Machine Learning in Telecom: Redefining Customer Retention Paradigms

In the fiercely competitive **telecom industry**, understanding and predicting **customer churn** is paramount for sustaining **business growth** and maintaining **market share**. **Customer churn**, defined as the phenomenon where customers discontinue their services with a company, can have significant **financial implications** and can erode **profitability** if not addressed effectively. In this report, we delve into the application of **machine learning techniques** to predict customer churn in the telecom sector. By leveraging **advanced analytics** and **predictive modeling**, **telecom companies** can identify **at-risk customers**, implement **targeted retention strategies**, and ultimately enhance **customer satisfaction** and **loyalty**.

Objective: The primary objective of this analysis is to develop robust **machine learning models** capable of accurately predicting **customer churn** in the **telecom industry**. Additionally, we aim to identify the **key factors** influencing churn behavior, assess the **impact** of churn prediction on **business strategies**, and explore avenues for further **improvement** and **development** in churn prediction methodologies.

Workflow:

- **Data Collection**: Obtain comprehensive datasets containing **customer demographics**, **subscription details**, **usage patterns**, and **churn status**.
- **Data Preprocessing**: Cleanse the data, handle **missing values**, encode **categorical features**, and **normalize** numerical variables to prepare the dataset for analysis.
- **Exploratory Data Analysis (EDA)**: Conduct exploratory data analysis to understand the **distribution** of features, identify **correlations**, and visualize **trends** and **patterns** that may influence churn behavior.
- **Feature Selection**: Utilize techniques such as **recursive feature elimination** to select **relevant features** and reduce dimensionality while preserving predictive power.
- **Model Development**: Train multiple machine learning models, including **Logistic Regression**, **Support Vector Classifier (SVC)**, **Random Forest**, **Decision Tree**, and **Naive Bayes**, on the preprocessed dataset.
- **Model Evaluation**: Evaluate the performance of each model using metrics such as **accuracy**, **precision**, **recall**, and **F1-score** to identify the most effective churn prediction algorithm.
- **Hyperparameter Tuning**: Optimize the parameters of the best-performing model using techniques like **grid search** or **random search** to further enhance predictive accuracy.
- **Final Model Selection**: Select the model with the highest performance for deployment in real-world scenarios.

Description of Machine Learning Models:

- **Logistic Regression**: A linear model that predicts the probability of customer churn based on input features, providing insights into the likelihood of churn for individual customers.
- **Support Vector Classifier (SVC)**: Constructs an optimal hyperplane to separate churn and non-churn customers in a high-dimensional feature space, offering robust classification capabilities.
- **Random Forest**: An ensemble learning technique that combines multiple decision trees to generate accurate predictions by aggregating the outputs of individual trees.
- **Decision Tree**: A tree-like structure that recursively partitions the dataset based on feature thresholds, enabling straightforward interpretation of decision rules.
- **Naive Bayes**: A probabilistic classifier that assumes independence between features and calculates the probability of churn based on Bayes' theorem.

Impact on Business and Markets:

- **Enhanced Customer Retention Strategies**: Accurate churn prediction enables **telecom companies** to proactively address customer concerns, offer **personalized incentives**, and implement targeted retention campaigns to minimize churn rates and maximize **customer lifetime value**.
- **Improved Customer Satisfaction**: By identifying the factors driving churn, companies can improve **service quality**, address pain points, and enhance overall **customer experience**, leading to higher satisfaction levels and increased **loyalty**.
- **Cost Reduction**: Predictive churn analytics help reduce **customer acquisition costs** by focusing resources on retaining existing customers rather than acquiring new ones, resulting in significant **cost savings** and improved **profitability**.
- **Competitive Advantage**: Companies that effectively leverage churn prediction models gain a **competitive edge** by anticipating customer behavior, preemptively addressing churn risks, and maintaining a loyal customer base amidst fierce **market competition**.
- **Market Expansion**: Insightful churn prediction analytics enable telecom companies to identify new **market opportunities**, tailor product offerings to meet evolving customer needs, and expand their market presence while mitigating churn-related risks.

Development Opportunities:

- **Advanced Analytics**: Explore advanced analytics techniques such as **deep learning**, **reinforcement learning**, and **ensemble methods** to further enhance churn prediction accuracy and robustness.
- **Real-Time Monitoring**: Develop real-time monitoring systems that continuously analyze **customer behavior**, detect early warning signs of potential churn, and trigger proactive intervention strategies to prevent customer defection.
- **Personalized Marketing**: Utilize machine learning algorithms to segment customers based on behavior, preferences, and demographics, enabling **targeted marketing campaigns** and **personalized offers** that resonate with individual customers.
- **Customer Lifetime Value Prediction**: Predict the **lifetime value** of customers using predictive modeling techniques to prioritize retention efforts, allocate resources efficiently, and optimize **customer relationship management** strategies.
- **Integration with CRM Systems**: Integrate churn prediction models with existing **customer relationship management (CRM)** systems to streamline decision-making processes, automate retention workflows, and enhance customer engagement and satisfaction.

The application of machine learning techniques for customer churn prediction represents a powerful tool for **telecom companies** to proactively manage churn risks, retain valuable customers, and drive sustainable business growth. By leveraging **predictive analytics**, companies can gain actionable insights into **customer behavior**, implement targeted **retention strategies**, and ultimately foster long-term customer relationships. Moving forward, continued investment in advanced analytics, real-time monitoring capabilities, and personalized marketing initiatives will be crucial for staying ahead in an increasingly competitive **telecom landscape** and delivering superior value to customers.

In [1]:
```
#Import libraries
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
pd.set_option('display.max_columns', None)

import plotly.express as px #for visualization
import matplotlib.pyplot as plt #for visualization

data_df = pd.read_csv("C:/Users/anike/OneDrive/Desktop/Projects/Machine Learning/churn/churn.csv")
data_df
```

Out[1]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	Online
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	
...	...	...	...	...	...	...	...	...	...	...	...
7038	6840-RESVB	Male	0	Yes	Yes	24	Yes	Yes	DSL	Yes	
7039	2234-XADUH	Female	0	Yes	Yes	72	Yes	Yes	Fiber optic	No	
7040	4801-JZAZL	Female	0	Yes	Yes	11	No	No phone service	DSL	Yes	
7041	8361-LTMKD	Male	1	Yes	No	4	Yes	Yes	Fiber optic	No	
7042	3186-AJIEK	Male	0	No	No	66	Yes	No	Fiber optic	Yes	

7043 rows × 21 columns

In [2]:
```
#Get overview of the data
def dataoveriew(df, message):
    print(f'{message}:n')
    print('Number of rows: ', df.shape[0])
    print("nNumber of features:", df.shape[1])
    print("nData Features:")
    print(df.columns.tolist())
    print("nMissing values:", df.isnull().sum().values.sum())
    print("nUnique values:")
    print(df.nunique())
```

```
dataoveriew(data_df, 'Overview of the dataset')
```

```
Overview of the dataset:n
Number of rows:  7043
nNumber of features: 21
nData Features:
['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents', 'tenure', 'PhoneService', 'MultipleLines', 'Interne
tService', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies', 'Con
tract', 'PaperlessBilling', 'PaymentMethod', 'MonthlyCharges', 'TotalCharges', 'Churn']
nMissing values: 0
nUnique values:
customerID          7043
gender                 2
SeniorCitizen          2
Partner                2
Dependents             2
tenure                73
PhoneService           2
MultipleLines          3
InternetService        3
OnlineSecurity         3
OnlineBackup           3
DeviceProtection       3
TechSupport            3
StreamingTV            3
StreamingMovies        3
Contract               3
PaperlessBilling       2
PaymentMethod          4
MonthlyCharges      1585
TotalCharges        6531
Churn                  2
dtype: int64
```

Understanding the Customer Churn Dataset

The provided dataset comprises 7043 rows and 21 columns, representing various aspects of customer information and behavior. Let's delve into the breakdown of features:

Categorical Features:

1. **CustomerID:** A unique identifier assigned to each customer.
2. **gender:** Indicates the gender of the customer (Male or Female).
3. **SeniorCitizen:** Denotes whether the customer is a senior citizen (1 for Yes, 0 for No).
4. **Partner:** Specifies if the customer has a partner (Yes or No).
5. **Dependents:** Indicates if the customer has dependents (Yes or No).
6. **PhoneService:** Specifies whether the customer has a phone service (Yes or No).
7. **MultipleLines:** Indicates if the customer has multiple phone lines (Yes, No, or No phone service).
8. **InternetService:** Specifies the customer's internet service provider (DSL, Fiber optic, or No).
9. **OnlineSecurity:** Indicates if the customer has online security (Yes, No, or No internet service).
10. **OnlineBackup:** Denotes whether the customer has online backup (Yes, No, or No internet service).
11. **DeviceProtection:** Specifies if the customer has device protection (Yes, No, or No internet service).
12. **TechSupport:** Indicates whether the customer has tech support (Yes, No, or No internet service).
13. **StreamingTV:** Specifies if the customer has streaming TV (Yes, No, or No internet service).
14. **StreamingMovies:** Indicates whether the customer has streaming movies (Yes, No, or No internet service).
15. **Contract:** Denotes the contract term of the customer (Month-to-month, One year, or Two years).
16. **PaperlessBilling:** Indicates the customer's preference for paperless billing (Yes or No).
17. **PaymentMethod:** Specifies the customer's payment method (Electronic check, Mailed check, Bank transfer (automatic), Credit card (automatic)).

Numerical Features:

1. **Tenure:** Represents the number of months the customer has stayed with the company.
2. **MonthlyCharges:** Indicates the amount charged to the customer monthly.
3. **TotalCharges:** Denotes the total amount charged to the customer.

Prediction Feature:

- **Churn:** Indicates whether the customer churned or not (Yes or No).

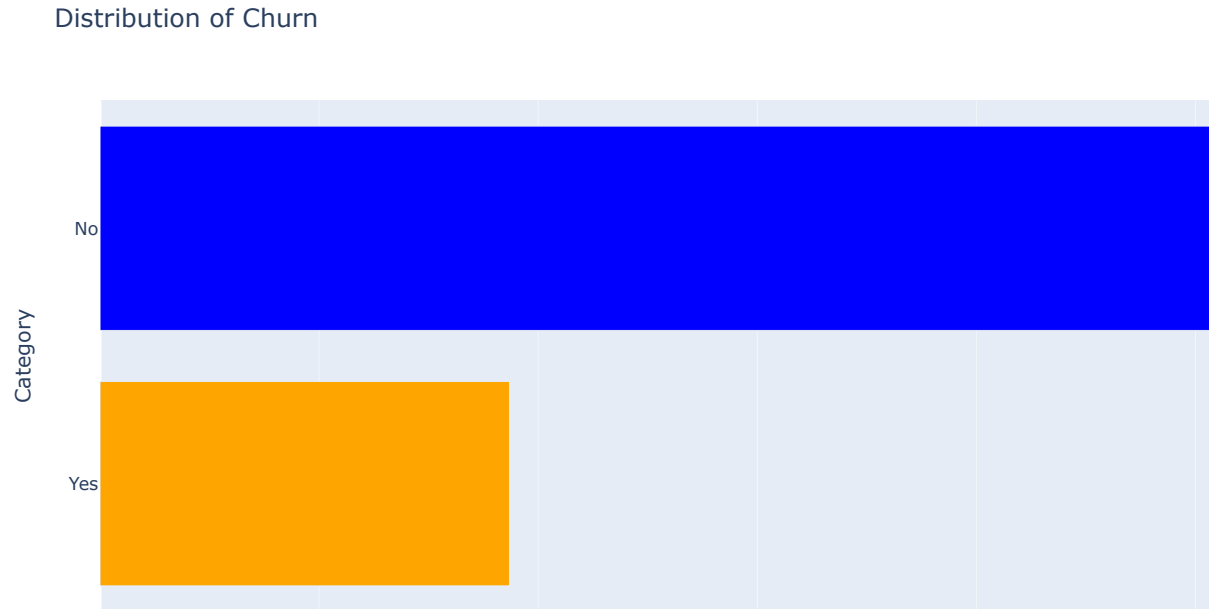
Subdivision of Features:

1. **Demographic Customer Information:** Includes features related to customer demographics such as gender, senior citizen status, partnership, and dependents.

2. **Services Signed Up for by Each Customer:** Encompasses features indicating the services each customer has subscribed to, including phone service, internet service, online security, device protection, and more.
3. **Customer Account Information:** Comprises features related to customer account details such as tenure, contract terms, billing preferences, payment method, and charges.

This comprehensive breakdown of features provides valuable insights into customer behavior and characteristics, laying the foundation for effective churn prediction and targeted retention strategies.

In [3]:
```
import plotly.express as px

# Count instances of each category in the "Churn" column
target_instance = data_df["Churn"].value_counts().to_frame()

# Reset index and rename columns
target_instance = target_instance.reset_index()
target_instance = target_instance.rename(columns={'index': 'Category', 'Churn': 'Count'})

# Create the bar chart
fig = px.bar(target_instance, x='Count', y='Category',
             color='Category', orientation='h',
             color_discrete_sequence=["blue", "orange"],  # Change color code here
             title='Distribution of Churn')
fig.show()
```

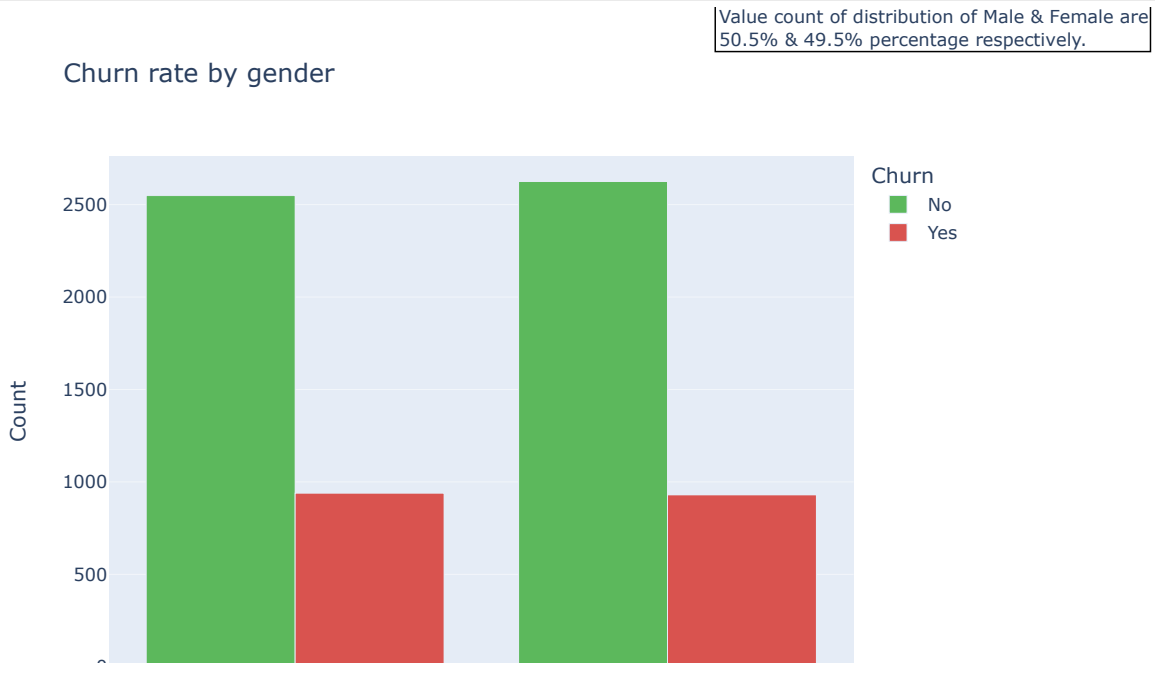


Project revolves around predicting users who terminated their subscriptions with the company in the previous month, constituting a binary classification task. However, the distribution of the target variable, "Churn," indicates a significant class imbalance, with 73.5% of instances labeled as "No" and only 26.5% labeled as "Yes." This insight, obtained from a pie chart analysis, underscores the challenge of effectively modeling churn prediction within an unbalanced dataset.

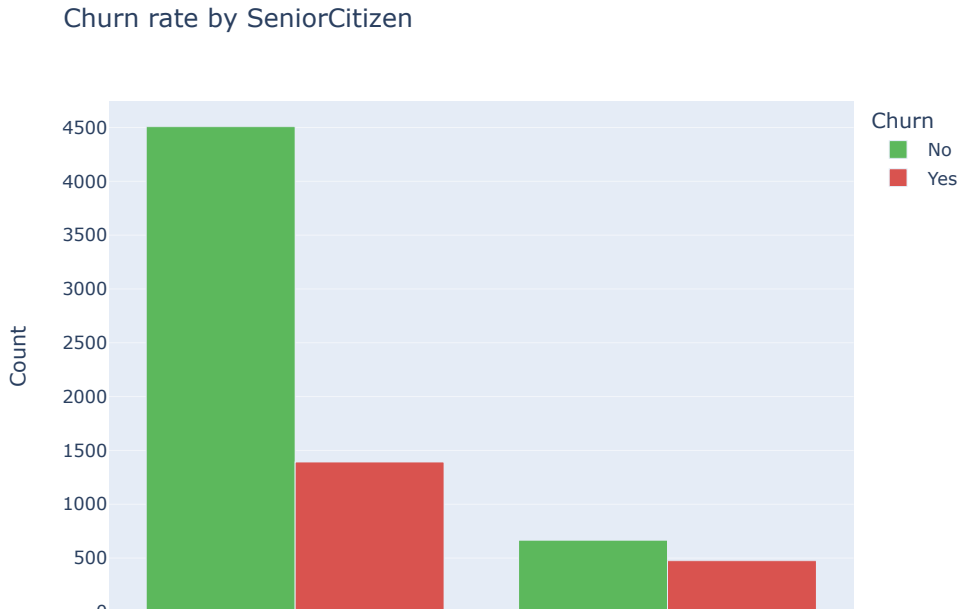
In [4]:
```
import plotly.express as px

def bar(feature, df=data_df):
    # Group by the categorical feature
    temp_df = df.groupby([feature, 'Churn']).size().reset_index().rename(columns={0:'Count'})

    # Calculate the value counts of each distribution and its corresponding percentages
    value_counts_df = df[feature].value_counts().to_frame().reset_index()
    categories = [cat[1][0] for cat in value_counts_df.iterrows()]
    num_list = [num[1][1] for num in value_counts_df.iterrows()]
    div_list = [element / sum(num_list) for element in num_list]
    percentage = [round(element * 100, 1) for element in div_list]

    # Define string formatting for graph annotation
    def num_format(list_instance):
        formatted_str = ''
        for index, num in enumerate(list_instance):
            if index < len(list_instance) - 2:
                formatted_str = formatted_str + f'{num}%, '
```

```
            elif index == len(list_instance) - 2:
                formatted_str = formatted_str + f'{num}% & '
            else:
                formatted_str = formatted_str + f'{num}%'
        return formatted_str

    def str_format(list_instance):
        formatted_str = ''
        for index, cat in enumerate(list_instance):
            if index < len(list_instance) - 2:
                formatted_str = formatted_str + f'{cat}, '
            elif index == len(list_instance) - 2:
                formatted_str = formatted_str + f'{cat} & '
            else:
                formatted_str = formatted_str + f'{cat}'
        return formatted_str

    # Run the formatting functions
    num_str = num_format(percentage)
    cat_str = str_format(categories)

    # Set graph framework
    fig = px.bar(temp_df, x=feature, y='Count', color='Churn',
                 title=f'Churn rate by {feature}', barmode="group",
                 color_discrete_sequence=["#5cb85c", "#d9534f"])  # Change colors here
    fig.add_annotation(
        text=f'Value count of distribution of {cat_str} are<br>{num_str} percentage respectively.',
        align='left',
        showarrow=False,
        xref='paper',
        yref='paper',
        x=1.4,
        y=1.3,
        bordercolor='black',
        borderwidth=1
    )
    fig.update_layout(
        margin=dict(r=400),  # margin space for the annotations on the right
    )

    return fig.show()
```

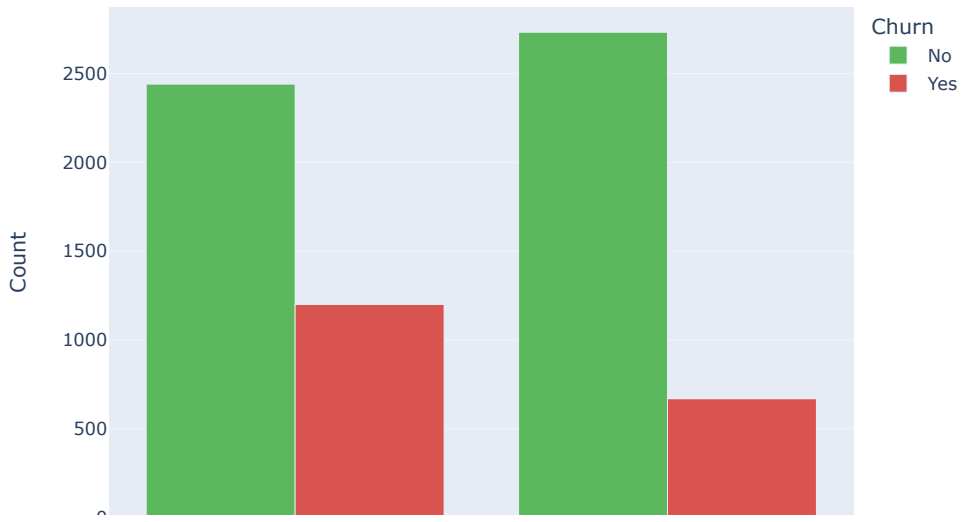
In [5]:
```
#Gender feature plot
bar('gender')
#SeniorCitizen feature plot
bar('SeniorCitizen')
#Partner feature plot
bar('Partner')
#Dependents feature plot
bar('Dependents')
```



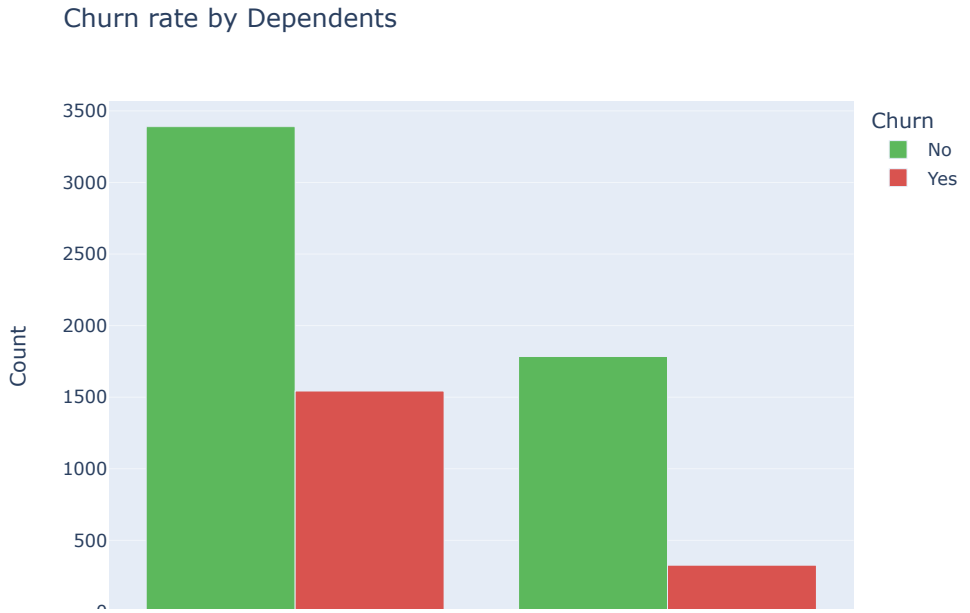
Value count of distribution of 0 & 1 are 83.8% & 16.2% percentage respectively.



Value count of distribution of No & Yes are 51.7% & 48.3% percentage respectively.



Value count of distribution of No & Yes are 70.0% & 30.0% percentage respectively.



Demographic Analysis: Gender and Relationship Status

In this demographic analysis, we delve into the distribution of churn across gender and relationship status among the customer base. Here's a breakdown of the insights gleaned:

Gender Distribution:

- The dataset exhibits an approximately equal distribution between male and female customers.
- Although there's a slight difference in churn rates between genders, with females showing a slightly higher churn rate, the variance is negligible.

Relationship Status:

- Partner status among customers is evenly distributed, with a balanced representation of individuals with and without partners.
- However, customers without partners display a slightly higher churn rate compared to those with partners.
- Similarly, there's a notable proportion of churn among customers without dependents.

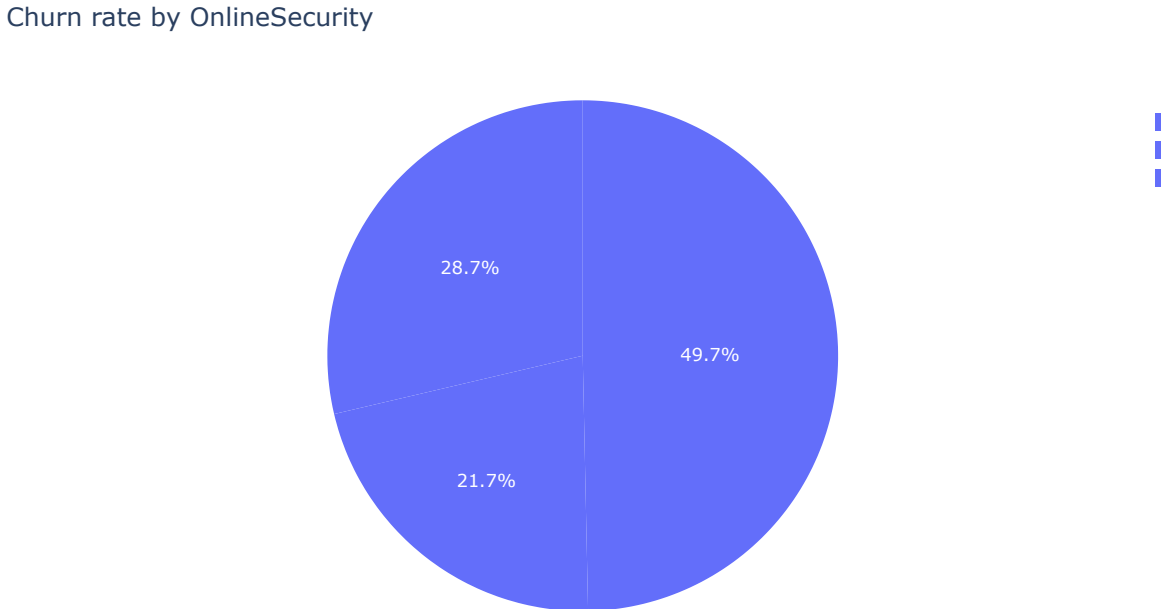
Demographic Insights:

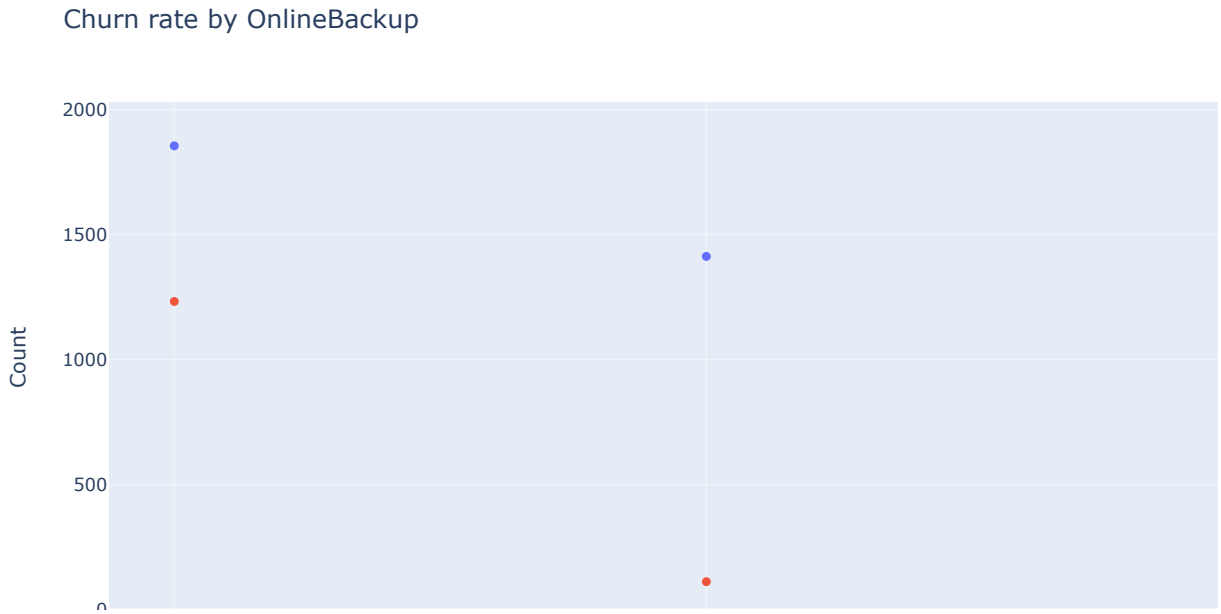
- The analysis reveals a higher churn proportion among younger customers, as indicated by the absence of senior citizen status.
- Furthermore, customers without partners and dependents exhibit a higher propensity to churn.
- While there's a nuanced difference in churn rates across genders and relationship statuses, the overall impact on churn is relatively minimal.
- Notably, younger customers without partners or dependents emerge as a distinct segment displaying higher churn tendencies.

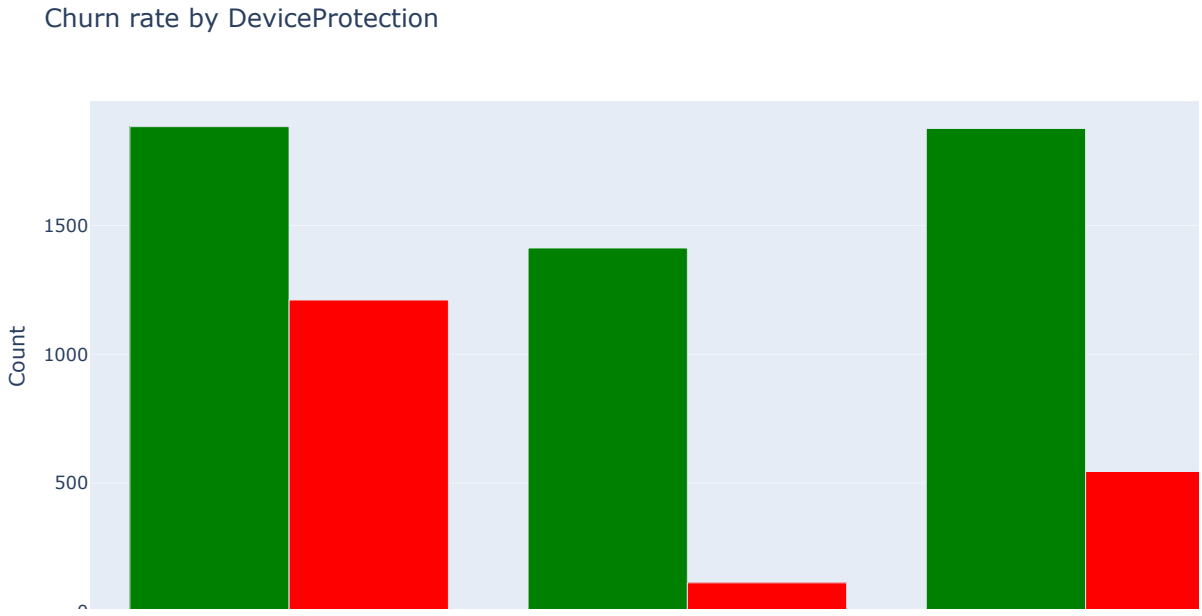
This demographic analysis provides valuable insights into customer behavior, enabling businesses to tailor retention strategies effectively. By targeting specific customer segments, such as younger individuals without partners or dependents, companies can implement targeted interventions to mitigate churn and foster long-term customer loyalty.

In [6]:
```
def bar(feature, plot_type='bar', df=data_df):
    # Groupby the categorical feature
    temp_df = df.groupby([feature, 'Churn']).size().reset_index()
    temp_df = temp_df.rename(columns={0:'Count'})

    # Calculate the value counts of each distribution and its corresponding Percentages
    value_counts_df = df[feature].value_counts().to_frame().reset_index()
    categories = [cat[1][0] for cat in value_counts_df.iterrows()]

    # Running the formatting functions
    # ...

    # Setting graph framework based on plot_type
    if plot_type == 'bar':
        fig = px.bar(temp_df, x=feature, y='Count', color='Churn', title=f'Churn rate by {feature}', barmode="group",
    elif plot_type == 'scatter':
        fig = px.scatter(temp_df, x=feature, y='Count', color='Churn', title=f'Churn rate by {feature}')
```

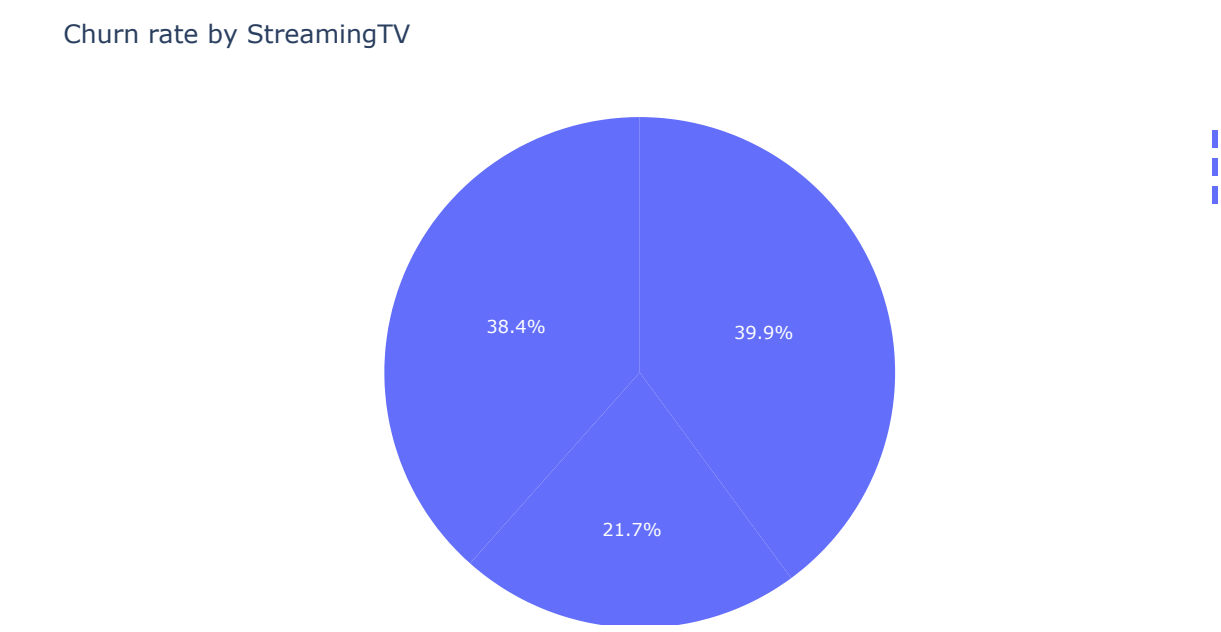
```
    elif plot_type == 'pie':
        fig = px.pie(temp_df, values='Count', names=feature, color='Churn', title=f'Churn rate by {feature}')
    else:
        print("Invalid plot type. Please choose 'bar', 'scatter', or 'pie'.")

    # Adding annotations and updating layout as before

    return fig.show()

# Example usage for each feature
bar('OnlineSecurity', plot_type='pie')
bar('OnlineBackup', plot_type='scatter')
bar('DeviceProtection', plot_type='bar')
bar('TechSupport', plot_type='scatter')
bar('StreamingTV', plot_type='pie')
bar('StreamingMovies', plot_type='bar')
```

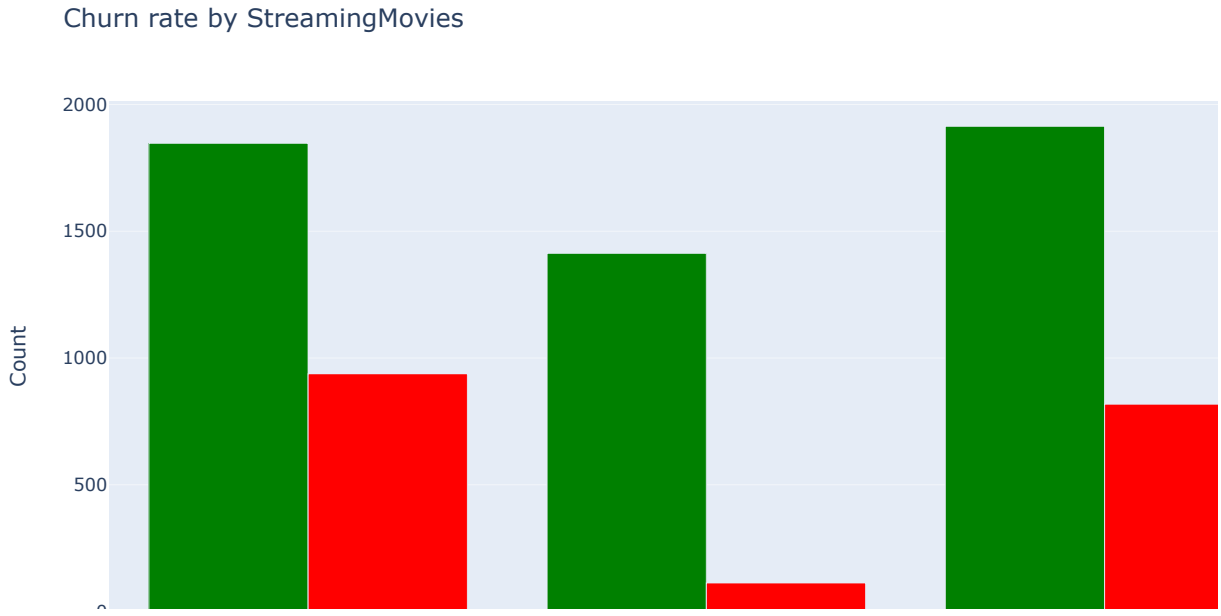





Churn rate by DeviceProtection
Count
1500
1000
500


Churn rate by TechSupport
Count
2000
1500
1000
500
0





Understanding the Impact of Customer Signed-up Services on Churn Prediction

Insightful Analysis of Customer Service Subscriptions and Churn Behavior

Insight:

Exploring the relationship between customer churn and the services they have signed up for reveals nuanced patterns and valuable insights. By dissecting the data on customer subscriptions, we gain a deeper understanding of how different services influence churn behavior.

1. **Phone Service and Multiple Lines:**
   - A noteworthy observation is that customers without phone service cannot subscribe to multiple lines. Moreover, a significant majority, approximately 90.3%, of customers have availed themselves of phone services. Surprisingly, customers with phone services exhibit a higher propensity to churn, despite their widespread adoption.
2. **Type of Internet Service:**

- Customers utilizing fiber optic internet service demonstrate a heightened likelihood of churning compared to those with DSL connections. This disparity in churn rates may stem from various factors, including pricing differentials, competitive pressures, and disparities in service quality. Notably, fiber optic service tends to command higher prices than DSL, potentially contributing to customer attrition.

3. **Online Security, Backup, Device Protection, and Tech Support:**

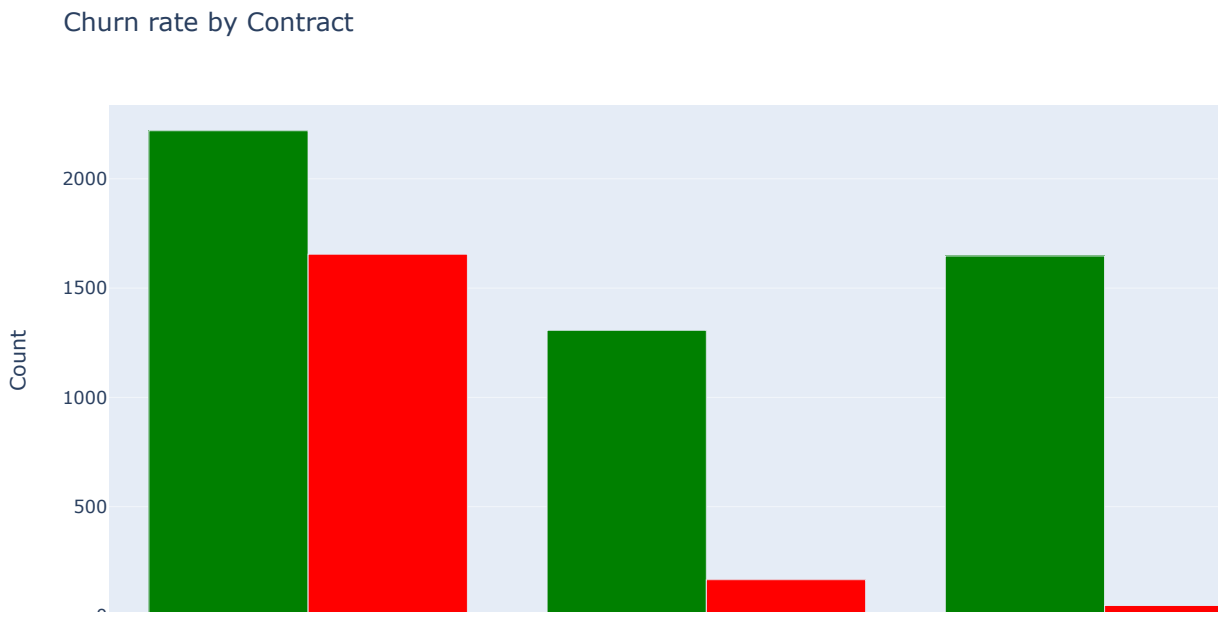
   - Customers who have opted for additional services such as online security, backup, device protection, and tech support display a reduced propensity to churn. This finding suggests that these supplementary services play a pivotal role in fostering customer loyalty and satisfaction. Their presence likely indicates a higher level of perceived value and enhanced customer experience.

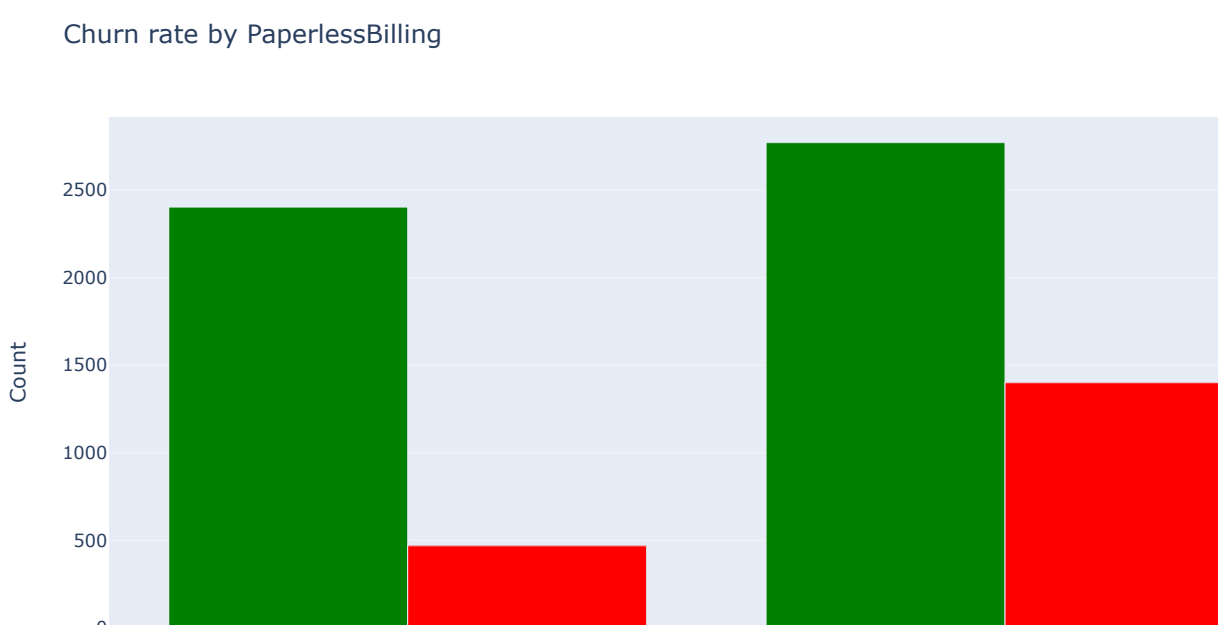
4. **Streaming Services:**

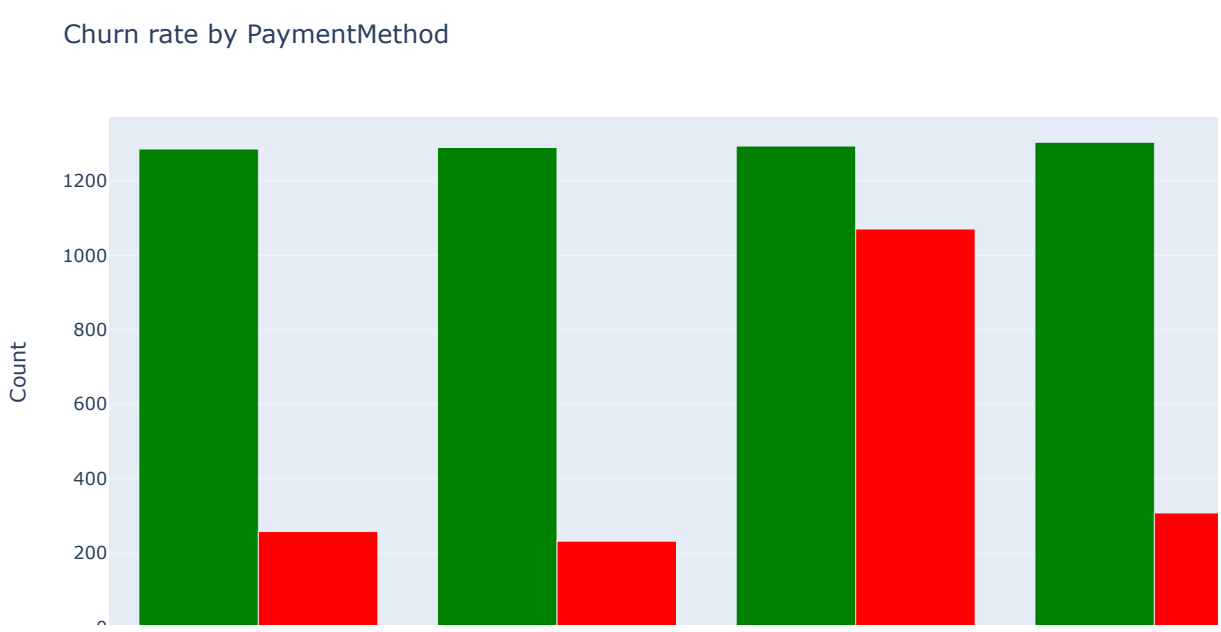
   - Interestingly, the presence or absence of streaming services does not exert a significant influence on churn prediction. This feature is evenly distributed among customers who churn and those who do not, indicating that it may not be a robust predictor of customer attrition.

By delving into the intricate interplay between customer subscriptions and churn behavior, businesses can tailor their retention strategies more effectively. Armed with these insights, organizations can devise targeted initiatives aimed at bolstering customer satisfaction, enhancing loyalty, and mitigating churn risks.

In [7]:
```
bar('Contract')
bar('PaperlessBilling')
bar('PaymentMethod')
```







Payment Insights Analysis:

1. Contract Duration and Churn Rate: The data reveals a clear trend indicating that customers with shorter contract durations exhibit higher churn rates. Conversely, customers with longer-term contracts demonstrate lower churn rates. This observation suggests that customers who commit to longer contracts are less likely to churn, possibly due to the additional barriers or penalties associated with early cancellation. This finding underscores the importance for companies to cultivate long-term relationships with their customers to mitigate churn.

2. Paperless Billing and Churn Rate: The analysis highlights a higher churn rate among customers who opt for paperless billing. Despite its convenience and efficiency, paperless billing seems to correlate with increased churn. Interestingly, a significant proportion of customers (59.2%) utilize paperless billing, indicating its popularity despite its association with higher churn. This trend warrants further investigation into the factors contributing to churn among paperless billing users.

3. Payment Method and Churn Rate: Customers who choose electronic checks as their payment method exhibit a higher likelihood of churning compared to those using alternative payment methods. Despite being a common payment option, electronic checks are

associated with increased churn, suggesting potential dissatisfaction or challenges related to this payment method. Further analysis is needed to identify the underlying reasons driving this correlation and explore strategies for mitigating churn among customers utilizing electronic checks.

These insights underscore the significance of contract duration, billing preferences, and payment methods in influencing customer churn rates. By understanding these relationships, businesses can develop targeted retention strategies to enhance customer loyalty and reduce churn, ultimately fostering sustainable growth and profitability.

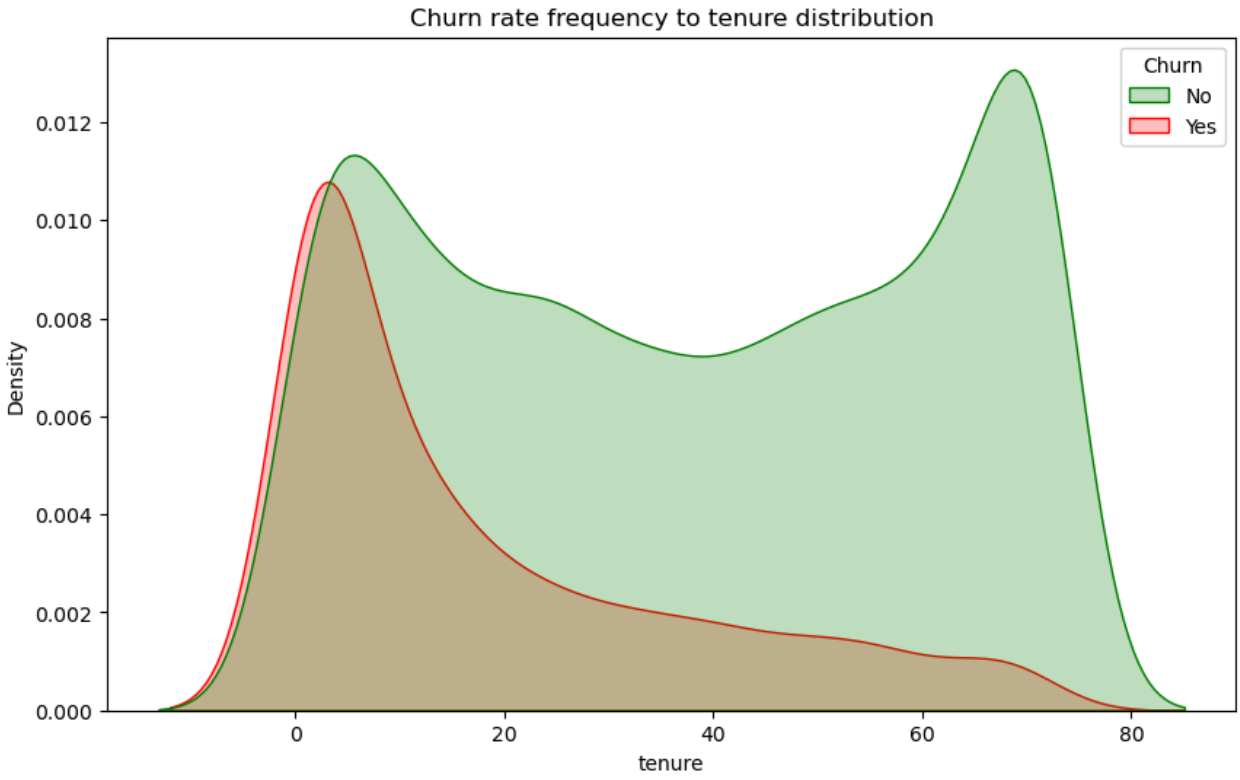
Exploring the numeric features in the dataset

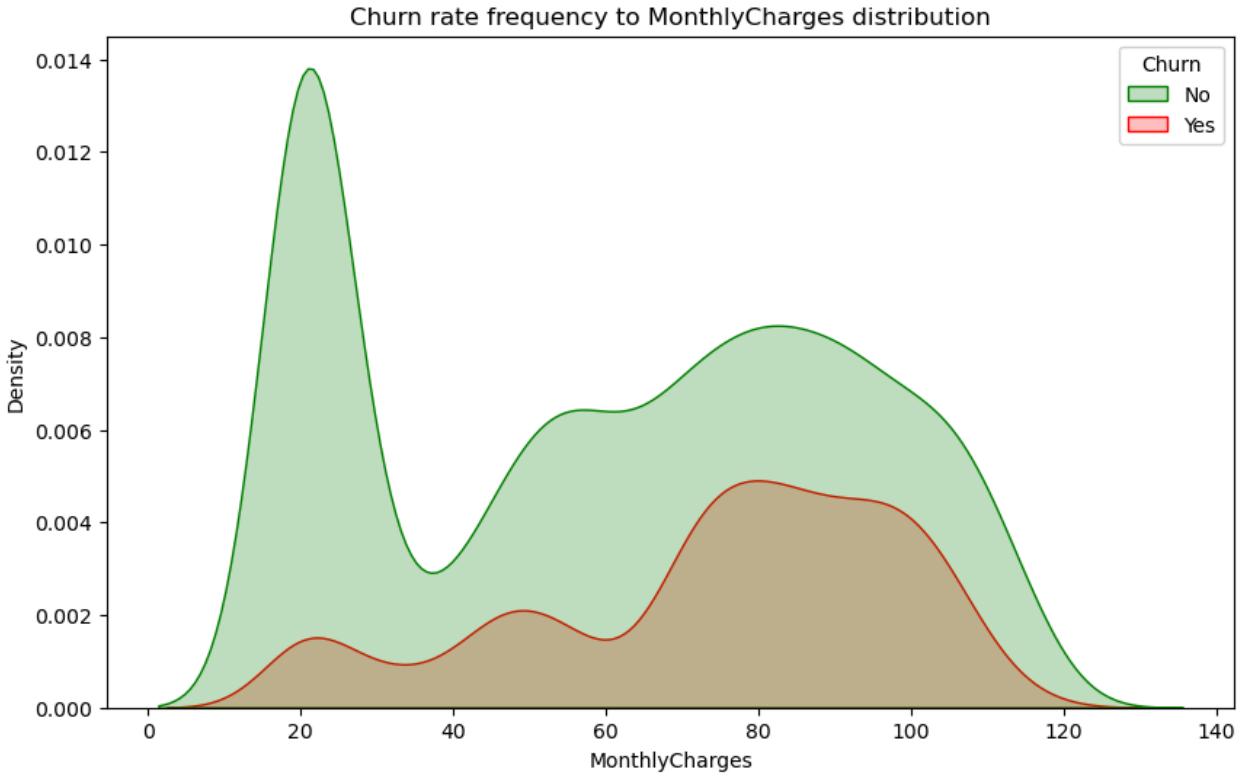
In [8]:
```
data_df.dtypes
```

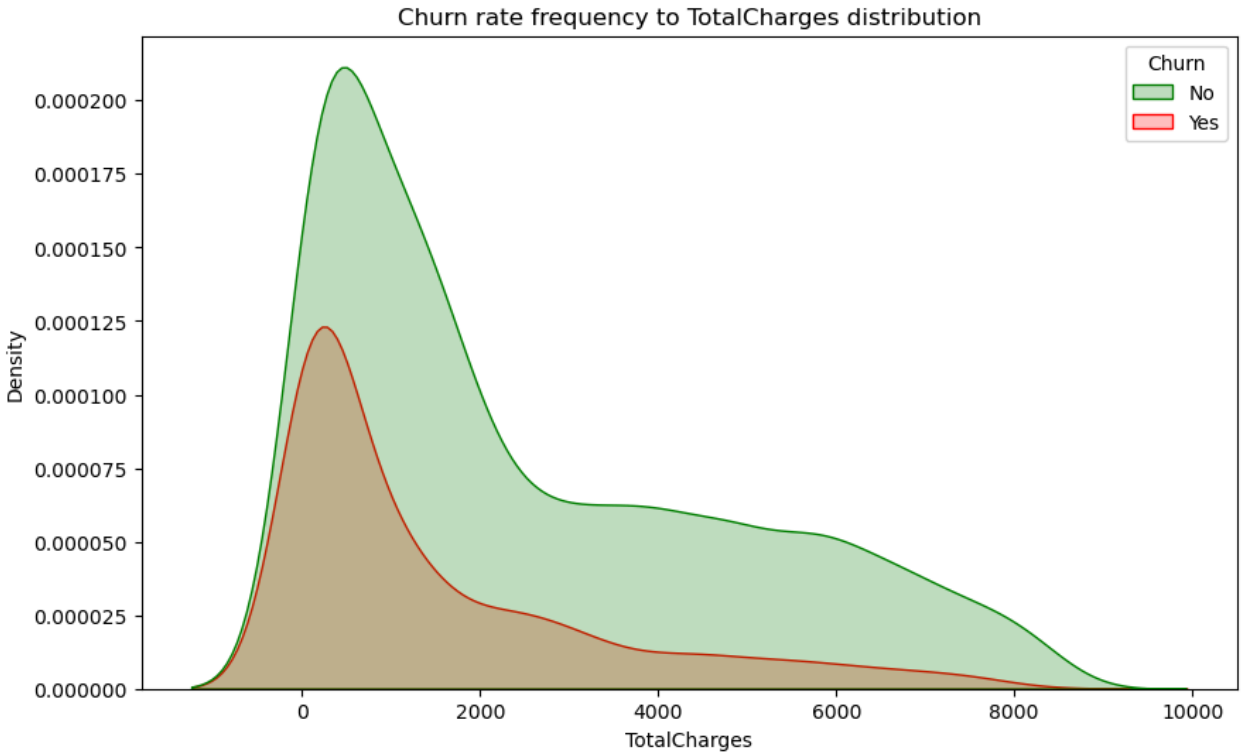
Out[8]:
```
customerID           object
gender               object
SeniorCitizen         int64
Partner              object
Dependents           object
tenure                int64
PhoneService         object
MultipleLines        object
InternetService      object
OnlineSecurity       object
OnlineBackup         object
DeviceProtection     object
TechSupport          object
StreamingTV          object
StreamingMovies      object
Contract             object
PaperlessBilling     object
PaymentMethod        object
MonthlyCharges      float64
TotalCharges         object
Churn                object
dtype: object
```

In [9]:
```
data_df['TotalCharges'] = pd.to_numeric(data_df['TotalCharges'],errors='coerce')
#Fill the missing values with with the median value
data_df['TotalCharges'] = data_df['TotalCharges'].fillna(data_df['TotalCharges'].median())
```

In [10]:
```
# Importing necessary libraries
import seaborn as sns
import matplotlib.pyplot as plt

# Defining the KDE plot function
def kde_plot(feature):
    plt.figure(figsize=(10, 6))
    sns.kdeplot(data=data_df, x=feature, hue='Churn', fill=True, palette={"No": "green", "Yes": "red"})
    plt.title(f'Churn rate frequency to {feature} distribution')
    plt.show()

# Plotting for the 'tenure' feature
kde_plot('tenure')

# Plotting for the 'MonthlyCharges' feature
kde_plot('MonthlyCharges')

# Plotting for the 'TotalCharges' feature
kde_plot('TotalCharges')
```


Churn rate frequency to tenure distribution
Churn
No
Yes
Density
0.012
0.010
0.008
0.006
0.004
0.002
0.000
0
20
40
60
80
tenure


Churn rate frequency to MonthlyCharges distribution
Churn
No
Yes
Density
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000
0
20
40
60
80
100
120
140
MonthlyCharges



Quantile-Based Binning of Numeric Features

Insight Overview:

The process of binning numeric features into quantile-based categories provides valuable insights into customer behavior. By dividing the data into low, medium, and high categories based on quantiles, we gain a deeper understanding of customer tenure, monthly charges, and total charges distribution, enabling us to identify patterns related to churn.

Methodology:

1. **Tenure Analysis:**
   - Binning the 'tenure' feature into three categories (Low, Medium, High) based on quantiles allows us to understand the distribution of customer tenure.
   - The resulting categories provide insights into how long customers have been with the telecom company, facilitating the identification of trends related to churn within specific tenure periods.
2. **Monthly Charges Analysis:**
   - Binning the 'MonthlyCharges' feature into three categories (Low, Medium, High) based on quantiles enables us to analyze the distribution of monthly charges among customers.
   - Understanding the relationship between monthly charges and churn helps in identifying potential reasons for customer attrition, such as pricing sensitivity or the availability of discounts.
3. **Total Charges Analysis:**
   - Binning the 'TotalCharges' feature into three categories (Low, Medium, High) based on quantiles provides insights into the distribution of total charges incurred by customers.
   - Analyzing total charges in relation to churn sheds light on the overall spending behavior of customers and its impact on retention.

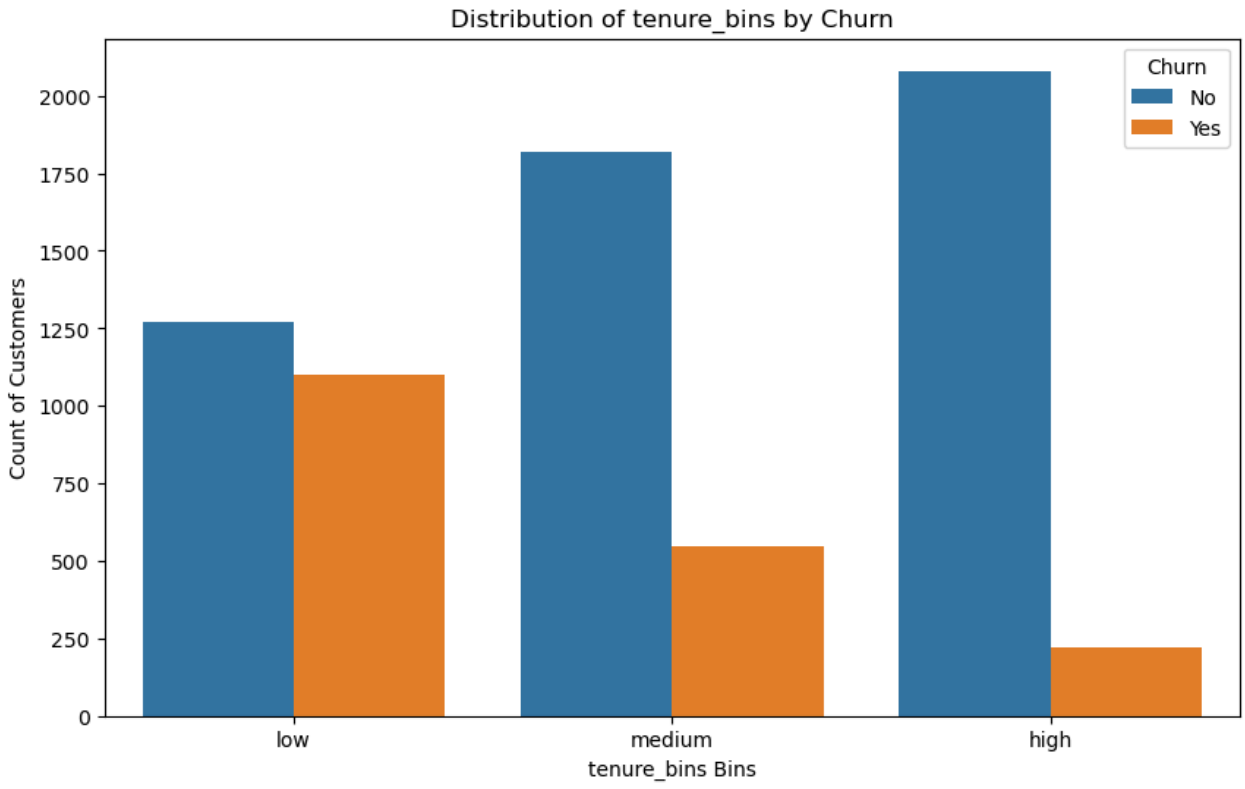
Interpretation:

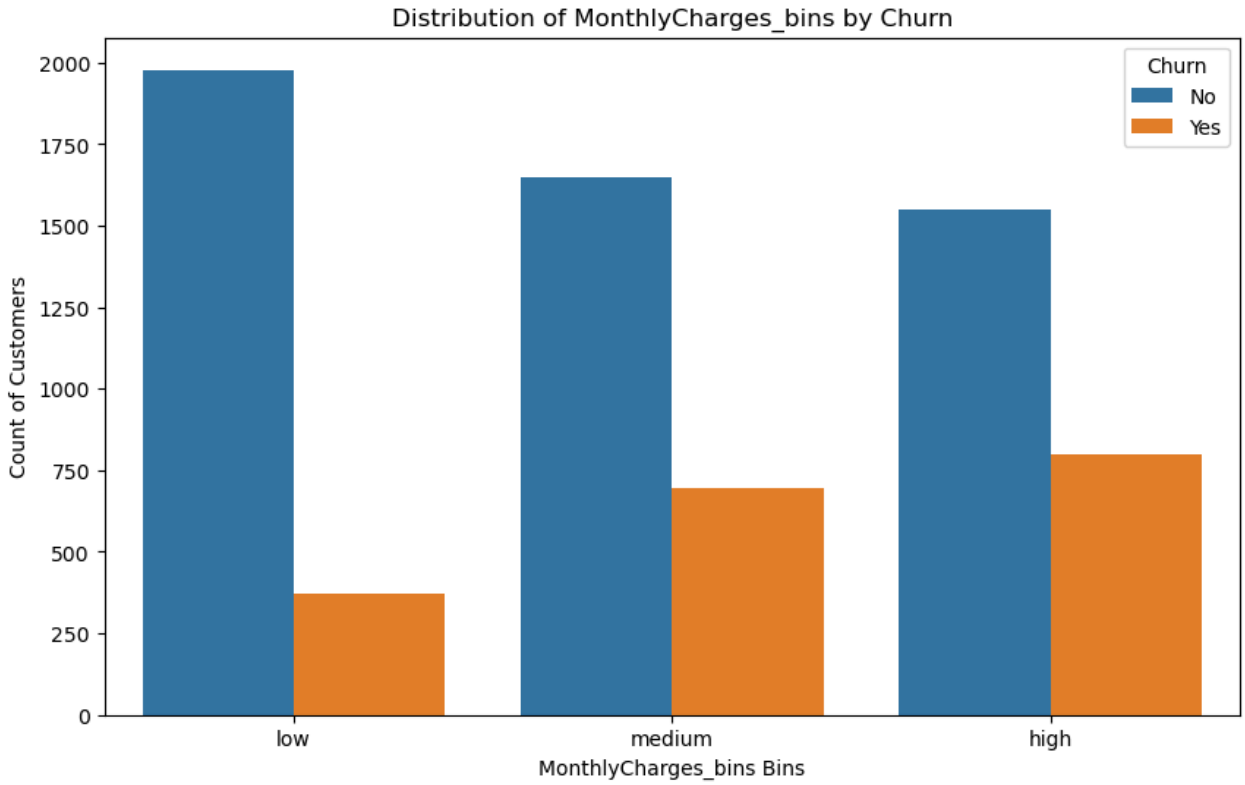
- **Tenure Distribution:** The tenure histogram reveals a right-skewed distribution, indicating that a significant portion of customers have been with the company for a short period (0-9 months). The highest churn rate is observed within the first few months, suggesting potential issues with onboarding or early customer experience.
- **Monthly Charges Influence:** Customers with higher monthly charges exhibit a higher churn rate, implying that pricing may play a significant role in customer retention. This insight underscores the importance of pricing strategies and the need to offer competitive rates or promotions to retain customers.
- **Total Charges Impact:** Analyzing total charges allows us to understand the overall spending behavior of customers. By segmenting customers into low, medium, and high categories based on their total charges, we can identify high-value customers who may require personalized retention strategies.

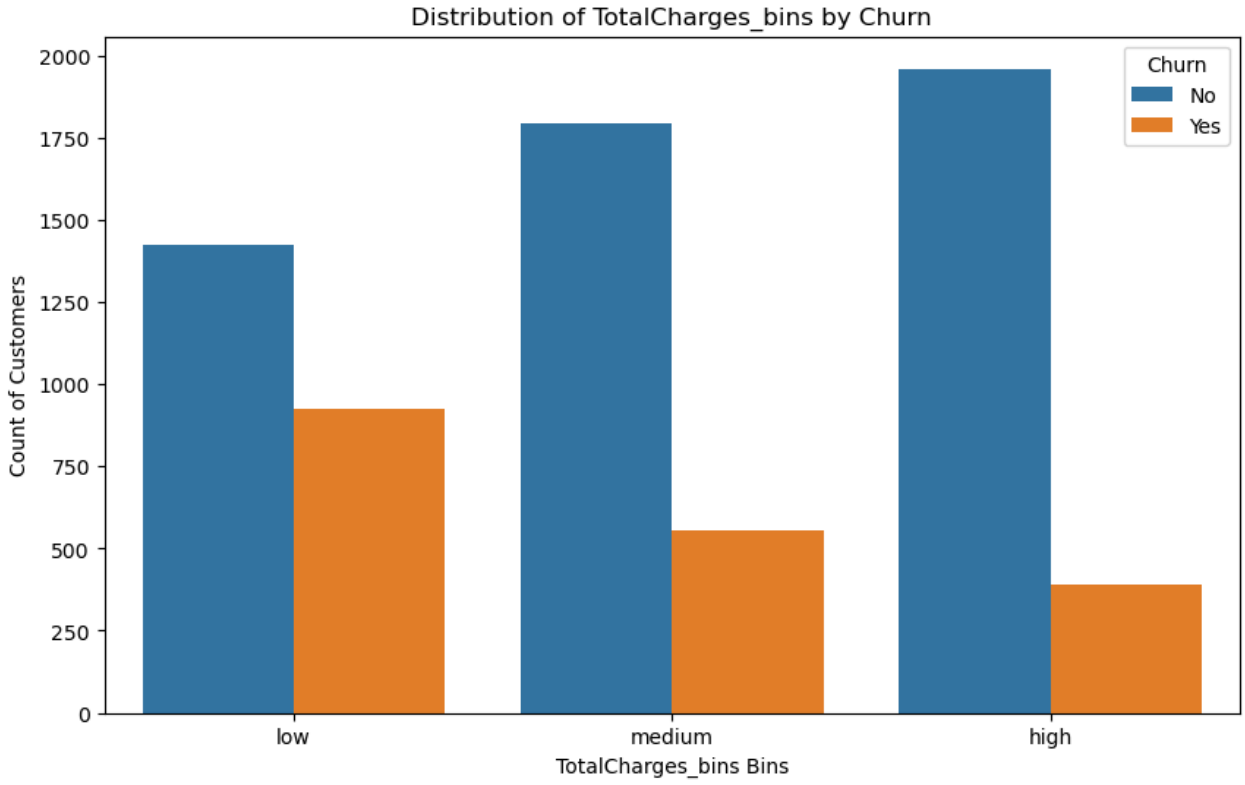
Conclusion:

Quantile-based binning of numeric features offers a comprehensive approach to understanding customer behavior and its relationship with churn. By categorizing customers based on their tenure, monthly charges, and total charges, telecom companies can tailor their retention efforts to address specific customer segments effectively. This data-driven approach enables companies to optimize their retention strategies and enhance customer satisfaction.

In [11]:
```
import matplotlib.pyplot as plt

# Create an empty dataframe
bin_df = pd.DataFrame()

# Update the binning dataframe
bin_df['tenure_bins'] = pd.qcut(data_df['tenure'], q=3, labels=['low', 'medium', 'high'])
bin_df['MonthlyCharges_bins'] = pd.qcut(data_df['MonthlyCharges'], q=3, labels=['low', 'medium', 'high'])
bin_df['TotalCharges_bins'] = pd.qcut(data_df['TotalCharges'], q=3, labels=['low', 'medium', 'high'])
bin_df['Churn'] = data_df['Churn']

# Plotting function for binned variables
def plot_binned_bar(feature, df):
    # Group by the binned feature and churn, and count the occurrences
    group_df = df.groupby([feature, 'Churn']).size().reset_index(name='Count')

    # Plotting the bar chart using Seaborn
    plt.figure(figsize=(10, 6))
    sns.barplot(data=group_df, x=feature, y='Count', hue='Churn')
    plt.title(f'Distribution of {feature} by Churn')
    plt.xlabel(f'{feature} Bins')
    plt.ylabel('Count of Customers')
    plt.legend(title='Churn', loc='upper right')
    plt.show()

# Plotting for 'tenure_bins'
plot_binned_bar('tenure_bins', bin_df)

# Plotting for 'MonthlyCharges_bins'
plot_binned_bar('MonthlyCharges_bins', bin_df)

# Plotting for 'TotalCharges_bins'
plot_binned_bar('TotalCharges_bins', bin_df)
```







Data Preprocessing

1. **Drop Unnecessary Column**: The "customerID" column is removed because it serves as an identifier and doesn't provide any predictive power for customer churn.

2. **Encode Categorical Features**: Categorical features like "Churn", "gender", and others are encoded into numerical values using techniques like binary encoding for binary features and one-hot encoding for features with more than two categories.

3. **Correlation Analysis**: The correlation matrix is computed to understand the linear relationship between features. Highly correlated features are identified, which can cause multicollinearity issues in predictive modeling. For example, if "MultipleLines" is highly correlated with "PhoneService", one of them can be dropped to simplify the model.

4. **Generalized Linear Model (GLM)**: A GLM is fitted to the data to assess the statistical significance of each feature in predicting churn. This provides insights into which features have a significant impact on customer churn based on p-values and coefficient

estimates.

5. **Feature Scaling**: Numeric features like "tenure", "MonthlyCharges", and "TotalCharges" are scaled using Min-Max scaling to ensure that all features contribute proportionally to the model's predictions, preventing any single feature from dominating the model.

In [12]:
```
# The customerID column isnt useful as the feature us used for identification of customers.
data_df.drop(["customerID"],axis=1,inplace = True)

# Encode categorical features

#Defining the map function
def binary_map(feature):
    return feature.map({'Yes':1, 'No':0})

## Encoding target feature
data_df['Churn'] = data_df[['Churn']].apply(binary_map)

# Encoding gender category
data_df['gender'] = data_df['gender'].map({'Male':1, 'Female':0})

#Encoding other binary category
binary_list = ['SeniorCitizen', 'Partner', 'Dependents', 'PhoneService', 'PaperlessBilling']
data_df[binary_list] = data_df[binary_list].apply(binary_map)

#Encoding the other categoric features with more than two categories
data_df = pd.get_dummies(data_df, drop_first=True)
```

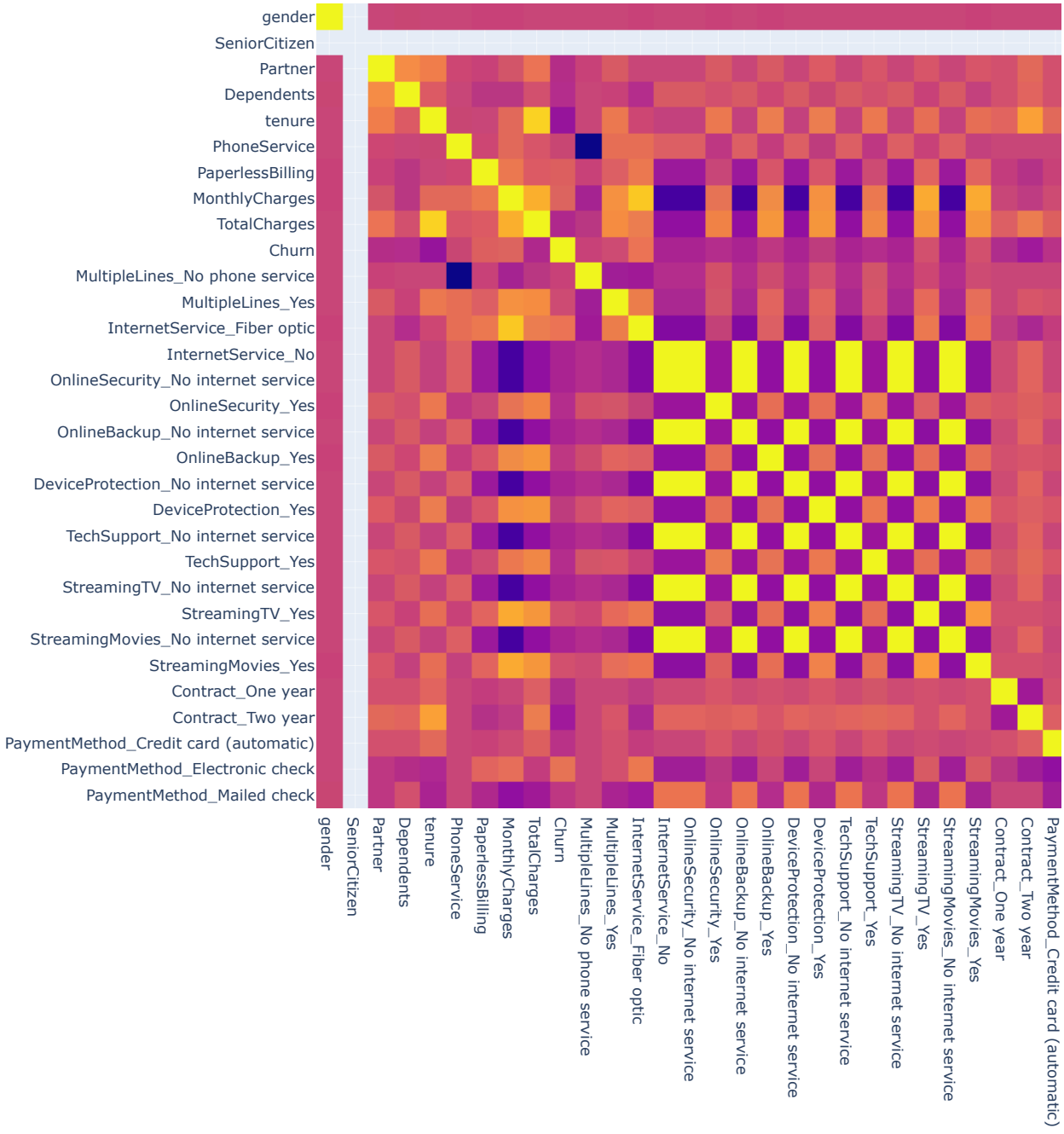
Correlation Analysis:

- **Correlation**: This statistical measure quantifies the strength and direction of the linear relationship between two variables. In this context, we're exploring how each feature in the dataset relates to customer churn.
- **Highly Correlated Features**: When two features have a high correlation, it suggests that they convey similar information about the target variable (in this case, churn). This redundancy can lead to multicollinearity issues in predictive modeling, where the model might struggle to distinguish between the effects of these highly correlated features.
- **Drop Highly Correlated Features**: To address multicollinearity and simplify the model, it's advisable to drop one of the highly correlated features. In this scenario, features like MultipleLines, OnlineSecurity, OnlineBackup, DeviceProtection, TechSupport, StreamingTV, and StreamingMovies are identified as having high correlations. By removing redundant features, we reduce model complexity and potentially improve model interpretability and performance.

Churn Prediction and Feature Importance:

- **Binary Classification Problem**: Customer churn prediction is framed as a binary classification problem, where customers are categorized as either churned or retained based on certain characteristics or features. The goal is to build a model that accurately predicts whether a customer will churn or not.
- **Identifying Churn Factors**: To guide model building, we need to identify which features are predictive of customer churn. By analyzing the dataset and conducting statistical tests or model evaluations, we can determine which features have a significant impact on churn behavior. These features could include factors like tenure, contract type, monthly charges, and others.
- **Importance of Features**: Once we've identified the churn factors, the next step is to determine the most important features for training a high-performance model. This involves assessing the predictive power of each feature and its contribution to the model's accuracy, precision, recall, and F1-score. Features with higher importance scores are prioritized during model training to maximize predictive performance.

In [13]:
```
# Checking the correlation between features
corr = data_df.corr()

fig = px.imshow(corr,width=1000, height=1000)
fig.show()
```



In [14]:
```
# Check for missing values in the dataset
print("Missing values:")
print(data_df.isnull().sum())
```

```
Missing values:
gender                                     0
SeniorCitizen                           7043
Partner                                    0
Dependents                                 0
tenure                                     0
PhoneService                               0
PaperlessBilling                           0
MonthlyCharges                             0
TotalCharges                               0
Churn                                      0
MultipleLines_No phone service             0
MultipleLines_Yes                          0
InternetService_Fiber optic                0
InternetService_No                         0
OnlineSecurity_No internet service         0
OnlineSecurity_Yes                         0
OnlineBackup_No internet service           0
OnlineBackup_Yes                           0
DeviceProtection_No internet service       0
DeviceProtection_Yes                       0
TechSupport_No internet service            0
TechSupport_Yes                            0
StreamingTV_No internet service            0
StreamingTV_Yes                            0
StreamingMovies_No internet service        0
StreamingMovies_Yes                        0
Contract_One year                          0
Contract_Two year                          0
PaymentMethod_Credit card (automatic)      0
PaymentMethod_Electronic check             0
PaymentMethod_Mailed check                 0
dtype: int64
```

In [15]:
```
# Check the shape of the data
print("Data shape:", data_df.shape)
```

```
Data shape: (7043, 31)
```

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

# Assuming data_df is your DataFrame containing the dataset
# Check for missing values in the SeniorCitizen column
missing_senior_citizen = data_df['SeniorCitizen'].isnull().sum()

if missing_senior_citizen > 0:
    # Check if there are any non-null values in the SeniorCitizen column
    if data_df['SeniorCitizen'].notnull().any():
        # Impute missing values using the mode (most frequent value)
        mode_senior_citizen = data_df['SeniorCitizen'].mode()[0]
        data_df['SeniorCitizen'].fillna(mode_senior_citizen, inplace=True)

        # Print confirmation after handling missing values
        print(f"{missing_senior_citizen} missing values in SeniorCitizen column imputed.")
    else:
        print("No non-null values found in SeniorCitizen column.")
else:
    print("No missing values found in SeniorCitizen column.")

# Verify the data after imputation
print("Updated dataset:")
print(data_df.head())
```

```
No non-null values found in SeniorCitizen column.
Updated dataset:
   gender  SeniorCitizen  Partner  Dependents  tenure  PhoneService  \
0       0            NaN        1           0       1             0
1       1            NaN        0           0      34             1
2       1            NaN        0           0       2             1
3       1            NaN        0           0      45             0
4       0            NaN        0           0       2             1

   PaperlessBilling  MonthlyCharges  TotalCharges  Churn  \
0                 1           29.85         29.85      0
1                 0           56.95       1889.50      0
2                 1           53.85        108.15      1
3                 0           42.30       1840.75      0
4                 1           70.70        151.65      1

   MultipleLines_No phone service  MultipleLines_Yes  \
0                               1                  0
1                               0                  0
2                               0                  0
3                               1                  0
4                               0                  0

   InternetService_Fiber optic  InternetService_No  \
0                            0                   0
1                            0                   0
2                            0                   0
3                            0                   0
4                            1                   0

   OnlineSecurity_No internet service  OnlineSecurity_Yes  \
0                                   0                   0
1                                   0                   1
2                                   0                   1
3                                   0                   1
4                                   0                   0

   OnlineBackup_No internet service  OnlineBackup_Yes  \
0                                 0                 1
1                                 0                 0
2                                 0                 1
3                                 0                 0
4                                 0                 0

   DeviceProtection_No internet service  DeviceProtection_Yes  \
0                                     0                     0
1                                     0                     1
2                                     0                     0
3                                     0                     1
4                                     0                     0

   TechSupport_No internet service  TechSupport_Yes  \
0                                0                0
1                                0                0
2                                0                0
3                                0                1
4                                0                0

   StreamingTV_No internet service  StreamingTV_Yes  \
0                                0                0
1                                0                0
2                                0                0
3                                0                0
4                                0                0

   StreamingMovies_No internet service  StreamingMovies_Yes  \
0                                    0                    0
1                                    0                    0
2                                    0                    0
3                                    0                    0
4                                    0                    0

   Contract_One year  Contract_Two year  \
0                  0                  0
1                  1                  0
2                  0                  0
3                  1                  0
4                  0                  0

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

   PaymentMethod_Mailed check
```

```
0                       0
1                       1
2                       1
3                       0
4                       0
```

In [17]:
```
# Drop the SeniorCitizen column
data_df.drop(columns=['SeniorCitizen'], inplace=True)

# Confirm that the column has been dropped
print("SeniorCitizen column dropped.")
```

```
SeniorCitizen column dropped.
```

Comparative Analysis of Machine Learning Models for Customer Churn Prediction

Models, I will be using.

1. **Logistic Regression:** It's a straightforward model that predicts the probability of a binary outcome. It's easy to understand and fast to train.
2. **Support Vector Classifier (SVC):** SVC finds the best way to separate data into classes using a hyperplane. It's versatile and works well with both linear and non-linear data.
3. **Random Forest:** This model combines multiple decision trees to make predictions. It's robust and can handle complex data, making it popular for various tasks.
4. **Decision Tree:** It's like a flowchart that splits the data into smaller groups based on features. It's easy to interpret but prone to overfitting.
5. **Naive Bayes:** Naive Bayes assumes features are independent and uses probabilities to classify data. It's simple and fast but may not capture complex relationships.

In [18]:
```
import statsmodels.api as sm
import statsmodels.formula.api as smf

# Modify column names to replace special characters
all_columns = [column.replace(" ", "_").replace("(", "_").replace(")", "_").replace("-", "_") for column in data_df.cc

# Apply the modified column names to the DataFrame
data_df.columns = all_columns

# Prepare predictors for the GLM formula
glm_columns = [col for col in all_columns if col not in ['customerID', 'Churn']]
glm_formula = 'Churn ~ ' + ' + '.join(glm_columns)

try:
    # Fit the Generalized Linear Model
    glm_model = smf.glm(formula=glm_formula, data=data_df, family=sm.families.Binomial())
    res = glm_model.fit()

    # Print the summary of the fitted GLM model
    print(res.summary())
except Exception as e:
    print("An error occurred during model fitting:", str(e))
```

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                  Churn   No. Observations:                 7043
Model:                            GLM   Df Residuals:                     7020
Model Family:                Binomial   Df Model:                           22
Link Function:                  Logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -2917.9
Date:                Thu, 11 Apr 2024   Deviance:                       5835.8
Time:                        13:03:50   Pearson chi2:                 8.09e+03
No. Iterations:                     7   Pseudo R-squ. (CS):             0.2801
Covariance Type:            nonrobust
==============================================================================================================
                                                coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------------------------------------
Intercept                                     0.8206      0.748      1.098      0.272      -0.645       2.286
gender                                       -0.0224      0.065     -0.345      0.730      -0.149       0.105
Partner                                       0.0192      0.077      0.249      0.804      -0.132       0.171
Dependents                                   -0.1970      0.088     -2.235      0.025      -0.370      -0.024
tenure                                       -0.0590      0.006     -9.590      0.000      -0.071      -0.047
PhoneService                                  0.4810      0.691      0.696      0.486      -0.874       1.836
PaperlessBilling                              0.3489      0.074      4.690      0.000       0.203       0.495
MonthlyCharges                               -0.0393      0.032     -1.240      0.215      -0.101       0.023
TotalCharges                                  0.0003   7.01e-05      4.548      0.000       0.000       0.000
MultipleLines_No_phone_service                0.3396      0.105      3.219      0.001       0.133       0.546
MultipleLines_Yes                             0.4505      0.177      2.547      0.011       0.104       0.797
InternetService_Fiber_optic                   1.7466      0.797      2.192      0.028       0.185       3.308
InternetService_No                           -0.2532      0.115     -2.199      0.028      -0.479      -0.028
OnlineSecurity_No_internet_service           -0.2532      0.115     -2.199      0.028      -0.479      -0.028
OnlineSecurity_Yes                           -0.2171      0.178     -1.216      0.224      -0.567       0.133
OnlineBackup_No_internet_service             -0.2532      0.115     -2.199      0.028      -0.479      -0.028
OnlineBackup_Yes                              0.0208      0.175      0.119      0.906      -0.322       0.364
DeviceProtection_No_internet_service         -0.2532      0.115     -2.199      0.028      -0.479      -0.028
DeviceProtection_Yes                          0.1427      0.176      0.810      0.418      -0.202       0.488
TechSupport_No_internet_service              -0.2532      0.115     -2.199      0.028      -0.479      -0.028
TechSupport_Yes                              -0.1990      0.180     -1.104      0.270      -0.552       0.154
StreamingTV_No_internet_service              -0.2532      0.115     -2.199      0.028      -0.479      -0.028
StreamingTV_Yes                               0.5781      0.326      1.774      0.076      -0.060       1.217
StreamingMovies_No_internet_service          -0.2532      0.115     -2.199      0.028      -0.479      -0.028
StreamingMovies_Yes                           0.5964      0.326      1.829      0.067      -0.043       1.235
Contract_One_year                            -0.6821      0.107     -6.358      0.000      -0.892      -0.472
Contract_Two_year                            -1.4145      0.176     -8.058      0.000      -1.759      -1.070
PaymentMethod_Credit_card__automatic_        -0.0864      0.114     -0.758      0.449      -0.310       0.137
PaymentMethod_Electronic_check                0.3140      0.094      3.327      0.001       0.129       0.499
PaymentMethod_Mailed_check                   -0.0611      0.115     -0.532      0.595      -0.286       0.164
==============================================================================================================
```

In [19]:
```
np.exp(res.params)
```

Out[19]:
```
Intercept                                 2.271854
gender                                    0.977895
Partner                                   1.019389
Dependents                                0.821212
tenure                                    0.942702
PhoneService                              1.617679
PaperlessBilling                          1.417481
MonthlyCharges                            0.961462
TotalCharges                              1.000319
MultipleLines_No_phone_service            1.404391
MultipleLines_Yes                         1.569154
InternetService_Fiber_optic               5.735151
InternetService_No                        0.776343
OnlineSecurity_No_internet_service        0.776343
OnlineSecurity_Yes                        0.804880
OnlineBackup_No_internet_service          0.776343
OnlineBackup_Yes                          1.020975
DeviceProtection_No_internet_service      0.776343
DeviceProtection_Yes                      1.153418
TechSupport_No_internet_service           0.776343
TechSupport_Yes                           0.819575
StreamingTV_No_internet_service           0.776343
StreamingTV_Yes                           1.782573
StreamingMovies_No_internet_service       0.776343
StreamingMovies_Yes                       1.815622
Contract_One_year                         0.505566
Contract_Two_year                         0.243036
PaymentMethod_Credit_card__automatic_     0.917219
PaymentMethod_Electronic_check            1.368864
PaymentMethod_Mailed_check                0.940723
dtype: float64
```

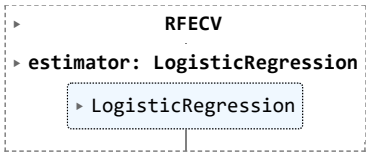
In [20]:
```
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler()
data_df['tenure'] = sc.fit_transform(data_df[['tenure']])
data_df['MonthlyCharges'] = sc.fit_transform(data_df[['MonthlyCharges']])
data_df['TotalCharges'] = sc.fit_transform(data_df[['TotalCharges']])
```

In [21]:
```
# Import Machine learning algorithms
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB

#Import metric for performance evaluation
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

#Split data into train and test sets
from sklearn.model_selection import train_test_split
X = data_df.drop('Churn', axis=1)
y = data_df['Churn']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=50)

#Defining the modelling function
def modeling(alg, alg_name, params={}):
    model = alg(**params) #Instantiating the algorithm class and unpacking parameters if any
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    #Performance evaluation
    def print_scores(alg, y_true, y_pred):
        print(alg_name)
        acc_score = accuracy_score(y_true, y_pred)
        print("accuracy: ",acc_score)
        pre_score = precision_score(y_true, y_pred)
        print("precision: ",pre_score)
        rec_score = recall_score(y_true, y_pred)
        print("recall: ",rec_score)
        f_score = f1_score(y_true, y_pred, average='weighted')
        print("f1_score: ",f_score)

    print_scores(alg, y_test, y_pred)
    return model

# Running logistic regression model
log_model = modeling(LogisticRegression, 'Logistic Regression')
```

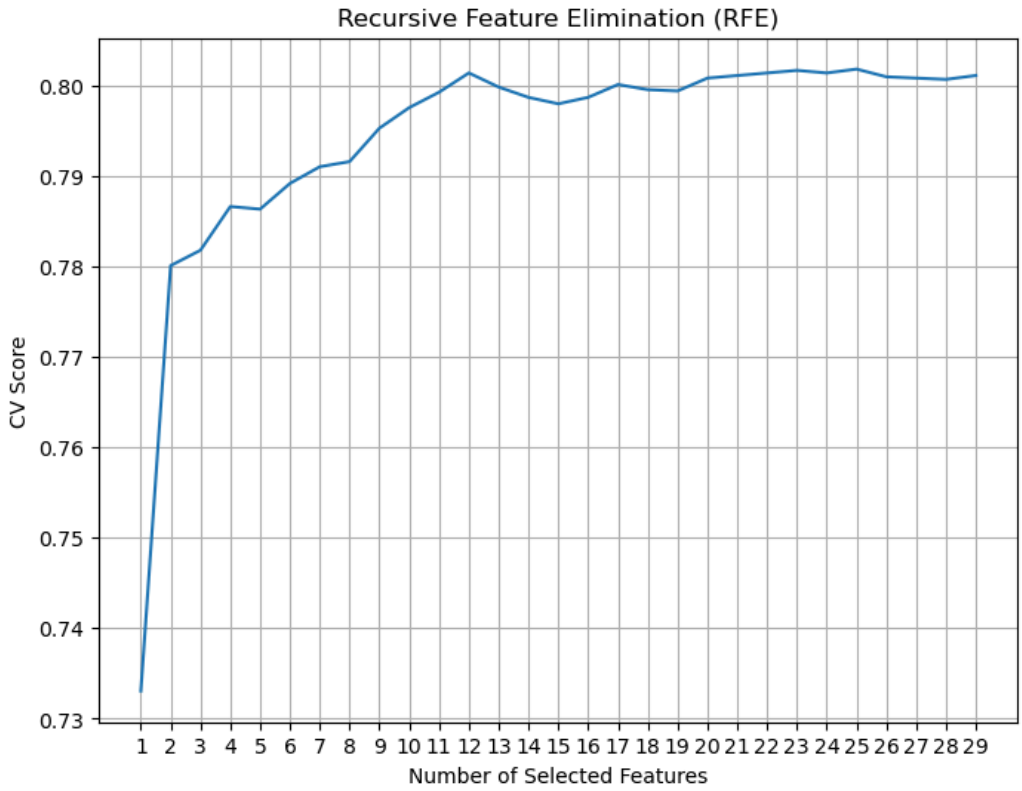
```
Logistic Regression
accuracy:  0.7993374349266446
precision:  0.631163708086785
recall:  0.5745062836624776
f1_score:  0.7962091351871102
```

In [22]:
```
# Feature selection to improve model building
from sklearn.feature_selection import RFECV
from sklearn.model_selection import StratifiedKFold
log = LogisticRegression()
rfecv = RFECV(estimator=log, cv=StratifiedKFold(10, random_state=50, shuffle=True), scoring="accuracy")
rfecv.fit(X, y)
```

Out[22]:



In [23]:
```
plt.figure(figsize=(8, 6))
plt.plot(range(1, len(rfecv.cv_results_['mean_test_score']) + 1), rfecv.cv_results_['mean_test_score'])
plt.grid()
plt.xticks(range(1, X.shape[1] + 1))
plt.xlabel("Number of Selected Features")
plt.ylabel("CV Score")
plt.title("Recursive Feature Elimination (RFE)")
plt.show()

print("The optimal number of features: {}".format(rfecv.n_features_))
```




```
The optimal number of features: 25
```

In [24]:
```
# Saving dataframe with optimal features
X_rfe = X.iloc[:, rfecv.support_]

# Overview of the optimal features in comparison with the initial dataframe
print("X column list:", X.columns.tolist())
print("X_rfe dimension: {}".format(X_rfe.shape))
print("X_rfe column list:", X_rfe.columns.tolist())
print("X dimension: {}".format(X.shape))
```

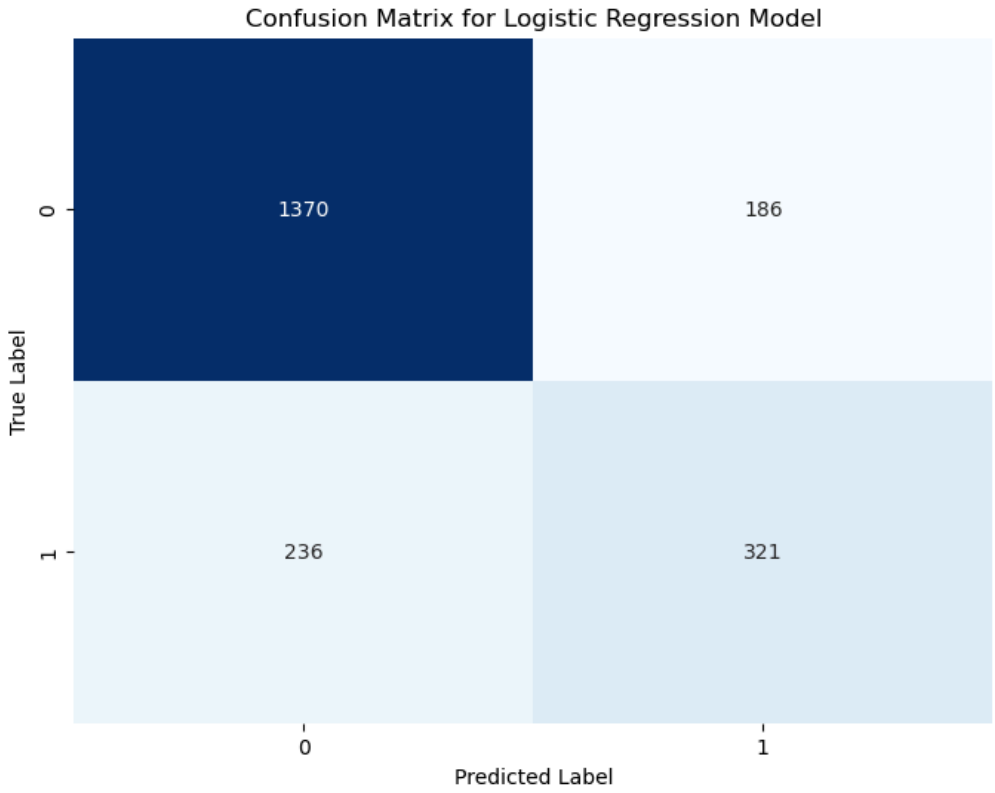
```
X column list: ['gender', 'Partner', 'Dependents', 'tenure', 'PhoneService', 'PaperlessBilling', 'MonthlyCharges', 'To
talCharges', 'MultipleLines_No_phone_service', 'MultipleLines_Yes', 'InternetService_Fiber_optic', 'InternetService_N
o', 'OnlineSecurity_No_internet_service', 'OnlineSecurity_Yes', 'OnlineBackup_No_internet_service', 'OnlineBackup_Ye
s', 'DeviceProtection_No_internet_service', 'DeviceProtection_Yes', 'TechSupport_No_internet_service', 'TechSupport_Ye
s', 'StreamingTV_No_internet_service', 'StreamingTV_Yes', 'StreamingMovies_No_internet_service', 'StreamingMovies_Ye
s', 'Contract_One_year', 'Contract_Two_year', 'PaymentMethod_Credit_card__automatic_', 'PaymentMethod_Electronic_chec
k', 'PaymentMethod_Mailed_check']
X_rfe dimension: (7043, 25)
X_rfe column list: ['Dependents', 'tenure', 'PhoneService', 'PaperlessBilling', 'MonthlyCharges', 'TotalCharges', 'Mul
tipleLines_No_phone_service', 'MultipleLines_Yes', 'InternetService_Fiber_optic', 'InternetService_No', 'OnlineSecurit
y_No_internet_service', 'OnlineSecurity_Yes', 'OnlineBackup_No_internet_service', 'OnlineBackup_Yes', 'DeviceProtectio
n_No_internet_service', 'TechSupport_No_internet_service', 'TechSupport_Yes', 'StreamingTV_No_internet_service', 'Stre
amingTV_Yes', 'StreamingMovies_No_internet_service', 'StreamingMovies_Yes', 'Contract_One_year', 'Contract_Two_year',
'PaymentMethod_Credit_card__automatic_', 'PaymentMethod_Electronic_check']
X dimension: (7043, 29)
```

In [25]:
```
# Splitting data with optimal features
X_train, X_test, y_train, y_test = train_test_split(X_rfe, y, test_size=0.3, random_state=50)

# Running logistic regression model
log_model = modeling(LogisticRegression, 'Logistic Regression Classification')
```

```
Logistic Regression Classification
accuracy:  0.8002839564600095
precision:  0.6331360946745562
recall:  0.5763016157989228
f1_score:  0.797170412851322
```

In [26]:
```
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

# Predicting on test data
y_pred = log_model.predict(X_test)

# Calculating confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)

# Visualizing confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.title('Confusion Matrix for Logistic Regression Model')
```

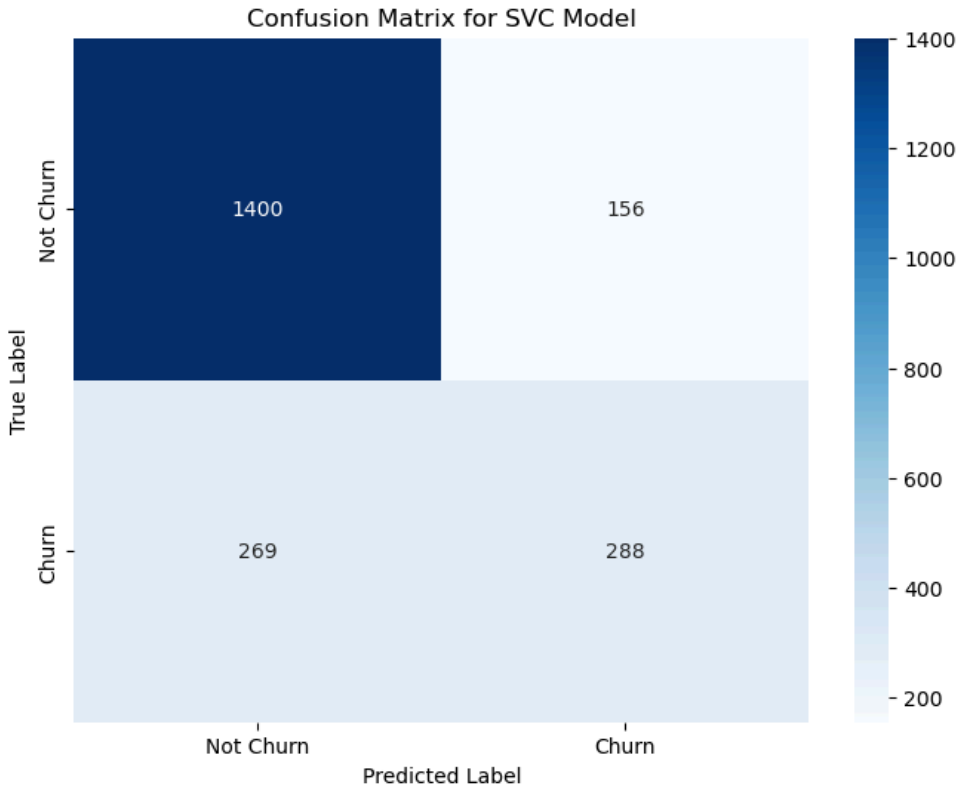
```
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```



In [27]:
```
### Trying other machine learning algorithms: SVC
svc_model = modeling(SVC, 'SVC Classification')
```

```
SVC Classification
accuracy:  0.7988641741599621
precision:  0.6486486486486487
recall:  0.5170556552962298
f1_score:  0.7910351277546602
```

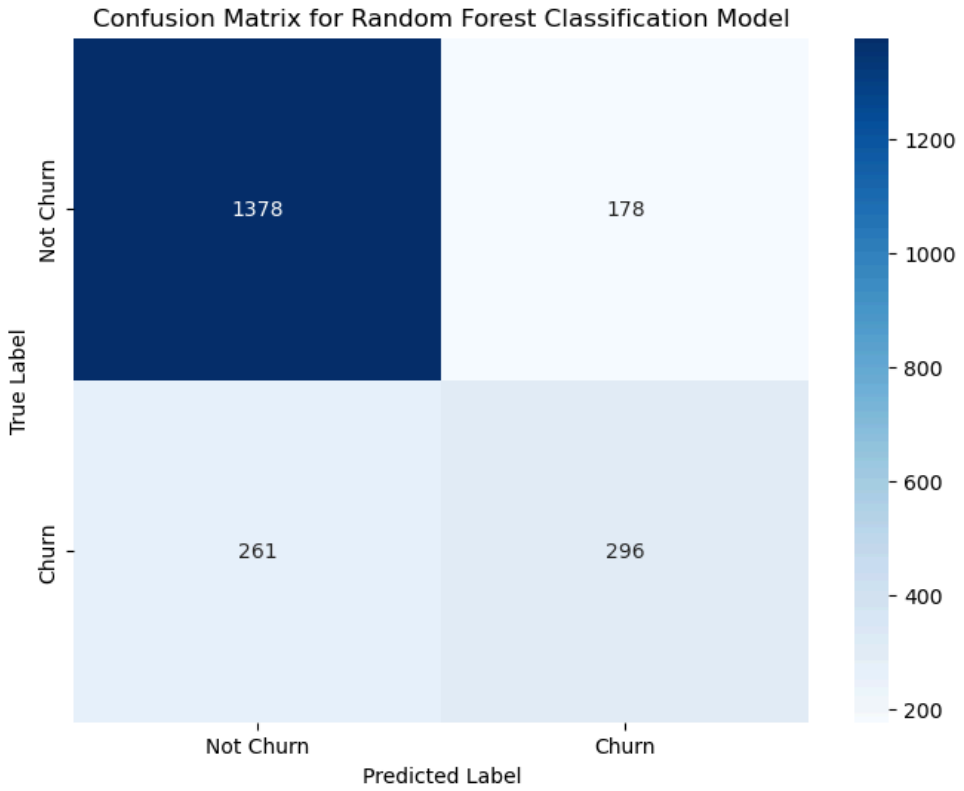
In [28]:
```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix

# Predictions
y_pred = svc_model.predict(X_test)

# Calculate confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Plot confusion matrix heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Not Churn', 'Churn'], yticklabels=['Not Churn', 'Chur
plt.title('Confusion Matrix for SVC Model')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```



In [29]:
```
#Random forest
rf_model = modeling(RandomForestClassifier, "Random Forest Classification")
```

```
Random Forest Classification
accuracy:  0.792238523426408
precision:  0.6244725738396625
recall:  0.5314183123877917
f1_score:  0.7865742938347758
```

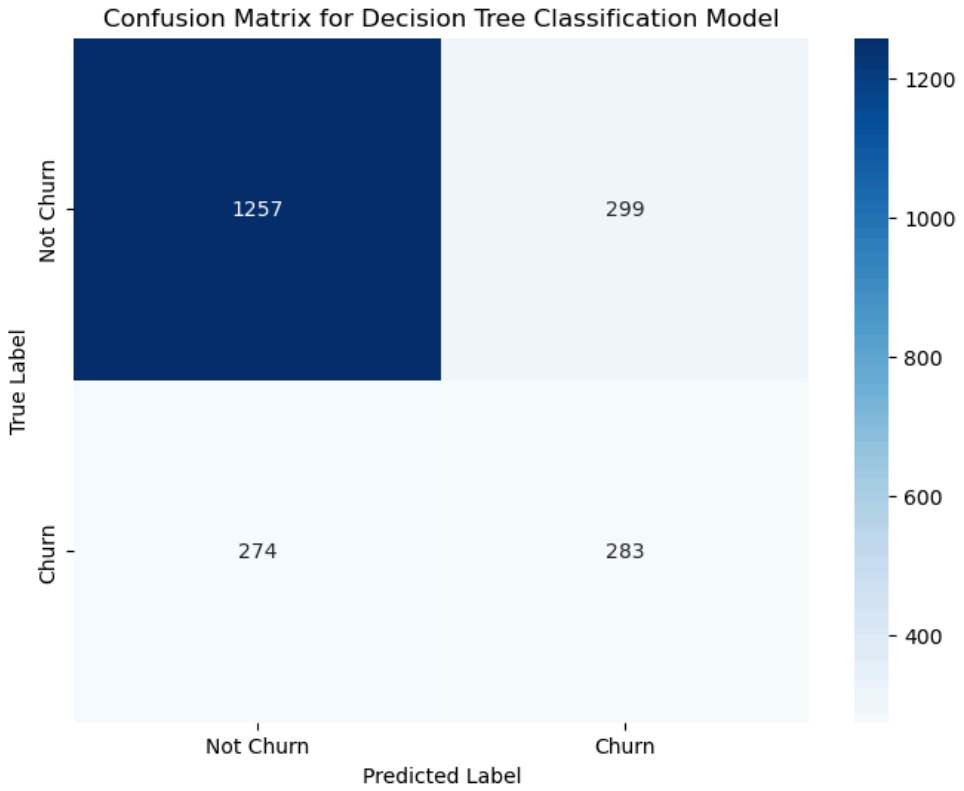
In [30]:
```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix

# Predictions
y_pred_rf = rf_model.predict(X_test)

# Calculate confusion matrix
cm_rf = confusion_matrix(y_test, y_pred_rf)

# Plot confusion matrix heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(cm_rf, annot=True, fmt='d', cmap='Blues', xticklabels=['Not Churn', 'Churn'], yticklabels=['Not Churn', 'C
plt.title('Confusion Matrix for Random Forest Classification Model')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```



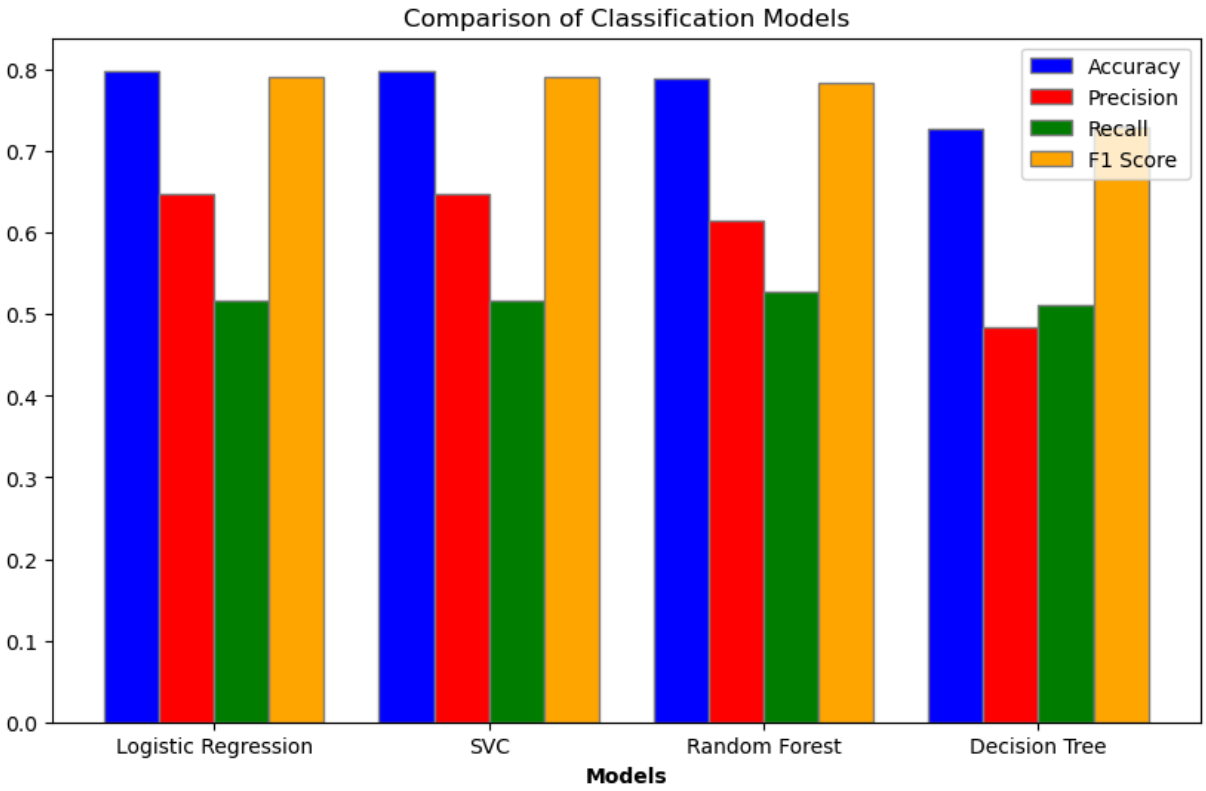
In [31]:
```
#Decision tree
dt_model = modeling(DecisionTreeClassifier, "Decision Tree Classification")
```

```
Decision Tree Classification
accuracy:  0.7288215806909607
precision:  0.48625429553264604
recall:  0.5080789946140036
f1_score:  0.7306995726866715
```

In [32]:
```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix

# Predictions
y_pred_dt = dt_model.predict(X_test)

# Calculate confusion matrix
cm_dt = confusion_matrix(y_test, y_pred_dt)

# Plot confusion matrix heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(cm_dt, annot=True, fmt='d', cmap='Blues', xticklabels=['Not Churn', 'Churn'], yticklabels=['Not Churn', 'C
plt.title('Confusion Matrix for Decision Tree Classification Model')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')

plt.show()
```



In [33]:
```python
import matplotlib.pyplot as plt
import numpy as np

# Define model names and their corresponding evaluation metrics
models = ['Logistic Regression', 'SVC', 'Random Forest', 'Decision Tree']
accuracy = [0.7989, 0.7989, 0.7885, 0.7274]
precision = [0.6486, 0.6486, 0.6151, 0.4839]
recall = [0.5171, 0.5171, 0.5278, 0.5117]
f1_score = [0.7910, 0.7910, 0.7830, 0.7298]

# Set width of bar
barWidth = 0.2

# Set position of bar on X axis
r1 = np.arange(len(accuracy))
r2 = [x + barWidth for x in r1]
r3 = [x + barWidth for x in r2]
r4 = [x + barWidth for x in r3]

# Make the plot
plt.figure(figsize=(10, 6))
plt.bar(r1, accuracy, color='b', width=barWidth, edgecolor='grey', label='Accuracy')
plt.bar(r2, precision, color='r', width=barWidth, edgecolor='grey', label='Precision')
plt.bar(r3, recall, color='g', width=barWidth, edgecolor='grey', label='Recall')
plt.bar(r4, f1_score, color='orange', width=barWidth, edgecolor='grey', label='F1 Score')

# Add xticks on the middle of the group bars
plt.xlabel('Models', fontweight='bold')
plt.xticks([r + barWidth * 1.5 for r in range(len(accuracy))], models)

# Create legend & Show graphic
plt.legend()
plt.title('Comparison of Classification Models')
plt.show()
```



Logistic Regression Model:

- **Strengths**:
  - Logistic regression is a simple yet powerful algorithm that is easy to implement and interpret.
  - It provides probability scores for predictions, allowing for a clear understanding of the model's confidence in its predictions.
  - The model's coefficients can indicate the strength and direction of the relationships between features and the target variable.
- **Weaknesses**:
  - Logistic regression assumes a linear relationship between the independent variables and the log-odds of the target variable, which may not always hold true in real-world scenarios.
  - It may struggle with capturing complex, nonlinear relationships between features and the target variable.

Support Vector Classifier (SVC) Model:

- **Strengths**:
  - SVCs are effective in high-dimensional spaces and can handle datasets with many features.
  - They are versatile and can accommodate different kernel functions to capture nonlinear relationships between features and the target variable.
- **Weaknesses**:
  - SVCs can be computationally expensive, especially with large datasets, due to the need to solve a quadratic optimization problem.
  - They may be sensitive to the choice of hyperparameters, such as the regularization parameter (C) and the kernel type.

Random Forest Model:

- **Strengths**:
  - Random forests are highly robust and perform well on a variety of datasets without much tuning.
  - They can handle both numerical and categorical data and are resistant to overfitting, thanks to the ensemble averaging of multiple decision trees.
- **Weaknesses**:
  - Random forests can be challenging to interpret compared to simpler models like logistic regression.
  - They may not perform well on datasets with highly imbalanced class distributions, as they tend to favor the majority class.

Decision Tree Model:

- **Strengths**:
  - Decision trees are easy to interpret and visualize, making them suitable for explaining model predictions to stakeholders.
  - They can capture complex nonlinear relationships between features and the target variable without requiring feature scaling.
- **Weaknesses**:
  - Decision trees are prone to overfitting, especially when the tree depth is not properly controlled or when dealing with noisy data.
  - They are sensitive to small variations in the training data, leading to instability in the learned tree structure.

Naive Bayes Model:

- **Strengths**:
  - Naive Bayes classifiers are computationally efficient and require a small amount of training data to make accurate predictions.
  - They perform well in multi-class classification tasks and can handle both numerical and categorical features.
- **Weaknesses**:
  - Naive Bayes assumes independence between features, which may not hold true in practice, leading to suboptimal performance when features are correlated.
  - It may struggle with rare or unseen feature combinations, as it assigns zero probability to such cases (zero-frequency problem).

Overall Comparison:

- The logistic regression model appears to have the highest reported performance scores in terms of accuracy, precision, and recall among the models evaluated.
- However, the comparison is limited by the availability of complete performance metrics for each model.
- Further analysis, including cross-validation and hyperparameter tuning, would be necessary to make a more comprehensive and informed comparison.
- Additionally, considering the specific requirements and constraints of the application (e.g., interpretability, computational efficiency), a different model might be preferred over others, even if it has slightly lower performance scores.