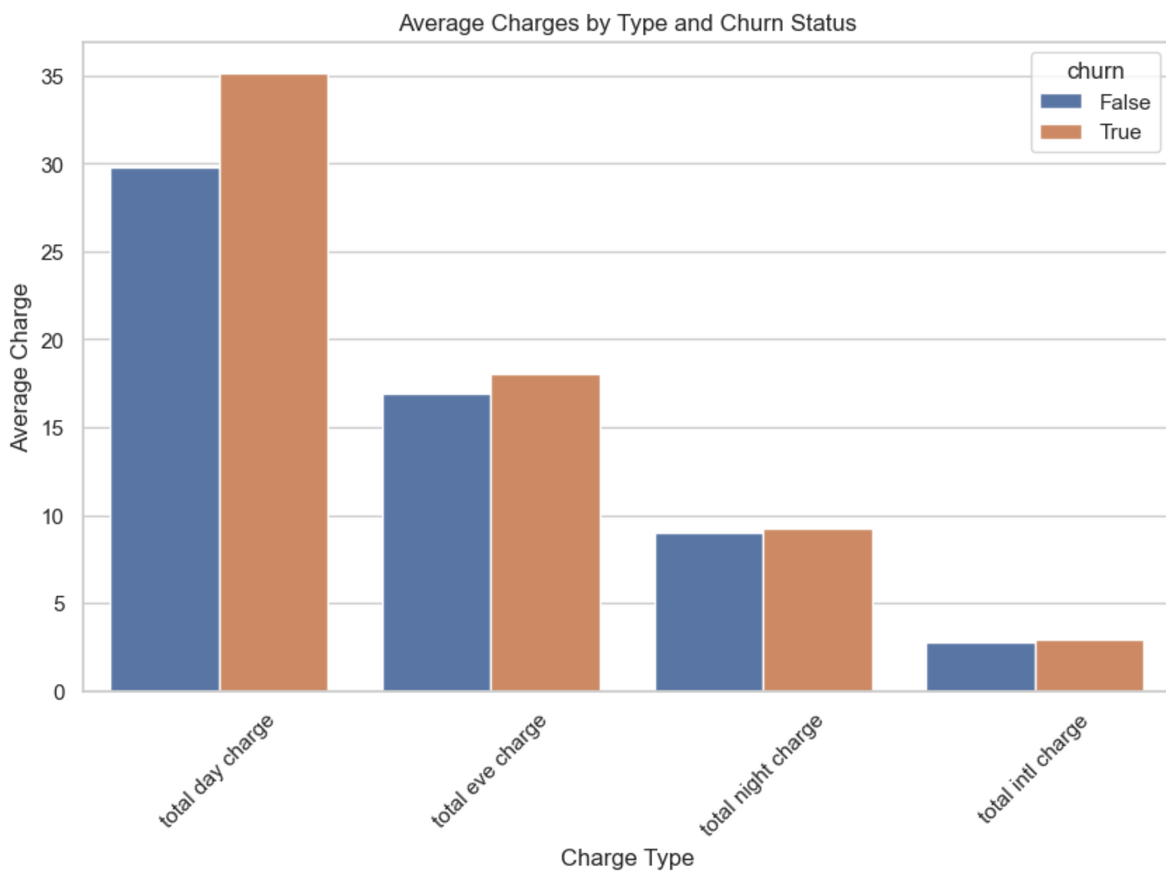
In [17]:
```python
# Calculate the correlation matrix for all numeric columns including 'churn'
corr_mat = df2.corr()
plt.subplots(figsize=(15,12))
sns.heatmap(corr_mat, annot=True, cmap='Blues', square=True, fmt='.2f')
plt.title('Correlation Heatmap')
plt.xticks(rotation=90)
plt.yticks(rotation=0)
plt.savefig('numeric_correlation_heatmap')
plt.show()
```



some of the features in the dataset demonstrate a perfect positive correlation, such as "Total day charge" and "Total day minutes", "Total eve charge" and "Total eve minutes", "Total night charge" and "Total night minutes", and "Total int charge" and "Total int minutes".They have a correlation coefficient of 1.00, indicating perfect multicollinearity. During modeling, perfect multicollinearity can cause issues, but its impact on the nonlinear models can vary.However when some models get affected by perfect multicollinearity, others may not be influenced significantly.

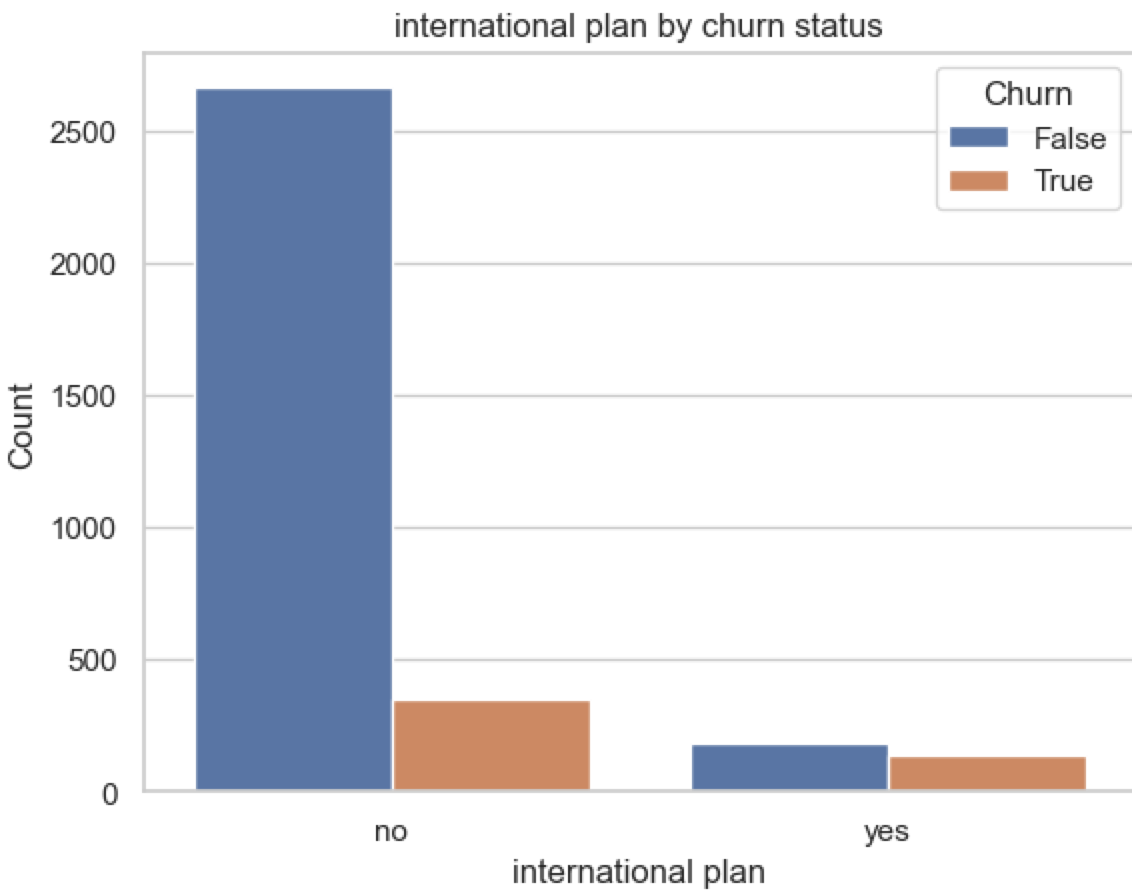
CHURN BY CHARGE STATUS

In [18]:
```
# Calculate average charges for each type of charge
average_charges = df2.groupby('churn')[['total day charge', 'total eve charge'

# Melt the dataframe for easier plotting
average_charges_melted = pd.melt(average_charges, id_vars='churn', var_name='c
# Plot the bar plot
plt.figure(figsize=(10, 6))
sns.barplot(x='charge_type', y='average_charge', hue='churn', data=average_cha
plt.title('Average Charges by Type and Churn Status')
plt.xlabel('Charge Type')
plt.ylabel('Average Charge')
plt.xticks(rotation=45)
plt.savefig('Average_charges_type_churn')
plt.show()
```



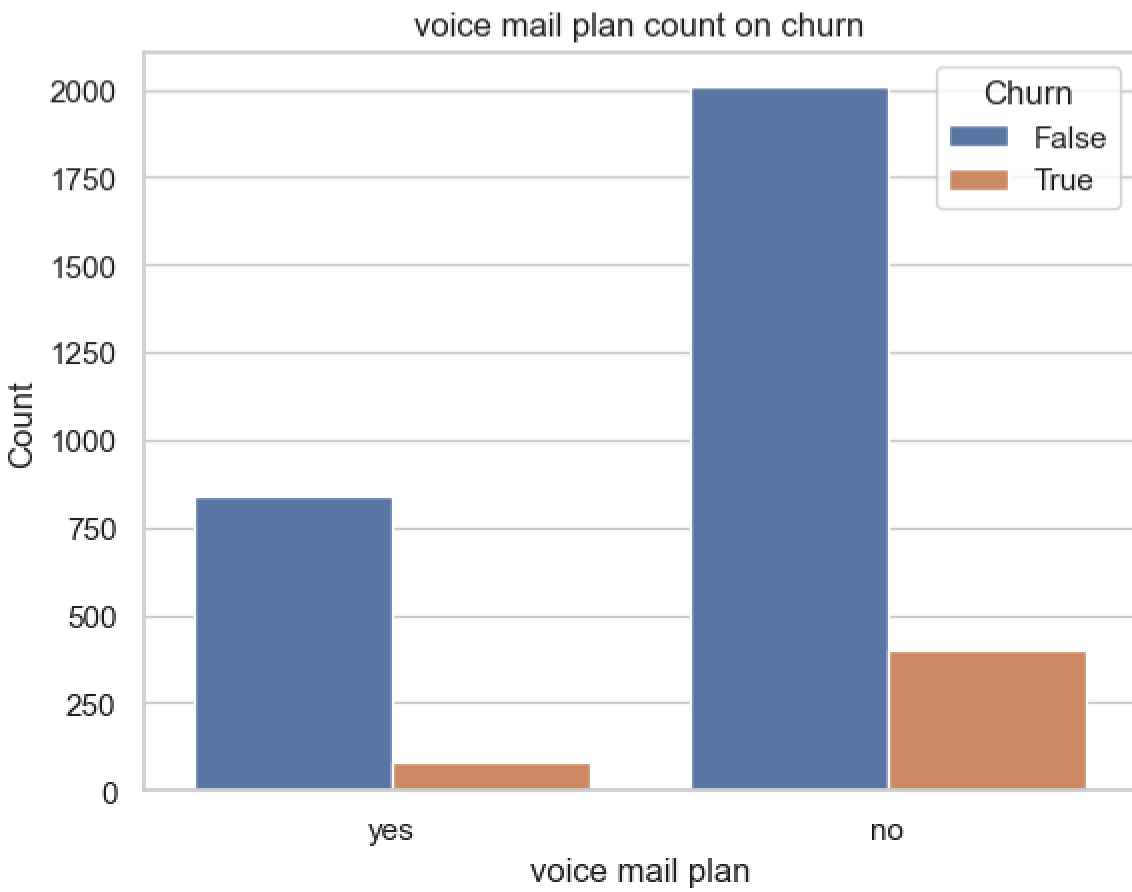
DISTRIBUTION OF CATEGORICAL VARIABLE

In [19]:
```
#cat_vars = ['state', 'international plan', 'voice mail plan']
sns.countplot(x='international plan', data=df2, hue='churn').set(title='intern
plt.xlabel('international plan')
plt.ylabel('Count')
plt.title('international plan by churn status')
plt.legend(title='Churn', loc='upper right')

# Save the plot
plt.savefig('international_plan_by_churn_status')
plt.show()
```



We have a small number of customers with an international plan howerver we observe a high churn rate among this group.Possible reasons for the high churn rate could be dissatisfaction with the international plan, international plan charges etc. Emphasizing a need for plan restructure.

In [20]:
```
sns.countplot(x='voice mail plan', data=df2, hue='churn').set(title='voice mai
plt.xlabel('voice mail plan')
plt.ylabel('Count')
plt.title('voice mail plan count on churn')
plt.legend(title='Churn', loc='upper right')

# Save the plot
plt.savefig('voice_mail_plan_count_on_churn')
plt.show()
```



We can observe from the plot above that there is a significantly low churn rate among customers with a voicemail plan.This indicates customers have a prefernce of using this plan.

DATA PREPROCESSING AND PREPARATION

In [21]:
```
print("original dataframe has {} columns.".format(df2.shape[1]))
# Calculate the correlation matrix and take the absolute value
corr_matrix = df2.corr().abs()

# Create a True/False mask and apply it
mask = np.triu(np.ones_like(corr_matrix, dtype=bool))
tri_df2 = corr_matrix.mask(mask)

# List column names of highly correlated features (r > 0.90)
to_drop = [c for c in tri_df2.columns if any(tri_df2[c] >  0.90)]

new_df2 = df2.drop(to_drop, axis=1) # Drop the features
print("new dataframe has {} columns.".format(new_df2.shape[1]))
```

```
original dataframe has 20 columns.
new dataframe has 16 columns.
```

Transform "churn"column from true and false to 0s and 1s

In [22]:
```
new_df2['churn'].value_counts()
```

Out[22]:
```
False    2850
True      483
Name: churn, dtype: int64
```

In [23]:
```
new_df2['churn'] = new_df2['churn'].map({True: 1, False: 0}).astype('int')
new_df2.head()
```

Out[23]:

	state	account length	area code	international plan	voice mail plan	number vmail messages	total day calls	total day charge	total eve calls	total eve charge	total night calls	t ni cha
0	KS	128	415	no	yes	25	110	45.07	99	16.78	91	1
1	OH	107	415	no	yes	26	123	27.47	103	16.62	103	1
2	NJ	137	415	no	no	0	114	41.38	110	10.30	104	7
3	OH	84	408	yes	no	0	71	50.90	88	5.26	89	8
4	OK	75	415	yes	no	0	113	28.34	122	12.61	121	8

ONE-HOT ENCODING CATEGORICAL FEATURES.

To be able to run a classification model categorical features are transformed into dummy variable values of 0 and 1

In [24]:
```
dummy_df2_area_code = pd.get_dummies(new_df2["area code"],dtype=np.int64,prefi
dummy_df2_international_plan = pd.get_dummies(new_df2["international plan"],dt
dummy_df2_voice_mail_plan = pd.get_dummies(new_df2["voice mail plan"],dtype=np


new_df2 = pd.concat([new_df2,dummy_df2_area_code,dummy_df2_international_plan,
new_df2 = new_df2.loc[:,~new_df2.columns.duplicated()]
new_df2 = new_df2.drop(['area code','international plan','voice mail plan'],ax
```

In [25]:
```
new_df2.head()
```

Out[25]:

	state	account length	number vmail messages	total day calls	total day charge	total eve calls	total eve charge	total night calls	total night charge	total intl calls	total intl charge	custom servic cal
0	KS	128	25	110	45.07	99	16.78	91	11.01	3	2.70	
1	OH	107	26	123	27.47	103	16.62	103	11.45	3	3.70	
2	NJ	137	0	114	41.38	110	10.30	104	7.32	5	3.29	
3	OH	84	0	71	50.90	88	5.26	89	8.86	7	1.78	
4	OK	75	0	113	28.34	122	12.61	121	8.41	3	2.73	

LABEL ENCODING STATE COLUMN

Label encoding is a technique used to convert categorical columns to numerical ones so that the can be fittd by machine learning models which only take numerical variables.

In [26]:
```
le = LabelEncoder()
le.fit(new_df2['state'])
new_df2['state'] = le.transform(new_df2['state'])
new_df2.head()
```

Out[26]:

	state	account length	number vmail messages	total day calls	total day charge	total eve calls	total eve charge	total night calls	total night charge	total intl calls	total intl charge	custom servic cal
0	16	128	25	110	45.07	99	16.78	91	11.01	3	2.70	
1	35	107	26	123	27.47	103	16.62	103	11.45	3	3.70	
2	31	137	0	114	41.38	110	10.30	104	7.32	5	3.29	
3	35	84	0	71	50.90	88	5.26	89	8.86	7	1.78	
4	36	75	0	113	28.34	122	12.61	121	8.41	3	2.73	

OUTLIER DETECTION AND TREATMENT

In [28]:
```
from scipy import stats
print("Before dropping numerical outliers, length of the dataframe is: ", len(

def drop_numerical_outliers(new_df2, z_thresh=3):


    # Use DataFrame.copy() to avoid modifying the original DataFrame
    new_df2 = new_df2.copy()

    # Apply z-score method to identify outliers
    constrains = new_df2.select_dtypes(include=[np.number]).apply(lambda x: np

    # Drop rows with outliers
    new_df2.drop(new_df2.index[~constrains], inplace=True)

    return new_df2

new_df2 = drop_numerical_outliers(new_df2)
print("After dropping numerical outliers, length of the dataframe is: ", len(n
```

```
Before dropping numerical outliers, length of the dataframe is:  3333
After dropping numerical outliers, length of the dataframe is:  2860
```

SCALING NUMERICAL FEATURE.

In [29]:
```
scaler = MinMaxScaler()

def scaling(columns):
    return scaler.fit_transform(new_df2[columns].values.reshape(-1,1))

for i in new_df2.select_dtypes(include=[np.number]).columns:
    new_df2[i] = scaling(i)
new_df2.head()
```

Out[29]:

	state	account length	number vmail messages	total day calls	total day charge	total eve calls	total eve charge	total night calls	total night charge	total intl calls
0	0.32	0.587963	0.510204	0.576271	0.773956	0.495652	0.490082	0.422414	0.644118	0.2
1	0.70	0.490741	0.530612	0.686441	0.450248	0.530435	0.483858	0.525862	0.676471	0.2
2	0.62	0.629630	0.000000	0.610169	0.706088	0.591304	0.238040	0.534483	0.372794	0.4
6	0.38	0.555556	0.489796	0.389831	0.627184	0.573913	0.989498	0.655172	0.538235	0.6
8	0.36	0.537037	0.000000	0.466102	0.521979	0.330435	1.000000	0.413793	0.548529	0.3

FEATURE SCALING

In [30]:
```
# create the X and Y variables (predict and target values)
y = new_df2['churn']

X = new_df2.drop(['churn'], axis=1)

#split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, rand

print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
print("y_train shape:", y_train.shape)
print("y_test shape:", y_test.shape)
```

```
X_train shape: (2145, 17)
X_test shape: (715, 17)
y_train shape: (2145,)
y_test shape: (715,)
```

SMOTE

SMOTE is a data resampling technique used to address class imbalance by generating synthetic samples for the minority class.In this case our minority is churned.

In [31]:
```
!pip install imbalanced-learn
```

```
Requirement already satisfied: imbalanced-learn in /Users/berylsaoke/anaconda
3/envs/learn-env/lib/python3.8/site-packages (0.12.0)
Requirement already satisfied: numpy>=1.17.3 in /Users/berylsaoke/anaconda3/e
nvs/learn-env/lib/python3.8/site-packages (from imbalanced-learn) (1.18.5)
Requirement already satisfied: scipy>=1.5.0 in /Users/berylsaoke/anaconda3/en
vs/learn-env/lib/python3.8/site-packages (from imbalanced-learn) (1.5.3)
Requirement already satisfied: scikit-learn>=1.0.2 in /Users/berylsaoke/anaco
nda3/envs/learn-env/lib/python3.8/site-packages (from imbalanced-learn) (1.3.
0)
Requirement already satisfied: joblib>=1.1.1 in /Users/berylsaoke/anaconda3/e
nvs/learn-env/lib/python3.8/site-packages (from imbalanced-learn) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /Users/berylsaoke/anac
onda3/envs/learn-env/lib/python3.8/site-packages (from imbalanced-learn) (2.
2.0)
```

In [32]:
```
#smote is used to address class imbalance in machine learning
from imblearn.over_sampling import SMOTE
oversample = SMOTE(k_neighbors=5)
X_smote, y_smote = oversample.fit_resample(X, y)
print(y_smote.value_counts())
```

```
0.0    2546
1.0    2546
Name: churn, dtype: int64
```

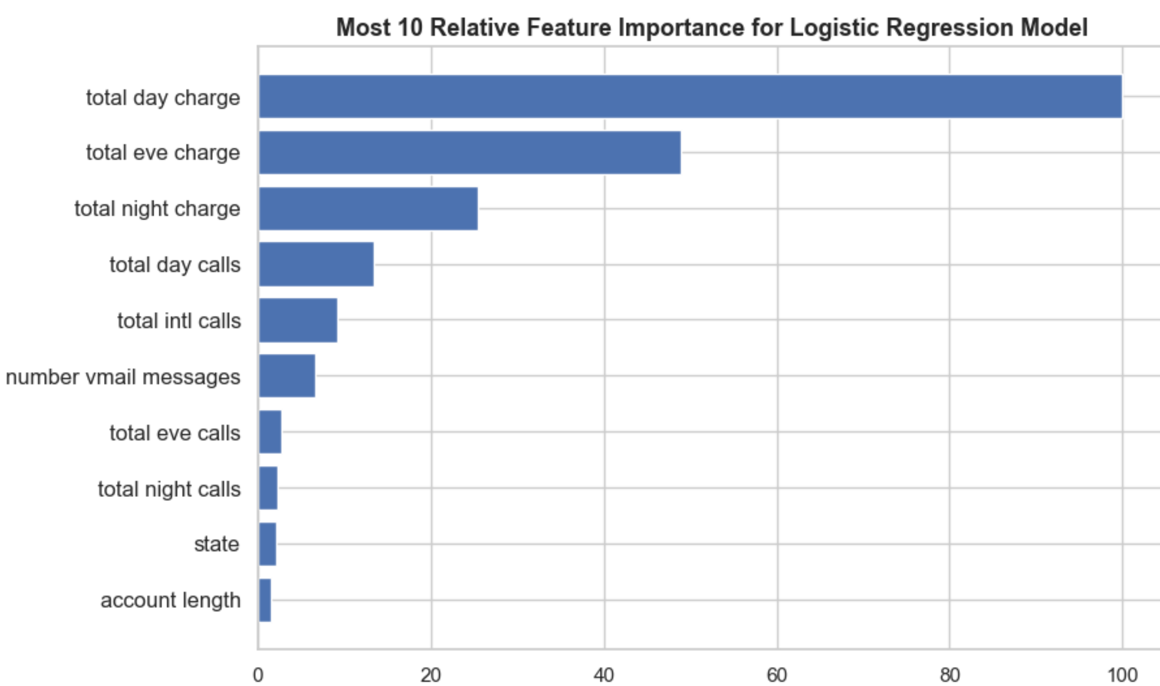
MODELS

LOGISTIC REGRESSION MODEL

In [33]:
```
# Create logistic regression model:
lr = LogisticRegression()

# Train the model:
lr.fit(X_train, y_train)

# Make predictions on the training and testing sets:
y_train_pred = lr.predict(X_train)
y_test_pred = lr.predict(X_test)
```

In [34]:
```
# Feature Importances
feature_importance = abs(lr.coef_[0])
feature_importance = 100.0 * (feature_importance / feature_importance.max())[0
sorted_idx = np.argsort(feature_importance)[0:10]
pos = np.arange(sorted_idx.shape[0]) + .5

featfig = plt.figure(figsize=(9, 6))
featax = featfig.add_subplot(1, 1, 1)
featax.barh(pos, feature_importance[sorted_idx], align='center')
plt.title('Most 10 Relative Feature Importance for Logistic Regression Model',
featax.set_yticks(pos)
featax.set_yticklabels(np.array(X.columns)[sorted_idx], fontsize=12)
```

Out[34]:
```
[Text(0, 0.5, 'account length'),
 Text(0, 1.5, 'state'),
 Text(0, 2.5, 'total night calls'),
 Text(0, 3.5, 'total eve calls'),
 Text(0, 4.5, 'number vmail messages'),
 Text(0, 5.5, 'total intl calls'),
 Text(0, 6.5, 'total day calls'),
 Text(0, 7.5, 'total night charge'),
 Text(0, 8.5, 'total eve charge'),
 Text(0, 9.5, 'total day charge')]
```



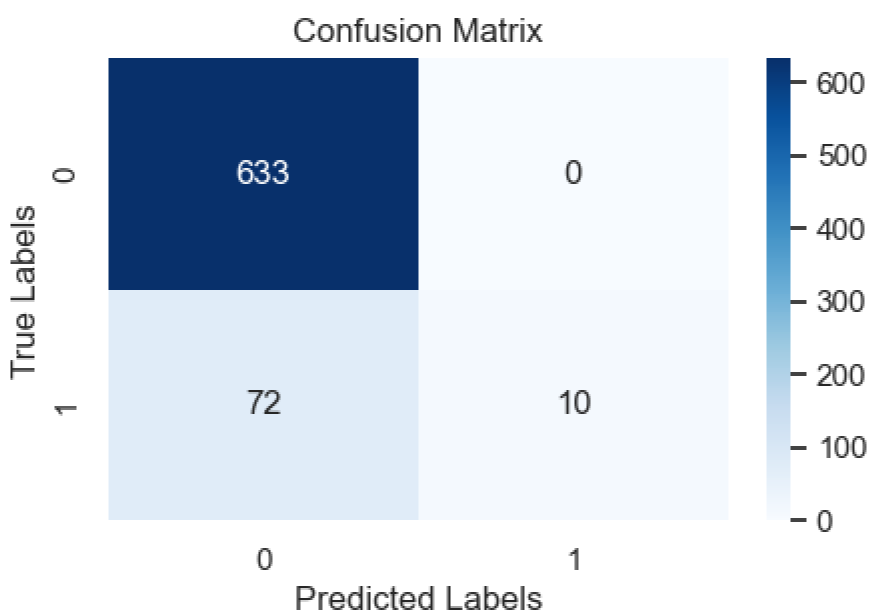
In [35]:
```
print(classification_report(y_test, y_test_pred, target_names=['0', '1']))
```

```
              precision    recall  f1-score   support

           0       0.90      1.00      0.95       633
           1       1.00      0.12      0.22        82

    accuracy                           0.90       715
   macro avg       0.95      0.56      0.58       715
weighted avg       0.91      0.90      0.86       715
```

In [36]:
```
print('Accuracy score for testing set: ',round(accuracy_score(y_test,y_test_pr
print('F1 score for testing set: ',round(f1_score(y_test,y_test_pred),5))
print('Recall score for testing set: ',round(recall_score(y_test,y_test_pred),
print('Precision score for testing set: ',round(precision_score(y_test,y_test_
cm_lr = confusion_matrix(y_test, y_test_pred)
f, ax= plt.subplots(1,1,figsize=(5,3))
sns.heatmap(cm_lr, annot=True, cmap='Blues', fmt='g', ax=ax)
ax.set_xlabel('Predicted Labels'); ax.set_ylabel('True Labels') ; ax.set_title
ax.xaxis.set_ticklabels(['0', '1']) ; ax.yaxis.set_ticklabels(['0', '1'])
```

```
Accuracy score for testing set:  0.8993
F1 score for testing set:  0.21739
Recall score for testing set:  0.12195
Precision score for testing set:  1.0
```

Out[36]:
```
[Text(0, 0.5, '0'), Text(0, 1.5, '1')]
```



Accuracy Score (0.8993):An accuracy of 0.8993 suggests that the model correctly predicted the churn status of approximately 89.9% of the customers in the testing set. F1 Score (0.21739):A low F1 score of 0.21739 indicates that the model's accuracy in predicting both churn and non-churn customers is relatively poor. Recall Score (0.12195):A recall score of 0.12195 means that the model correctly identified only about 12.2% of the churn cases in the testing set. Precision

Score (1.0):A precision score of 1.0 suggests that when the model predicts a customer will churn, it is always correct. However, this high precision score seems unusual and may indicate

MODEL 2; DECISION TREE MODEL

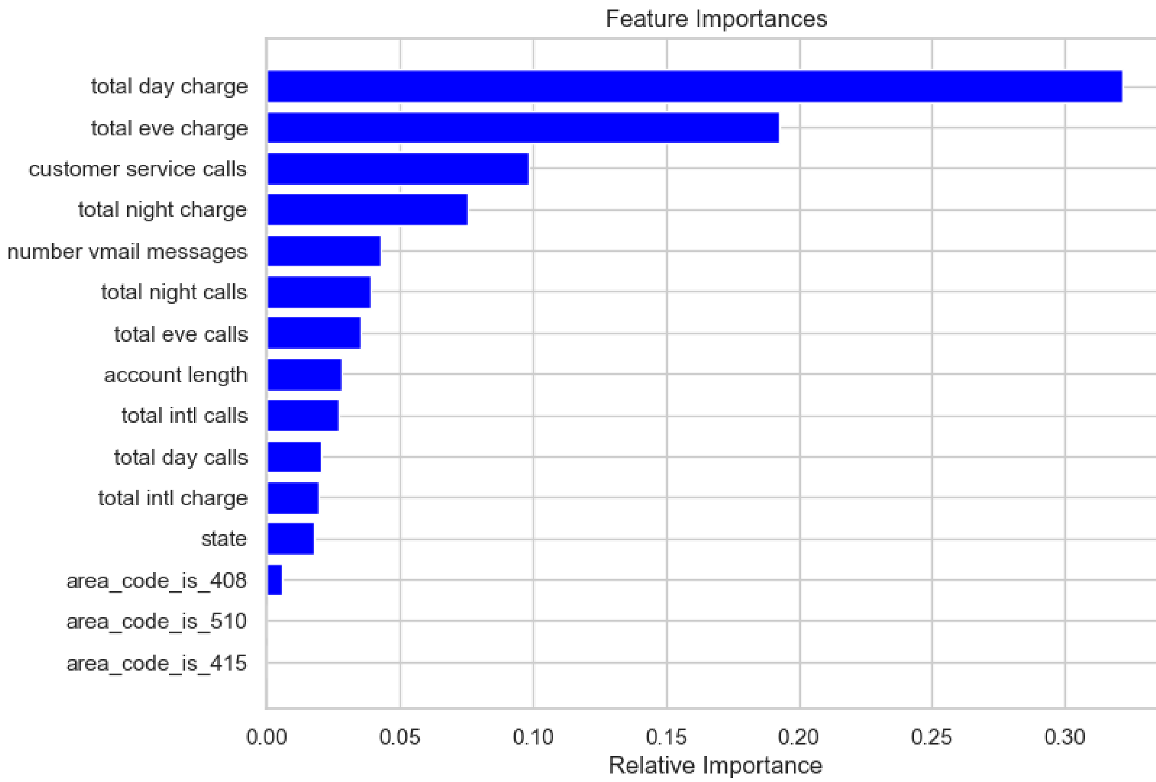
In [37]:
```
# Create logistic regression model:
dt = DecisionTreeClassifier()

# Train the model:
dt.fit(X_train, y_train)

# Make predictions on the training and testing sets:
y_train_pred = dt.predict(X_train)
y_test_pred = dt.predict(X_test)
```

Feature Importance

The chart below shows the top 10 features and their importance levels determined by the hyperparameter tuned Decision tree model. The importance values indicate the relative significance of each feature in predicting the target variable.

Analyzing feature importance helps identify the most influential factors in the model's decision-making process. This information guides feature selection and highlights areas for further investigation or model improvement.

In [38]:
```
feature_names = list(X_train.columns)
importances = dt.feature_importances_[0:15]
indices = np.argsort(importances)

plt.figure(figsize=(8,6))
plt.title('Feature Importances')
plt.barh(range(len(indices)), importances[indices], color='Blue', align='cente
plt.yticks(range(len(indices)), [feature_names[i] for i in indices])
plt.xlabel('Relative Importance')
plt.show()
```



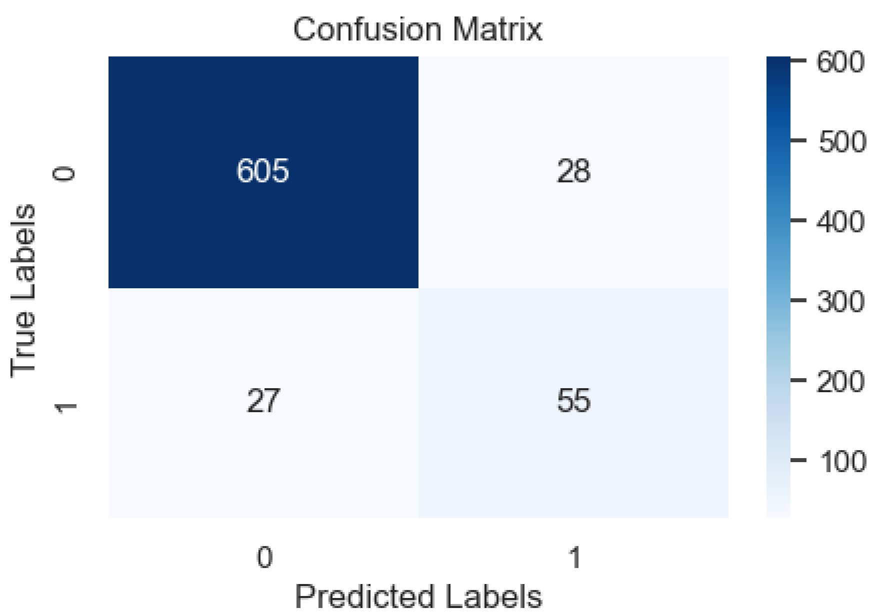
Classification Report

In [39]:
```
print(classification_report(y_test, y_test_pred, target_names=['0', '1']))
```

```
              precision    recall  f1-score   support

           0       0.96      0.96      0.96       633
           1       0.66      0.67      0.67        82

    accuracy                           0.92       715
   macro avg       0.81      0.81      0.81       715
weighted avg       0.92      0.92      0.92       715
```

In [40]:
```
print('Accuracy score for testing set: ',round(accuracy_score(y_test,y_test_pr
print('F1 score for testing set: ',round(f1_score(y_test,y_test_pred),5))
print('Recall score for testing set: ',round(recall_score(y_test,y_test_pred),
print('Precision score for testing set: ',round(precision_score(y_test,y_test_
cm_dt = confusion_matrix(y_test, y_test_pred)
f, ax= plt.subplots(1,1,figsize=(5,3))
sns.heatmap(cm_dt, annot=True, cmap='Blues', fmt='g', ax=ax)
ax.set_xlabel('Predicted Labels'); ax.set_ylabel('True Labels') ; ax.set_title
ax.xaxis.set_ticklabels(['0', '1']) ; ax.yaxis.set_ticklabels(['0', '1'])
plt.show();
```

```
Accuracy score for testing set:  0.92308
F1 score for testing set:  0.66667
Recall score for testing set:  0.67073
Precision score for testing set:  0.66265
```



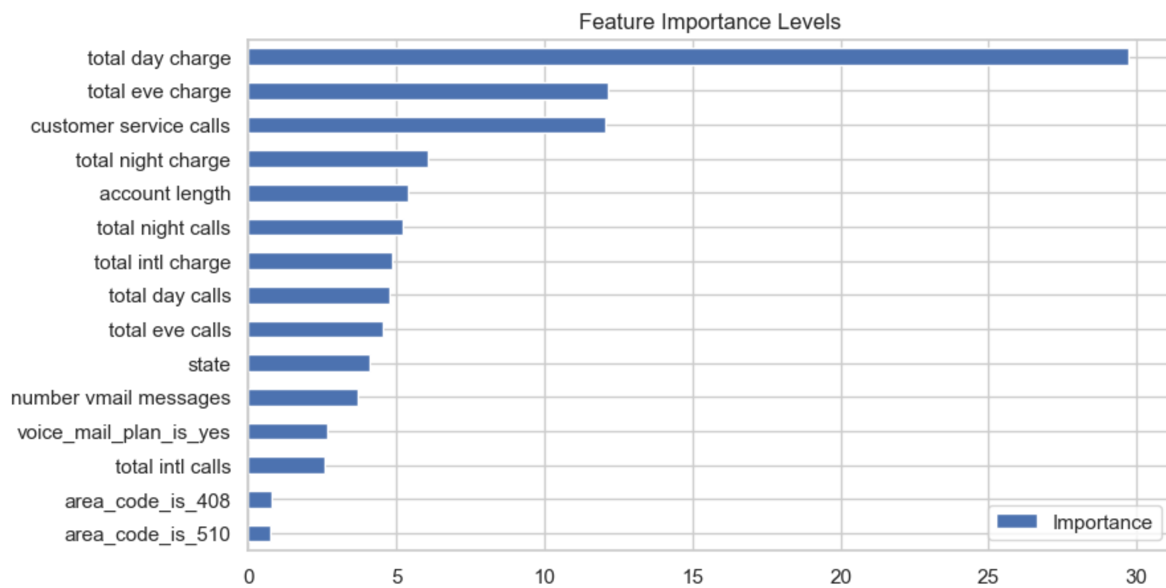
Accuracy Score (0.92308):An accuracy of 0.92308 suggests that the model correctly predicted the churn status of approximately 92.3% of the customers in the testing set. F1 Score (0.68874):A higher F1 score of 0.66667 indicates that the model's accuracy in predicting both churn and non-churn customers is relatively good. Recall Score (0.67073):A recall score of 0.67073 means that the model correctly identified approximately 67.0% of the churn cases in the testing set. Precision Score (0.66265):A precision score of 0.66265 suggests that when the model predicts a customer will churn, it is correct approximately 66.2% of the time.

MODEL 3; RANDOM FOREST CLASSIFIER

In [41]:
```
# Create logistic regression model:
rf = RandomForestClassifier()

# Train the model:
rf.fit(X_train, y_train)

# Make predictions on the training and testing sets:
y_train_pred = rf.predict(X_train)
y_test_pred = rf.predict(X_test)
```

Random Forest Feature Importance

In [42]:
```
Importance =pd.DataFrame({"Importance": rf.feature_importances_*100},index = X
Importance.sort_values(by = "Importance", axis = 0, ascending = True).tail(15)
plt.title("Feature Importance Levels");
plt.show()
```



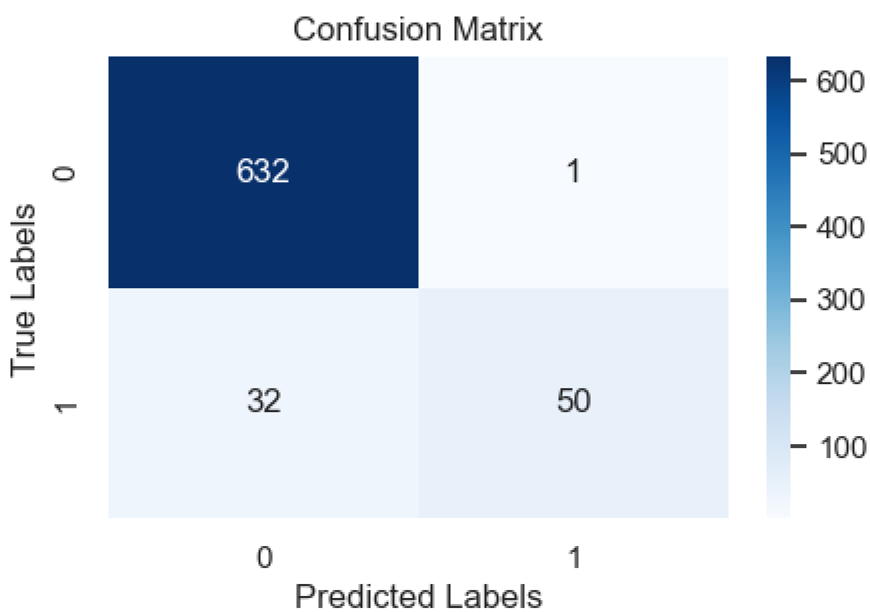
Classification Report

In [43]:
```
print(classification_report(y_test, y_test_pred, target_names=['0', '1']))
```

```
              precision    recall  f1-score   support

           0       0.95      1.00      0.97       633
           1       0.98      0.61      0.75        82

    accuracy                           0.95       715
   macro avg       0.97      0.80      0.86       715
weighted avg       0.96      0.95      0.95       715
```

In [44]:
```
print('Accuracy score for testing set: ',round(accuracy_score(y_test,y_test_pr
print('F1 score for testing set: ',round(f1_score(y_test,y_test_pred),5))
print('Recall score for testing set: ',round(recall_score(y_test,y_test_pred),
print('Precision score for testing set: ',round(precision_score(y_test,y_test_
cm_rf = confusion_matrix(y_test, y_test_pred)
f, ax= plt.subplots(1,1,figsize=(5,3))
sns.heatmap(cm_rf, annot=True, cmap='Blues', fmt='g', ax=ax)
ax.set_xlabel('Predicted Labels'); ax.set_ylabel('True Labels') ; ax.set_title
ax.xaxis.set_ticklabels(['0', '1']) ; ax.yaxis.set_ticklabels(['0', '1'])
plt.show();
```

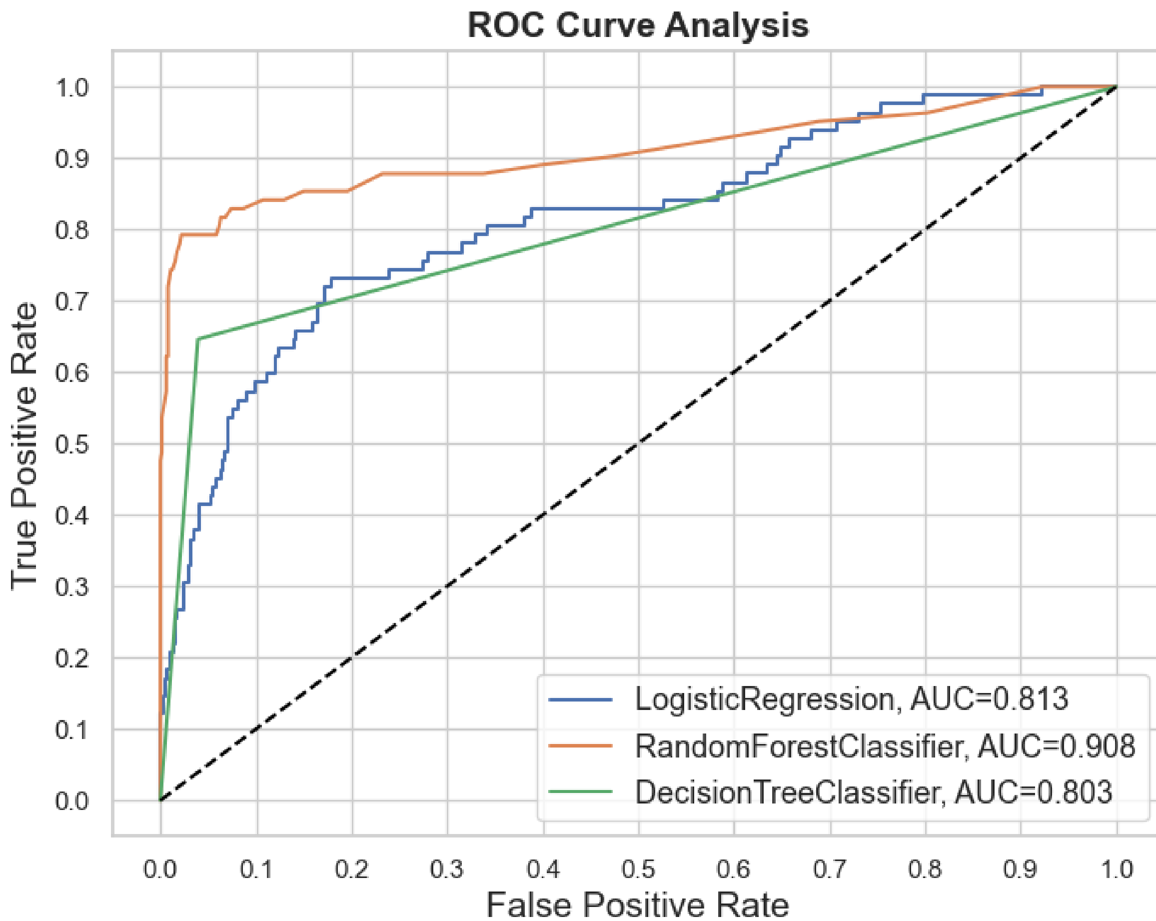
```
Accuracy score for testing set:  0.95385
F1 score for testing set:  0.75188
Recall score for testing set:  0.60976
Precision score for testing set:  0.98039
```



Accuracy Score (0.95385):An accuracy of 0.95385 suggests that the model correctly predicted the churn status of approximately 95.3% of the customers in the testing set. F1 Score (0.75188):A higher F1 score of 0.75188 indicates that the model's accuracy in predicting both churn and non-churn customers is relatively good. Recall Score (0.60976):A recall score of 0.60976 means that the model correctly identified approximately 60.9% of the churn cases in the testing set. Precision Score (0.98039):A precision score of 0.98039 suggests that when the model predicts a customer will churn, it is correct approximately 98.0% of the time.

CLASSIFICATION MODEL COMPARISON

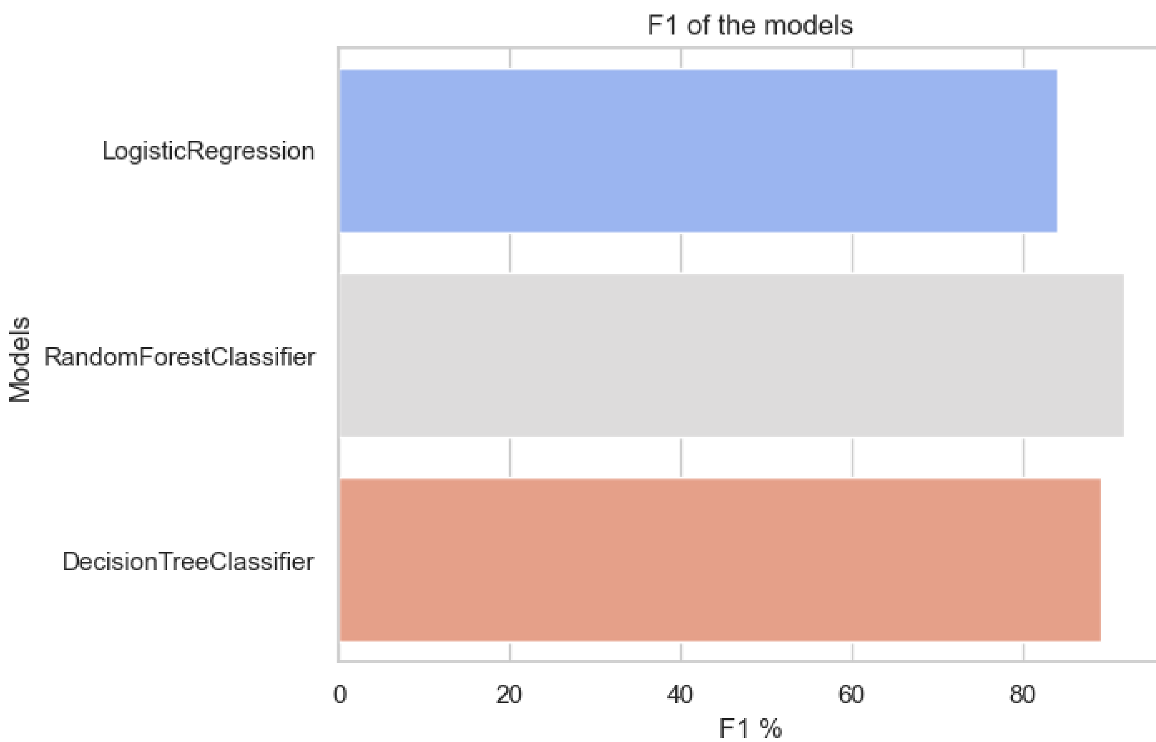
In [45]:
```
classifiers = [LogisticRegression(),
               RandomForestClassifier(),
               DecisionTreeClassifier()]


# Define a result table as a DataFrame
result_table = pd.DataFrame(columns=['classifiers', 'fpr','tpr','auc'])

# Train the models and record the results
for cls in classifiers:
    model = cls.fit(X_train, y_train)
    yproba = model.predict_proba(X_test)[::,1]

    fpr, tpr, _ = roc_curve(y_test,  yproba)
    auc = roc_auc_score(y_test, yproba)

    result_table = result_table.append({'classifiers':cls.__class__.__name__,
                                        'fpr':fpr,
                                        'tpr':tpr,
                                        'auc':auc}, ignore_index=True)

# Set name of the classifiers as index labels
result_table.set_index('classifiers', inplace=True)

fig = plt.figure(figsize=(8,6))

for i in result_table.index:
    plt.plot(result_table.loc[i]['fpr'],
             result_table.loc[i]['tpr'],
             label="{}, AUC={:.3f}".format(i, result_table.loc[i]['auc']))

plt.plot([0,1], [0,1], color='black', linestyle='--')

plt.xticks(np.arange(0.0, 1.1, step=0.1))
plt.xlabel("False Positive Rate", fontsize=15)

plt.yticks(np.arange(0.0, 1.1, step=0.1))
plt.ylabel("True Positive Rate", fontsize=15)

plt.title('ROC Curve Analysis', fontweight='bold', fontsize=15)
plt.legend(prop={'size':13}, loc='lower right')

plt.show()
```



The ROC curve is a plot of the true positive rate against the false positive rate of our classifier. The best performing models will have a curve that hugs the upper left of the graph, which is the the random forest classifier in this case.

MODEL COMPARISON (F1 SCORE)

In [46]:
```
models = [lr,rf,dt]

result = []
results = pd.DataFrame(columns= ["Models","F1"])

for model in models:
    names = model.__class__.__name__
    y_pred = model.predict(X_test)
    f1 = cross_val_score(model,X_test,y_test,cv=10,scoring="f1_weighted").mean
    result = pd.DataFrame([[names, f1*100]], columns= ["Models","F1"])
    results = results.append(result)

sns.barplot(x= 'F1', y = 'Models', data=results, palette="coolwarm")
plt.xlabel('F1 %')
plt.title('F1 of the models');
```



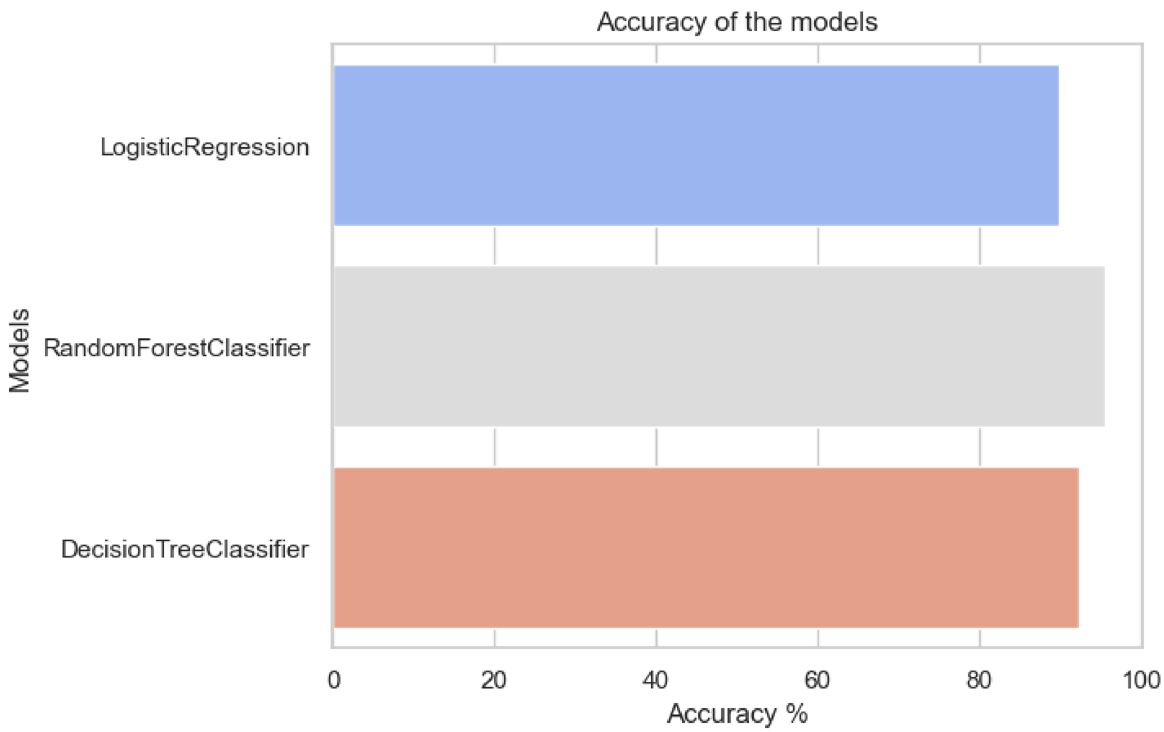
In [47]:
```
results.sort_values(by="F1",ascending=False)
```

Out[47]:

	Models	F1
0	RandomForestClassifier	91.785362
0	DecisionTreeClassifier	89.133241
0	LogisticRegression	84.158830

MODEL COMPARISON (ACCURACY)

In [48]:
```
models = [lr,rf,dt]
result = []
results = pd.DataFrame(columns= ["Models","Accuracy"])

for model in models:
    names = model.__class__.__name__
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    result = pd.DataFrame([[names, accuracy*100]], columns= ["Models","Accurac
    results = results.append(result)


sns.barplot(x= 'Accuracy', y = 'Models', data=results, palette="coolwarm")
plt.xlabel('Accuracy %')
plt.title('Accuracy of the models');
```



In [49]:
```
results.sort_values(by="Accuracy",ascending=False)
```

Out[49]:

	Models	Accuracy
0	RandomForestClassifier	95.384615
0	DecisionTreeClassifier	92.307692
0	LogisticRegression	89.930070

We are searching for a model that can predict with high accuracy and precision random forest classifier fits those requirements

MODEL HYPERPARAMETER TUNING RANDOM FOREST CLASSIFIER

In [50]:
```
# grid search to find the best hyperparameter combination for the model.

rf_params = {"max_depth": [8,15,20],
             "n_estimators":[500,1000],
             "min_samples_split":[5,10,15],
             "criterion":['entropy','gini']}
```

In [51]:
```
rf_model2 = RandomForestClassifier()
rf_cv_model = GridSearchCV(rf_model2,rf_params,cv=3,n_jobs=-1,verbose=False)
rf_cv_model.fit(X_train,y_train)
print("Best parameters:"+str(rf_cv_model.best_params_))
```

```
Best parameters:{'criterion': 'entropy', 'max_depth': 20, 'min_samples_spli
t': 10, 'n_estimators': 500}
```

In [52]:
```
rf_model_GridSearchCV_Applied = RandomForestClassifier(criterion='gini', max_d
rf_model_GridSearchCV_Applied.fit(X_train, y_train)
y_pred_rf_GridSearchCV_Applied = rf_model_GridSearchCV_Applied.predict(X_test)
```

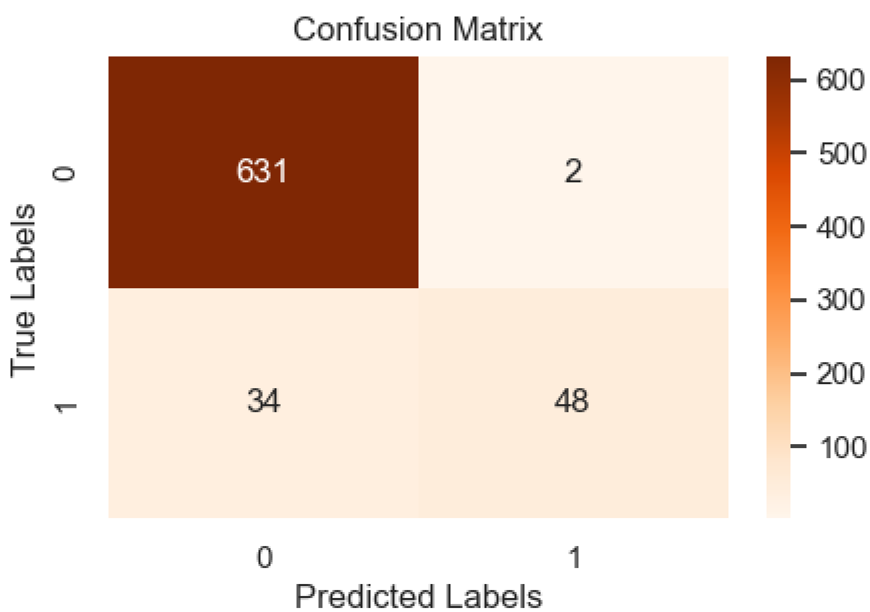
Classification Report(Hyperparameter Tuned Random Forest Model)

In [53]:
```
print(classification_report(y_test, y_pred_rf_GridSearchCV_Applied))
```

```
              precision    recall  f1-score   support

         0.0       0.95      1.00      0.97       633
         1.0       0.96      0.59      0.73        82

    accuracy                           0.95       715
   macro avg       0.95      0.79      0.85       715
weighted avg       0.95      0.95      0.94       715
```

In [54]:
```
print('Accuracy: ',round(accuracy_score(y_test, y_pred_rf_GridSearchCV_Applied
print('F1 score: ',round(f1_score(y_test, y_pred_rf_GridSearchCV_Applied),5))
print('Recall: ',round(recall_score(y_test, y_pred_rf_GridSearchCV_Applied),5)
print('Precision: ',round(precision_score(y_test, y_pred_rf_GridSearchCV_Appli
cm_rf = confusion_matrix(y_test, y_pred_rf_GridSearchCV_Applied)
f, ax= plt.subplots(1,1,figsize=(5,3))
sns.heatmap(cm_rf, annot=True, cmap='Oranges', fmt='g', ax=ax);
ax.set_xlabel('Predicted Labels'); ax.set_ylabel('True Labels') ; ax.set_title
ax.xaxis.set_ticklabels(['0', '1']) ; ax.yaxis.set_ticklabels(['0', '1'])
plt.show();
```

```
Accuracy:  0.94965
F1 score:  0.72727
Recall:  0.58537
Precision:  0.96
```



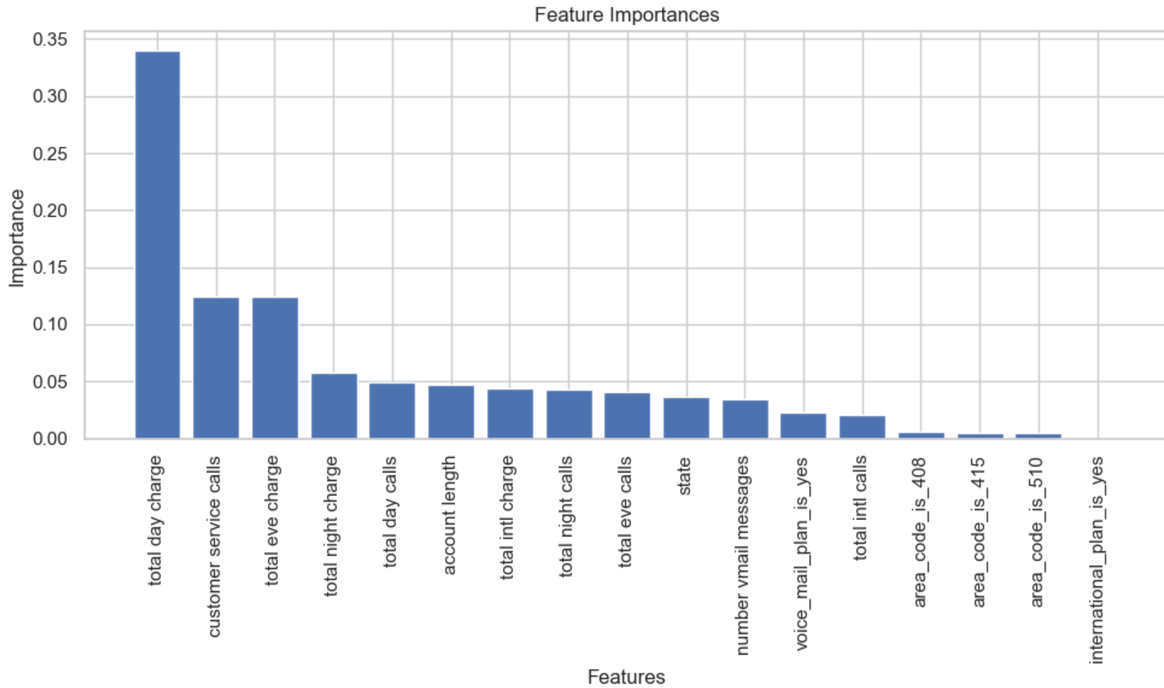
Accuracy (0.94965):An accuracy of 0.94965 suggests that the model correctly predicted the churn status of approximately 94.1% of the customers in the dataset. This indicates a high overall correctness of the model's predictions.A F1 score of 0.72727 indicates a balanced evaluation of the model's accuracy in predicting both churn and non-churn customers.A recall of 0.58537 means that the model correctly identified approximately 58.5% of the churn cases.A precision of 0.96 suggests that when the model predicts a customer will churn, it is correct approximately 96% of the time.This high precision score indicates that the model is very good at minimizing false positive predictions, which is desirable in many scenarios.

MODEL CONCLUSION;

In summary, the model demonstrates high accuracy and precision, indicating a strong ability to correctly classify churn and non-churn instances. However, there is room for improvement in terms of recall, suggesting that the model could better identify churn cases.

In [55]:
```
# Get feature importances
feature_importances = rf_model_GridSearchCV_Applied.feature_importances_

# Get feature names
feature_names = X_train.columns

# Sort feature importances in descending order
indices = np.argsort(feature_importances)[::-1]

# Print feature importances
print("Feature importances:")
for i in range(len(feature_names)):
    print(f"{feature_names[i]}: {feature_importances[i]}")

# Plot feature importances
plt.figure(figsize=(10, 6))
plt.title("Feature Importances")
plt.bar(range(X_train.shape[1]), feature_importances[indices], align="center")
plt.xticks(range(X_train.shape[1]), feature_names[indices], rotation=90)
plt.xlabel("Features")
plt.ylabel("Importance")
plt.tight_layout()
plt.show()
```

```
Feature importances:
state: 0.03622444242586477
account length: 0.04691512652961922
number vmail messages: 0.03428854146749858
total day calls: 0.048865484932782886
total day charge: 0.33937082615662745
total eve calls: 0.041327456271070324
total eve charge: 0.12378835612932615
total night calls: 0.043089864725316986
total night charge: 0.05817424700212419
total intl calls: 0.02070203931389702
total intl charge: 0.044091141980136125
customer service calls: 0.12422321731705803
area_code_is_408: 0.005571421240019709
area_code_is_415: 0.005312578062902207
area_code_is_510: 0.005003131472320722
international_plan_is_yes: 0.0
voice_mail_plan_is_yes: 0.023052124973435742
```



FINDINGS:

Based on the feature importances, the most influential factors contributing to customer churn are: . Total day charge . Customer service calls . Total eve charge These factors suggest that high charges for daytime and evening usage, as well as increased customer service interactions, are associated with higher churn rates.

As seen on the feature importance location can influence the likelihood of customer churn for example,Urban customers might value convenience, fast-paced service, and access to a wide range of products or services. On the other hand, rural customers might prioritize personalized service, community engagement, and affordability.

Based on the provided feature importances, specific contract terms or pricing plans are not directly identified as contributors to customer churn.However, certain features indirectly related to pricing or service plans, such as "total day charge," "total eve charge," and "customer service calls," are significant predictors of churn.

After modeling Random Forest Classifier was identified as the best model that best predicts likelihood of a customer to churn with a percentage 96% precision rate.

RECOMMENDATIONS

1.Focus on Total Day Charge and Total Evening Charge:

Features such as "total day charge" and "total eve charge" have significant importance in predicting churn. The company should analyze pricing strategies for daytime and evening usage, ensuring they are competitive and aligned with customer expectations. Consider offering customizable plans or incentives to reduce charges during peak hours.

2.Improve Customer Service Quality:

"Customer service calls" emerged as a crucial predictor of churn. Enhance customer service quality by investing in training, technology, and support resources. Proactively address customer issues and complaints to minimize the need for repeated service calls, ultimately improving customer satisfaction and retention.

3.Encourage Voice Mail Plan Adoption:

While "voice_mail_plan_is_yes" has moderate importance, it still contributes to predicting churn. Develop targeted marketing campaigns to promote voice mail plan adoption among customers. Highlight the benefits of voice mail services, such as message storage and accessibility, to increase their perceived value and encourage uptake.

4.Optimize International Calling Services:

Features related to international calling, such as "total intl calls" and "total intl charge," exhibit some importance. Review international calling rates, explore partnerships with global carriers, and introduce cost-effective international calling plans or bundles to attract and retain customers who frequently make international calls.

5.Address Area Code Specific Concerns:

While area code features have relatively low importance, they still contribute to predicting churn. Conduct targeted surveys or customer outreach to identify any area-specific issues or preferences. Tailor marketing strategies or service offerings to address the unique needs of customers in specific geographic areas, potentially improving customer satisfaction and loyalty.

CONCLUSION

In conclusion, the project provides a solid foundation for the telecommunications company to develop and implement data-driven strategies aimed at reducing churn, improving customer satisfaction, and ultimately driving business growth.

NEXT STEPS;

We can improve the model accuracy by establishing a process for continuous monitoring and maintenance of the model once deployed. Regularly evaluate its performance, update it with new data as it becomes available, and refine it based on evolving business needs and customer behaviors.