index

June 10, 2025

# 1 Telecom Churn Prediction - Phase 3 Classification Project

## 1.1 Introduction

## 1.2 Overview

Churn prediction is essential for customer retention in telecom. This project targets business analysts and marketing teams aiming to improve retention strategies. The objective is to build an interpretable and actionable machine learning model that identifies high-risk customers based on their usage behavior, service plans, and interaction history.

## 1.3 Business Problem

The telecom provider is experiencing customer churn but lacks predictive tools to proactively engage at-risk users. By identifying patterns that signal potential churn, the company can reduce losses and improve customer engagement.

## 1.4 Objectives

1. Explore and clean the dataset to ensure it's suitable for modeling.
2. Perform exploratory data analysis to understand relationships between features and churn.
3. Apply preprocessing steps to prevent leakage and standardize inputs.
4. Train and compare logistic regression and random forest models.
5. Evaluate models using appropriate metrics (precision, recall, ROC AUC).
6. Generate recommendations that the business can act on.

## 1.5 Analysis Steps

- **Data Loading & Cleaning**: Dropped irrelevant columns, handled categorical and numerical features.
- **EDA**: Investigated class imbalance and correlated variables.
- **Modeling**: Logistic Regression for interpretability; Random Forest for capturing nonlinear patterns.
- **Evaluation**: Used confusion matrix, classification report, and ROC curve.
- **Feature Importance**: Ranked drivers of churn.
- **Recommendations**: Offered data-driven strategies for churn mitigation.

## 2 1. Load Dataset & Import Libraries

[1]:
```python
# Import necessary libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix,␣
↪roc_auc_score, roc_curve
```

[3]:
```python
# Load the dataset
df = pd.read_csv("Churn.csv")
df.head()
```

[3]:
```
  state  account length  area code phone number international plan  \
0    KS             128        415     382-4657                 no
1    OH             107        415     371-7191                 no
2    NJ             137        415     358-1921                 no
3    OH              84        408     375-9999                yes
4    OK              75        415     330-6626                yes

  voice mail plan  number vmail messages  total day minutes  total day calls  \
0             yes                     25              265.1              110
1             yes                     26              161.6              123
2              no                      0              243.4              114
3              no                      0              299.4               71
4              no                      0              166.7              113

   total day charge  …  total eve calls  total eve charge  \
0             45.07  …               99             16.78
1             27.47  …              103             16.62
2             41.38  …              110             10.30
3             50.90  …               88              5.26
4             28.34  …              122             12.61

   total night minutes  total night calls  total night charge  \
0                244.7                 91               11.01
1                254.4                103               11.45
2                162.6                104                7.32
3                196.9                 89                8.86
4                186.9                121                8.41
```

```
   total intl minutes  total intl calls  total intl charge  \
0                10.0                 3               2.70
1                13.7                 3               3.70
2                12.2                 5               3.29
3                 6.6                 7               1.78
4                10.1                 3               2.73

   customer service calls  churn
0                       1  False
1                       1  False
2                       0  False
3                       2  False
4                       3  False

[5 rows x 21 columns]
```

[5]:
```python
# Preview the dataset
print("\nDataset Info:")
print(df.info())
print("\nSummary Statistics:")
print(df.describe(include='all'))
```

```
Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3333 entries, 0 to 3332
Data columns (total 21 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   state                   3333 non-null   object
 1   account length          3333 non-null   int64
 2   area code               3333 non-null   int64
 3   phone number            3333 non-null   object
 4   international plan      3333 non-null   object
 5   voice mail plan         3333 non-null   object
 6   number vmail messages   3333 non-null   int64
 7   total day minutes       3333 non-null   float64
 8   total day calls         3333 non-null   int64
 9   total day charge        3333 non-null   float64
 10  total eve minutes       3333 non-null   float64
 11  total eve calls         3333 non-null   int64
 12  total eve charge        3333 non-null   float64
 13  total night minutes     3333 non-null   float64
 14  total night calls       3333 non-null   int64
 15  total night charge      3333 non-null   float64
 16  total intl minutes      3333 non-null   float64
 17  total intl calls        3333 non-null   int64
```

```
 18  total intl charge       3333 non-null   float64
 19  customer service calls  3333 non-null   int64
 20  churn                   3333 non-null   bool
dtypes: bool(1), float64(8), int64(8), object(4)
memory usage: 524.2+ KB
None

Summary Statistics:
       state  account length    area code phone number international plan  \
count   3333     3333.000000  3333.000000         3333               3333
unique    51             NaN          NaN         3333                  2
top       WV             NaN          NaN     382-4657                 no
freq     106             NaN          NaN            1               3010
mean     NaN      101.064806   437.182418          NaN                NaN
std      NaN       39.822106    42.371290          NaN                NaN
min      NaN        1.000000   408.000000          NaN                NaN
25%      NaN       74.000000   408.000000          NaN                NaN
50%      NaN      101.000000   415.000000          NaN                NaN
75%      NaN      127.000000   510.000000          NaN                NaN
max      NaN      243.000000   510.000000          NaN                NaN

       voice mail plan  number vmail messages  total day minutes  \
count             3333            3333.000000        3333.000000
unique               2                    NaN                NaN
top                 no                    NaN                NaN
freq              2411                    NaN                NaN
mean               NaN               8.099010         179.775098
std                NaN              13.688365          54.467389
min                NaN               0.000000           0.000000
25%                NaN               0.000000         143.700000
50%                NaN               0.000000         179.400000
75%                NaN              20.000000         216.400000
max                NaN              51.000000         350.800000

        total day calls  total day charge  …  total eve calls  \
count       3333.000000       3333.000000  …      3333.000000
unique              NaN               NaN  …              NaN
top                 NaN               NaN  …              NaN
freq                NaN               NaN  …              NaN
mean         100.435644         30.562307  …       100.114311
std           20.069084          9.259435  …        19.922625
min            0.000000          0.000000  …         0.000000
25%           87.000000         24.430000  …        87.000000
50%          101.000000         30.500000  …       100.000000
75%          114.000000         36.790000  …       114.000000
max          165.000000         59.640000  …       170.000000

        total eve charge  total night minutes  total night calls  \
```

```
count       3333.000000          3333.000000        3333.000000
unique              NaN                  NaN                NaN
top                 NaN                  NaN                NaN
freq                NaN                  NaN                NaN
mean          17.083540           200.872037         100.107711
std            4.310668            50.573847          19.568609
min            0.000000            23.200000          33.000000
25%           14.160000           167.000000          87.000000
50%           17.120000           201.200000         100.000000
75%           20.000000           235.300000         113.000000
max           30.910000           395.000000         175.000000

        total night charge  total intl minutes  total intl calls  \
count          3333.000000         3333.000000       3333.000000
unique                 NaN                 NaN               NaN
top                    NaN                 NaN               NaN
freq                   NaN                 NaN               NaN
mean              9.039325           10.237294          4.479448
std               2.275873            2.791840          2.461214
min               1.040000            0.000000          0.000000
25%               7.520000            8.500000          3.000000
50%               9.050000           10.300000          4.000000
75%              10.590000           12.100000          6.000000
max              17.770000           20.000000         20.000000

        total intl charge  customer service calls  churn
count         3333.000000             3333.000000   3333
unique                NaN                     NaN      2
top                   NaN                     NaN  False
freq                  NaN                     NaN   2850
mean             2.764581                1.562856    NaN
std              0.753773                1.315491    NaN
min              0.000000                0.000000    NaN
25%              2.300000                1.000000    NaN
50%              2.780000                1.000000    NaN
75%              3.270000                2.000000    NaN
max              5.400000                9.000000    NaN

[11 rows x 21 columns]
```

[6]:
```
df.isna().sum()
```

[6]:
```
state                     0
account length            0
area code                 0
phone number              0
international plan        0
```

```
voice mail plan           0
number vmail messages     0
total day minutes         0
total day calls           0
total day charge          0
total eve minutes         0
total eve calls           0
total eve charge          0
total night minutes       0
total night calls         0
total night charge        0
total intl minutes        0
total intl calls          0
total intl charge         0
customer service calls    0
churn                     0
dtype: int64
```

The dataset has no missing values.

## 3 2. Data Cleaning

[ ]:
```python
# Drop irrelevant columns
# 'phone number' is an identifier and not useful for prediction

df.drop(['phone number'], axis=1, inplace = True)
```

[19]:
```python
# Encode binary categorical features
# Map 'yes'/'no' to 1/0 for plan columns

df['international plan'] = df['international plan'].map({'yes': 1, 'no': 0})
df['voice mail plan'] = df['voice mail plan'].map({'yes': 1, 'no': 0})
```

[ ]:
```python
# Convert churn to integer label for classification

df['churn'] = df['churn'].astype(int)
```

[ ]:
```python
# One-hot encode 'state' and 'area code' to avoid ordinal assumptions

df = pd.get_dummies(df, columns=['state', 'area code'], drop_first=True)
```

[20]:
```python
# Verify updated dataset to check that categorical encoding was successful

print("\nCleaned Dataset Preview:")
print(df.head())
```

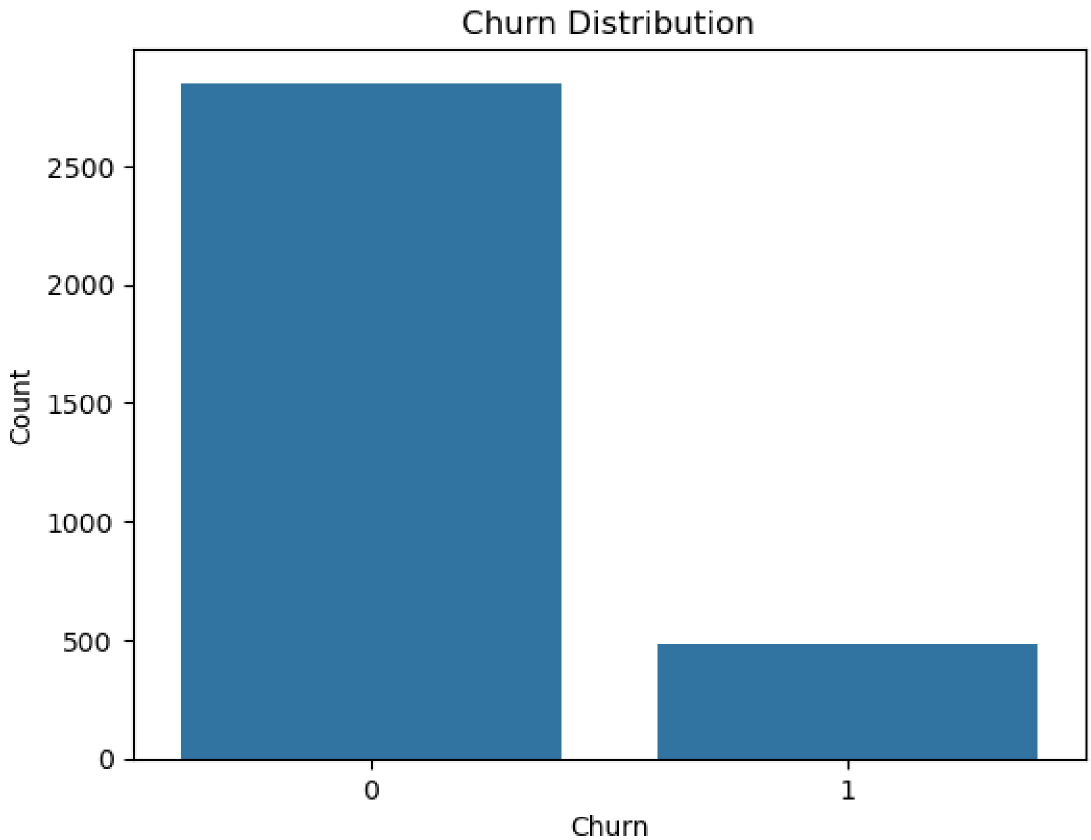
```
Cleaned Dataset Preview:
```

```
   account length  international plan  voice mail plan  number vmail messages  \
0             128                 NaN              NaN                     25   
1             107                 NaN              NaN                     26   
2             137                 NaN              NaN                      0   
3              84                 NaN              NaN                      0   
4              75                 NaN              NaN                      0   

   total day minutes  total day calls  total day charge  total eve minutes  \
0              265.1              110             45.07              197.4   
1              161.6              123             27.47              195.5   
2              243.4              114             41.38              121.2   
3              299.4               71             50.90               61.9   
4              166.7              113             28.34              148.3   

   total eve calls  total eve charge  …  state_TX  state_UT  state_VA  \
0               99             16.78  …     False     False     False   
1              103             16.62  …     False     False     False   
2              110             10.30  …     False     False     False   
3               88              5.26  …     False     False     False   
4              122             12.61  …     False     False     False   

   state_VT  state_WA  state_WI  state_WV  state_WY  area code_415  \
0     False     False     False     False     False           True   
1     False     False     False     False     False           True   
2     False     False     False     False     False           True   
3     False     False     False     False     False          False   
4     False     False     False     False     False           True   

   area code_510  
0          False  
1          False  
2          False  
3          False  
4          False  

[5 rows x 70 columns]
```

## 4 3. Exploratory Data Analysis

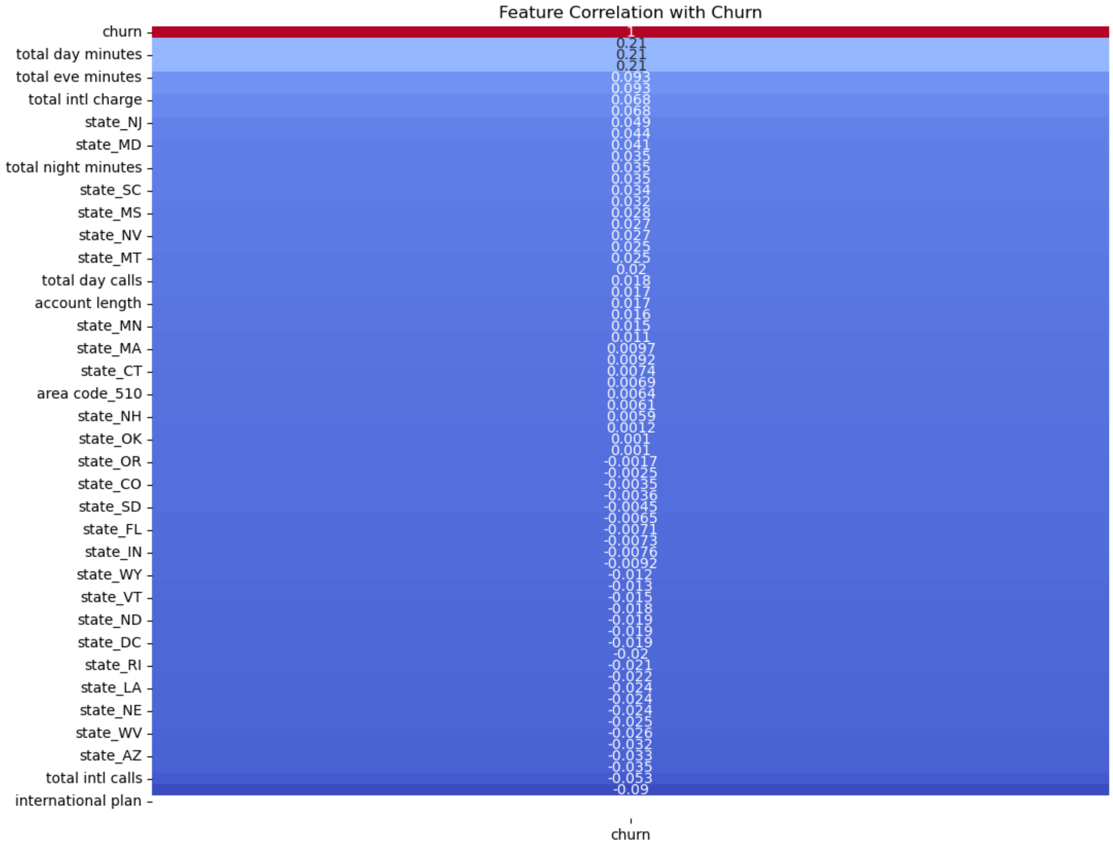
[46]:
```
# Plot churn distribution to check class imbalance

sns.countplot(x='churn', data=df)
plt.title('Churn Distribution')
plt.xlabel('Churn')
plt.ylabel('Count')

# Save the plot generated
```

```
plt.savefig("images/churn_distribution.png")
plt.show()
```




Observation: Majority of customers have not churned, showing a class imbalance. This could bias the model if not addressed.

[23]:
```
# Plot correlation of features with churn
plt.figure(figsize=(12, 10))
sns.heatmap(df.corr(numeric_only=True)[['churn']].sort_values(by='churn',␣
↪ascending=False),
            annot=True, cmap='coolwarm', cbar=False)
plt.title('Feature Correlation with Churn')
plt.show()
```




Observation: Features such as 'international plan', 'total day charge', and 'customer service calls have relatively high correlation with churn. These are important features for model inputs.

# 5 4. Train-Test Split and Scaling

[14]:
```
# Separate features and target variable
X = df.drop('churn', axis = 1)
y = df['churn']
```

[16]:
```
# Split into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42, stratify=y)
```

[24]:
```
# Normalise the feature values to ensure unfair weighting
# Fit the scaler on training data only to prevent data leakage
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

[26]:
```python
# Output shape and balance checks
# Shape of training features
print("\nTraining set size:", X_train.shape)

# Shape of test features
print("Test set size:", X_test.shape)
print("\nTarget distribution in training set:")

# Shows churn vs non-churn proportions in training data
print(y_train.value_counts(normalize=True))
```

```
Training set size: (2666, 69)
Test set size: (667, 69)

Target distribution in training set:
churn
0    0.855214
1    0.144786
Name: proportion, dtype: float64
```

This shows that 85.5% of customs in the training set did not churn, while 14.5% did indicating a class imbalance that should be considered during modeling.

# 6 5. Train & Evaluate Models

## 6.1 5.1 Logistic Regression Model

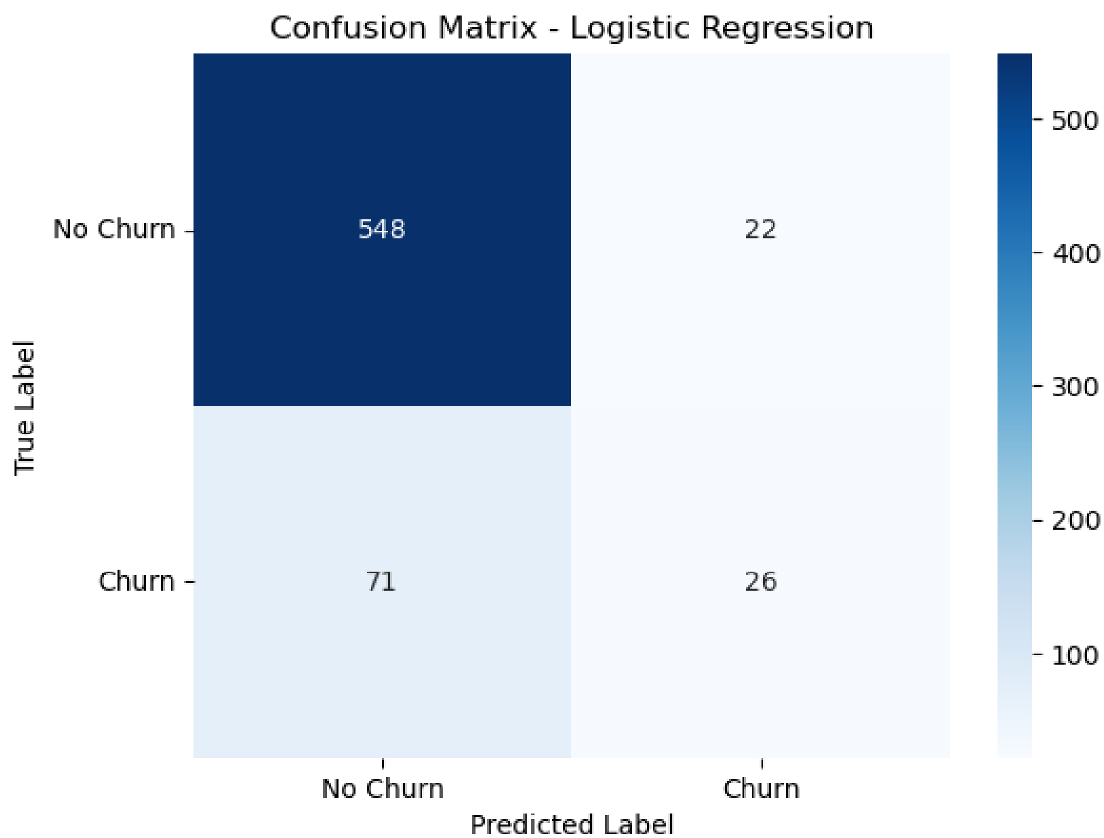
[27]:
```python
logreg = LogisticRegression(max_iter=1000)
logreg.fit(X_train_scaled, y_train)
y_pred_log = logreg.predict(X_test_scaled)
y_proba_log = logreg.predict_proba(X_test_scaled)[:, 1]
```

[33]:
```python
print("\nLogistic Regression Classification Report:")
print(classification_report(y_test, y_pred_log))
```

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

           0       0.89      0.96      0.92       570
           1       0.54      0.27      0.36        97

    accuracy                           0.86       667
   macro avg       0.71      0.61      0.64       667
weighted avg       0.84      0.86      0.84       667
```

The classification report shows: - Precision: 54% of churn predictions were correct. - Recall: The model caught 21% of real churners. - F1-score: A score of 0.36 shows the overall balance of precision and recall is moderate.

[47]:
```
from sklearn.metrics import ConfusionMatrixDisplay

# Visual confusion matrix for Logistic Regression
cm_log = confusion_matrix(y_test, y_pred_log)
sns.heatmap(cm_log, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix - Logistic Regression')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.xticks(ticks=[0.5, 1.5], labels=['No Churn', 'Churn'])
plt.yticks(ticks=[0.5, 1.5], labels=['No Churn', 'Churn'], rotation=0)

# Save plot generated
plt.savefig("images/confusion_matrix_logreg.png")
plt.show()
```




Confusion matrix breakdown: - True Negatives (top-left): 548 Non-churners correctly predicted. - False Positives (top-right): 22 Non-churners wrongly predicted as churners. - False Negatives (bottom-left): 71 Churners wrongly predicted as non-churners. - True Positives (bottom-right): 26

Churners correctly predicted.

[35]:
```python
print("ROC AUC Score:", roc_auc_score(y_test, y_proba_log))
```

```
ROC AUC Score: 0.8006872852233677
```

The ROC AUC score measures the model's ability to separate churners from non-churners across all thresholds.

Logistic Regression ROC AUC of 0.80 indicates strong ability to rank churners above non-churners.

### 6.2 5.2 Random Forest Model

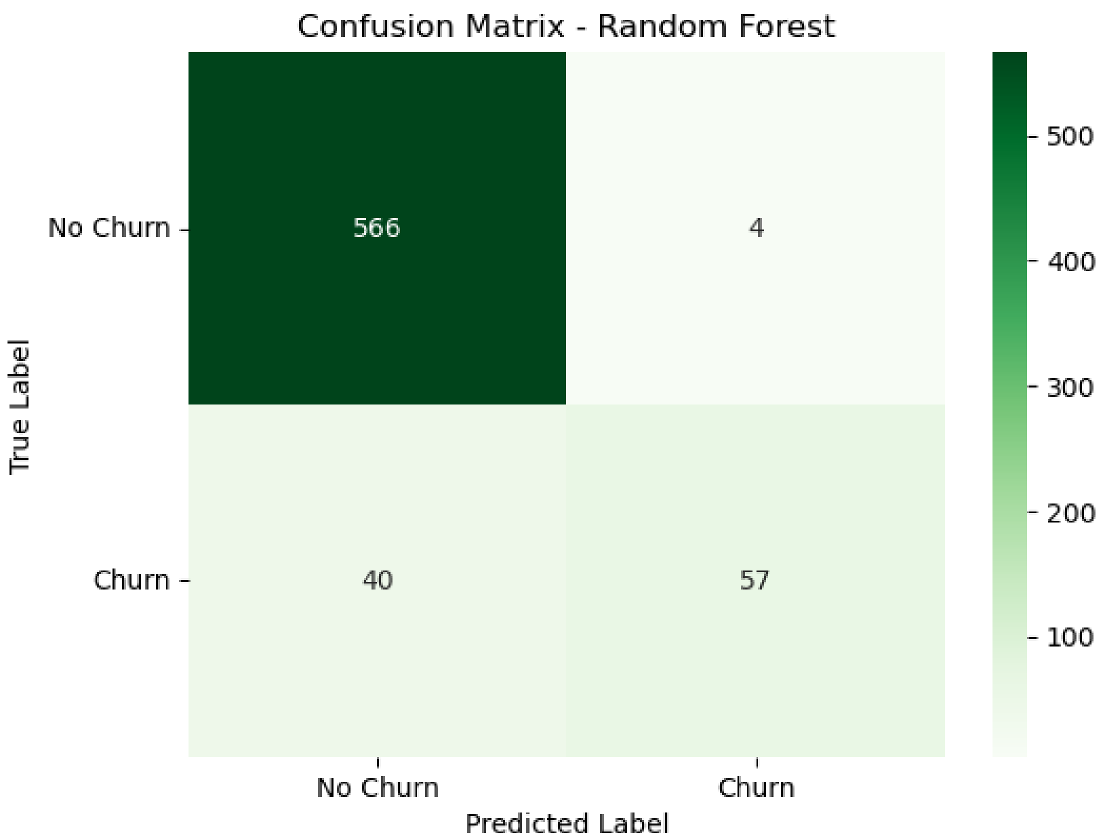
[37]:
```python
rf = RandomForestClassifier(random_state=42)
rf.fit(X_train, y_train)  # Tree-based models don't need scaled input
y_pred_rf = rf.predict(X_test)
y_proba_rf = rf.predict_proba(X_test)[:, 1]
```

[39]:
```python
print(classification_report(y_test, y_pred_rf))
```

```
              precision    recall  f1-score   support

           0       0.93      0.99      0.96       570
           1       0.93      0.59      0.72        97

    accuracy                           0.93       667
   macro avg       0.93      0.79      0.84       667
weighted avg       0.93      0.93      0.93       667
```

Interpretation: - Precision: Of all predicted churners, 93% were actual churners. - Recall: The model caught 59% of all churners. - F1-score: High average of 0.72 between precision and recall.

[49]:
```python
# Visual confusion matrix for Random Forest

cm_rf = confusion_matrix(y_test, y_pred_rf)
sns.heatmap(cm_rf, annot=True, fmt='d', cmap='Greens')
plt.title('Confusion Matrix - Random Forest')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.xticks(ticks=[0.5, 1.5], labels=['No Churn', 'Churn'])
plt.yticks(ticks=[0.5, 1.5], labels=['No Churn', 'Churn'], rotation=0)

# Save plot generated
plt.savefig("images/confusion_matrix_randforest.png")
plt.show()
```




Confusion Matrix Breakdown: - True Negatives (TN): 566 - False Positives (FP): 4 - False Negatives (FN): 40 - True Positives (TP): 57

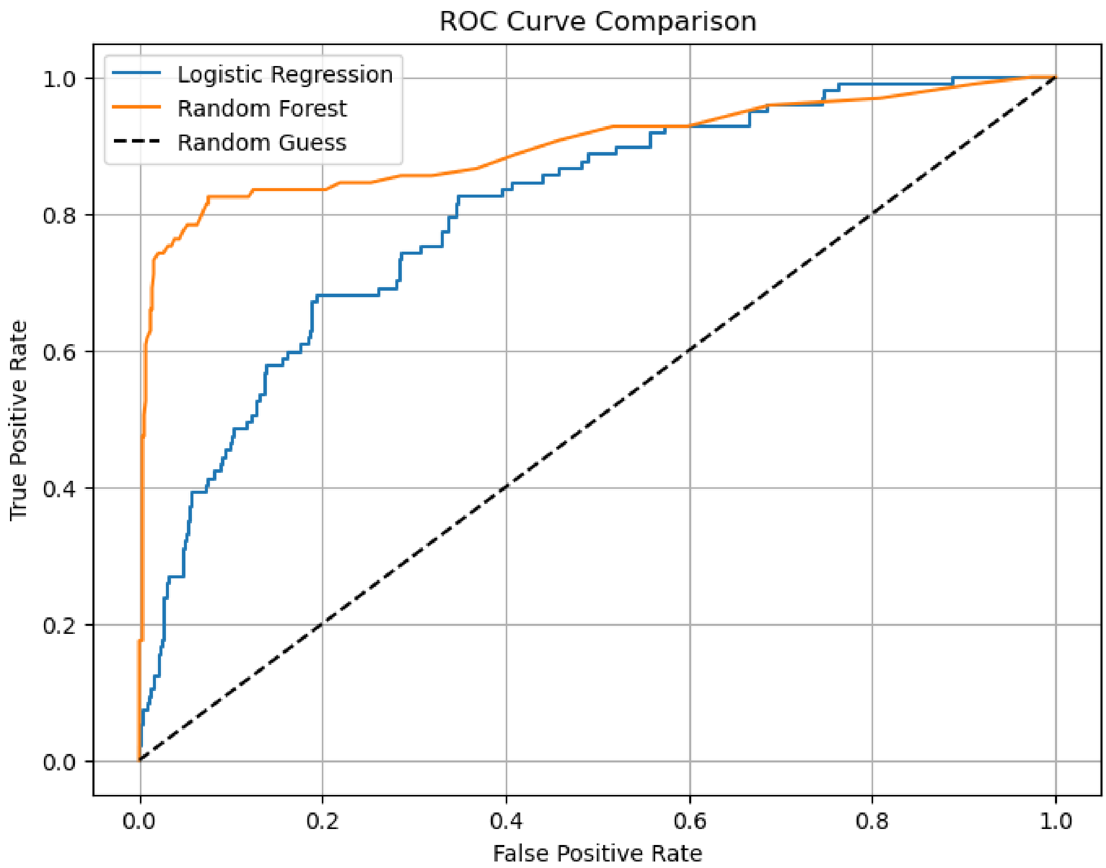
[41]:
```python
print("ROC AUC Score:", roc_auc_score(y_test, y_proba_rf))
```

```
ROC AUC Score: 0.899864351600651
```

ROC AUC Score: 0.89 shows excellent separation between churn and non-churn classes

[50]:
```python
# Visualize ROC Curves
fpr_log, tpr_log, _ = roc_curve(y_test, y_proba_log)
fpr_rf, tpr_rf, _ = roc_curve(y_test, y_proba_rf)

plt.figure(figsize=(8, 6))
plt.plot(fpr_log, tpr_log, label='Logistic Regression')
plt.plot(fpr_rf, tpr_rf, label='Random Forest')
plt.plot([0, 1], [0, 1], 'k--', label='Random Guess')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve Comparison')
plt.legend()
```

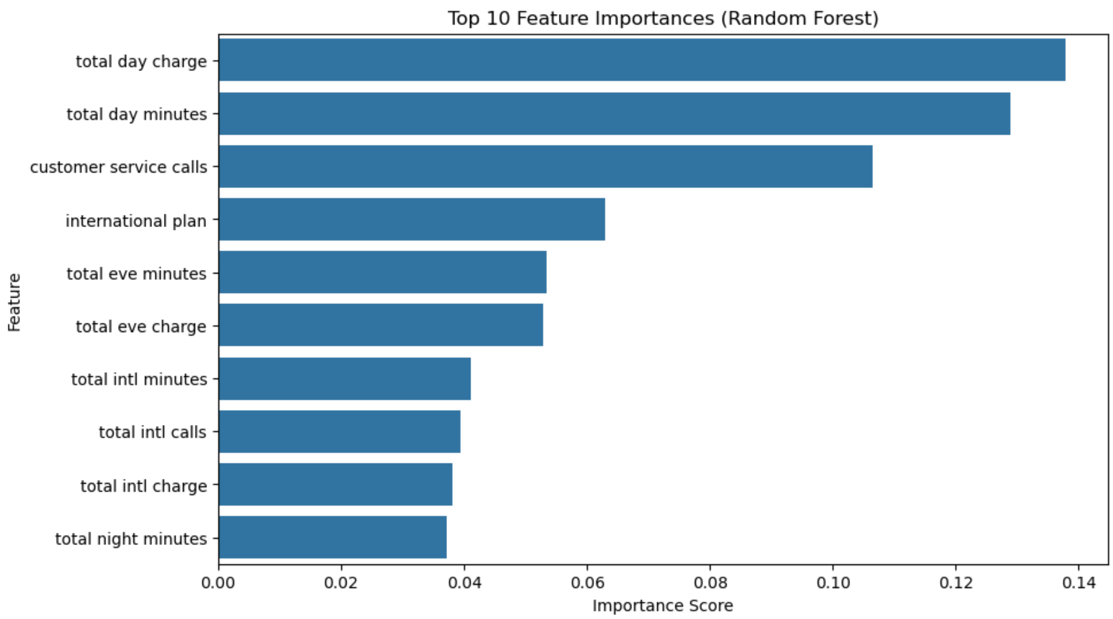
```
plt.grid(True)

# Save plot generated
plt.savefig("images/ROC_Curve.png")
plt.show()
```




Observation: The ROC curve shows Random Forest slightly outperforming Logistic Regression. Both models perform significantly better than random guessing.

[51]:
```
# Feature Importance from Random Forest

importances = pd.Series(rf.feature_importances_, index=X.columns)
importances_sorted = importances.sort_values(ascending=False)[:10]   # Top 10␣
↪features

plt.figure(figsize=(10, 6))
sns.barplot(x=importances_sorted.values, y=importances_sorted.index)
plt.title('Top 10 Feature Importances (Random Forest)')
plt.xlabel('Importance Score')
```

```python
plt.ylabel('Feature')

# Save plot generated
plt.savefig("images/features.png")
plt.show()
```




Observation: The most important features driving churn predictions include: - international plan - total day charge & minutes - number of customer service calls

These align well with EDA findings and should be the focus of retention strategies.

# 7 6.Recommendations

1. Focus retention efforts on customers with an international plan and high day charges.
2. Monitor customers making frequent customer service calls as they're more likely to churn.
3. Logistic regression gives interpretable results; random forest gives higher accuracy and handles feature interactions.
4. Use model predictions to proactively offer targeted promotions to high-risk customers.

[ ]:
```python

```