SyriaTel Customer Churn

Problem Statement

SyriaTel, a telecommunications company, is experiencing customer churn, which impacts its revenue and growth. To address this issue, the company seeks to predict whether a customer will soon stop using its services. By identifying the patterns and factors associated with customer churn, SyriaTel can take proactive measures to retain customers and reduce the churn rate.

This analysis aims to develop a machine learning classifier to predict customer churn based on various features such as customer demographics, usage patterns, and service details. The goal is to provide actionable insights and recommendations to SyriaTel's management, enabling them to implement targeted retention strategies and improve customer satisfaction.

In [16]:
```
import pandas as pd

# Load the dataset
data = pd.read_csv('SyriaTel_Customer_Data.csv')

# Inspect the dataset
data.head()
```

Out[16]:

	state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total day charge	...	total eve calls	total eve charge
0	KS	128	415	382-4657	no	yes	25	265.1	110	45.07	...	99	16.78
1	OH	107	415	371-7191	no	yes	26	161.6	123	27.47	...	103	16.62
2	NJ	137	415	358-1921	no	no	0	243.4	114	41.38	...	110	10.30
3	OH	84	408	375-9999	yes	no	0	299.4	71	50.90	...	88	5.26
4	OK	75	415	330-6626	yes	no	0	166.7	113	28.34	...	122	12.61

5 rows × 21 columns

In [17]:
```
data.isnull().sum()
```

Out[17]:
```
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
```

In [18]:
```
data.dtypes
```

Out[18]:
```
state                     object
account length             int64
area code                  int64
phone number              object
international plan        object
voice mail plan           object
number vmail messages      int64
total day minutes        float64
total day calls            int64
total day charge         float64
total eve minutes        float64
total eve calls            int64
total eve charge         float64
total night minutes      float64
total night calls          int64
total night charge       float64
total intl minutes       float64
total intl calls           int64
total intl charge        float64
customer service calls     int64
churn                       bool
dtype: object
```

In [19]:
```
data = data.drop('phone number', axis=1)
data
```

Out[19]:

	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	total eve charge
0	KS	128	415	no	yes	25	265.1	110	45.07	197.4	99	16.78
1	OH	107	415	no	yes	26	161.6	123	27.47	195.5	103	16.62
2	NJ	137	415	no	no	0	243.4	114	41.38	121.2	110	10.30
3	OH	84	408	yes	no	0	299.4	71	50.90	61.9	88	5.26
4	OK	75	415	yes	no	0	166.7	113	28.34	148.3	122	12.61
...	...	...	...	...	...	...	...	...	...	...	...	...
3328	AZ	192	415	no	yes	36	156.2	77	26.55	215.5	126	18.32
3329	WV	68	415	no	no	0	231.1	57	39.29	153.4	55	13.04
3330	RI	28	510	no	no	0	180.8	109	30.74	288.8	58	24.55
3331	CT	184	510	yes	no	0	213.8	105	36.35	159.6	84	13.57
3332	TN	74	415	no	yes	25	234.4	113	39.85	265.9	82	22.60

3333 rows × 20 columns

In [22]:
```
# Identify categorical columns to encode
categorical_columns = ['international plan', 'voice mail plan']

# One-hot encode the specified categorical variables
data_encoded = pd.get_dummies(data, columns=categorical_columns, drop_first=True)
data_encoded.head(10)
```

Out[22]:

	state	account length	area code	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	tota night charge
0	KS	128	415	25	265.1	110	45.07	197.4	99	16.78	244.7	91	11.01
1	OH	107	415	26	161.6	123	27.47	195.5	103	16.62	254.4	103	11.45
2	NJ	137	415	0	243.4	114	41.38	121.2	110	10.30	162.6	104	7.32
3	OH	84	408	0	299.4	71	50.90	61.9	88	5.26	196.9	89	8.86
4	OK	75	415	0	166.7	113	28.34	148.3	122	12.61	186.9	121	8.41
5	AL	118	510	0	223.4	98	37.98	220.6	101	18.75	203.9	118	9.18
6	MA	121	510	24	218.2	88	37.09	348.5	108	29.62	212.6	118	9.57
7	MO	147	415	0	157.0	79	26.69	103.1	94	8.76	211.8	96	9.53
8	LA	117	408	0	184.5	97	31.37	351.6	80	29.89	215.8	90	9.71
9	WV	141	415	37	258.6	84	43.96	222.0	111	18.87	326.4	97	14.69

In [23]:
```
# Normalize numerical variables
from sklearn.preprocessing import StandardScaler
numerical_columns = ['account length', 'number vmail messages', 'total day minutes', '
                     'total day charge', 'total eve minutes', 'total eve calls', 'total
                     'total night minutes', 'total night calls', 'total night charge',
                     'total intl calls', 'total intl charge', 'customer service calls']

scaler = StandardScaler()
data_encoded[numerical_columns] = scaler.fit_transform(data_encoded[numerical_columns]
```

In [24]:
```
data_encoded.head()
```

Out[24]:

	state	account length	area code	number vmail messages	total day minutes	total day calls	total day charge	total eve minutes	total eve calls	total eve charge	min
0	KS	0.676489	415	1.234883	1.566767	0.476643	1.567036	-0.070610	-0.055940	-0.070427	0.86
1	OH	0.149065	415	1.307948	-0.333738	1.124503	-0.334013	-0.108080	0.144867	-0.107549	1.05
2	NJ	0.902529	415	-0.591760	1.168304	0.675985	1.168464	-1.573383	0.496279	-1.573900	-0.75
3	OH	-0.428590	408	-0.591760	2.196596	-1.466936	2.196759	-2.742865	-0.608159	-2.743268	-0.07
4	OK	-0.654629	415	-0.591760	-0.240090	0.626149	-0.240041	-1.038932	1.098699	-1.037939	-0.27

In [27]:
```
# Split into features and target
X = data_encoded.drop('churn', axis=1)  # Features
y = data_encoded['churn']  # Target

# Check the shapes of X and y
print("Shape of X:", X.shape)
print("Shape of y:", y.shape)
```

```
Shape of X: (3333, 19)
Shape of y: (3333,)
```

In [28]:
```
# Train-test split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=

# Check the shapes of the train and test sets
print("Train set shape - X:", X_train.shape, " y:", y_train.shape)
print("Test set shape - X:", X_test.shape, " y:", y_test.shape)
```

```
Train set shape - X: (2666, 19)  y: (2666,)
Test set shape - X: (667, 19)  y: (667,)
```

In [29]:
```
# Output prepared data to CSV for modeling
data_encoded.to_csv('Prepared_SyriaTel_Customer_Data.csv', index=False)
```

Exploratory Data Analysis

We'll start by loading the dataset and then explore its structure, summary statistics, distribution of variables, and correlations.

In [31]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Load the prepared dataset
data = pd.read_csv('Prepared_SyriaTel_Customer_Data.csv')

# 2. Understand the Structure
data.shape
data.dtypes
```

Out[31]:
```
state                     object
account length           float64
area code                  int64
number vmail messages    float64
total day minutes        float64
total day calls          float64
total day charge         float64
total eve minutes        float64
total eve calls          float64
total eve charge         float64
total night minutes      float64
total night calls        float64
total night charge       float64
total intl minutes       float64
total intl calls         float64
total intl charge        float64
customer service calls   float64
churn                       bool
international plan_yes     int64
voice mail plan_yes        int64
dtype: object
```

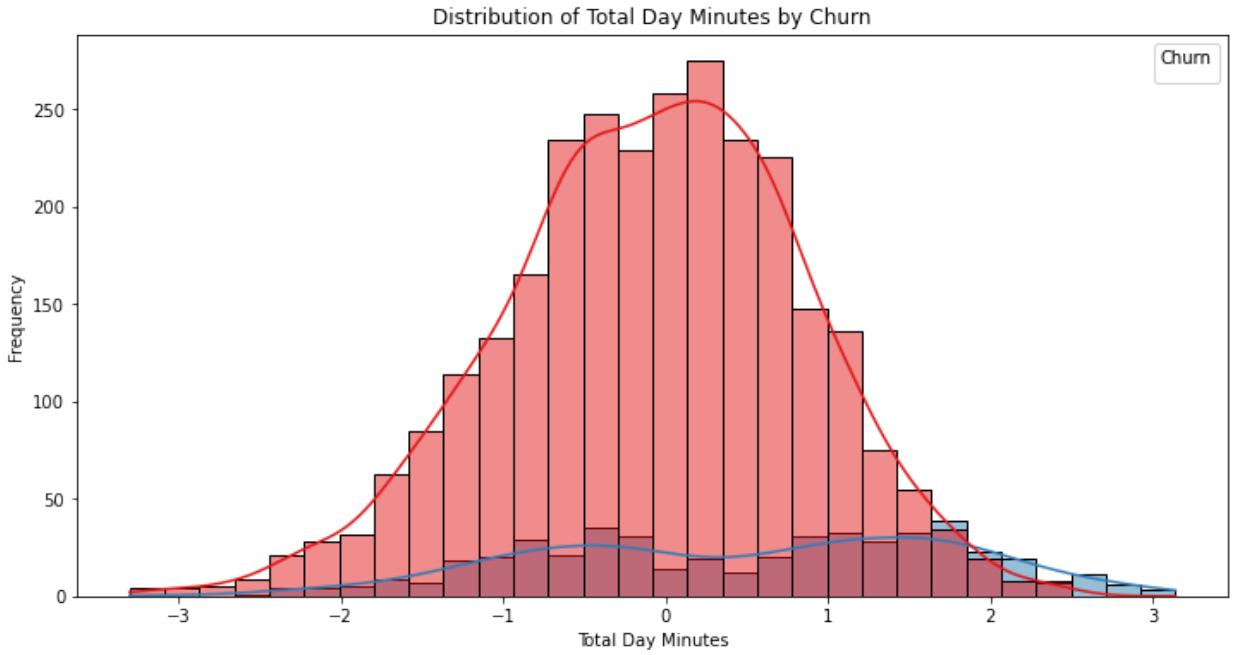
In [32]:
```
# 3. Summary Statistics
data.describe()
```

Out[32]:

total eve calls	total eve charge	total night minutes	total night calls	total night charge	total intl minutes	total intl calls	
3.333000e+03	3.333000e+03	3.333000e+03	3.333000e+03	3.333000e+03	3.333000e+03	3333.000000	3.3
3.267047e-16	1.343060e-16	7.461445e-17	-5.542788e-17	-5.116419e-17	-3.421605e-16	0.000000	2.7
1.000150e+00	1.000150e+00	1.000150e+00	1.000150e+00	1.000150e+00	1.000150e+00	1.000150	1.0
-5.025911e+00	-3.963679e+00	-3.513648e+00	-3.429870e+00	-3.515366e+00	-3.667413e+00	-1.820289	-3.6
-6.583610e-01	-6.783123e-01	-6.698545e-01	-6.699340e-01	-6.676792e-01	-6.223690e-01	-0.601195	-6.1
-5.738630e-03	8.459274e-03	6.485803e-03	-5.505089e-03	4.691242e-03	2.246393e-02	-0.194831	2.0
6.970854e-01	6.766695e-01	6.808485e-01	6.589239e-01	6.814562e-01	6.672969e-01	0.617898	6.7
3.508382e+00	3.207980e+00	3.839081e+00	3.827739e+00	3.836763e+00	3.497397e+00	6.307001	3.4

In [38]:
```
# 4. Univariate Analysis
# Distribution of Total Day Minutes for churned and non-churned customers
plt.figure(figsize=(12, 6))
sns.histplot(data=data, x='total day minutes', hue='churn', bins=30, kde=True, palette=
plt.title('Distribution of Total Day Minutes by Churn')
plt.xlabel('Total Day Minutes')
plt.ylabel('Frequency')
plt.legend(title='Churn', loc='upper right')
plt.show()
```

```
No handles with labels found to put in legend.
```



To explain the distribution of *Total Day Minutes* for churned and non-churned customers, we can analyze the histograms plotted for each group. The histograms show the distribution of total day minutes for both churned and non-churned customers separately.

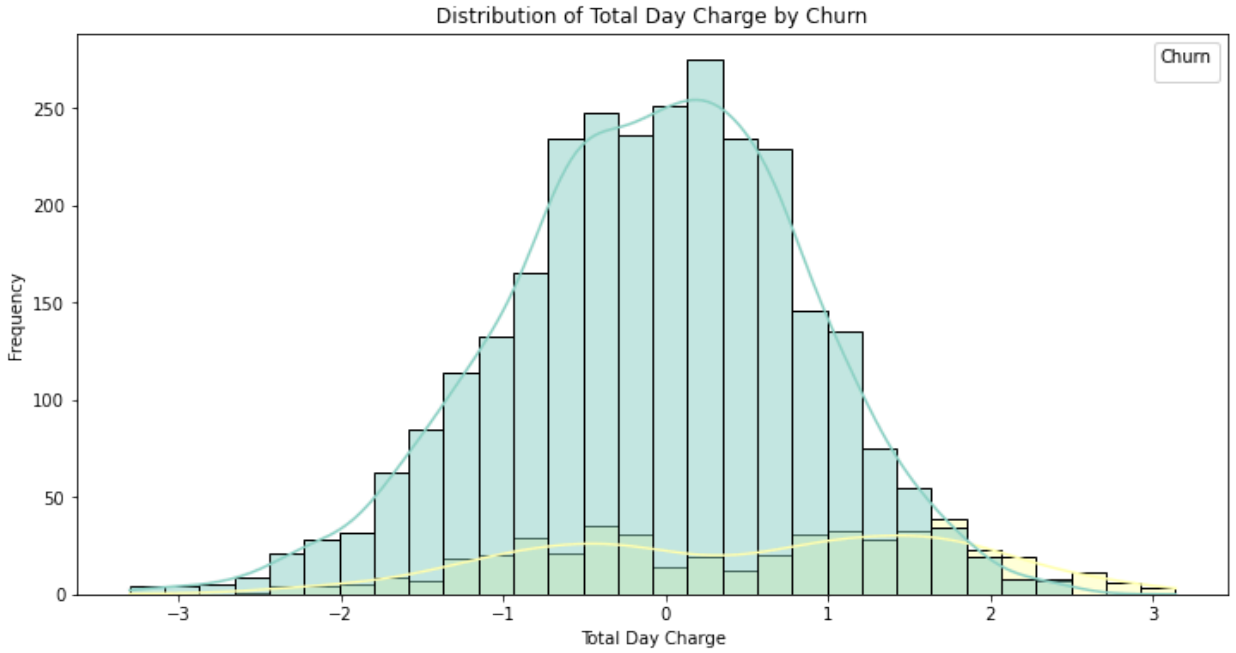
From the histograms, we can observe the following:

For churned customers (indicated by the blue bars), the distribution of total day minutes appears to be skewed towards lower values. This suggests that churned customers tend to have lower total day minutes compared to non-churned customers.

Conversely, for non-churned customers (indicated by the red bars), the distribution of total day minutes appears to be more spread out, with a peak at higher values. This indicates that non-churned customers tend to have higher total day minutes compared to churned customers.

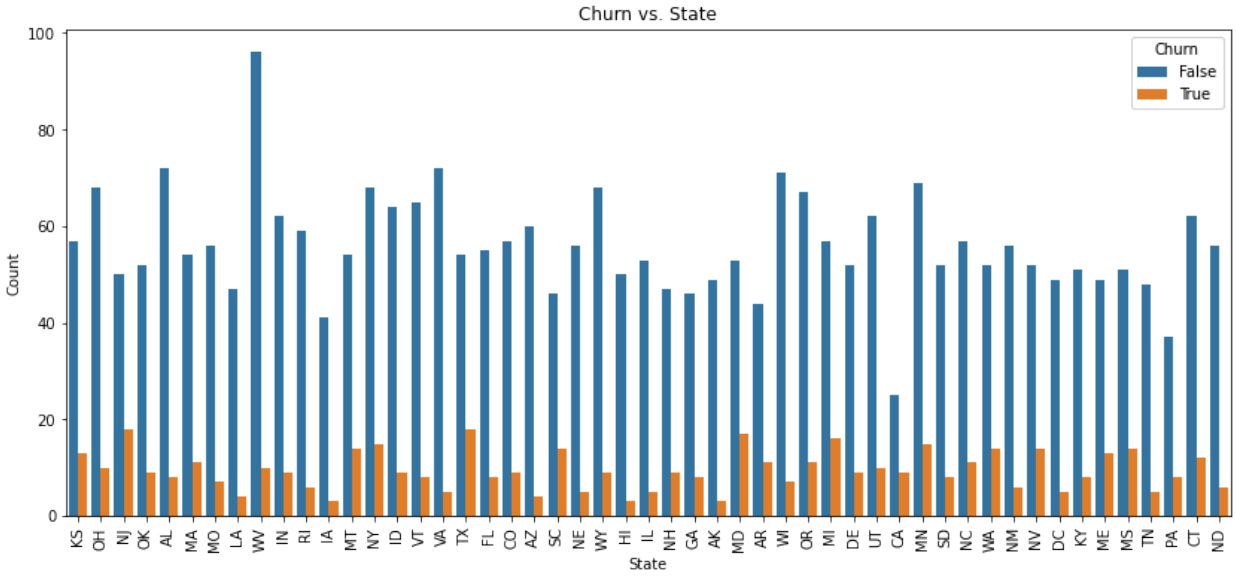
In [37]:
```
# Distribution of Total Day Charge for churned and non-churned customers
plt.figure(figsize=(12, 6))
sns.histplot(data=data, x='total day charge', hue='churn', bins=30, kde=True, palette=
plt.title('Distribution of Total Day Charge by Churn')
plt.xlabel('Total Day Charge')
plt.ylabel('Frequency')
plt.legend(title='Churn', loc='upper right')
plt.show()
```

```
No handles with labels found to put in legend.
```



A similar analysis of **Distribution of Total Day Charge by Churn** displays similar trends in skewness.

In [43]:
```
# 5.Bivariate Analysis
# Churn vs. State/Area Code (Considering State)
plt.figure(figsize=(14, 6))
sns.countplot(data=data, x='state', hue='churn')
plt.title('Churn vs. State')
plt.xlabel('State')
plt.ylabel('Count')
plt.legend(title='Churn', loc='upper right')
plt.xticks(rotation=90)
plt.show()
```



To explain the patterns observed in the histogram showing churn rates across different states, we can analyze the data provided:

1. **State with Higher Churn Rates**: From the histogram, we can identify states where the proportion of churned customers (blue bars) is notably higher compared to non-churned customers (orange bars). This indicates that customers from those states are more likely to churn. For example, states with a higher concentration of blue bars relative to orange bars may have higher churn rates.

2. **State with Lower Churn Rates**: Conversely, states where the proportion of non-churned customers (orange bars) is higher compared to churned customers (blue bars) indicate lower churn rates. In these states, customers are less likely to churn, resulting in a higher proportion of non-churned customers.

3. **Regional Patterns**: It's possible to observe regional patterns where neighboring states exhibit similar churn behavior. This could be due to various factors such as market dynamics, competition, or demographic differences.

4. **Outliers**: Some states may stand out as outliers, either with exceptionally high or low churn rates compared to neighboring states. These outliers could be investigated further to understand the underlying factors contributing to their churn behavior.

Modeling

In [62]:
```
import numpy as np
from sklearn.compose import ColumnTransformer, make_column_selector
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder


# Define numerical and categorical columns
numerical_columns = make_column_selector(dtype_include=np.number)(X_train)
categorical_columns = make_column_selector(dtype_exclude=np.number)(X_train)

# Create preprocessing pipelines for numerical and categorical data
numerical_pipeline = Pipeline([
    ('scaler', StandardScaler())
])

categorical_pipeline = Pipeline([
    ('encoder', OneHotEncoder(handle_unknown='ignore'))
])

# Combine preprocessing pipelines
preprocessor = ColumnTransformer([
    ('num', numerical_pipeline, numerical_columns),
    ('cat', categorical_pipeline, categorical_columns)
])

# Apply preprocessing to the training data
X_train_preprocessed = preprocessor.fit_transform(X_train)
X_test_preprocessed = preprocessor.transform(X_test)

# Train models using preprocessed data
from sklearn.linear_model import LogisticRegression

logistic_regression_model = LogisticRegression(max_iter=1000, random_state=42)
logistic_regression_model.fit(X_train_preprocessed, y_train)

from sklearn.ensemble import RandomForestClassifier


random_forest_model = RandomForestClassifier(n_estimators=100, random_state=42)
random_forest_model.fit(X_train_preprocessed, y_train)

# Make predictions using preprocessed data
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix

logistic_regression_predictions = logistic_regression_model.predict(X_test_preprocesse
random_forest_predictions = random_forest_model.predict(X_test_preprocessed)

# Evaluate model performance
print("Logistic Regression Model Performance:")
print("Accuracy:", accuracy_score(y_test, logistic_regression_predictions))
print("Classification Report:\n", classification_report(y_test, logistic_regression_pre
print("Confusion Matrix:\n", confusion_matrix(y_test, logistic_regression_predictions)

print("\nRandom Forest Model Performance:")
print("Accuracy:", accuracy_score(y_test, random_forest_predictions))
print("Classification Report:\n", classification_report(y_test, random_forest_predictio
```

```
print("Confusion Matrix:\n", confusion_matrix(y_test, random_forest_predictions))
```

```
Logistic Regression Model Performance:
Accuracy: 0.8545727136431784
Classification Report:
              precision    recall  f1-score   support

       False       0.87      0.97      0.92       566
        True       0.56      0.20      0.29       101

    accuracy                           0.85       667
   macro avg       0.71      0.58      0.61       667
weighted avg       0.82      0.85      0.82       667

Confusion Matrix:
 [[550  16]
 [ 81  20]]

Random Forest Model Performance:
Accuracy: 0.9370314842578711
Classification Report:
              precision    recall  f1-score   support

       False       0.93      1.00      0.96       566
        True       1.00      0.58      0.74       101

    accuracy                           0.94       667
   macro avg       0.97      0.79      0.85       667
weighted avg       0.94      0.94      0.93       667

Confusion Matrix:
 [[566   0]
 [ 42  59]]
```

Analysis:

The `logistic regression model` demonstrates decent accuracy but struggles with identifying churned customers, as evidenced by its lower recall and precision scores for the 'True' class.

In contrast, the `random forest model` outperforms `logistic regression` in terms of accuracy and performs significantly better in identifying churned customers, as indicated by its higher recall and precision scores for the 'True' class.

Therefore, the random forest model appears to be more effective in predicting customer churn based on the provided performance metrics. Its higher accuracy and better performance in identifying churned customers make it a preferable choice for the data analysis objective.

Feature Importance

Given what has been stated above, we shall use the `Random Forest Model` to identify the top factors contributing the most to predicting customer churn.

In [66]:
```
# Extract feature names from the preprocessor
numerical_feature_names = preprocessor.transformers_[0][2]
categorical_feature_names = list(preprocessor.named_transformers_['cat']['encoder'].ca
feature_names = numerical_feature_names + categorical_feature_names

# Create a dataframe to store feature importances and feature names
random_forest_feature_importances_df = pd.DataFrame({'Feature': feature_names, 'Importa

# Sort the dataframe by feature importances in descending order
random_forest_feature_importances_df = random_forest_feature_importances_df.sort_value

# Display top contributing factors for random forest
print("\nTop contributing factors for Random Forest:")
print(random_forest_feature_importances_df.head(10))
```

```
Top contributing factors for Random Forest:
                   Feature  Importance
5         total day charge    0.130333
3        total day minutes    0.118165
15  customer service calls    0.104667
16  international plan_yes    0.065503
6        total eve minutes    0.057134
8         total eve charge    0.056102
12      total intl minutes    0.042507
14       total intl charge    0.041763
13        total intl calls    0.040128
11      total night charge    0.038842
```

Recommendations

Model Selection: The `random forest model` outperformed the `logistic regression model` in predicting customer churn, achieving higher accuracy and better performance in identifying churned customers. Therefore, stakeholders should consider adopting the `random forest model` for predicting customer churn in future analyses.

Customer Segmentation: Utilize the insights gained from exploratory data analysis, such as regional patterns in churn rates, to segment customers based on their characteristics and behavior. By understanding the unique needs and preferences of different customer segments, stakeholders can tailor retention strategies more effectively.

Continuous Monitoring: Customer behavior and preferences may evolve over time, so it's essential to continuously monitor churn rates and model performance. Stakeholders should regularly update the predictive model and refine strategies based on new data and insights to ensure they remain effective in reducing churn.

Feedback Loop: Establish a feedback loop to gather input from customers who churned and those who were retained. Understanding the reasons behind customer churn and satisfaction can provide valuable insights for improving products, services, and customer experiences.

Based on the top contributing factors identified by the `Random Forest Model`, here are some **ACTIONABLE INSIGHTS** for stakeholders:

1. **Total Day Charge and Total Day Minutes**: These two factors have the highest importance according to the model. It suggests that the charges incurred during daytime calls and the duration of these calls significantly influence customer churn. Stakeholders should focus on optimizing pricing plans and service quality during peak hours to retain customers.
2. **Customer Service Calls**: The number of customer service calls is another crucial factor contributing to customer churn. This indicates that customers who frequently contact customer service may have unresolved issues or dissatisfaction with the service. Stakeholders should prioritize improving customer service experiences and resolving issues promptly to reduce churn.
3. **International Plan**: Whether a customer has an international plan or not also plays a significant role in predicting churn. This suggests that customers with international calling needs may have specific requirements or expectations that are not being met. Stakeholders should tailor their offerings and communication strategies to better serve customers with international calling needs.
4. **Total Evening Minutes and Total Evening Charge**: Similar to daytime usage, evening call usage and associated charges impact churn. Stakeholders should consider offering competitive pricing and attractive packages for evening usage to enhance customer loyalty during off-peak hours.
5. **Total International Minutes, Total International Calls, and Total International Charge**: These factors reflect customers' usage and charges associated with international calls. Stakeholders should analyze international calling patterns and preferences to tailor international calling plans and improve overall satisfaction among international callers.