



# DOMAIN-ORIENTED CASE STUDY TELECOM CHURN

LE THANH DAT





# DATA CLEANING & OUTLIERS HANDLING



# DATA CLEANING

- I drop some columns below because each of them store the same data for all row (e.g circle\_id are always 109, last\_date\_of\_month\_6 are always 6/30/2014, ...)
- After that, all the remaining data are numeric. Therefore, I fill 0 to all missing data

```
# Drop non-essential columns
columns_to_drop = [
    'circle_id', 'last_date_of_month_6', 'last_date_of_month_7',
    'last_date_of_month_8', 'last_date_of_month_9',
    'date_of_last_rech_6', 'date_of_last_rech_7',
    'date_of_last_rech_8', 'date_of_last_rech_9',
    'date_of_last_rech_data_6', 'date_of_last_rech_data_7',
    'date_of_last_rech_data_8', 'date_of_last_rech_data_9'
]

data = data.drop(columns=columns_to_drop)

# Fill all missing values with 0
data = data.fillna(0)
data.head()
```

# OUTLIERS HANDLING

Set the mobile\_number as data index

- Calculate the IQR for each feature to identify outliers.
- Cap values at the 5th and 95th percentiles for selected features, which helps retain data while limiting the impact of extreme values.

```
data.set_index('mobile_number', inplace=True)

# Define a function to handle outliers by capping at percentiles
def handle_outliers(df, columns, lower_percentile=0.05, upper_percentile=0.95):
    for col in columns:
        lower_bound = df[col].quantile(lower_percentile)
        upper_bound = df[col].quantile(upper_percentile)
        # Cap the outliers
        df[col] = np.where(df[col] < lower_bound, lower_bound, df[col])
        df[col] = np.where(df[col] > upper_bound, upper_bound, df[col])
    return df

# Apply outlier handling
data = handle_outliers(data, data.columns)

# Check data statistics after handling outliers
print(data[data.columns].describe())
```



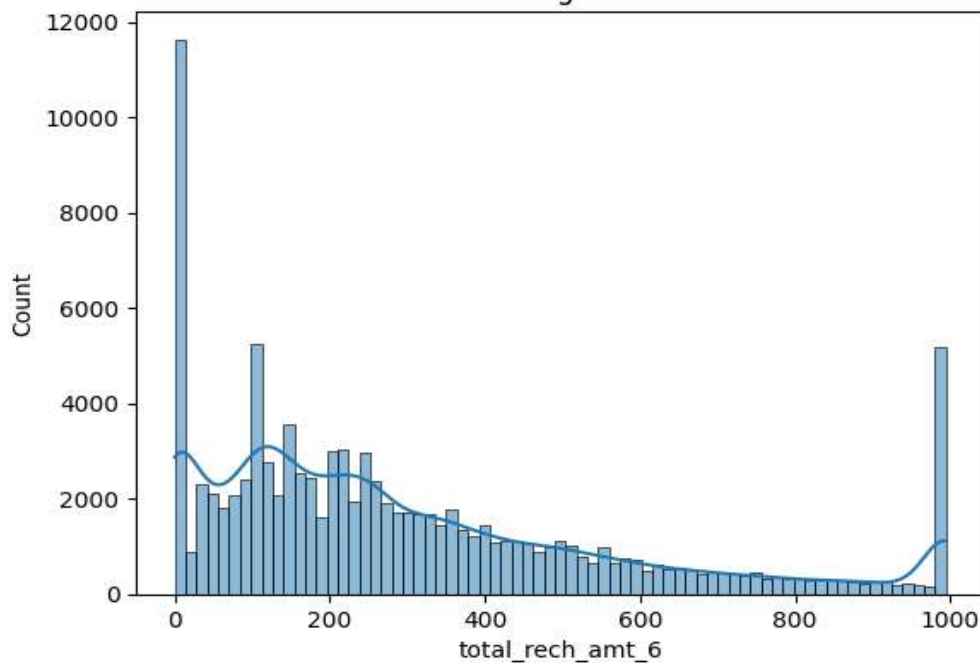
# EXPLANATORY DATA ANALYSIS (EDA)



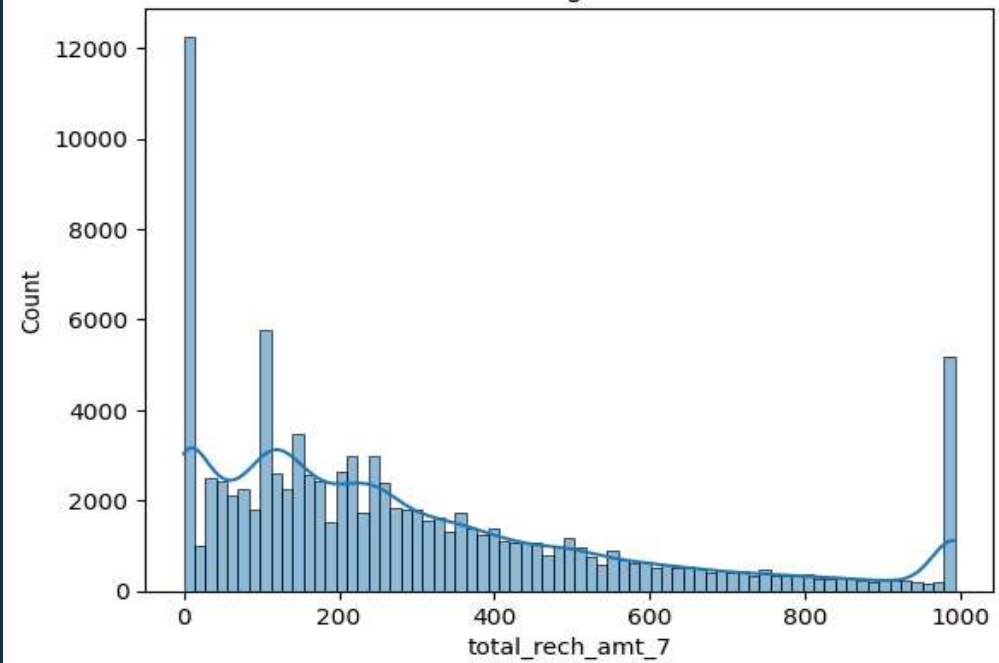
# Distribution of Recharge Amount in June and July

Checking the distribution and trend of recharge amount to see the its pattern over time

Distribution of Recharge Amount in Month 6



Distribution of Recharge Amount in Month 7



# Trend Analysis of Total Recharge Patterns

Checking if there are consistent patterns in recharge behavior that distinguish churners from non-churners.

```
# Add the 'churn' column if it's already created
data['churn'] = np.where(
    (data['total_ic_mou_9'] == 0) &
    (data['total_og_mou_9'] == 0) &
    (data['vol_2g_mb_9'] == 0) &
    (data['vol_3g_mb_9'] == 0), 1, 0
)

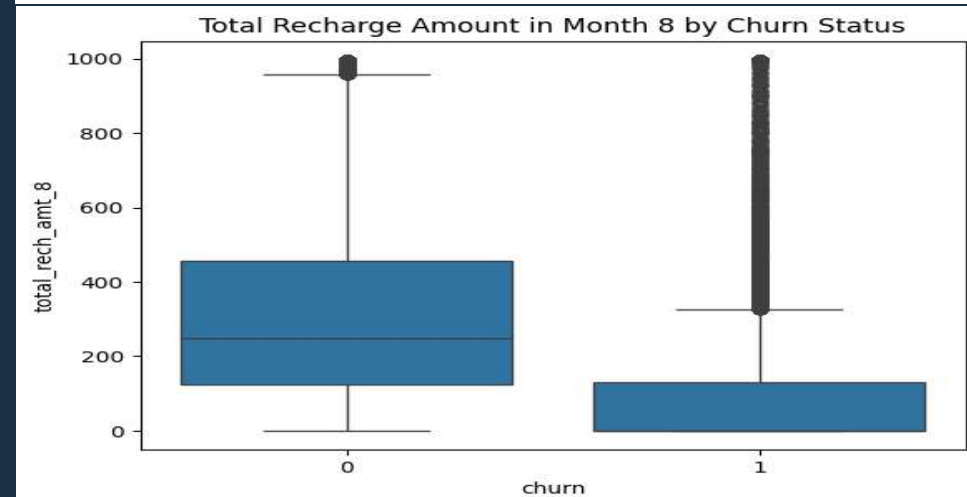
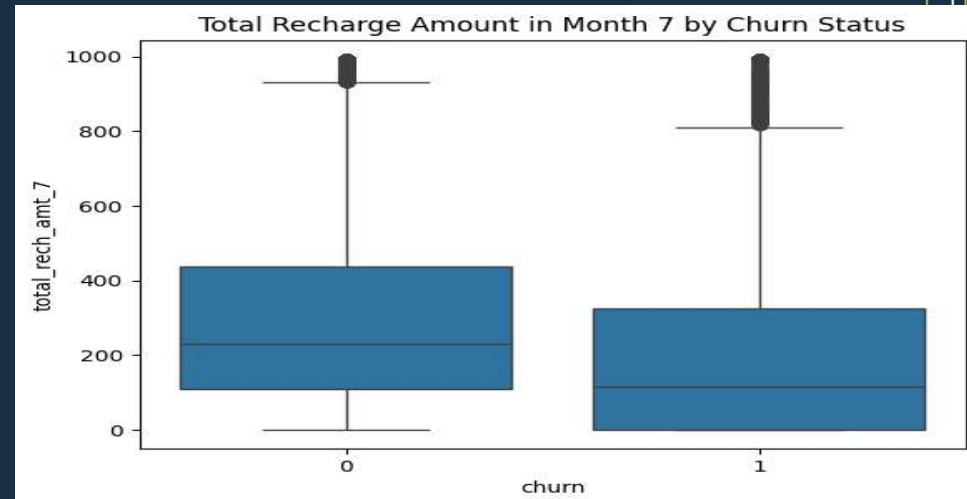
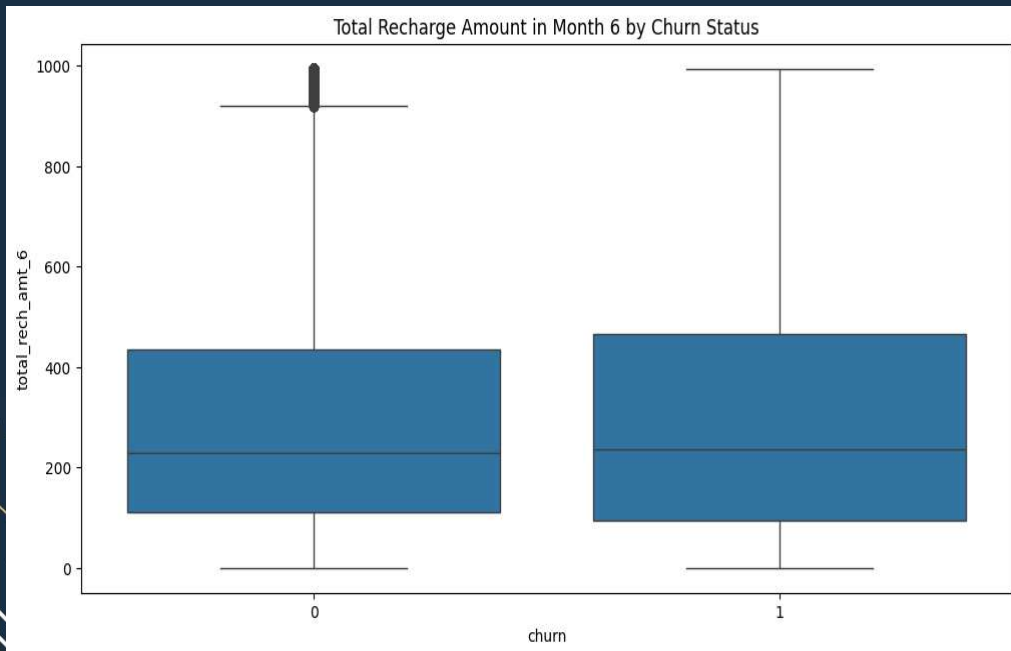
# Compare recharge trends for churners vs. non-churners
plt.figure(figsize=(12, 6))
sns.boxplot(x='churn', y='total_rech_amt_6', data=data)
plt.title('Total Recharge Amount in Month 6 by Churn Status')
plt.show()

sns.boxplot(x='churn', y='total_rech_amt_7', data=data)
plt.title('Total Recharge Amount in Month 7 by Churn Status')
plt.show()

sns.boxplot(x='churn', y='total_rech_amt_8', data=data)
plt.title('Total Recharge Amount in Month 8 by Churn Status')
plt.show()
```

# Trend Analysis of Total Recharge Patterns

A decline in total recharge amount over months might indicate churn.





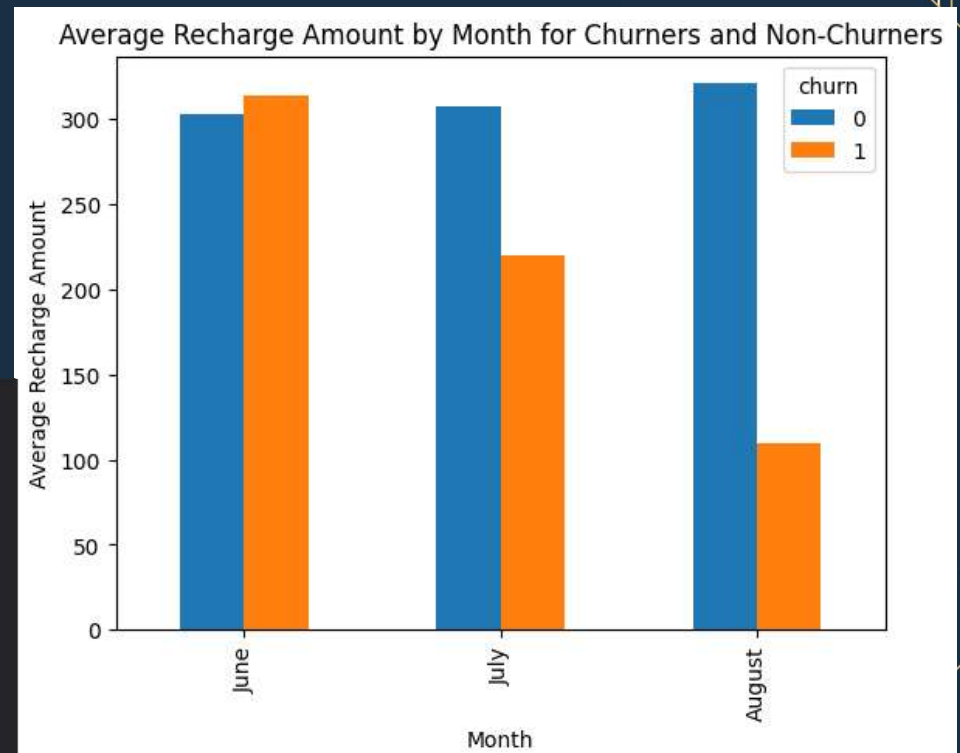
# Trend Analysis of Average Recharge Patterns

Checking and looking for any time-based trends such as differences in average recharge behavior by month. A decline in average recharge amounts over recent months might signal disengagement, which can precede churn.

```
avg_rech_amt_by_month = data.groupby(['churn']).agg({
    'total_rech_amt_6': 'mean',
    'total_rech_amt_7': 'mean',
    'total_rech_amt_8': 'mean'
}).T

# Rename the index to show the appropriate month numbers
avg_rech_amt_by_month.index = ['June', 'July', 'August']

# Plot average recharge amount per month for churners vs. non-churners
avg_rech_amt_by_month.plot(kind='bar')
plt.title('Average Recharge Amount by Month for Churners and Non-Churners')
plt.xlabel('Month')
plt.ylabel('Average Recharge Amount')
plt.show()
```



# Usage Pattern Analysis

Investigating whether changes in call usage (incoming/outgoing, local/roaming) predict churn. Often, churners show a drop in usage prior to churning.

Method: Plot call minutes over time and compare them for churners vs. non-churners.

A decline in usage (e.g., minutes call or data) over recent months might signal disengagement, which can precede churn.

```
# Local incoming minutes comparison
sns.boxplot(x='churn', y='loc_ic_mou_8', data=data)
plt.title('Local Incoming Minutes of Use (MOU) in Month 8 by Churn Status')
plt.show()

# Local outgoing minutes comparison
sns.boxplot(x='churn', y='loc_og_mou_8', data=data)
plt.title('Local Outgoing Minutes of Use (MOU) in Month 8 by Churn Status')
plt.show()

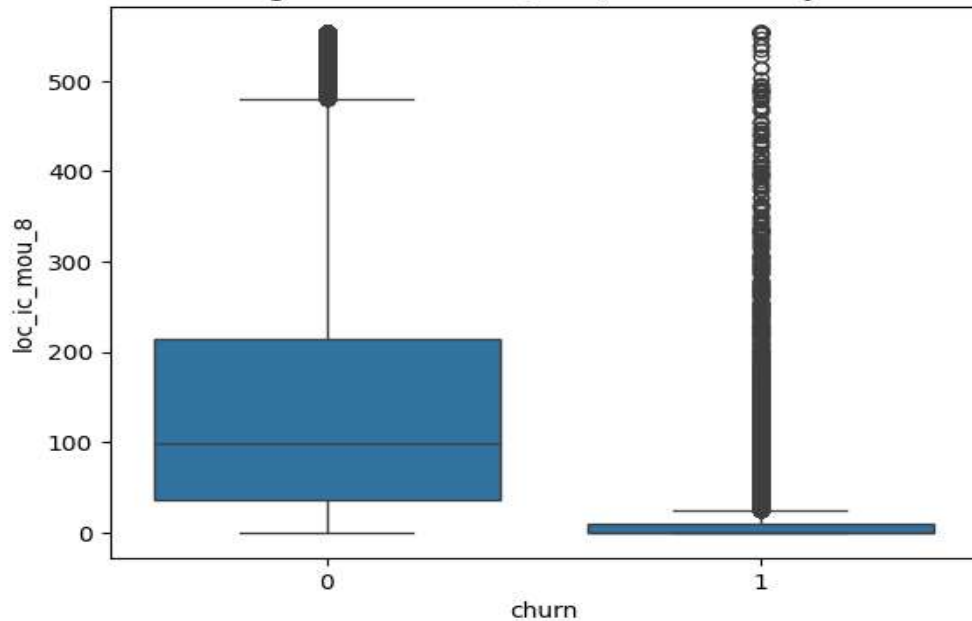
# Roaming minutes comparison
sns.boxplot(x='churn', y='roam_ic_mou_8', data=data)
plt.title('Roaming Incoming Minutes of Use (MOU) in Month 8 by Churn Status')
plt.show()

sns.boxplot(x='churn', y='roam_og_mou_8', data=data)
plt.title('Roaming Outgoing Minutes of Use (MOU) in Month 8 by Churn Status')
plt.show()
```

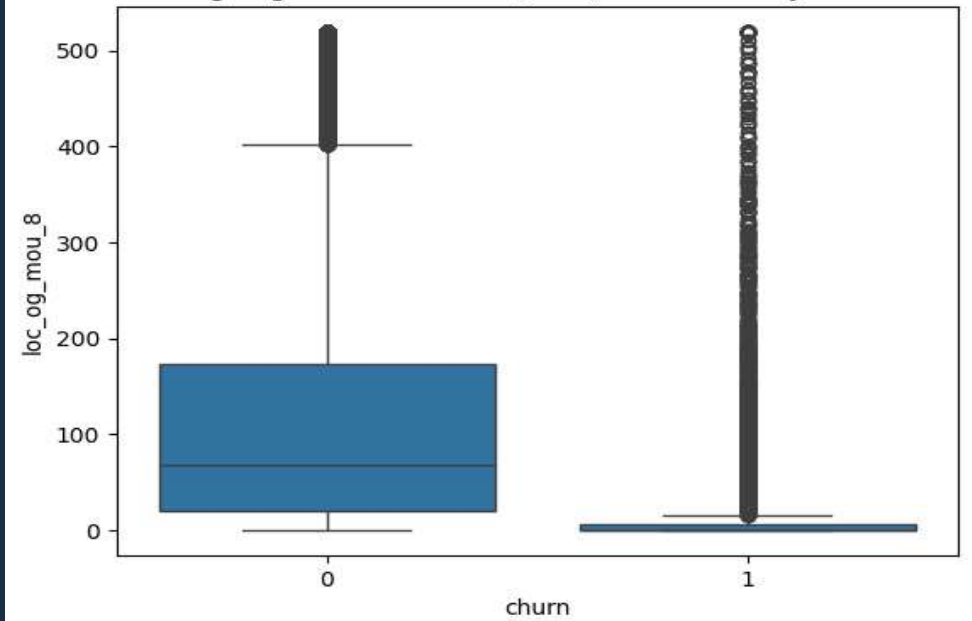
# Usage Pattern Analysis

A decline in usage (e.g., minutes call or data) over recent months might signal disengagement, which can precede churn.

Local Incoming Minutes of Use (MOU) in Month 8 by Churn Status



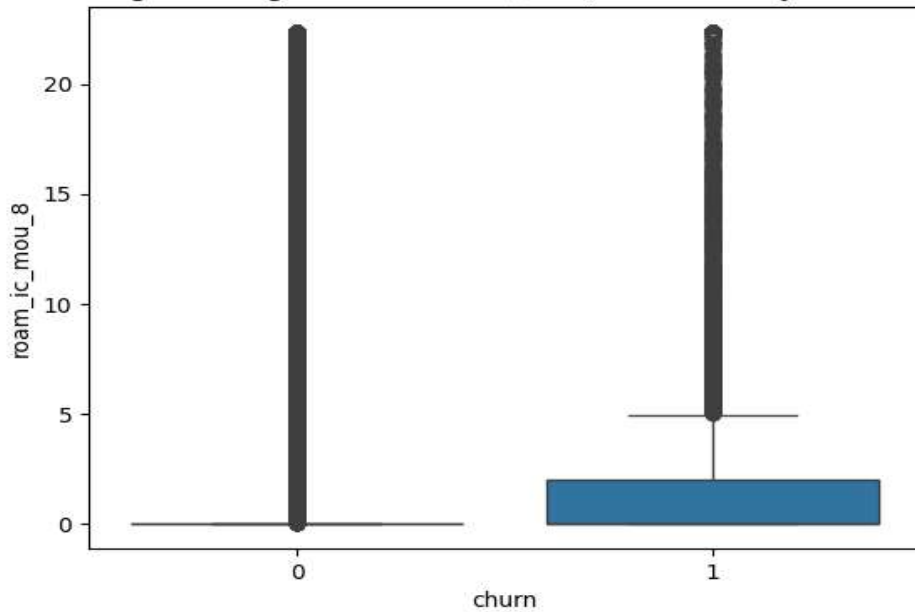
Local Outgoing Minutes of Use (MOU) in Month 8 by Churn Status



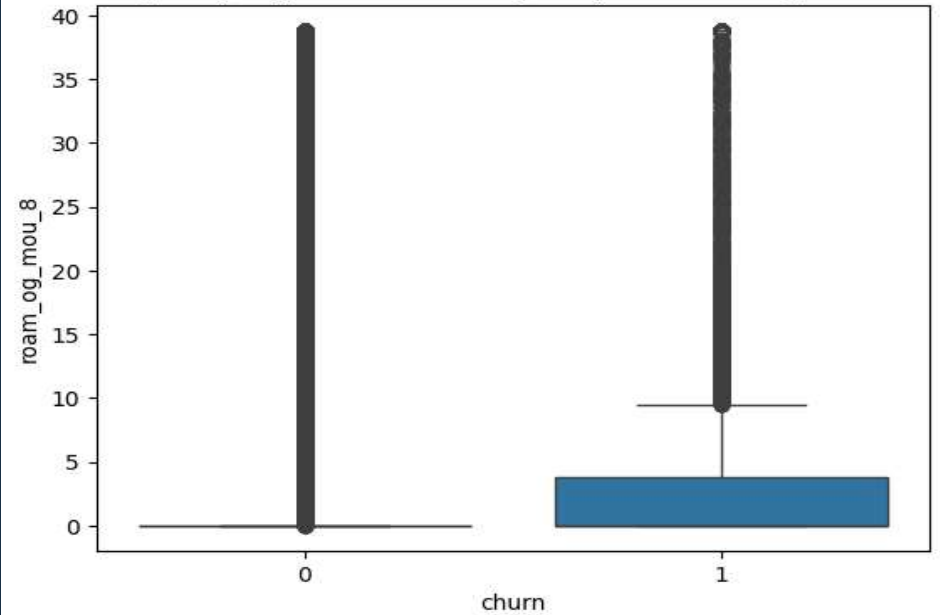
# Usage Pattern Analysis

A decline in usage (e.g., minutes call or data) over recent months might signal disengagement, which can precede churn.

Roaming Incoming Minutes of Use (MOU) in Month 8 by Churn Status



Roaming Outgoing Minutes of Use (MOU) in Month 8 by Churn Status



# Revenue Contribution Analysis

Checking if high-value customers are less likely to churn compared to low-value customers, by analyzing metrics like ARPU (Average Revenue Per User).

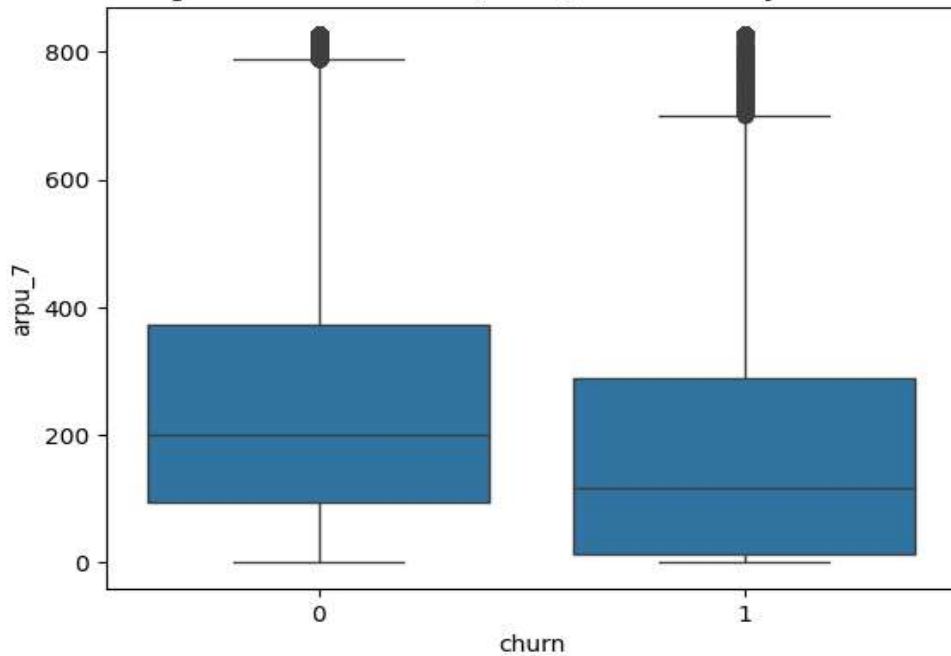
```
# ARPU analysis for churners vs non-churners
sns.boxplot(x='churn', y='arpu_8', data=data)
plt.title('Average Revenue Per User (ARPU) in Month 8 by Churn Status')
plt.show()

# Check revenue trends in earlier months as well
sns.boxplot(x='churn', y='arpu_7', data=data)
plt.title('Average Revenue Per User (ARPU) in Month 7 by Churn Status')
plt.show()
```

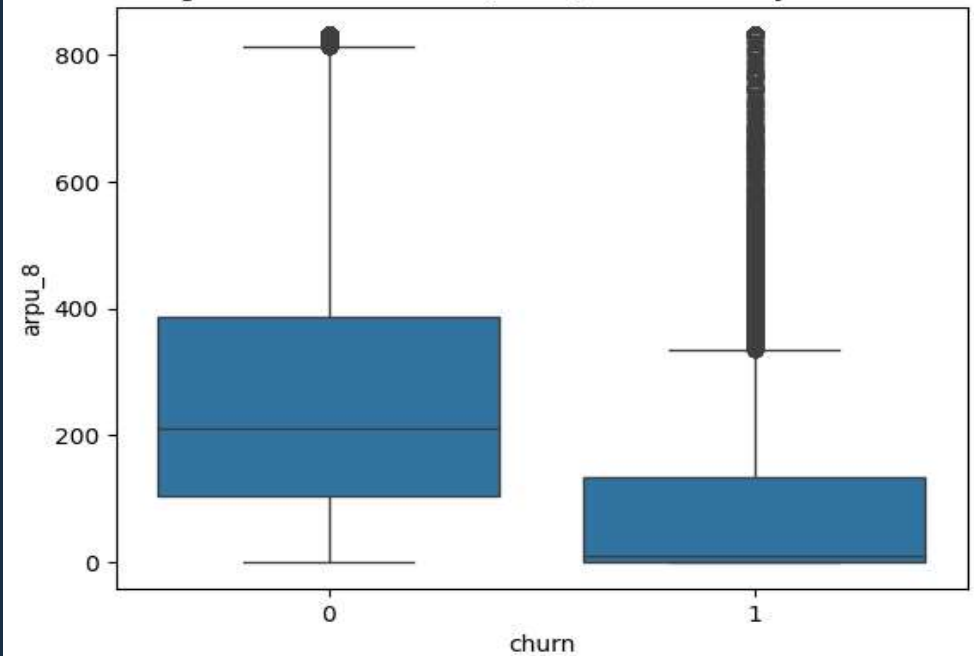
# Revenue Contribution Analysis

Customers with low ARPU are more likely to churn, possibly due to perceived lack of value.

Average Revenue Per User (ARPU) in Month 7 by Churn Status



Average Revenue Per User (ARPU) in Month 8 by Churn Status



# Correlation Analysis of Potential Features

Different features correlate with each other and with the churn variable. High correlation among features might suggest multicollinearity issues

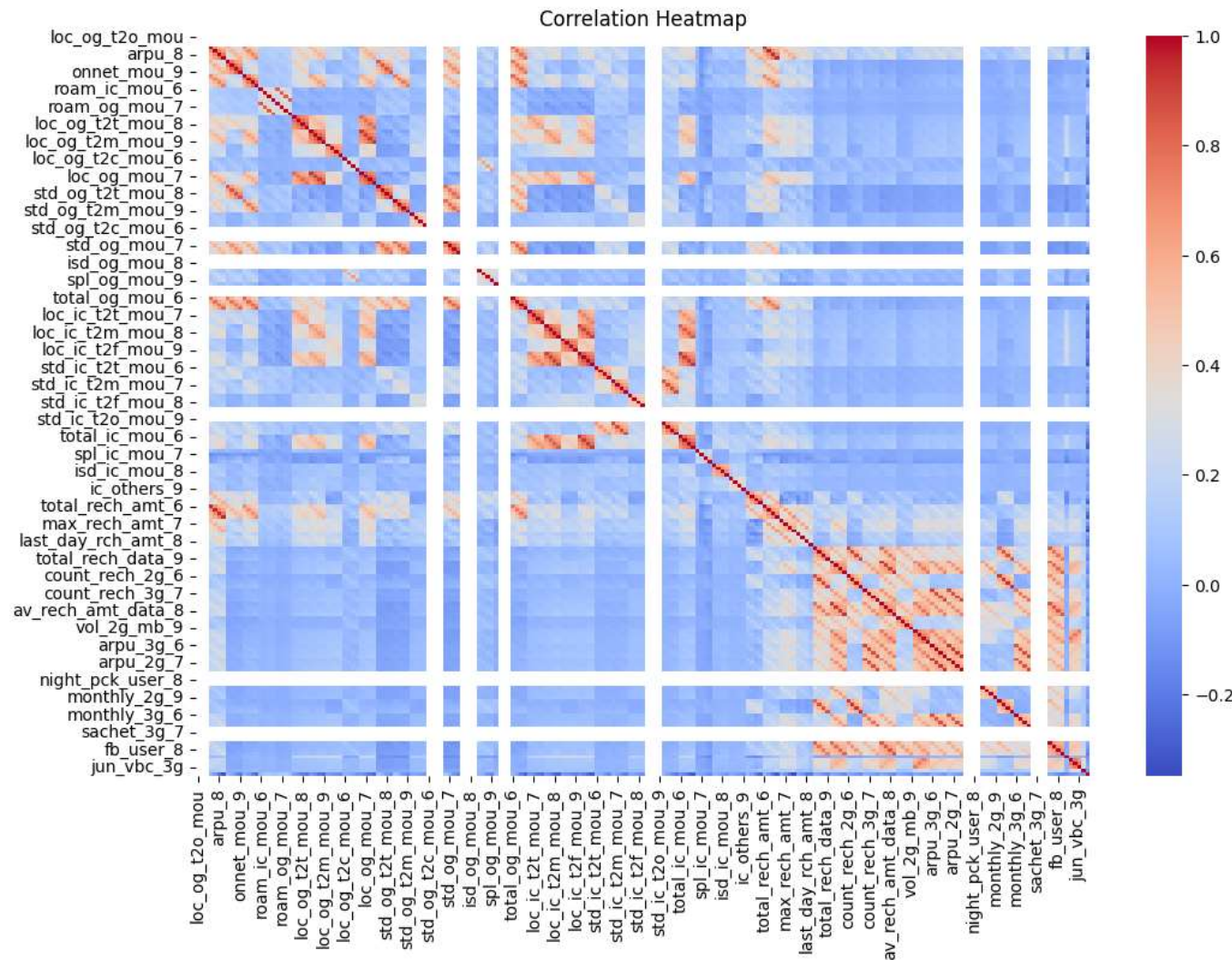
```
# Correlation heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(data.corr(), cmap="coolwarm", annot=False)
plt.title('Correlation Heatmap')
plt.show()

# Correlation with churn
churn_corr = data.corr()['churn'].sort_values(ascending=False)
print("Correlation with churn:")
print(churn_corr)
```



# Correlation Analysis of Potential Features

Identifying high correlation can help us address multicollinearity, especially important if we use logistic regression.







# CHURN DATA PREPARATION



# Data preparation steps

Prepare the data following steps of the business problem

```
# Step 1: Identify High-Value Customers
```

```
data['avg_rech_amt_good_phase'] = (data['total_rech_amt_6'] + data['total_rech_amt_7']) / 2
```

```
X = data['avg_rech_amt_good_phase'].quantile(0.7)
```

```
data['high_value_customer'] = np.where(data['avg_rech_amt_good_phase'] >= X, 1, 0)
```

```
# Filter high-value customers, making an independent copy to avoid warnings
```

```
high_value_data = data[data['high_value_customer'] == 1].copy()
```

```
# Step 2: Tag Churners based on zero usage in month 9
```

```
high_value_data.loc[:, 'churn'] = np.where(  
    (high_value_data['total_ic_mou_9'] == 0) &  
    (high_value_data['total_og_mou_9'] == 0) &  
    (high_value_data['vol_2g_mb_9'] == 0) &  
    (high_value_data['vol_3g_mb_9'] == 0), 1, 0  
)
```

```
# Step 3: Drop month 9 columns to prevent data leakage
```

```
churn_data = high_value_data.drop([col for col in high_value_data.columns if '_9' in col], axis=1)  
churn_data.head()
```

# Data preparation steps

Split the data to 80% and 20% for training and testing respectively. Also plot the class distribution

```
# Split into train and test
from sklearn.model_selection import train_test_split
# Separate features and target variable
X = churn_data.drop(columns=['churn'])
y = churn_data['churn']

# Split data (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y, random_state=42)

# Calculate the class distribution in the target variable
churn_counts = churn_data['churn'].value_counts(normalize=True) * 100
print("Class Distribution (%):")
print(churn_counts)

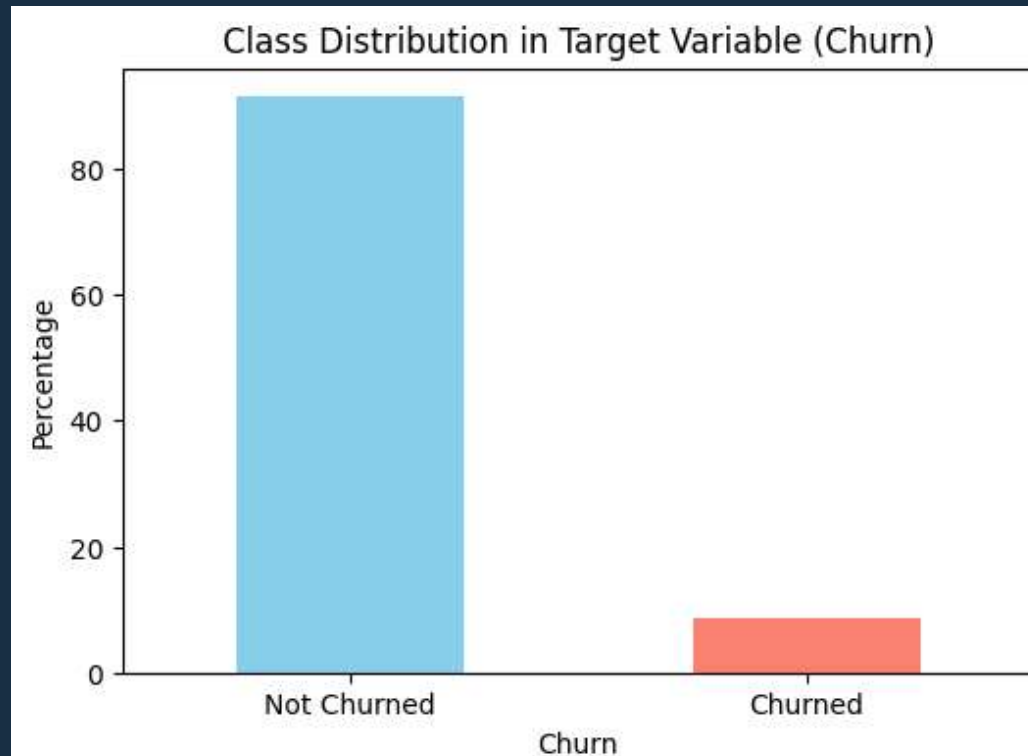
# Plot the distribution for a visual overview
import matplotlib.pyplot as plt

plt.figure(figsize=(6, 4))
churn_counts.plot(kind='bar', color=['skyblue', 'salmon'])
plt.title('Class Distribution in Target Variable (Churn)')
plt.xlabel('Churn')
plt.ylabel('Percentage')
plt.xticks(ticks=[0, 1], labels=['Not Churned', 'Churned'], rotation=0)
plt.show()
```

# Data preparation steps

According to the distribution graph, since the rate of churn is typically low (about 8.64%), this is class imbalance.

Therefore, when training the model in section 5, I need to input balanced class weight flag for the training function to handle it.





# VIF FEATURE ENGINEERING



# VIF feature engineering

We calculated the VIF for each predictor variable. Variables with high VIF values ( $>10$ ) were considered highly collinear.

```
from statsmodels.stats.outliers_influence import variance_inflation_factor

X_train = X_train.select_dtypes(include=['number'])
# If necessary, drop any rows with NaN values
X_train = X_train.dropna()

# Calculating VIF
vif_data = pd.DataFrame()
vif_data['feature'] = X_train.columns
vif_data['VIF'] = [variance_inflation_factor(X_train.values, i) for i in range(X_train.shape[1])]

# Convert VIF column to numeric, setting errors='coerce' to handle any non-numeric values
vif_data['VIF'] = pd.to_numeric(vif_data['VIF'], errors='coerce')

# Display features with VIF > 10 (indicating high multicollinearity)
high_vif = vif_data[vif_data['VIF'] > 10]

# Show the result
high_vif
```

	feature	VIF
5	arpu_8	11.139893
6	onnet_mou_6	16.631573
7	onnet_mou_7	23.258547
8	onnet_mou_8	21.420441
9	offnet_mou_6	14.338468
...	...	...
144	sachet_2g_6	18.925173
145	sachet_2g_7	38.899245
146	sachet_2g_8	48.671198
161	avg_rech_amt_good_phase	inf
162	high_value_customer	29.998350

69 rows × 2 columns



# MODEL BUILDING





# Logistic Regression

Since we are doing binary classification to predict churned or not churned, Logistic regression is a must-have method for this scenario

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, roc_auc_score

# Train logistic regression with class weights
log_reg = LogisticRegression(class_weight='balanced', random_state=42, max_iter=1000)
log_reg.fit(X_train, y_train)

# Predictions and evaluation on test data
y_pred = log_reg.predict(X_test)
print("Logistic Regression Classification Report:\n", classification_report(y_test, y_pred))
print("Logistic Regression AUC-ROC:", roc_auc_score(y_test, y_pred))
```

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

     0       0.98         0.84         0.91         5484
     1       0.33         0.83         0.48          519

 accuracy          0.84          6003
 macro avg         0.66         0.84         0.69          6003
 weighted avg      0.93         0.84         0.87          6003
```

```
Logistic Regression AUC-ROC: 0.8368116601948706
```



# Random Forest

Random Forest is highly effective for accurate predictions and identifying key predictors in complex datasets, especially in scenarios with potential non-linearity and numerous features

```
from sklearn.ensemble import RandomForestClassifier

# Train random forest with class weights
rf = RandomForestClassifier(class_weight='balanced', random_state=42)
rf.fit(X_train, y_train)

# Predictions and evaluation on test data
y_pred_rf = rf.predict(X_test)
print("Random Forest Classification Report:\n", classification_report(y_test, y_pred_rf))
print("Random Forest AUC-ROC:", roc_auc_score(y_test, y_pred_rf))
```

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

     0       0.95         0.99         0.97         5484
     1       0.79         0.44         0.56          519

 accuracy          0.94          0.94          0.94         6003
 macro avg       0.87         0.71         0.77         6003
 weighted avg    0.93         0.94         0.93         6003
```

```
Random Forest AUC-ROC: 0.7140003710215319
```

# Evaluation Metrics for both methods

## Overall Insights

- Random Forest Outperforms Logistic Regression:  
Based on both the AUC-ROC and F1 score, the Random Forest model performs better in distinguishing between churners and non-churners and handling the imbalanced data. It provides both higher discrimination (AUC-ROC) and a more balanced prediction (F1 score).
- Model Choice for Deployment: Given the higher AUC-ROC and F1 score, the Random Forest model appears to be the better choice for deployment if the goal is to accurately predict churn. This model would likely provide better results in real-world usage, especially in targeting high-risk customers.

```
from sklearn.metrics import roc_auc_score, f1_score
```

```
# Logistic Regression metrics
```

```
roc_auc_log = roc_auc_score(y_test, log_reg.predict_proba(X_test)[:, 1])
```

```
f1_log = f1_score(y_test, y_pred)
```

```
# Random Forest metrics
```

```
roc_auc_rf = roc_auc_score(y_test, rf.predict_proba(X_test)[:, 1])
```

```
f1_rf = f1_score(y_test, y_pred_rf)
```

```
print("Logistic Regression AUC-ROC:", roc_auc_log)
```

```
print("Logistic Regression F1 Score:", f1_log)
```

```
print("Random Forest AUC-ROC:", roc_auc_rf)
```

```
print("Random Forest F1 Score:", f1_rf)
```

```
Logistic Regression AUC-ROC: 0.8983411542985796
```

```
Logistic Regression F1 Score: 0.47624309392265196
```

```
Random Forest AUC-ROC: 0.9275622620508215
```

```
Random Forest F1 Score: 0.5636588380716935
```



# FEATURE IMPORTANCE VISUALIZATION



# Logistic Regression Coefficients

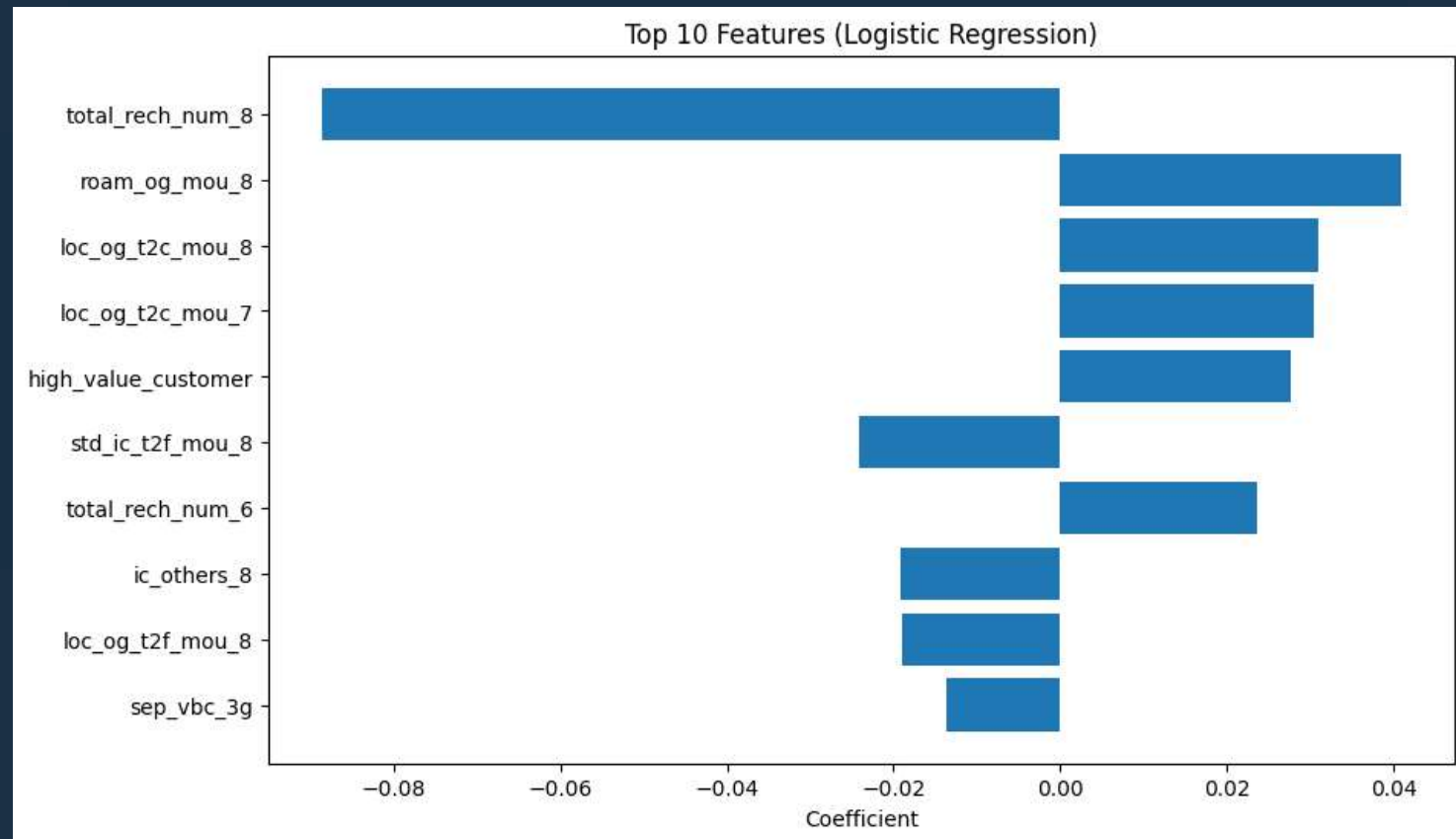
```
import matplotlib.pyplot as plt
import numpy as np

# Logistic regression feature importance (coefficients)
coefficients = log_reg.coef_[0]
features = X_train.columns

# Sort by absolute value to highlight most impactful features
coef_df = pd.DataFrame({'Feature': features, 'Coefficient': coefficients})
coef_df['AbsCoefficient'] = np.abs(coef_df['Coefficient'])
coef_df = coef_df.sort_values(by='AbsCoefficient', ascending=False)

plt.figure(figsize=(10, 6))
plt.barh(coef_df['Feature'][:10], coef_df['Coefficient'][:10])
plt.xlabel('Coefficient')
plt.title('Top 10 Features (Logistic Regression)')
plt.gca().invert_yaxis()
plt.show()
```

# Logistic Regression Coefficients



# Logistic Regression Coefficients

According to the coefficients graph:

- The positive coefficient on variables related to roaming and specific call usage (e.g., `loc_og_t2c_mou_8`, `std_ic_t2f_mou_8`) might indicate a segment of customers who use the network intensively in specific ways but are at a high risk of churning.
- High engagement in month 8 seems to reduce churn risk, suggesting that recency in recharges and usage plays an important role in retention.

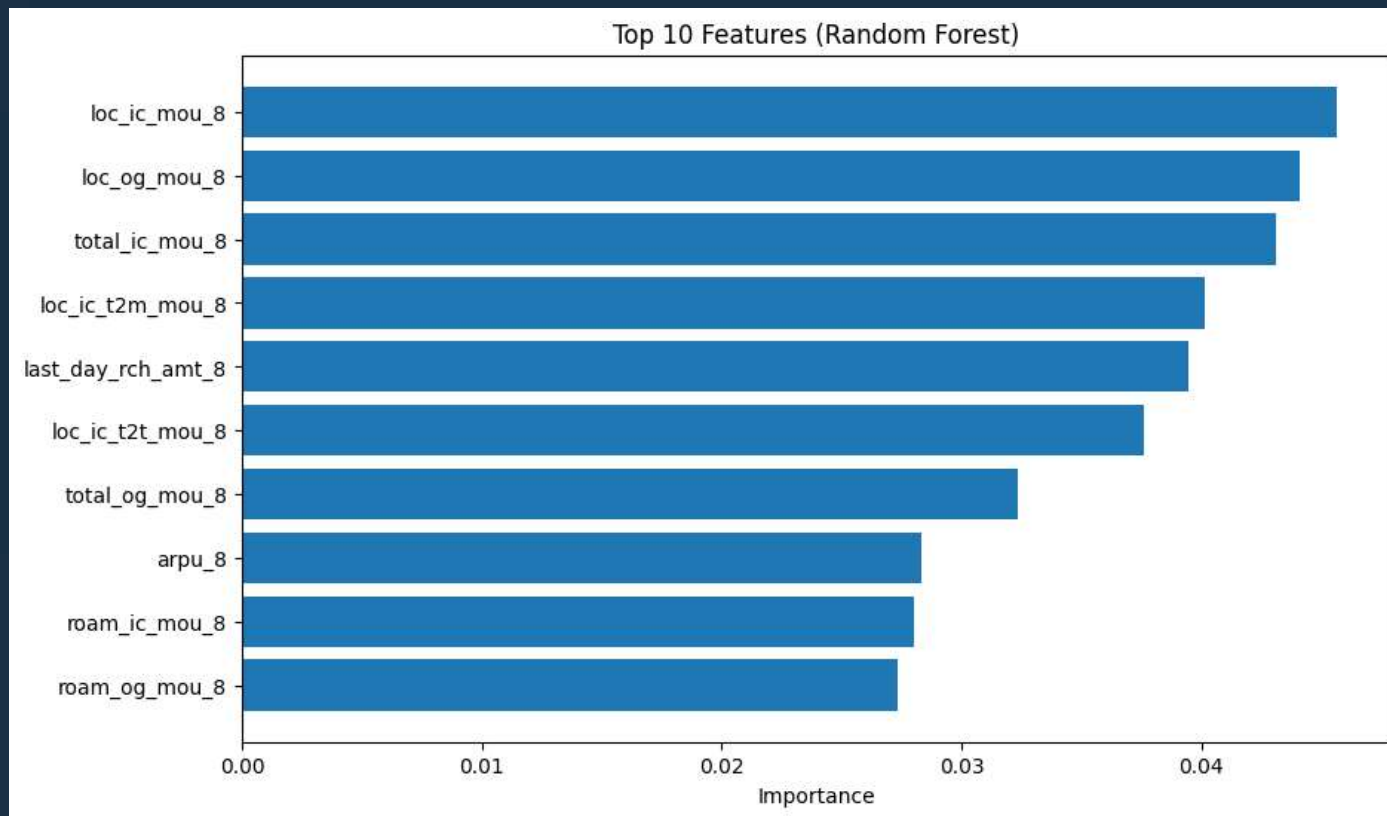


# Random Forest Feature Importances

```
# Random forest feature importance
importances = rf.feature_importances_
feature_names = X_train.columns
importance_df = pd.DataFrame({'Feature': feature_names, 'Importance': importances})

# Sort and plot top 10 features
importance_df = importance_df.sort_values(by='Importance', ascending=False)
plt.figure(figsize=(10, 6))
plt.barh(importance_df['Feature'][:10], importance_df['Importance'][:10])
plt.xlabel('Importance')
plt.title('Top 10 Features (Random Forest)')
plt.gca().invert_yaxis()
plt.show()
```

# Random Forest Feature Importances



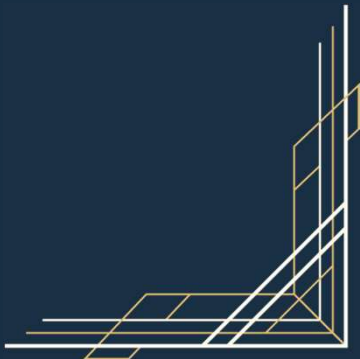
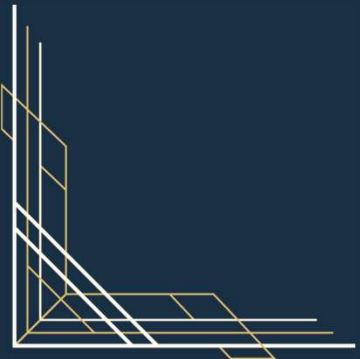




# Random Forest Feature Importances

According to the top 10 features of random forest graph:

While logistic regression highlighted features related to recharges and high-value customer indicators, the random forest model places more emphasis on minutes of use (MOU) metrics, covering a broader spectrum of incoming and outgoing call behaviors. This difference indicates that random forests capture nonlinear relationships more effectively, allowing for a wider range of feature influences.





# STRATEGY CONTRIBUTION



# Strategy Contribution

## Key Churn Indicators:

- **Recharge Amounts and Usage Patterns:** The model indicates that churners tend to have lower average recharge amounts and lower call or data usage in recent months. This trend suggests that customers may be disengaging with the service before fully churning.
- **Roaming and International Usage:** High usage of roaming and international call minutes (as shown in the Random Forest feature importance) appears to be a significant churn indicator. Customers with high roaming usage might be at risk if they find better international plans elsewhere.
- **Last Interaction Metrics:** Features such as last recharge amount and frequency of recent transactions have strong predictive power. Customers who have lower recharges in the most recent months may signal a risk of disengagement.

# Strategy Contribution

## Segmented Retention Strategies:

- **Low Recharge and Low Usage Customers:** For customers identified with low recharge and usage patterns, offering personalized incentives such as bonus minutes, additional data, or limited-time discounts could encourage continued engagement.
- **High Roaming Customers:** For customers with high roaming or international usage, targeted offers that provide better roaming rates or international calling plans can increase loyalty and reduce the likelihood of churn.
- **Infrequent Rechargers:** Customers who recharge infrequently might benefit from reminders or special offers before they are due for recharge, potentially preventing disengagement.

# Strategy Contribution

## Proactive Customer Engagement:

- **Predictive Alerts for At-Risk Customers:** Implement an early warning system based on the model's churn probabilities to flag high-risk customers. Customer service teams can proactively reach out to these customers to offer tailored retention plans or solve potential pain points.
- **Loyalty Programs:** Offer loyalty points or rewards for long-standing customers or those with high usage, especially among the features identified as churn drivers. Rewarding loyalty can improve customer satisfaction and reduce the likelihood of churn among high-value customers.

# Strategy Contribution

## Product and Service Improvements:

- **Enhanced Data and Voice Plans:** Based on the significant influence of usage features, the company could develop data and voice plans that better fit the needs of its customer base, particularly addressing patterns of high roaming or domestic usage.
- **Feedback Collection:** For customers identified as high-risk, collect feedback on what improvements or additional features they would like. Understanding customer needs could help reduce churn by adapting services accordingly.