In [1]:
```
# pip install pandas
```

In [2]:
```
# pip install pandas numpy matplotlib seaborn scikit-learn
```

Explore the Dataset:

In [3]:
```
import pandas as pd

# Read dataset
df = pd.read_csv('https://github.com/YBIFoundation/Dataset/raw/main/TelecomCusto

# Display the first few rows of the dataset
print(df.head())

# Check the shape of the dataset
print("Shape of the dataset:", df.shape)

# Get information about the dataset
print(df.info())
```

```
   customerID  Gender  SeniorCitizen Partner Dependents  Tenure PhoneService  \
0  7590-VHVEG  Female              0     Yes         No       1           No
1  5575-GNVDE    Male              0      No         No      34          Yes
2  3668-QPYBK    Male              0      No         No       2          Yes
3  7795-CFOCW    Male              0      No         No      45           No
4  9237-HQITU  Female              0      No         No       2          Yes

  MultipleLines InternetService OnlineSecurity  ... DeviceProtection  \
0            No             DSL             No  ...               No
1            No             DSL            Yes  ...              Yes
2            No             DSL            Yes  ...               No
3            No             DSL            Yes  ...              Yes
4            No     Fiber optic             No  ...               No

  TechSupport StreamingTV StreamingMovies  Contract PaperlessBilling  \
0          No          No              No   Monthly              Yes
1          No          No              No  One year               No
2          No          No              No   Monthly              Yes
3         Yes          No              No  One year               No
4          No          No              No   Monthly              Yes

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

[5 rows x 21 columns]
Shape of the dataset: (7043, 21)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   customerID        7043 non-null   object
 1   Gender            7043 non-null   object
 2   SeniorCitizen     7043 non-null   int64
 3   Partner           7043 non-null   object
 4   Dependents        7043 non-null   object
 5   Tenure            7043 non-null   int64
 6   PhoneService      7043 non-null   object
 7   MultipleLines     7043 non-null   object
 8   InternetService   7043 non-null   object
 9   OnlineSecurity    7043 non-null   object
 10  OnlineBackup      7043 non-null   object
 11  DeviceProtection  7043 non-null   object
 12  TechSupport       7043 non-null   object
 13  StreamingTV       7043 non-null   object
 14  StreamingMovies   7043 non-null   object
 15  Contract          7043 non-null   object
 16  PaperlessBilling  7043 non-null   object
 17  PaymentMethod     7043 non-null   object
 18  MonthlyCharges    7043 non-null   float64
 19  TotalCharges      7043 non-null   object
 20  Churn             7043 non-null   object
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB
None
```

Data Cleaning:

In [4]:
```
# Check for missing values
print("Missing values in each column:\n", df.isnull().sum())
```

```
Missing values in each column:
 customerID          0
Gender              0
SeniorCitizen       0
Partner             0
Dependents          0
Tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges        0
Churn               0
dtype: int64
```

In [5]:
```
df1= df
```

In [6]:
```
# Fill missing values
df.fillna(method='ffill', inplace=True)
```

In [7]:
```
# Remove duplicates
df.drop_duplicates(inplace=True)
```

In [8]:
```
# Identify outliers using IQR
# Convert 'TotalCharges' column to numeric, handling errors
df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')

# Drop rows with missing values in 'TotalCharges' after conversion
df.dropna(subset=['TotalCharges'], inplace=True)

Q1 = df['TotalCharges'].quantile(0.25)
Q3 = df['TotalCharges'].quantile(0.75)
```

Data Preprocessing:

In [9]:
```
#Encoding Categorical Variables:
# One-hot encoding for categorical variables
df = pd.get_dummies(df, drop_first=True)
```

In [10]:
```
#Feature Scaling
from sklearn.preprocessing import StandardScaler
```

```
scaler = StandardScaler()
df[['MonthlyCharges', 'TotalCharges']] = scaler.fit_transform(df[['MonthlyCharge
```

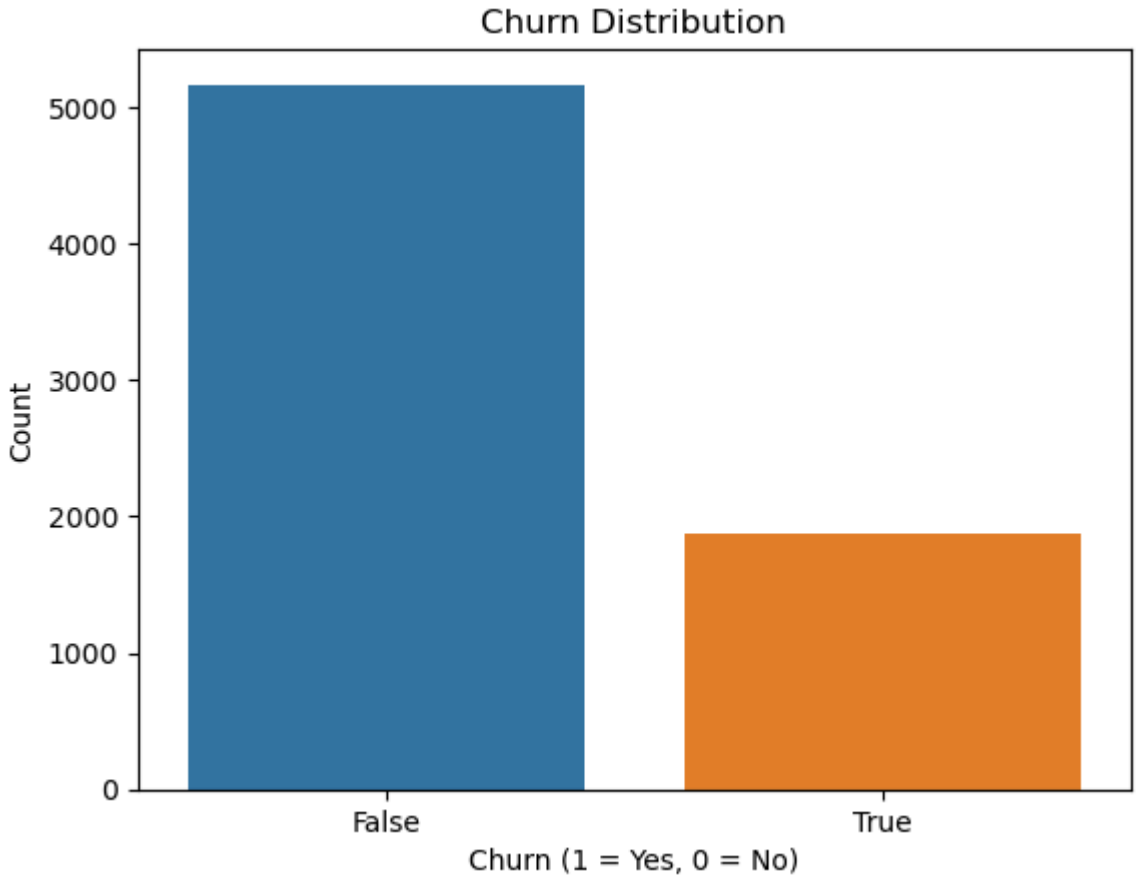
Exploratory Data Analysis (EDA)

In [11]:
```
# Summary statistics
print(df.describe())
```

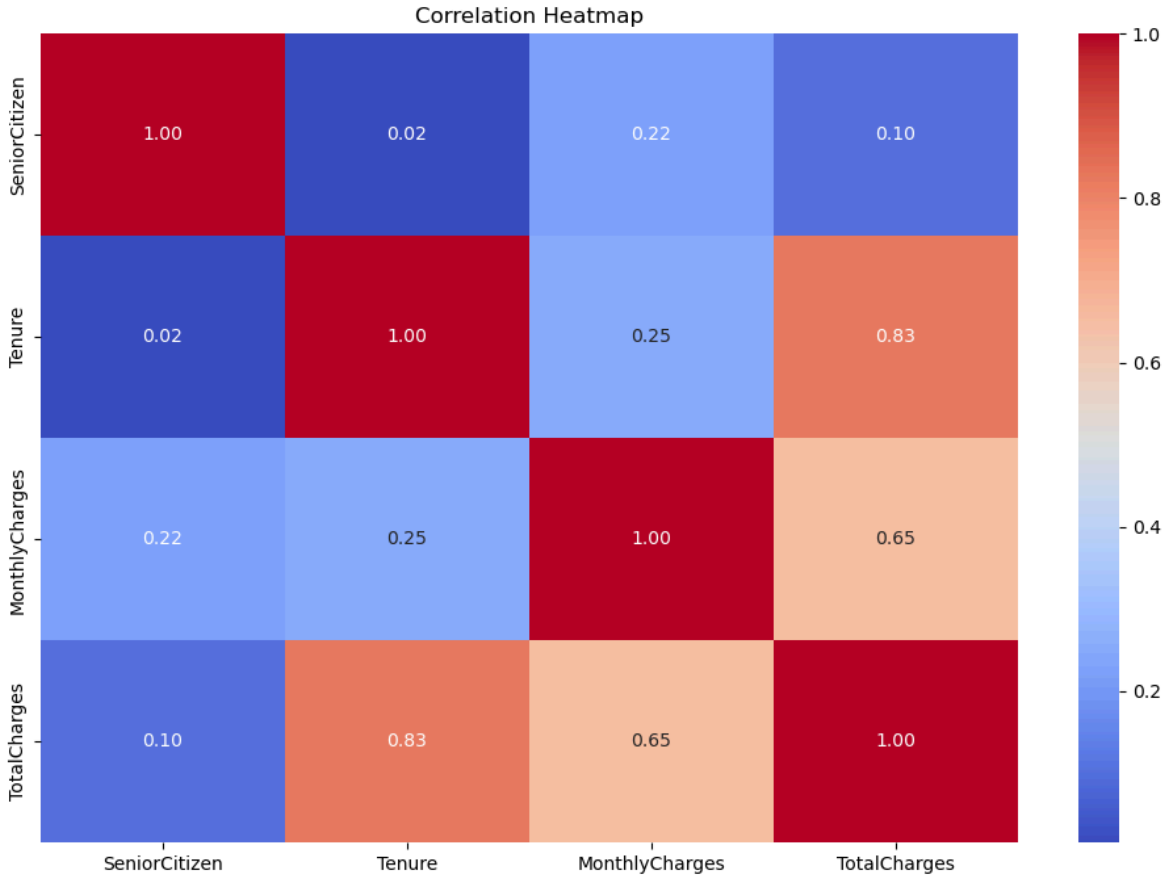
```
       SeniorCitizen       Tenure  MonthlyCharges  TotalCharges
count    7032.000000  7032.000000    7.032000e+03  7.032000e+03
mean        0.162400    32.421786    6.062651e-17 -1.119064e-16
std         0.368844    24.545260    1.000071e+00  1.000071e+00
min         0.000000     1.000000   -1.547283e+00 -9.990692e-01
25%         0.000000     9.000000   -9.709769e-01 -8.302488e-01
50%         0.000000    29.000000    1.845440e-01 -3.908151e-01
75%         0.000000    55.000000    8.331482e-01  6.668271e-01
max         1.000000    72.000000    1.793381e+00  2.824261e+00
```

Data Visualizations:

In [12]:
```
#Churn Distribution:
import matplotlib.pyplot as plt
import seaborn as sns

# Churn distribution
sns.countplot(x='Churn_Yes', data=df)
plt.title('Churn Distribution')
plt.xlabel('Churn (1 = Yes, 0 = No)')
plt.ylabel('Count')
plt.show()
```



In [13]:
```
# Correlation heatmap
plt.figure(figsize=(12, 8))
# Calculate correlation only for numeric features
numeric_df = df.select_dtypes(include=['number'])  # Select only numeric columns
sns.heatmap(numeric_df.corr(), annot=True, fmt='.2f', cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```



In [14]:
```
# Display the first few rows of the dataset
print(df1.head(2))
```

```
   customerID  Gender  SeniorCitizen Partner Dependents  Tenure PhoneService  \
0  7590-VHVEG  Female              0     Yes         No       1           No
1  5575-GNVDE    Male              0      No         No      34          Yes

  MultipleLines InternetService OnlineSecurity  ... DeviceProtection  \
0            No             DSL             No  ...               No
1            No             DSL            Yes  ...              Yes

  TechSupport StreamingTV StreamingMovies  Contract PaperlessBilling  \
0          No          No              No   Monthly              Yes
1          No          No              No  One year               No

  PaymentMethod MonthlyCharges  TotalCharges  Churn
0        Manual          29.85         29.85     No
1        Manual          56.95       1889.50     No

[2 rows x 21 columns]
```

Model Selection and Building

In [15]:
```
#Split the Data:
# Check the columns in the DataFrame
print(df.columns)
```

```
Index(['SeniorCitizen', 'Tenure', 'MonthlyCharges', 'TotalCharges',
       'customerID_0003-MKNFE', 'customerID_0004-TLHLJ',
       'customerID_0011-IGKFF', 'customerID_0013-EXCHZ',
       'customerID_0013-MHZWF', 'customerID_0013-SMEOE',
       ...
       'DeviceProtection_Yes', 'TechSupport_Yes', 'StreamingTV_Yes',
       'StreamingMovies_Yes', 'Contract_One year', 'Contract_Two year',
       'PaperlessBilling_Yes', 'PaymentMethod_Credit card (automatic)',
       'PaymentMethod_Manual', 'Churn_Yes'],
      dtype='object', length=7054)
```

In [16]:
```
# Assuming the churn column is named 'Churn'
X = df1.drop(['customerID', 'Churn'], axis=1)  # Features
y = df1['Churn']  # Target variable

# Split the data
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_
```

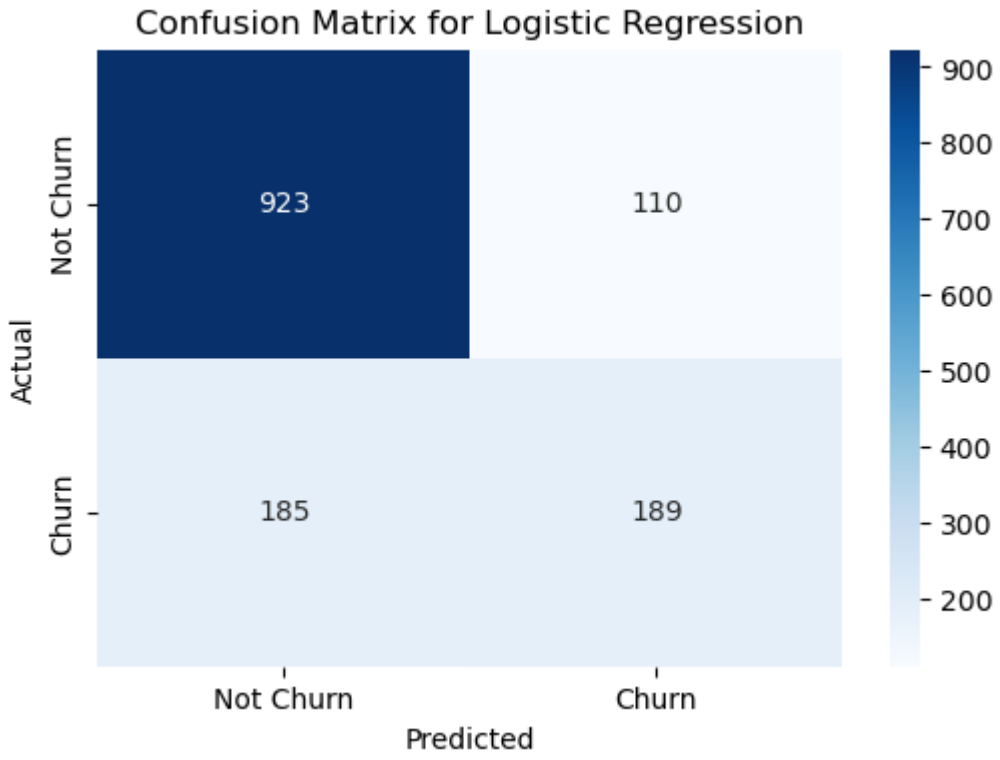
In [17]:
```
# Check the shape of the DataFrame
print("DataFrame Shape:", df.shape)

# Check the columns in the DataFrame
print("DataFrame Columns:", df.columns)

# Display the first few rows of the DataFrame
print(df.head())

# Check if 'customerID' and 'Churn' are in the DataFrame
if 'customerID' not in df.columns:
    print("'customerID' column is missing.")
if 'Churn' not in df.columns:
    print("'Churn' column is missing.")
```

```
DataFrame Shape: (7032, 7054)
DataFrame Columns: Index(['SeniorCitizen', 'Tenure', 'MonthlyCharges', 'TotalChar
ges',
       'customerID_0003-MKNFE', 'customerID_0004-TLHLJ',
       'customerID_0011-IGKFF', 'customerID_0013-EXCHZ',
       'customerID_0013-MHZWF', 'customerID_0013-SMEOE',
       ...
       'DeviceProtection_Yes', 'TechSupport_Yes', 'StreamingTV_Yes',
       'StreamingMovies_Yes', 'Contract_One year', 'Contract_Two year',
       'PaperlessBilling_Yes', 'PaymentMethod_Credit card (automatic)',
       'PaymentMethod_Manual', 'Churn_Yes'],
      dtype='object', length=7054)
   SeniorCitizen  Tenure  MonthlyCharges  TotalCharges  customerID_0003-MKNFE  \
0              0       1       -1.161694     -0.994194                  False
1              0      34       -0.260878     -0.173740                  False
2              0       2       -0.363923     -0.959649                  False
3              0      45       -0.747850     -0.195248                  False
4              0       2        0.196178     -0.940457                  False

   customerID_0004-TLHLJ  customerID_0011-IGKFF  customerID_0013-EXCHZ  \
0                  False                  False                  False
1                  False                  False                  False
2                  False                  False                  False
3                  False                  False                  False
4                  False                  False                  False

   customerID_0013-MHZWF  customerID_0013-SMEOE  ...  DeviceProtection_Yes  \
0                  False                  False  ...                 False
1                  False                  False  ...                  True
2                  False                  False  ...                 False
3                  False                  False  ...                  True
4                  False                  False  ...                 False

   TechSupport_Yes  StreamingTV_Yes  StreamingMovies_Yes  Contract_One year  \
0            False            False                False              False
1            False            False                False               True
2            False            False                False              False
3             True            False                False               True
4            False            False                False              False

   Contract_Two year  PaperlessBilling_Yes  \
0              False                  True
1              False                 False
2              False                  True
3              False                 False
4              False                  True

   PaymentMethod_Credit card (automatic)  PaymentMethod_Manual  Churn_Yes
0                                  False                  True      False
1                                  False                  True      False
2                                  False                  True       True
3                                  False                 False      False
4                                  False                  True       True

[5 rows x 7054 columns]
'customerID' column is missing.
'Churn' column is missing.
```

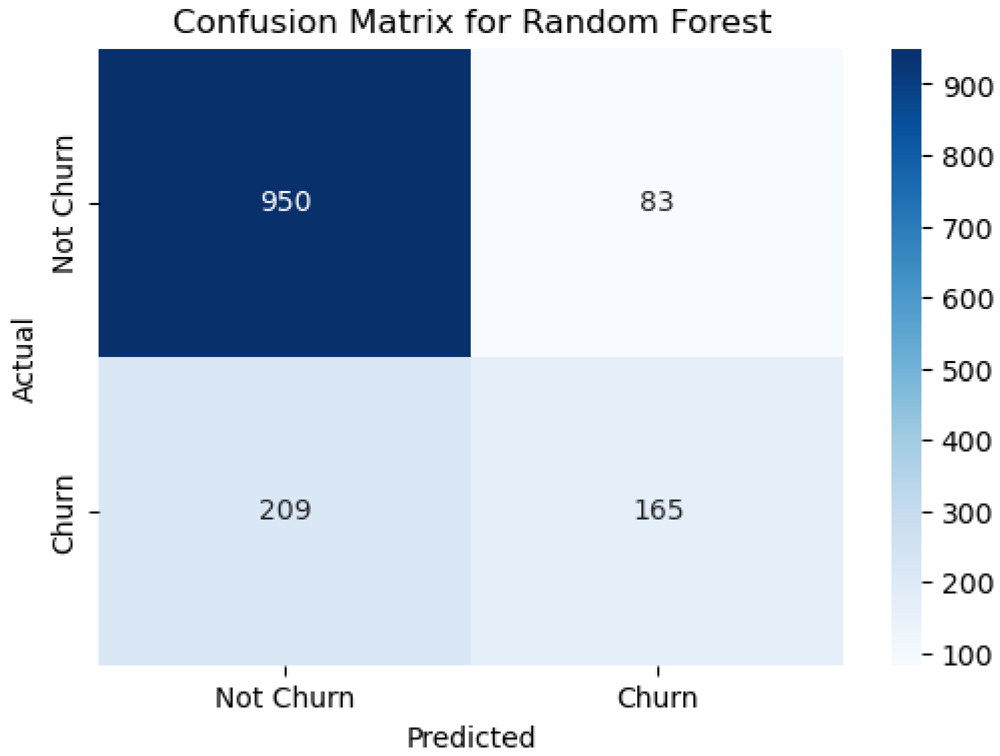
In [18]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, confusion_mat
import seaborn as sns
import matplotlib.pyplot as plt

# Read dataset
df = pd.read_csv('https://github.com/YBIFoundation/Dataset/raw/main/TelecomCusto

# Data preprocessing steps (as previously discussed)
df.fillna(method='ffill', inplace=True)
df.drop_duplicates(inplace=True)
df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')
df.dropna(subset=['TotalCharges'], inplace=True)
df = pd.get_dummies(df, drop_first=True)

# Define features and target variable
X = df.drop('Churn_Yes', axis=1)  # Assuming 'Churn_Yes' is the target variable
y = df['Churn_Yes']

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_

# Initialize models
models = {
    'Logistic Regression': LogisticRegression(max_iter=1000),
    'Random Forest': RandomForestClassifier(),
    'Support Vector Classifier': SVC()
}

# Train and validate models
results = {}
for model_name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    # Calculate accuracy
    accuracy = accuracy_score(y_test, y_pred)
    results[model_name] = accuracy

    # Print classification report
    print(f"Classification Report for {model_name}:\n", classification_report(y_

    # Confusion Matrix
    cm = confusion_matrix(y_test, y_pred)
    plt.figure(figsize=(6, 4))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Not Churn',
    plt.title(f'Confusion Matrix for {model_name}')
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.show()

# Compare model performances
print("Model Performance Comparison:")
for model_name, accuracy in results.items():
    print(f"{model_name}: {accuracy:.4f}")
```

```
Classification Report for Logistic Regression:
              precision    recall  f1-score   support

       False       0.83      0.89      0.86      1033
        True       0.63      0.51      0.56       374

    accuracy                           0.79      1407
   macro avg       0.73      0.70      0.71      1407
weighted avg       0.78      0.79      0.78      1407
```

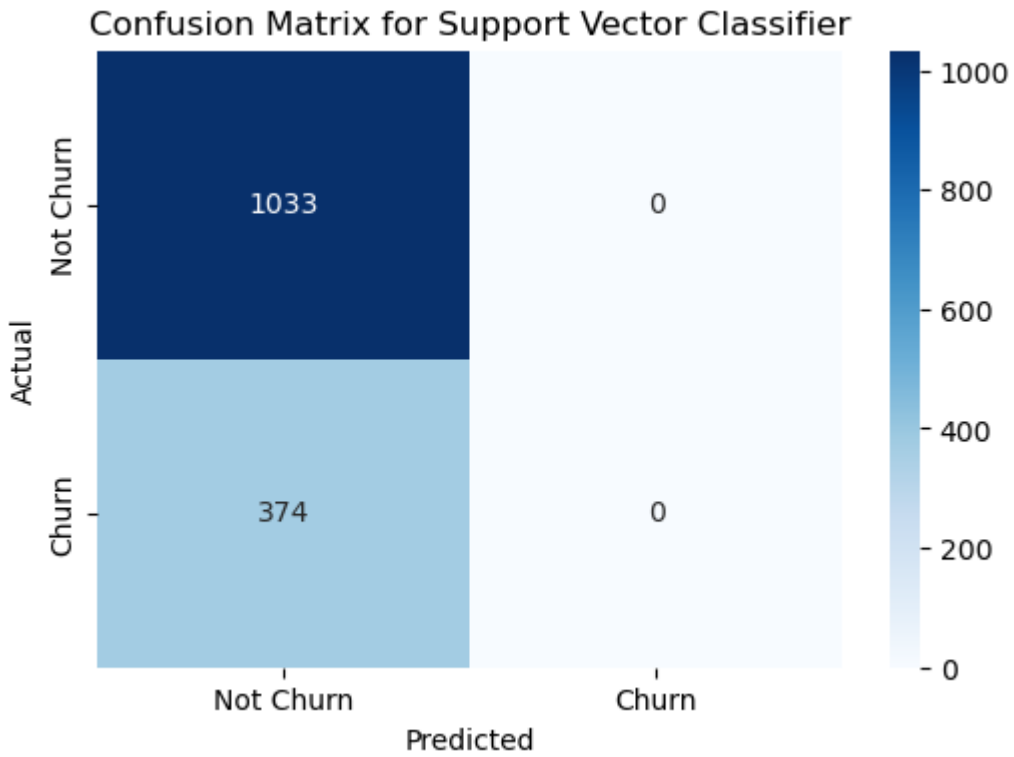



```
Classification Report for Random Forest:
              precision    recall  f1-score   support

       False       0.82      0.92      0.87      1033
        True       0.67      0.44      0.53       374

    accuracy                           0.79      1407
   macro avg       0.74      0.68      0.70      1407
weighted avg       0.78      0.79      0.78      1407
```




```
Classification Report for Support Vector Classifier:
              precision    recall  f1-score   support

       False       0.73      1.00      0.85      1033
        True       0.00      0.00      0.00       374

    accuracy                           0.73      1407
   macro avg       0.37      0.50      0.42      1407
weighted avg       0.54      0.73      0.62      1407
```

```
c:\Users\johnh\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1469: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\johnh\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1469: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\johnh\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1469: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```




```
Model Performance Comparison:
Logistic Regression: 0.7903
Random Forest: 0.7925
Support Vector Classifier: 0.7342
```

In [19]:

```python
from sklearn.model_selection import RandomizedSearchCV

# Simplified Hyperparameter Grid
rf_param_grid = {
    'n_estimators': [50, 100],
    'max_depth': [None, 10],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2]
}

# RandomizedSearchCV with fewer iterations
rf_random_search = RandomizedSearchCV(
    RandomForestClassifier(),
    param_distributions=rf_param_grid,
    n_iter=10,  # Test only 10 random combinations
    cv=3,       # Reduce cross-validation folds
    n_jobs=-1,  # Use all CPU cores
    verbose=1
)

rf_random_search.fit(X_train, y_train)

# Best parameters and model evaluation
print("Best parameters for Random Forest:", rf_random_search.best_params_)
best_rf_model = rf_random_search.best_estimator_
y_pred_rf = best_rf_model.predict(X_test)
print("Random Forest Test Accuracy:", accuracy_score(y_test, y_pred_rf))
print("Classification Report:\n", classification_report(y_test, y_pred_rf))
```

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits
Best parameters for Random Forest: {'n_estimators': 100, 'min_samples_split': 2,
'min_samples_leaf': 1, 'max_depth': None}
Random Forest Test Accuracy: 0.7917555081734187
Classification Report:
              precision    recall  f1-score   support

       False       0.82      0.92      0.87      1033
        True       0.66      0.44      0.53       374

    accuracy                           0.79      1407
   macro avg       0.74      0.68      0.70      1407
weighted avg       0.78      0.79      0.78      1407
```

In [20]:
```
pip install Flask
```

```
Requirement already satisfied: Flask in c:\users\johnh\anaconda3\lib\site-package
s (2.2.2)
Requirement already satisfied: Werkzeug>=2.2.2 in c:\users\johnh\anaconda3\lib\si
te-packages (from Flask) (2.2.3)
Requirement already satisfied: Jinja2>=3.0 in c:\users\johnh\anaconda3\lib\site-p
ackages (from Flask) (3.1.2)
Requirement already satisfied: itsdangerous>=2.0 in c:\users\johnh\anaconda3\lib
\site-packages (from Flask) (2.0.1)
Requirement already satisfied: click>=8.0 in c:\users\johnh\anaconda3\lib\site-pa
ckages (from Flask) (8.0.4)
Requirement already satisfied: colorama in c:\users\johnh\anaconda3\lib\site-pack
ages (from click>=8.0->Flask) (0.4.6)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\johnh\anaconda3\lib\si
te-packages (from Jinja2>=3.0->Flask) (2.1.1)
Note: you may need to restart the kernel to use updated packages.
```

In [21]:
```
import joblib

# Save the best model
joblib.dump(best_rf_model, 'best_rf_model.pkl')
```

Out[21]:
```
['best_rf_model.pkl']
```

In [23]:
```
from flask import Flask, request, jsonify
import joblib
import pandas as pd

# Load the trained model
model = joblib.load('best_rf_model.pkl')

# Initialize Flask app
app = Flask(__name__)

# Define a route for predictions
@app.route('/predict', methods=['POST'])
def predict():
    try:
        # Get the JSON data from the request
        data = request.get_json(force=True)

        # Check if data is valid
        if not data:
            return jsonify({'error': 'No input data provided'}), 400
```

```
        # Convert the data into a DataFrame
        input_data = pd.DataFrame(data, index=[0])

        # Ensure that the input features match the model's expected features
        expected_columns = model.feature_names_in_  # Use the features used duri
        if list(input_data.columns) != list(expected_columns):
            return jsonify({'error': 'Input features do not match model expectat

        # Make predictions
        prediction = model.predict(input_data)
        prediction_proba = model.predict_proba(input_data)[:, 1]  # Probability

        # Return the prediction and probability
        return jsonify({
            'prediction': int(prediction[0]),
            'probability': float(prediction_proba[0])
        })

    except Exception as e:
        return jsonify({'error': str(e)}), 500

# Run the app
if __name__ == '__main__':
    app.run(debug=True)
```

```
 * Serving Flask app '__main__'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with watchdog (windowsapi)
An exception has occurred, use %tb to see the full traceback.

SystemExit: 1
```