Churn Prediction and Business Insight with Data Science at MTN Nigeria

This project was built using the “MTN Nigeria Customer Churn” dataset taken from Kaggle (https://www.kaggle.com/datasets/oluwademiladeadeniyi/mtn-nigeria-customer-churn/). MTN Nigeria is one of the largest telecommunications service providers in Nigeria and the main branch of the MTN group, with millions of customers across the country.

With this workflow, I hope to generate actionable insights to help MTN Nigeria reduce churn and improve customer retention.

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import random
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import StratifiedKFold
from sklearn.feature_selection import RFECV
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.metrics import confusion_matrix
import warnings
from imblearn.over_sampling import SMOTE
```

In [2]:
```
data = pd.read_csv('Dataset\mtn_customer_churn.csv')

df = data.copy()
```

```
<>:1: SyntaxWarning: invalid escape sequence '\m'
<>:1: SyntaxWarning: invalid escape sequence '\m'
C:\Users\azizl\AppData\Local\Temp\ipykernel_15584\2209089821.py:1: SyntaxWarning: in
valid escape sequence '\m'
  data = pd.read_csv('Dataset\mtn_customer_churn.csv')
```

Problem Understanding

In [3]:
```
df.head()
```

Out[3]:

	Customer ID	Full Name	Date of Purchase	Age	State	MTN Device	Gender	Satisfaction Rate	Customer Review
0	CUST0001	Ngozi Berry	Jan-25	27	Kwara	4G Router	Male	2	Fair
1	CUST0002	Zainab Baker	Mar-25	16	Abuja (FCT)	Mobile SIM Card	Female	2	Fair
2	CUST0003	Saidu Evans	Mar-25	21	Sokoto	5G Broadband Router	Male	1	Poor
3	CUST0003	Saidu Evans	Mar-25	21	Sokoto	Mobile SIM Card	Male	1	Poor
4	CUST0003	Saidu Evans	Mar-25	21	Sokoto	Broadband MiFi	Male	1	Poor

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 974 entries, 0 to 973
Data columns (total 17 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   Customer ID                974 non-null    object 
 1   Full Name                  974 non-null    object 
 2   Date of Purchase           974 non-null    object 
 3   Age                        974 non-null    int64  
 4   State                      974 non-null    object 
 5   MTN Device                 974 non-null    object 
 6   Gender                     974 non-null    object 
 7   Satisfaction Rate          974 non-null    int64  
 8   Customer Review            974 non-null    object 
 9   Customer Tenure in months  974 non-null    int64  
 10  Subscription Plan          974 non-null    object 
 11  Unit Price                 974 non-null    int64  
 12  Number of Times Purchased  974 non-null    int64  
 13  Total Revenue              974 non-null    int64  
 14  Data Usage                 974 non-null    float64
 15  Customer Churn Status      974 non-null    object 
 16  Reasons for Churn          284 non-null    object 
dtypes: float64(1), int64(6), object(10)
memory usage: 129.5+ KB
```

In [5]:

```
df['Customer Churn Status'].value_counts()
```

Out[5]:
```
Customer Churn Status
No     690
Yes    284
Name: count, dtype: int64
```

In [6]:
```
persentage_churn = (round((len(df[df['Customer Churn Status'] == 'Yes']) / len(df))

print(f'Percentage of customers who churned: {persentage_churn}%')
```

```
Percentage of customers who churned: 29.16%
```

Customer churn is one of the biggest concerns for telecom companies, especially in competitive markets like Nigeria.

This issue requires serious attention. We need to address it effectively or at least reduce the churn rate. But how? First, we must gain a deeper understanding of two critical metrics:

1. Customer Satisfaction Score (CSAT)
2. Monthly Recurring Revenue

These metrics are key to identifying the underlying issues. To do this, we should begin by visualizing the data to uncover trends and patterns. Understanding why customers are more likely to churn is essential. Currently, the churn rate stands at 29%, which is significantly high for telecom companies in Nigeria.

For context, the average churn rate for telecom companies in the United States is around 21%, according to ExplodingTopics.com.

Data Preparation

We need to first understand the data and identify what insights can be extracted from it.

In [7]:
```
# Check for missing values
missing_values = df.isnull().sum()
print("Missing values in each column:\n", missing_values)
```

```
Missing values in each column:
 Customer ID                    0
Full Name                      0
Date of Purchase               0
Age                            0
State                          0
MTN Device                     0
Gender                         0
Satisfaction Rate              0
Customer Review                0
Customer Tenure in months      0
Subscription Plan              0
Unit Price                     0
Number of Times Purchased      0
Total Revenue                  0
Data Usage                     0
Customer Churn Status          0
Reasons for Churn            690
dtype: int64
```

In [8]:

```
# Duplicate values
duplicate_values = df.duplicated().sum()
print(f"Number of duplicate rows: {duplicate_values}")
```

```
Number of duplicate rows: 0
```

In [9]:

```
# Check for inconsistent data
data.describe()
```

Out[9]:

	Age	Satisfaction Rate	Customer Tenure in months	Unit Price	Number of Times Purchased	Total Revenue	
count	974.000000	974.000000	974.000000	974.000000	974.000000	9.740000e+02	974.0
mean	48.043121	2.947639	31.422998	19196.663244	10.564682	2.046696e+05	99.3
std	17.764307	1.384219	17.191256	25586.726985	5.709427	3.247855e+05	57.7
min	16.000000	1.000000	1.000000	350.000000	1.000000	3.500000e+02	0.8
25%	32.000000	2.000000	17.000000	5500.000000	5.000000	3.300000e+04	47.6
50%	49.000000	3.000000	31.000000	14500.000000	11.000000	1.080000e+05	103.3
75%	63.750000	4.000000	47.000000	24000.000000	15.000000	2.610000e+05	149.6
max	80.000000	5.000000	60.000000	150000.000000	20.000000	3.000000e+06	200.0

In [10]:

```
# Get information about the metrics (CSAT and MRR Churn)
mrr_churn = data[data['Customer Churn Status'] == 'Yes']['Total Revenue'].sum()
mrr_total = data['Total Revenue'].sum()
mrr_churn_percentage = (mrr_churn / mrr_total) * 100
print(f"Monthly Recurring Revenue (MRR) Churn: {mrr_churn_percentage:.2f}%")

csat = data['Satisfaction Rate'].value_counts()
```

```
csat_good = csat[csat.index.isin([4,5])].sum()
csat_total = data['Satisfaction Rate'].notna().sum()
csat_percentage = (csat_good / csat_total) * 100
print(f"Customer Satisfaction (CSAT): {csat_percentage:.2f}%")
```

```
Monthly Recurring Revenue (MRR) Churn: 29.09%
Customer Satisfaction (CSAT): 38.81%
```

There is nothing concerning in the data quality itself (the dataset is clean). However, from a business perspective, we should be concerned. Metrics such as MRR (Monthly Recurring Revenue) and CSAT (Customer Satisfaction Score) indicate that the business is in poor condition. Therefore, it is crucial to find a solution, starting with visualizing the data for deeper insights.

In [11]:
```
numerical = data.select_dtypes(include=['number']).columns
data[numerical].columns
```

Out[11]:
```
Index(['Age', 'Satisfaction Rate', 'Customer Tenure in months', 'Unit Price',
       'Number of Times Purchased', 'Total Revenue', 'Data Usage'],
      dtype='object')
```

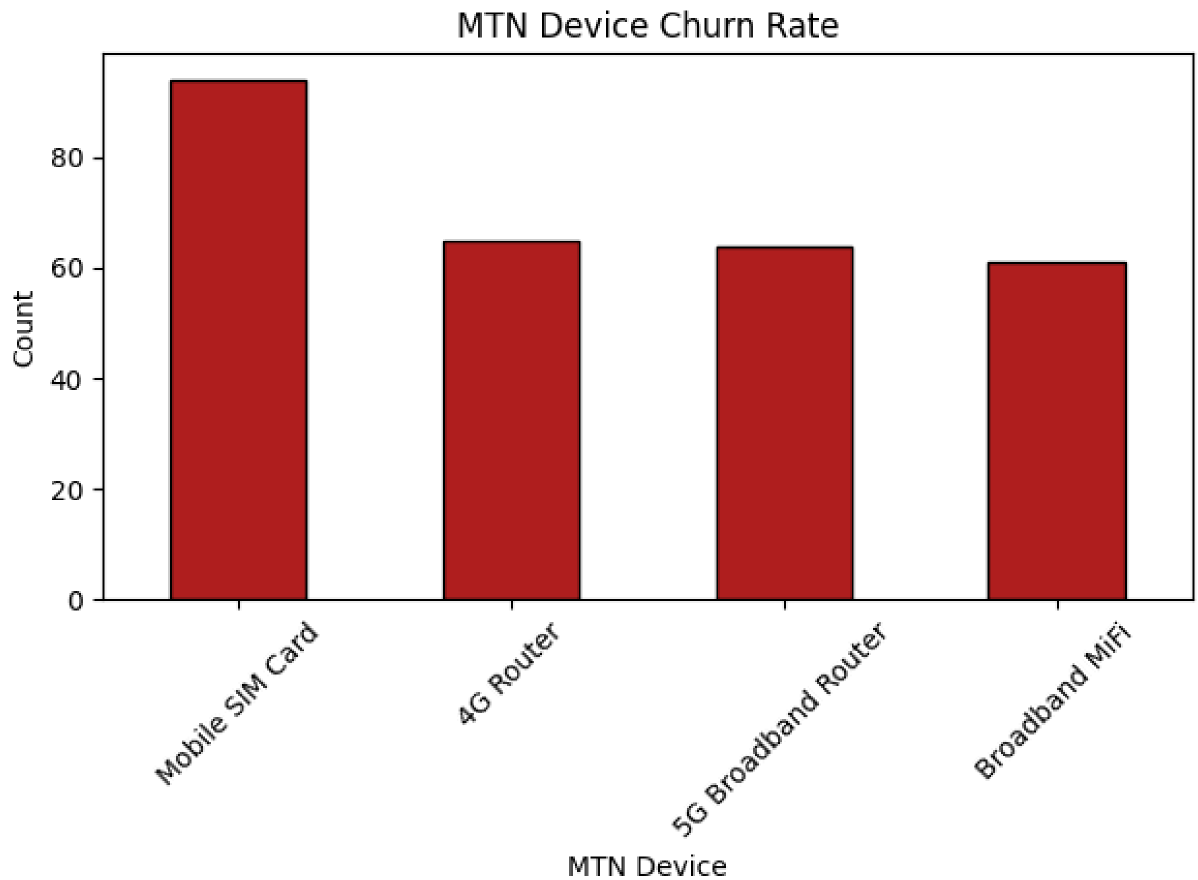
In [12]:
```
categorical = data.select_dtypes(include=['object']).columns
data[categorical].columns
```

Out[12]:
```
Index(['Customer ID', 'Full Name', 'Date of Purchase', 'State', 'MTN Device',
       'Gender', 'Customer Review', 'Subscription Plan',
       'Customer Churn Status', 'Reasons for Churn'],
      dtype='object')
```

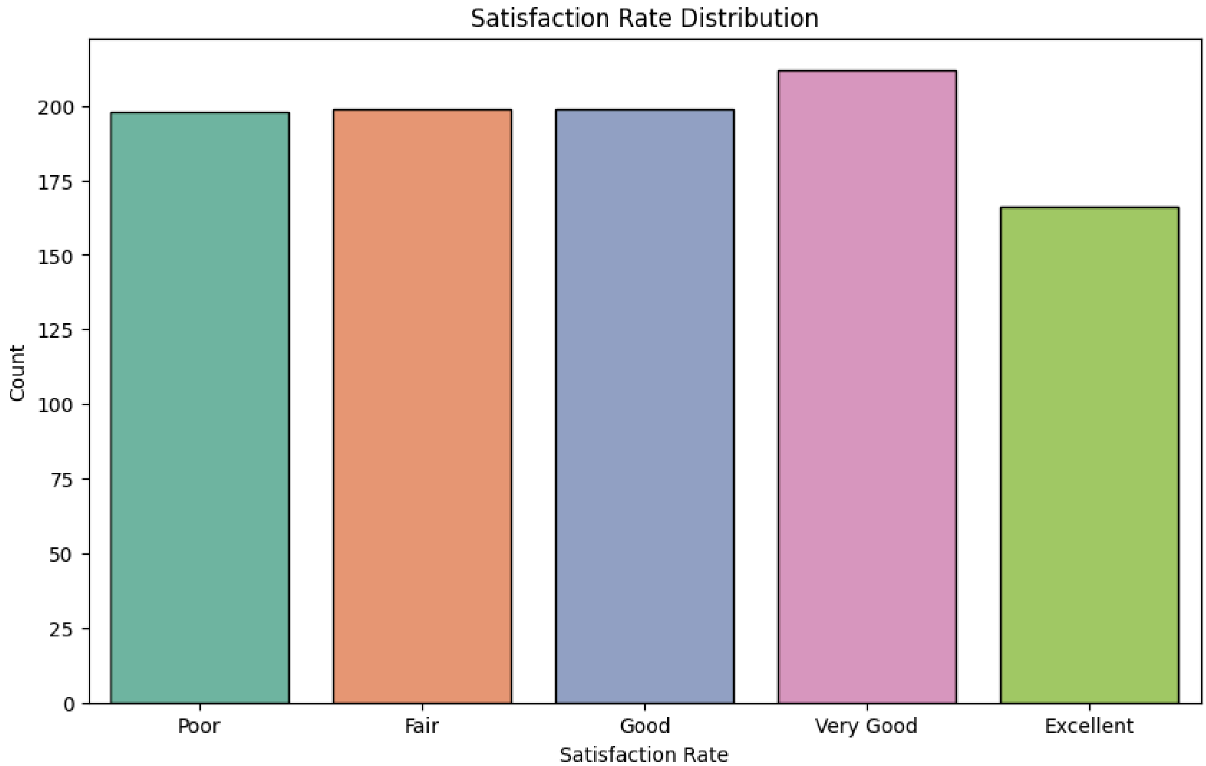
In [13]:
```
data['Customer Review'].value_counts()
```

Out[13]:
```
Customer Review
Very Good    212
Fair         199
Good         199
Poor         198
Excellent    166
Name: count, dtype: int64
```

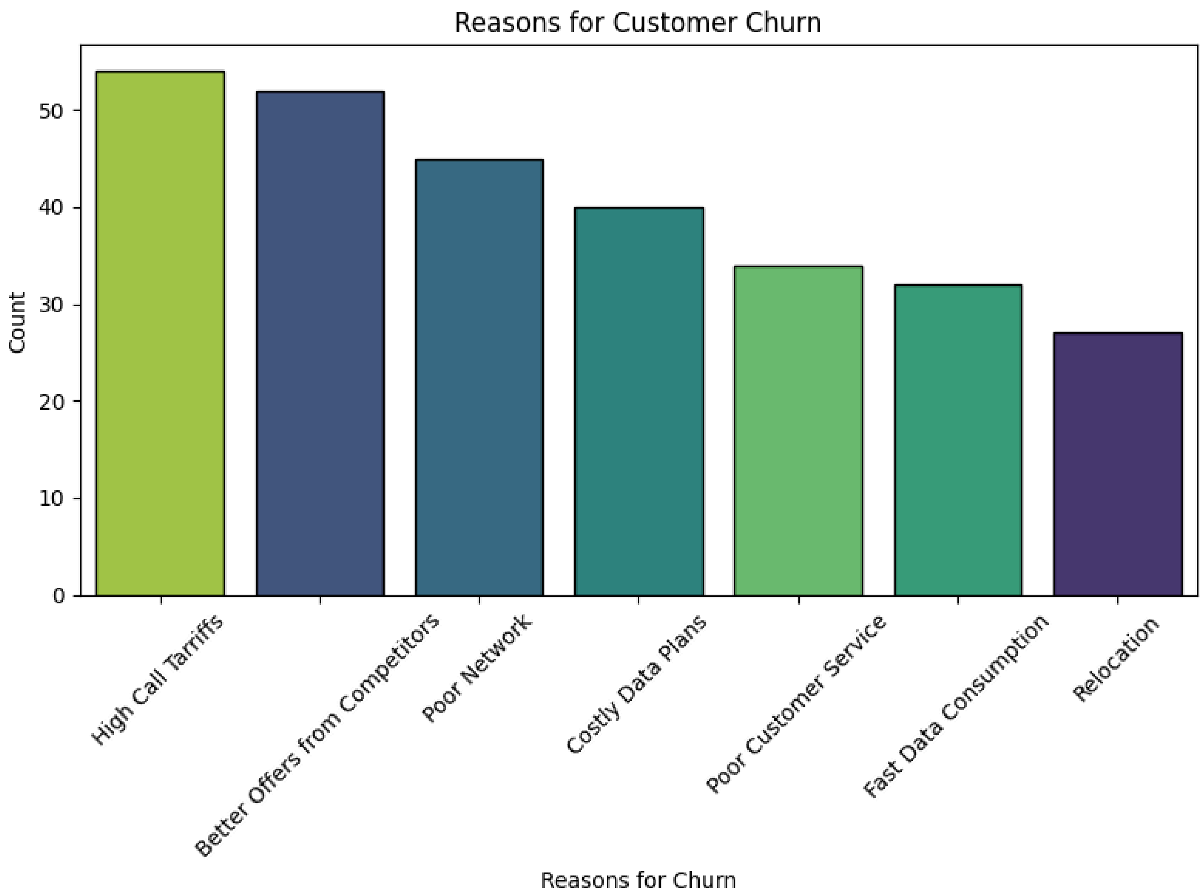
In [14]:
```
mtn_churn = data[data['Customer Churn Status'] == 'Yes']['MTN Device'].value_counts
mtn_churn.plot(kind='bar', color='firebrick', edgecolor='black')
plt.title('MTN Device Churn Rate')
plt.xlabel('MTN Device')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



In [15]:
```
satisfaction_rate = data['Satisfaction Rate'].value_counts()
plt.figure(figsize=(10, 6))
sns.countplot(x='Satisfaction Rate', hue='Satisfaction Rate' ,data=data, palette='S
plt.title('Satisfaction Rate Distribution')
plt.xlabel('Satisfaction Rate')
plt.ylabel('Count')
plt.xticks(ticks=[0, 1, 2, 3, 4], labels=["Poor", "Fair", "Good", "Very Good", "Exc
plt.show()
```



In [16]:
```
order = data[data['Customer Churn Status'] == 'Yes']['Reasons for Churn'].value_cou
plt.figure(figsize=(8, 6))
sns.countplot(data=data[data['Customer Churn Status'] == 'Yes'], x='Reasons for Chu
plt.title('Reasons for Customer Churn')
plt.xlabel('Reasons for Churn')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



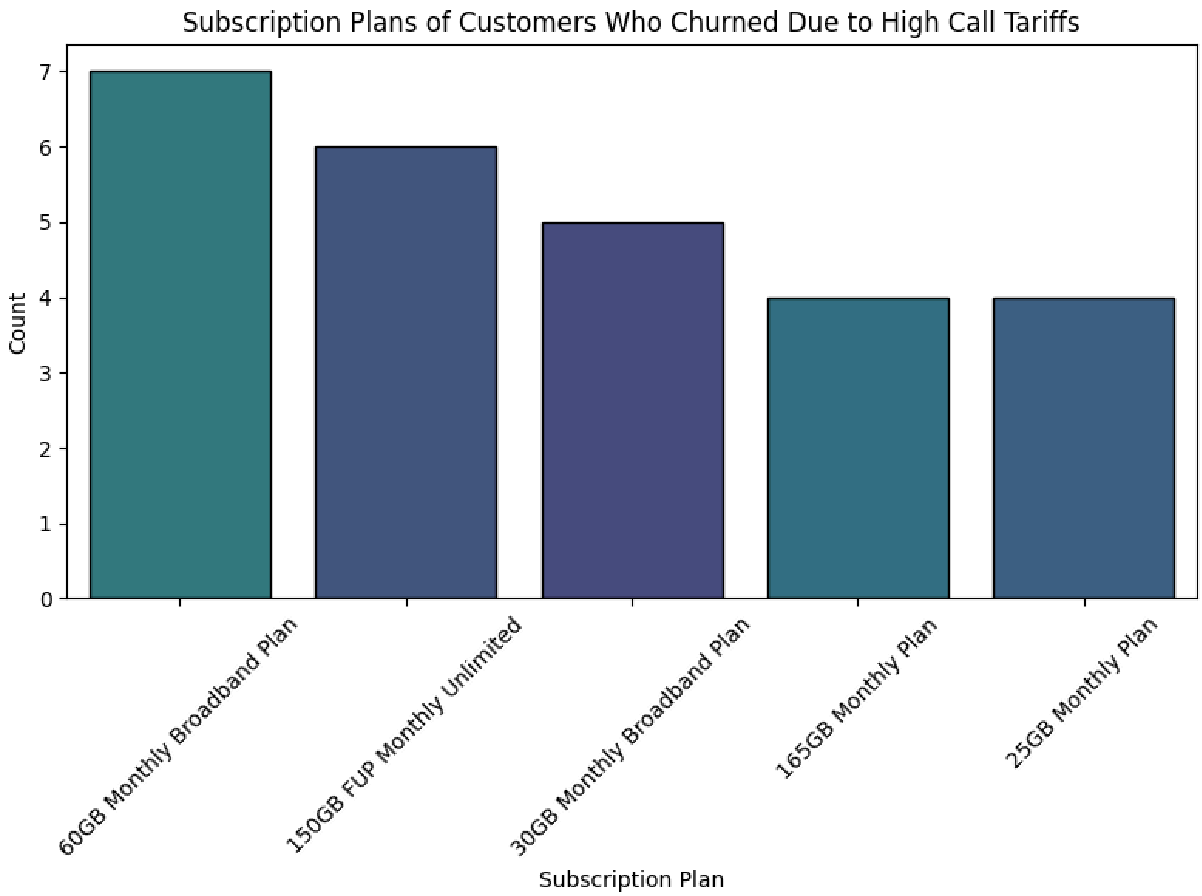
In [17]:
```
high_call = data[data['Customer Churn Status'] == 'Yes'][data[data['Customer Churn
print(f"Number of customers who churned due to high call tariffs: {len(high_call)}"
high_call.head(2)
```

```
Number of customers who churned due to high call tariffs: 54
```

Out[17]:

	Customer ID	Full Name	Date of Purchase	Age	State	MTN Device	Gender	Satisfaction Rate	Custome Reviev
183	CUST0097	Ese Perez	Feb-25	55	Zamfara	5G Broadband Router	Female	3	Goo
220	CUST0116	Bola Garcia	Mar-25	30	Benue	Mobile SIM Card	Male	4	Ver Goo

In [18]:
```
order = high_call['Subscription Plan'].value_counts().head(5).index
plt.figure(figsize=(8, 6))
sns.countplot(data=high_call, x='Subscription Plan', hue='Subscription Plan', order
plt.title('Subscription Plans of Customers Who Churned Due to High Call Tariffs')
plt.xlabel('Subscription Plan')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



In [19]:
```
len(data[data['Customer Churn Status'] == 'Yes']['Customer Tenure in months'])
```

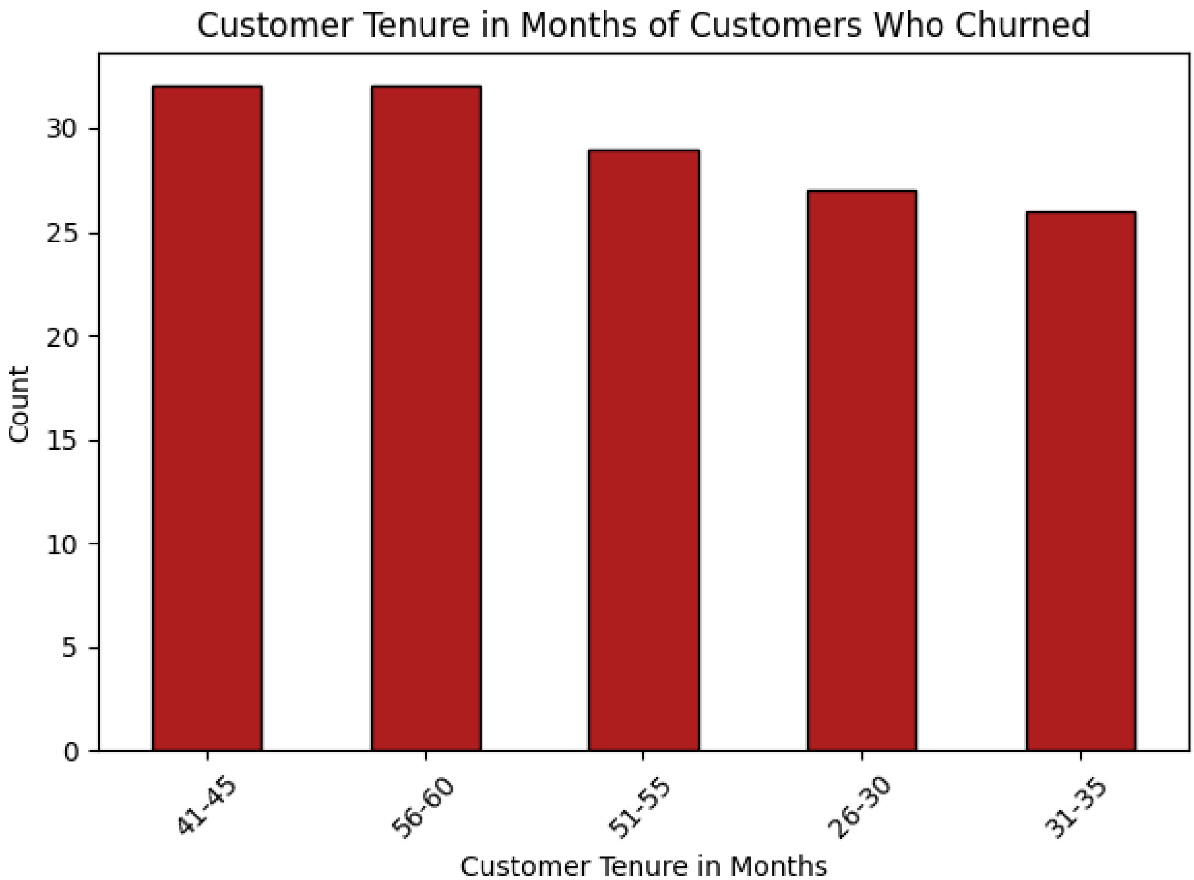
Out[19]:
```
284
```

In [20]:
```
bins = [0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60]
labels = ['0-5', '6-10', '11-15', '16-20', '21-25', '26-30', '31-35', '36-40', '41-

tenure_binned = pd.cut(data[data['Customer Churn Status'] == 'Yes']['Customer Tenur
tenure_binned.value_counts()
```

Out[20]:
```
Customer Tenure in months
41-45    32
56-60    32
51-55    29
26-30    27
31-35    26
6-10     24
16-20    24
36-40    23
21-25    19
46-50    18
11-15    17
0-5      13
Name: count, dtype: int64
```

In [21]:
```
tenure_binned.value_counts().head(5).plot(kind='bar', color='firebrick', edgecolor=
plt.title('Customer Tenure in Months of Customers Who Churned')
plt.xlabel('Customer Tenure in Months')
```

```
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



In [22]:
```
state_churn = data[data['Customer Churn Status'] == 'Yes']['State'].value_counts()
state_churn.columns = ['State', 'Count']
state_churn = state_churn.reset_index()
state_churn.head(5)
```

Out[22]:

	State	count
0	Abuja (FCT)	15
1	Imo	15
2	Kebbi	14
3	Yobe	13
4	Benue	13

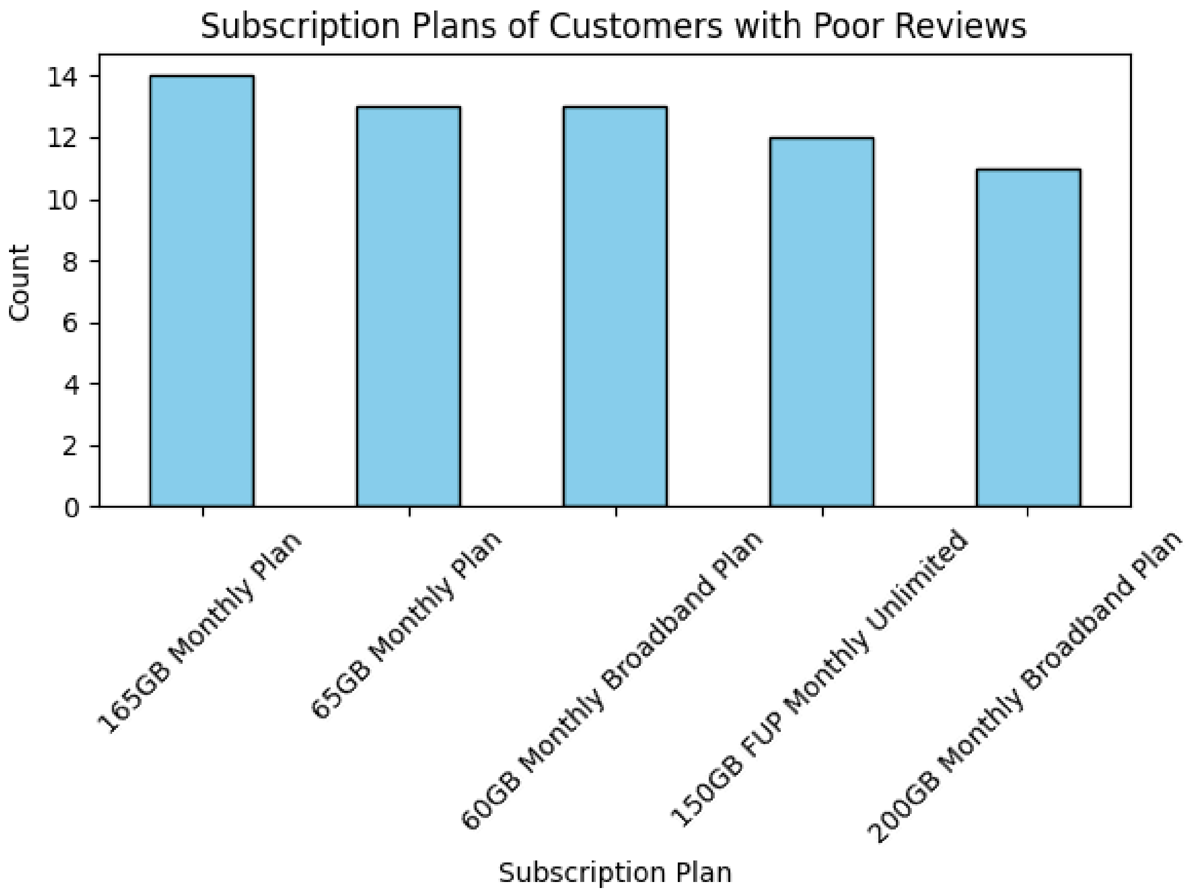
In [23]:
```
bottom_reviews = data[data['Customer Churn Status'] == 'Yes'][data[data['Customer C
bottom_reviews.head()
```

Out[23]:

	Customer ID	Full Name	Date of Purchase	Age	State	MTN Device	Gender	Satisfaction Rate	Customer Review
0	CUST0001	Ngozi Berry	Jan-25	27	Kwara	4G Router	Male	2	Fair
1	CUST0002	Zainab Baker	Mar-25	16	Abuja (FCT)	Mobile SIM Card	Female	2	Fair
16	CUST0011	Ejiro Griffith	Feb-25	72	Bauchi	4G Router	Female	2	Fair
17	CUST0011	Ejiro Griffith	Feb-25	72	Bauchi	5G Broadband Router	Female	2	Fair
18	CUST0011	Ejiro Griffith	Feb-25	72	Bauchi	Broadband MiFi	Female	2	Fair

In [24]:

```
bottom_reviews['Subscription Plan'].value_counts().head(5).plot(kind='bar', color='
plt.title('Subscription Plans of Customers with Poor Reviews')
plt.xlabel('Subscription Plan')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



In [25]:
```
from sklearn.preprocessing import LabelEncoder

encoder = LabelEncoder()
```

In [26]:
```
data.head(2)
```

Out[26]:

	Customer ID	Full Name	Date of Purchase	Age	State	MTN Device	Gender	Satisfaction Rate	Customer Review	Cust Tenu mo
0	CUST0001	Ngozi Berry	Jan-25	27	Kwara	4G Router	Male	2	Fair	
1	CUST0002	Zainab Baker	Mar-25	16	Abuja (FCT)	Mobile SIM Card	Female	2	Fair	

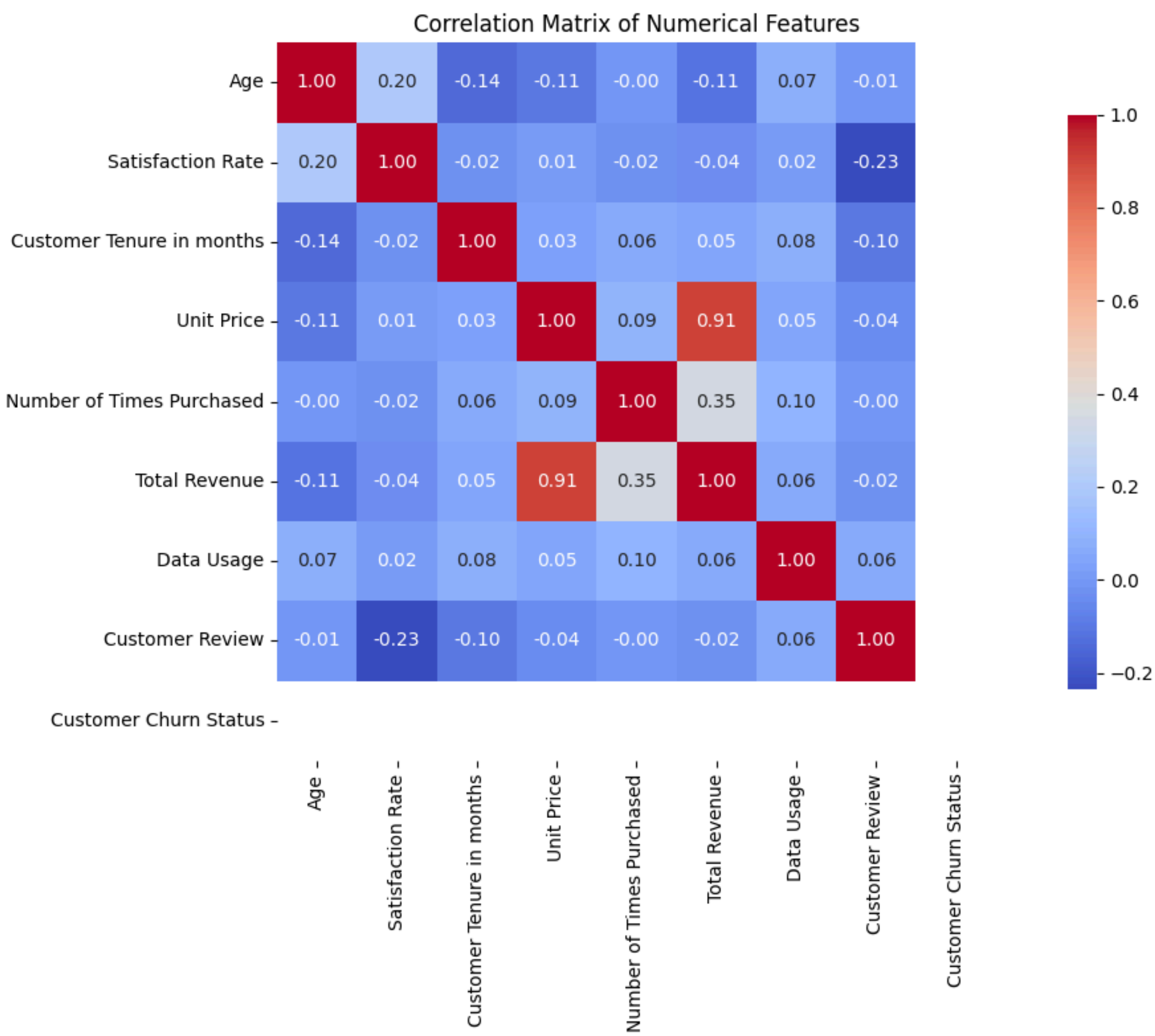
In [27]:
```
data_encode = data[["Customer Review", "Customer Churn Status"]].apply(encoder.fit_

data_matrix = pd.concat([data[numerical], data_encode], axis=1)
data_matrix.head()
```

Out[27]:

	Age	Satisfaction Rate	Customer Tenure in months	Unit Price	Number of Times Purchased	Total Revenue	Data Usage	Customer Review	Customer Churn Status
0	27	2	2	35000	19	665000	44.48	1	1
1	16	2	22	5500	12	66000	19.79	1	1
2	21	1	60	20000	8	160000	9.64	3	0
3	21	1	60	500	8	4000	197.05	3	0
4	21	1	60	9000	15	135000	76.34	3	0

In [28]:

```
matrix = data_matrix[data_matrix['Customer Churn Status'] == 1].corr()
plt.figure(figsize=(12, 8))
sns.heatmap(matrix, annot=True, fmt='.2f', cmap='coolwarm', square=True, cbar_kws={
plt.title('Correlation Matrix of Numerical Features')
plt.tight_layout()
plt.show()
```



Machine Learning Workflow

We aim to predict customers who are at risk of churning.

In [29]:
```
data.head(2)
```

Out[29]:

	Customer ID	Full Name	Date of Purchase	Age	State	MTN Device	Gender	Satisfaction Rate	Customer Review	Cust Tenu mo
0	CUST0001	Ngozi Berry	Jan-25	27	Kwara	4G Router	Male	2	Fair	
1	CUST0002	Zainab Baker	Mar-25	16	Abuja (FCT)	Mobile SIM Card	Female	2	Fair	

In [30]:
```
data[numerical].head(2)
```

Out[30]:

	Age	Satisfaction Rate	Customer Tenure in months	Unit Price	Number of Times Purchased	Total Revenue	Data Usage
0	27	2	2	35000	19	665000	44.48
1	16	2	22	5500	12	66000	19.79

In [31]:
```
data_numerical = data[numerical]
data_numerical = data_numerical.drop(columns=['Age', 'Number of Times Purchased'])
data_numerical.head(2)
```

Out[31]:

	Satisfaction Rate	Customer Tenure in months	Unit Price	Total Revenue	Data Usage
0	2	2	35000	665000	44.48
1	2	22	5500	66000	19.79

In [32]:
```
data_encode = data[["State", "Subscription Plan", "Customer Review", "Customer Chur
new_data = pd.concat([data_numerical, data_encode], axis=1)
new_data.head()
```

Out[32]:

	Satisfaction Rate	Customer Tenure in months	Unit Price	Total Revenue	Data Usage	State	Subscription Plan	Customer Review	Custom Chu Stat
0	2	2	35000	665000	44.48	22	7	1	
1	2	22	5500	66000	19.79	1	3	1	
2	1	60	20000	160000	9.64	31	5	3	
3	1	60	500	4000	197.05	31	8	3	
4	1	60	9000	135000	76.34	31	15	3	

In [33]:

```
X_train, X_test, y_train, y_test = train_test_split(new_data.drop(columns=['Custome
```

In [34]:

```
def evaluate_model(model, X_train, y_train, X_test, y_test):
    cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
    selector = RFECV(estimator=model, step=1, cv=cv, scoring='accuracy')
    selector.fit(X_train, y_train)
    selected_features = X_train.columns[selector.support_]
    X_train_sel = X_train[selected_features]
    X_test_sel = X_test[selected_features]
    final_model = model
    final_model.fit(X_train_sel, y_train)
    score = final_model.score(X_test_sel, y_test)
    return score

def evaluate_model_2(model, X_train, y_train, X_test, y_test):
    selector = SelectKBest(score_func=f_classif, k='all')
    selector.fit(X_train, y_train)
    selected_features = X_train.columns[selector.get_support()]
    X_train_sel = X_train[selected_features]
    X_test_sel = X_test[selected_features]
    final_model = model
    final_model.fit(X_train_sel, y_train)
    score = final_model.score(X_test_sel, y_test)
    return score
```

In [35]:

```
model_dt = DecisionTreeClassifier(random_state=42)

score = evaluate_model(model_dt, X_train, y_train, X_test, y_test)
print(f"Decision Tree Classifier Accuracy: {score:.2f}")
```

```
Decision Tree Classifier Accuracy: 0.62
```

In [36]:

```
model_rf = RandomForestClassifier(random_state=42)

score = evaluate_model(model_rf,  X_train, y_train, X_test, y_test)
print(f"Random Forest Classifier Accuracy: {score:.2f}")
```

```
Random Forest Classifier Accuracy: 0.71
```

In [37]:

```
model_gbc = GradientBoostingClassifier(n_estimators=100, random_state=42)
```

```
score = evaluate_model_2(model_gbc,  X_train, y_train, X_test, y_test)
print(f"Random Gradient Boosting Classifier Accuracy: {score:.2f}")
```

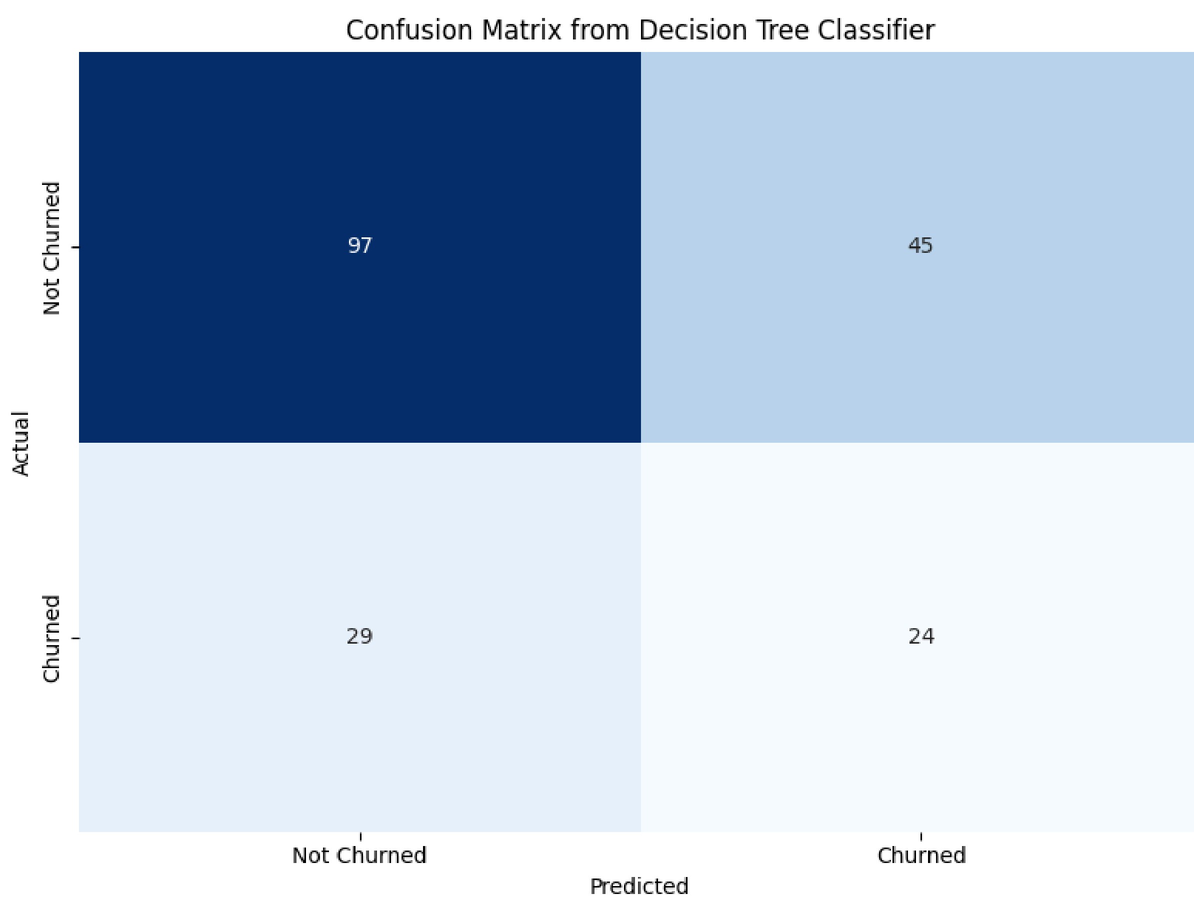
```
Random Gradient Boosting Classifier Accuracy: 0.71
```

In [38]:
```
model_knn = KNeighborsClassifier(n_neighbors=5)

score = evaluate_model_2(model_knn,  X_train, y_train, X_test, y_test)
print(f"Random KNN Accuracy: {score:.2f}")
```

```
Random KNN Accuracy: 0.73
```

Evaluation Model

In [39]:
```
cm_dt = confusion_matrix(y_test, model_dt.predict(X_test))
plt.figure(figsize=(8, 6))
sns.heatmap(cm_dt, annot=True, fmt='d', cmap='Blues', cbar=False, xticklabels=['Not
plt.title('Confusion Matrix from Decision Tree Classifier')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.tight_layout()
plt.show()
```

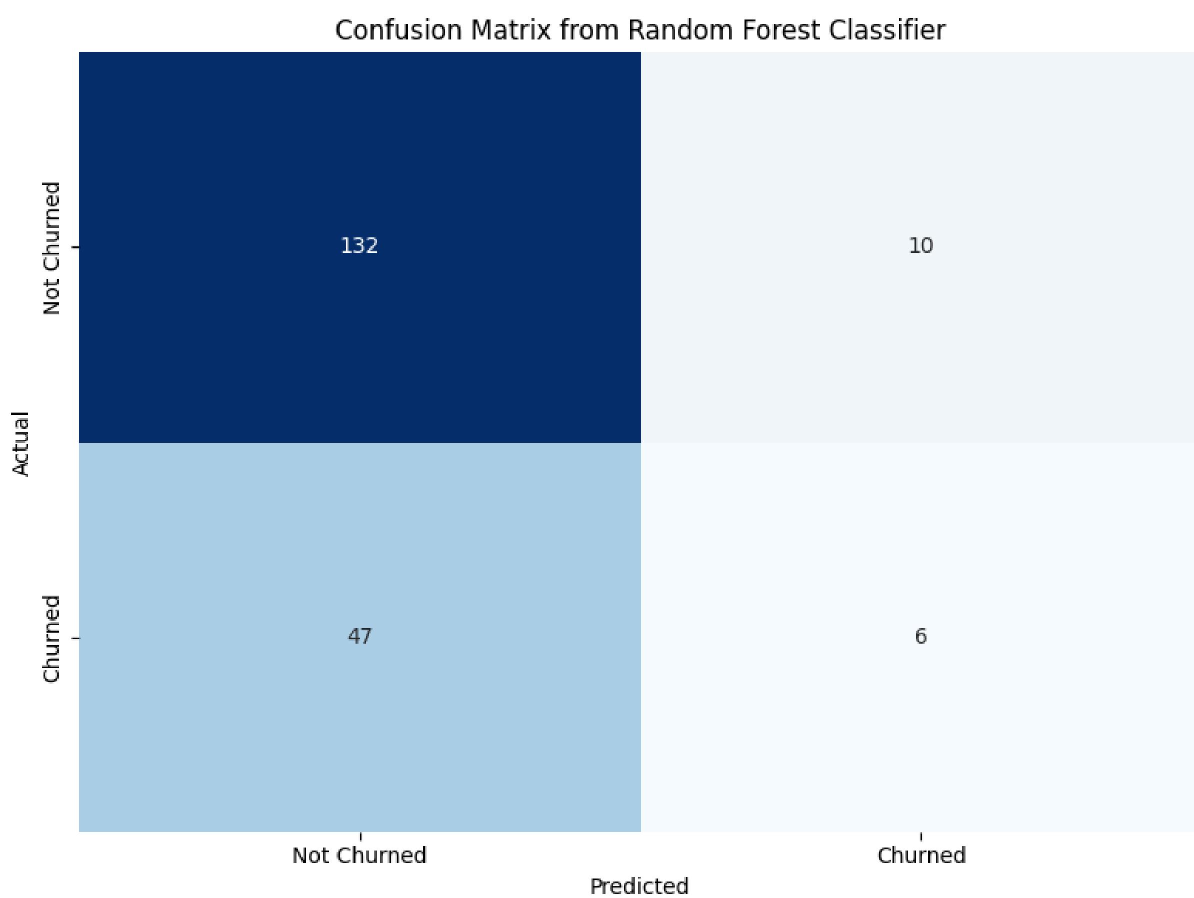


In [40]:
```
recall_churn_dt = cm_dt[1, 1] / (cm_dt[1, 1] + cm_dt[1, 0])
precision_churn_dt = cm_dt[1, 1] / (cm_dt[1, 1] + cm_dt[0, 1])
f1_churn_dt = 2 * (recall_churn_dt * precision_churn_dt) / (recall_churn_dt + preci

print("Decision Tree Classifier Metrics:")
```

```
print("---------------------------------")
print(f"Recall for Churned Customers: {recall_churn_dt:.2f}")
print(f"Precision for Churned Customers: {precision_churn_dt:.2f}")
print(f"F1 Score for Churned Customers: {f1_churn_dt:.2f}")
```

```
Decision Tree Classifier Metrics:
---------------------------------
Recall for Churned Customers: 0.45
Precision for Churned Customers: 0.35
F1 Score for Churned Customers: 0.39
```

In [41]:
```
cm_rf = confusion_matrix(y_test, model_rf.predict(X_test))
plt.figure(figsize=(8, 6))
sns.heatmap(cm_rf, annot=True, fmt='d', cmap='Blues', cbar=False, xticklabels=['Not
plt.title('Confusion Matrix from Random Forest Classifier')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.tight_layout()
plt.show()
```

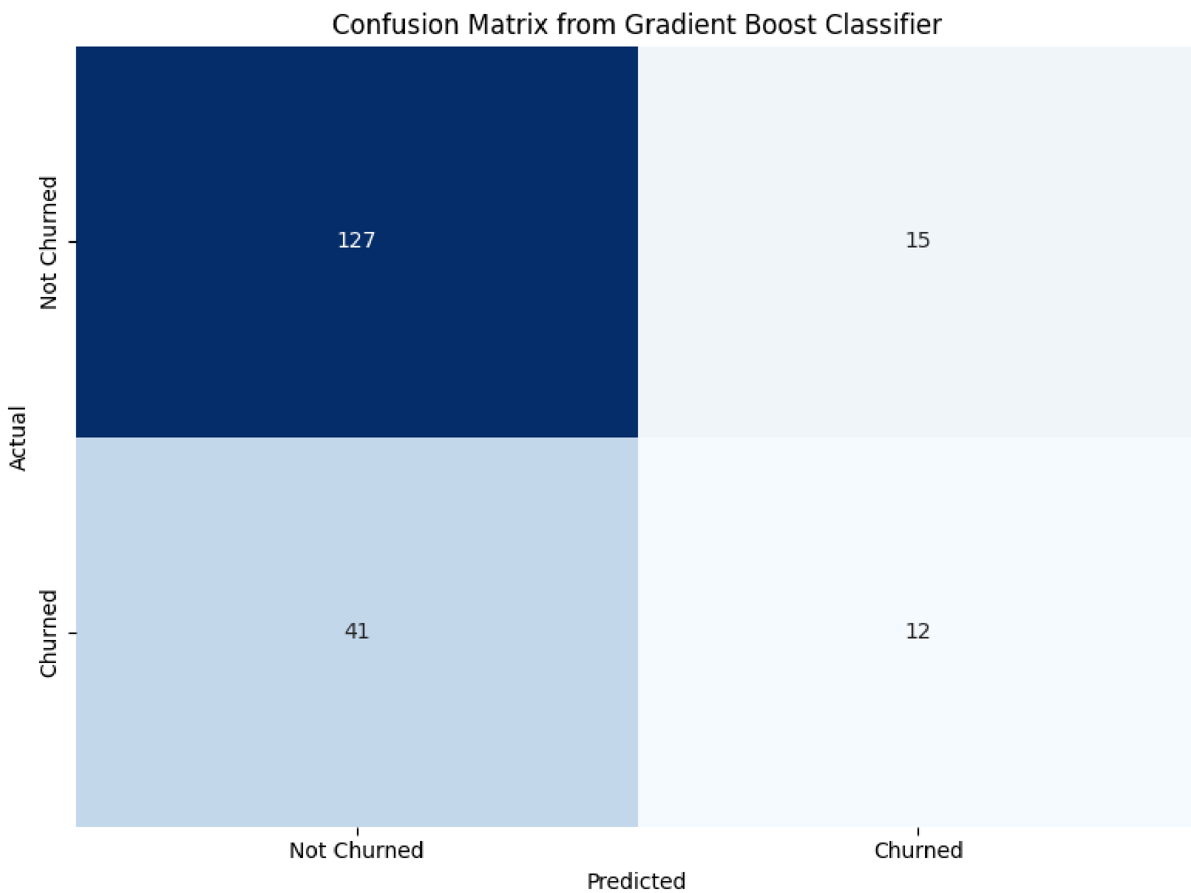


In [42]:
```
recall_churn_rf = cm_rf[1, 1] / (cm_rf[1, 1] + cm_rf[1, 0])
precision_churn_rf = cm_rf[1, 1] / (cm_rf[1, 1] + cm_rf[0, 1])
f1_churn_rf = 2 * (recall_churn_rf * precision_churn_rf) / (recall_churn_rf + preci

print("Random Forest Classifier Metrics:")
print("---------------------------------")
print(f"Recall for Churned Customers: {recall_churn_rf:.2f}")
```

```python
print(f"Precision for Churned Customers: {precision_churn_rf:.2f}")
print(f"F1 Score for Churned Customers: {f1_churn_rf:.2f}")
```

```
Random Forest Classifier Metrics:
---------------------------------
Recall for Churned Customers: 0.11
Precision for Churned Customers: 0.38
F1 Score for Churned Customers: 0.17
```

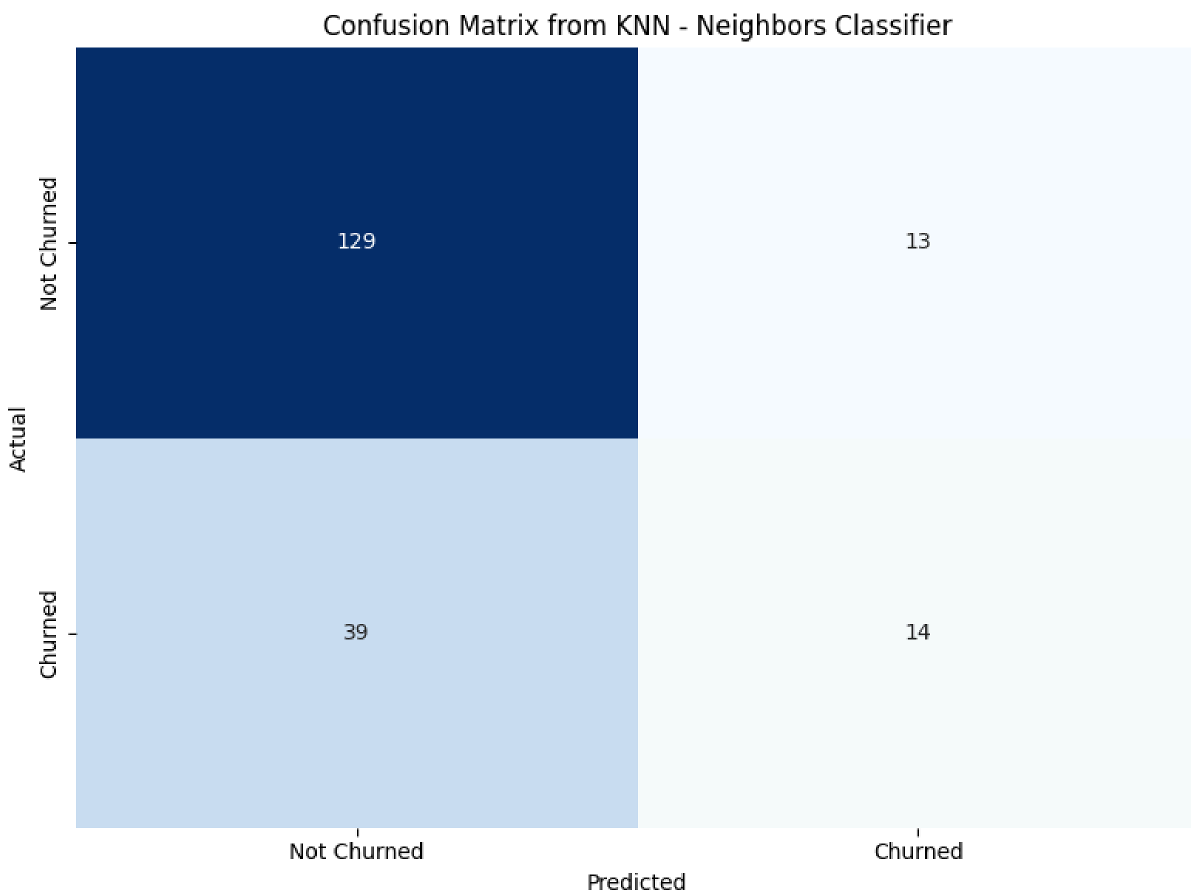
In [43]:
```python
cm_gbc = confusion_matrix(y_test, model_gbc.predict(X_test))
plt.figure(figsize=(8, 6))
sns.heatmap(cm_gbc, annot=True, fmt='d', cmap='Blues', cbar=False, xticklabels=['No
plt.title('Confusion Matrix from Gradient Boost Classifier')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.tight_layout()
plt.show()
```



In [44]:
```python
recall_churn_gbc = cm_gbc[1, 1] / (cm_gbc[1, 1] + cm_gbc[1, 0])
precision_churn_gbc = cm_gbc[1, 1] / (cm_gbc[1, 1] + cm_gbc[0, 1])
f1_churn_gbc = 2 * (recall_churn_gbc * precision_churn_gbc) / (recall_churn_gbc + p

print("Gradient Boost Classifier Metrics:")
print("---------------------------------")
print(f"Recall for Churned Customers: {recall_churn_gbc:.2f}")
print(f"Precision for Churned Customers: {precision_churn_gbc:.2f}")
print(f"F1 Score for Churned Customers: {f1_churn_gbc:.2f}")
```

```
Gradient Boost Classifier Metrics:
---------------------------------
Recall for Churned Customers: 0.23
Precision for Churned Customers: 0.44
F1 Score for Churned Customers: 0.30
```

In [45]:
```python
cm_knn = confusion_matrix(y_test, model_knn.predict(X_test))
plt.figure(figsize=(8, 6))
sns.heatmap(cm_knn, annot=True, fmt='d', cmap='Blues', cbar=False, xticklabels=['No
plt.title('Confusion Matrix from KNN - Neighbors Classifier')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.tight_layout()
plt.show()
```



In [46]:
```python
recall_churn_knn = cm_knn[1, 1] / (cm_knn[1, 1] + cm_knn[1, 0])
precision_churn_knn = cm_knn[1, 1] / (cm_knn[1, 1] + cm_knn[0, 1])
f1_churn_knn = 2 * (recall_churn_knn * precision_churn_knn) / (recall_churn_knn + p

print("KNN Classifier Metrics:")
print("-----------------------")
print(f"Recall for Churned Customers: {recall_churn_knn:.2f}")
print(f"Precision for Churned Customers: {precision_churn_knn:.2f}")
print(f"F1 Score for Churned Customers: {f1_churn_knn:.2f}")
```

```
KNN Classifier Metrics:
-----------------------
Recall for Churned Customers: 0.26
Precision for Churned Customers: 0.52
F1 Score for Churned Customers: 0.35
```

Prediction

We will chose Decision Tree as the base model for our approach. Why? Because our primary goal is to quickly identify users who are likely to churn (Decision tree has higher Recall compared to other models). Even if the model misclassifies some retained users as potential churners, it's acceptable. Our objective is to retain as many users as possible.

While this approach may lead to an increase in marketing costs due to targeting users who may not actually churn, it can still effectively help reduce the overall churn rate. Our target is to bring the current churn rate down from 29% to below the 21% industry benchmark.

In [47]:
```
def predict_churn(model, x_sample):
    y_sample = model.predict([x_sample])
    return y_sample
```

In [48]:
```
random_number = random.randrange(1,len(X_test)-1)
x_sample = X_test.iloc[random_number]
y_sample = y_test.iloc[random_number]
print(f"Sample Data:\n{x_sample}\n")
print(f"Sample Data Churn Status: {y_sample}\n")
```

```
Sample Data:
Satisfaction Rate              3.00
Customer Tenure in months     51.00
Unit Price                 16000.00
Total Revenue              16000.00
Data Usage                    99.44
State                          7.00
Subscription Plan             19.00
Customer Review                2.00
Name: 951, dtype: float64

Sample Data Churn Status: 1
```

In [49]:
```
warnings.filterwarnings("ignore", category=UserWarning)
prediction = predict_churn(model_dt, x_sample)
if prediction[0] == 1:
    print("🚨 Risk Alert: This customer is likely to churn.")
    print("Recommended Strategy: Proactively reach out with a personalized retentio
else:
    print("✅ Customer Retention Likely.")
    print("Recommended Strategy: Maintain engagement with regular updates and value
```

```
🚨 Risk Alert: This customer is likely to churn.
Recommended Strategy: Proactively reach out with a personalized retention offer or e
xclusive service benefits.
```