Project Overview

- SyriaTel Communications, a leading telecommunications provider, is facing challenges related to customer churn—the phenomenon where customers discontinue their services. Churn not only results in direct revenue loss but also incurs additional costs related to acquiring new customers.

- To address this, the project aims to build a machine learning model capable of predicting customer churn. This involves performing Exploratory Data Analysis (EDA) to uncover key behavioral patterns and using classification algorithms such as Logistic Regression and Decision Trees to identify customers at risk of leaving. By proactively identifying potential churners, SyriaTel can implement targeted retention strategies such as personalized offers or improved customer service, ultimately improving customer loyalty and reducing churn rates.

- This project demonstrates how data-driven insights can enable strategic decision-making, helping SyriaTel optimize customer retention efforts and enhance long-term profitability.

1. Business Understanding

- The primary objective is to develop a machine learning model that accurately predicts customer churn. This predictive capability will enable SyriaTel to:

- Recognize patterns leading to churn.

- Implement targeted retention strategies.

- Ultimately reduce customer turnover and associated costs.

Problem statement

- SyriaTel lacks a predictive mechanism to identify which customers are likely to leave the service. As a result, retention efforts are reactive rather than proactive, leading to higher operational costs and loss of revenue. The problem at hand is to:

- Build a classification model that can predict whether a customer is likely to churn, based on their behavioral and account-related data.

Objective

- To determine distribution of the churn
- To establish correlation matrix for numerical features
- To determine relationship of churn in relation to important features
- To predict whether a customer will "soon" stop doing business with SyriaTel telecommunications company.

- To find a way to prevent customer churn.
- To find a way of reducing revenue loss incase of customer churn.

Challanges

- Data Quality & Imbalance: Churn datasets often contain imbalanced classes, where non-churners vastly outnumber churners.
- Feature Relevance: Determining which customer attributes (e.g., usage patterns, service interactions) are most predictive of churn.
- Interpretability: Business stakeholders require not just predictions, but understandable insights to act upon.
- Cost Sensitivity: Misclassifying a churner as a non-churner has higher financial consequences than the reverse.

Proposed Solution

- Exploratory Data Analysis (EDA): Understand key trends and correlations in customer data.
- Predictive Modeling: Use Logistic Regression and Decision Tree Classifiers to model churn risk.
- Feature Engineering: Identify and transform relevant features such as call durations, customer service interactions, and usage patterns.
- Model Evaluation: Emphasize recall (true positive rate) to minimize false negatives —critical for catching churners early.
- Actionable Insights: Derive key drivers of churn to support targeted interventions.

Conclusion

- The successful implementation of a churn prediction model can provide SyriaTel with actionable insights into customer behavior, empowering the company to intervene before customers leave. This data-driven approach not only improves retention but also enhances customer satisfaction and reduces operational costs related to acquiring new clients. Through a blend of analytics and strategic action, SyriaTel can transform churn management into a competitive advantage.

2. Data Understanding

- Here we explore the dataset to understand its structure, content, and quality to assess its suitability for predicting customer churn for SyriaTel.

Data Source:

- The dataset is sourced from: https://www.kaggle.com/datasets/becksddf/churn-in-telecoms-dataset/data.

Import Relevant Libraries and Load Dataset

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
```

```
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import RandomizedSearchCV
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report, accuracy_score
from sklearn.metrics import accuracy_score, r2_score, recall_score,
precision_score, roc_curve, roc_auc_score, f1_score
from sklearn.metrics import classification_report, confusion_matrix
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import mean_absolute_error, mean_squared_error,
r2_score
from sklearn.metrics import ConfusionMatrixDisplay

churn_data = pd.read_csv('bigml_59c28831336c6604c800002a.csv')
churn_data.head()

  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]
```

```
churn_data.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
```

Dataset Overview

- The dataset used in this project contains detailed customer information from SyriaTel Communications. It includes demographic, service usage, and account activity features that are essential for understanding customer behavior and predicting churn. Each row in the dataset represents a unique customer, and the key target variable is churn, a binary label indicating whether a customer has left the service (True) or remained (False).

Key Features in the Dataset:

- **Customer Service Interactions:** Number of calls made to customer support—frequent calls may indicate dissatisfaction.
- **Account Information:** Such as contract type, monthly charges, total charges, and tenure (length of customer relationship).
- **Usage Metrics:** Includes usage of different telecom services (e.g., data, calls).
- **Demographics:** Gender, senior citizen status, and other personal attributes.
- **Billing and Payment Methods:** Whether customers use electronic billing or automatic payments.

Target Variable:

- **churn:** Boolean value (True = customer has churned, False = customer is retained).

Initial Observations:

- The dataset is imbalanced, with a significantly higher number of non-churned customers compared to churned ones.
- Data preprocessing steps such as normalization, encoding of categorical features, and handling missing values should be applied to prepare the data for modeling.

-This comprehensive dataset forms the foundation for developing predictive models and extracting actionable insights to reduce customer churn.

3. Data Preparation

Objective

- Prepare the dataset for modeling by cleaning, transforming, and structuring the data to ensure it is suitable for training **Logistic Regression** and **Decision Tree** classifiers to predict customer churn for SyriaTel.
- The goal is to address issues identified in the **Data Understanding** phase, such as categorical variables, irrelevant features, multicollinearity, and class imbalance, while ensuring the data is in a format compatible with both models.

Handle Missing Values

- Check for missing values in all columns.

```
# Check for null values
churn_data.isna().sum()

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
```

There are no missing values in our dataset

```
# Drop the 'state' and 'phone number' columns
# Not useful for the analysis
churn_data_df = churn_data.drop(['state', 'phone number'], axis=1)
```

Checking for duplicates

```
churn_data.duplicated().sum()

0
```

There are duplicates in our dataset

Handle Multicollinearity

- Features like `total day minutes` and `total day charge` are likely highly correlated as charges are derived from minutes.
- To avoid multicollinearity in Logistic Regression, we can drop charge-related features (`total day charge`, `total eve charge`, `total night charge`, `total intl charge`) and keep minutes, as they capture similar information.

```
# Handle multicollinearity by dropping charge-related features
churn_data_df = churn_data_df.drop(['total day charge', 'total eve
charge', 'total night charge', 'total intl charge'], axis=1)

churn_data_df.head()

   account length  area code international plan voice mail plan  \
0             128        415                 no             yes
1             107        415                 no             yes
2             137        415                 no              no
3              84        408                yes              no
4              75        415                yes              no

   number vmail messages  total day minutes  total day calls  \
0                     25              265.1              110
1                     26              161.6              123
2                      0              243.4              114
3                      0              299.4               71
4                      0              166.7              113

   total eve minutes  total eve calls  total night minutes  total
night calls  \
0              197.4               99                244.7
91
1              195.5              103                254.4
103
2              121.2              110                162.6
104
3               61.9               88                196.9
89
4              148.3              122                186.9
121

   total intl minutes  total intl calls  customer service calls  churn

0                10.0                 3                       1  False

1                13.7                 3                       1  False

2                12.2                 5                       0  False

3                 6.6                 7                       2  False

4                10.1                 3                       3  False
```

Encode Categorical Variables

- Convert categorical features (`international plan`, `voice mail plan`, `area code`) into numerical format.

- Use **Label Encoding** for binary categorical variables (`international plan`, `voice mail plan`) and the target (`churn`).

```
# Encode categorical variables
le = LabelEncoder()
churn_data_df['international plan'] =
le.fit_transform(churn_data_df['international plan'])  # yes=1, no=0
churn_data_df['voice mail plan'] =
le.fit_transform(churn_data_df['voice mail plan'])  # yes=1, no=0
churn_data_df['churn'] = le.fit_transform(churn_data_df['churn'])  #
True=1, False=0

# One-hot encode area code
churn_data_df = pd.get_dummies(churn_data_df, columns=['area code'],
prefix='area_code')


churn_data_df.head()

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

   total day minutes  total day calls  total eve minutes  total eve
calls  \
0              265.1              110              197.4
99
1              161.6              123              195.5
103
2              243.4              114              121.2
110
3              299.4               71               61.9
88
4              166.7              113              148.3
122

   total night minutes  total night calls  total intl minutes  \
0                244.7                 91                10.0
1                254.4                103                13.7
2                162.6                104                12.2
3                196.9                 89                 6.6
```

```
4                186.9                121                 10.1

   total intl calls  customer service calls  churn  area_code_408  \
0                 3                       1      0          False
1                 3                       1      0          False
2                 5                       0      0          False
3                 7                       2      0           True
4                 3                       3      0          False

   area_code_415  area_code_510
0           True          False
1           True          False
2           True          False
3          False          False
4           True          False
```

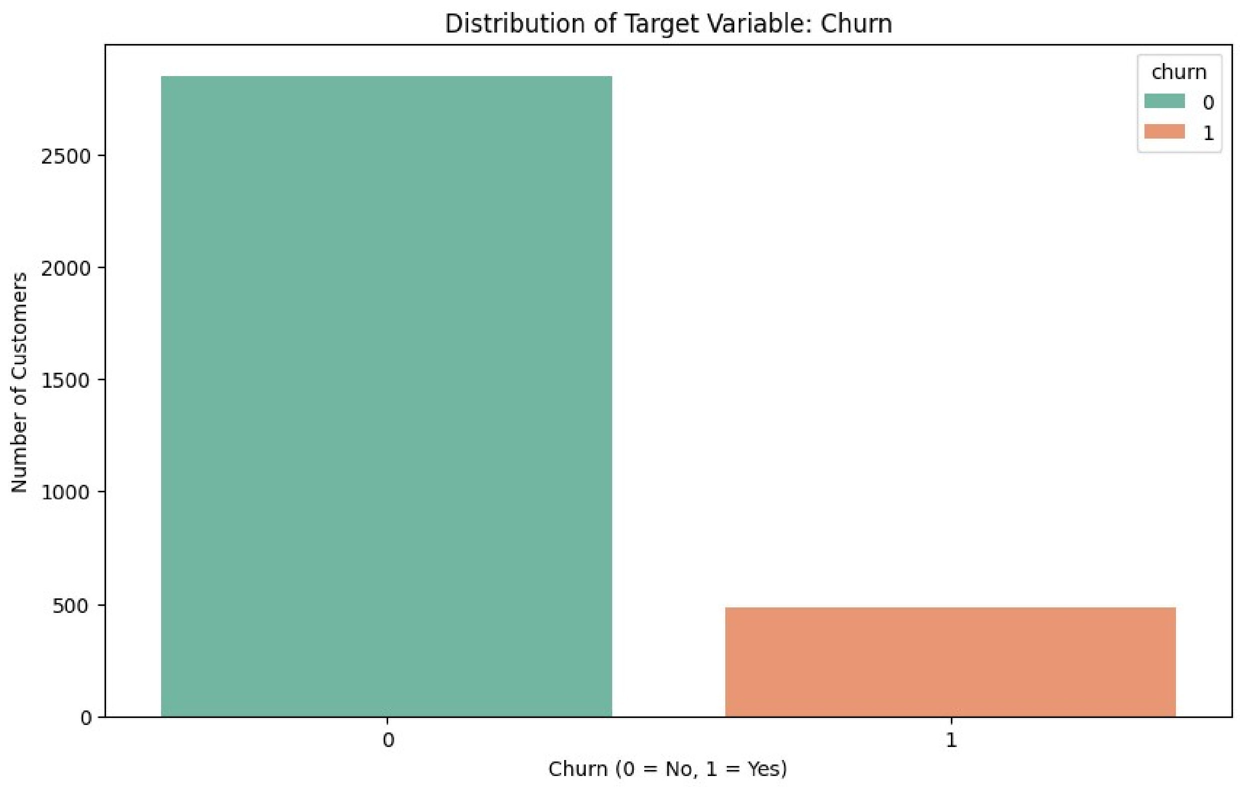
```
churn_data['churn'].value_counts()
```

```
churn
False    2850
True      483
Name: count, dtype: int64
```

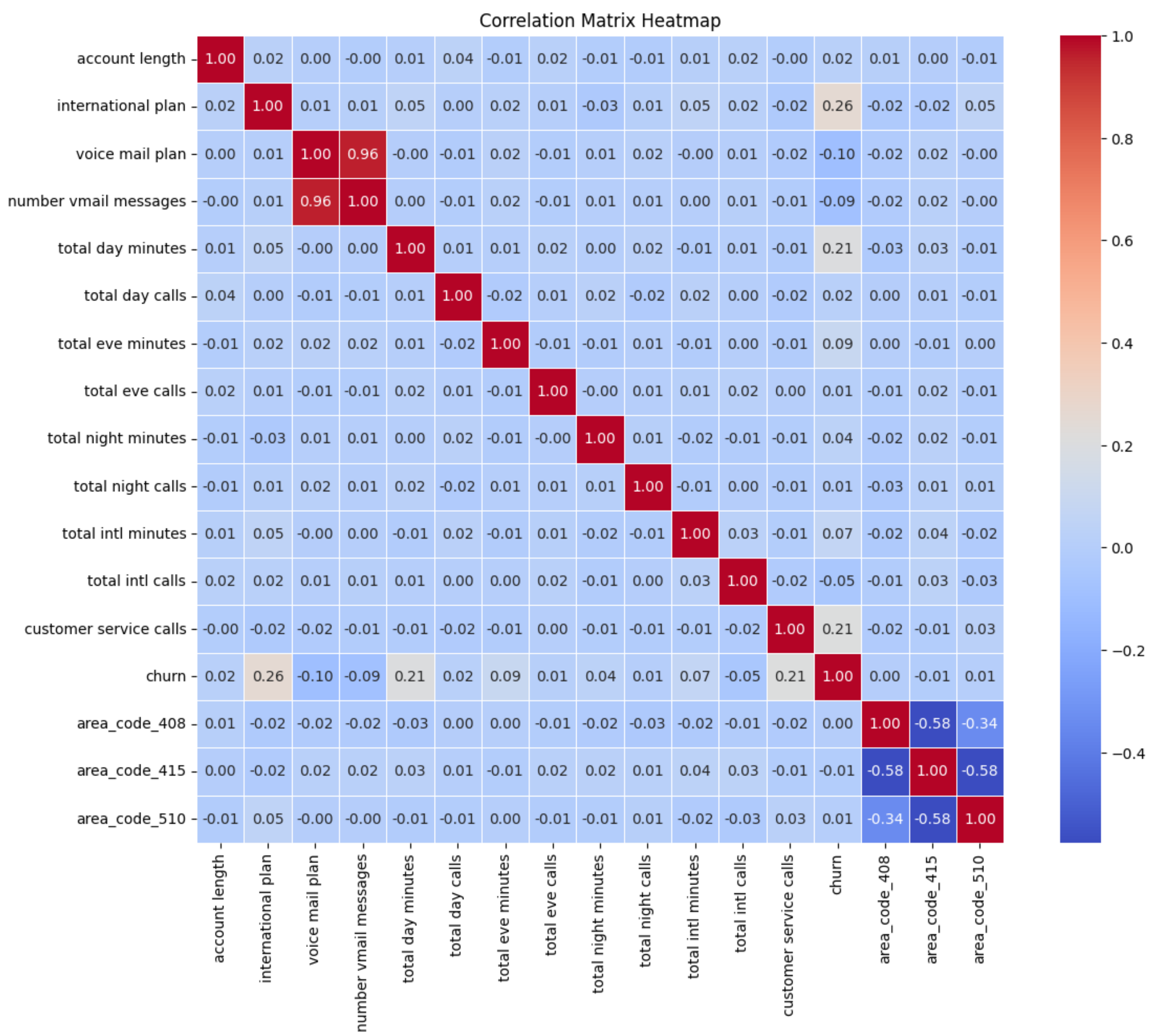
```
# Check class balance for the target variable
print(churn_data_df['churn'].value_counts(normalize=True)*100)
```

```
churn
0    85.508551
1    14.491449
Name: proportion, dtype: float64
```

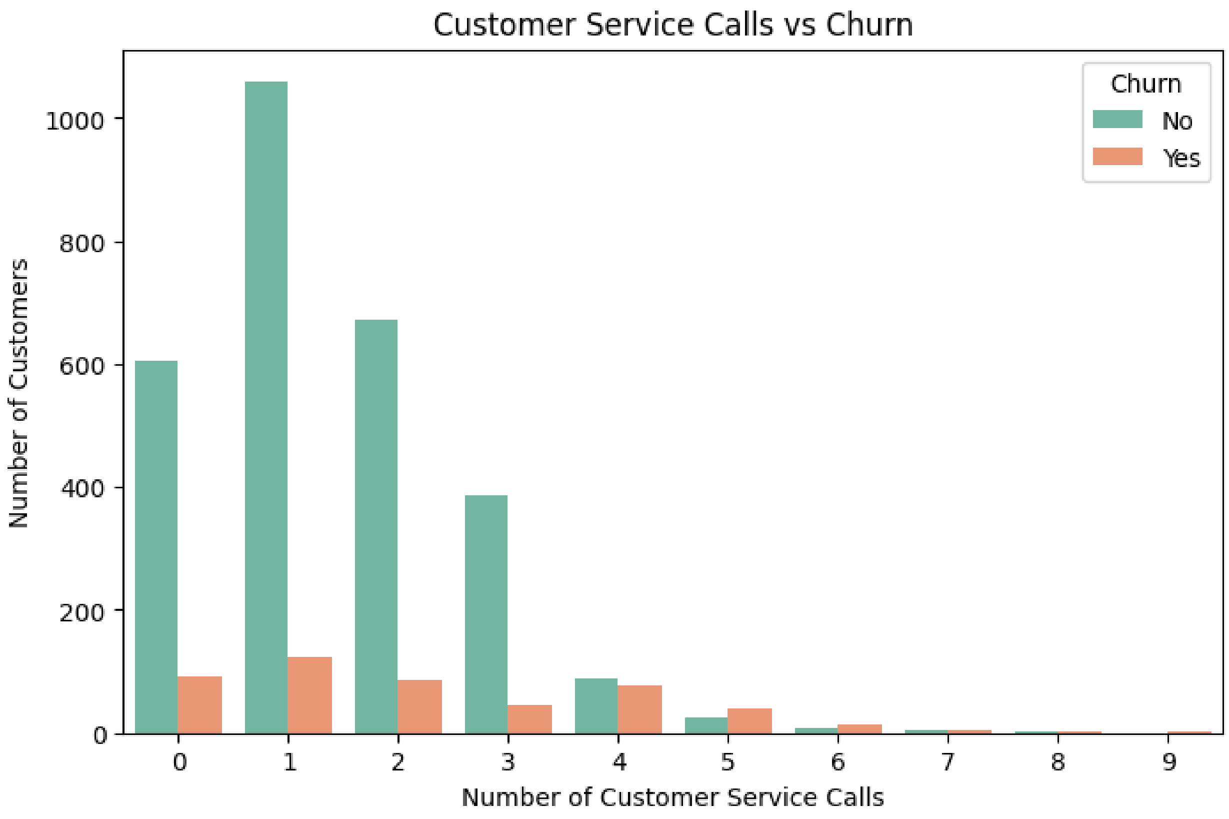
```
# Plot the distribution of the churn variable
plt.figure(figsize=(10,6))
sns.countplot(x='churn', data=churn_data_df,
palette='Set2',hue='churn')
plt.title('Distribution of Target Variable: Churn')
plt.xlabel('Churn (0 = No, 1 = Yes)')
plt.ylabel('Number of Customers')
legend = False
plt.show()
```

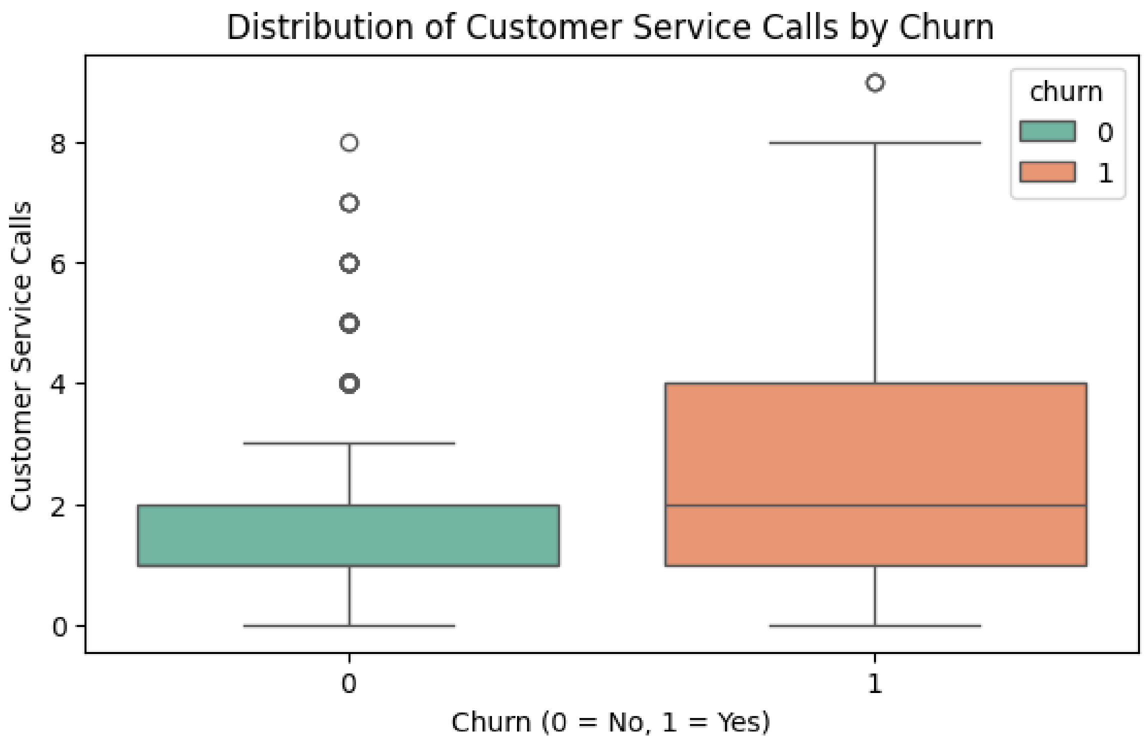


```python
# Compute correlation matrix for numerical features
corr_matrix = churn_data_df.corr(numeric_only=True)

plt.figure(figsize=(14,10))
sns.heatmap(corr_matrix, annot=True, fmt=".2f", cmap='coolwarm',
square=True, linewidths=.5)
plt.title('Correlation Matrix Heatmap')
plt.show()
```

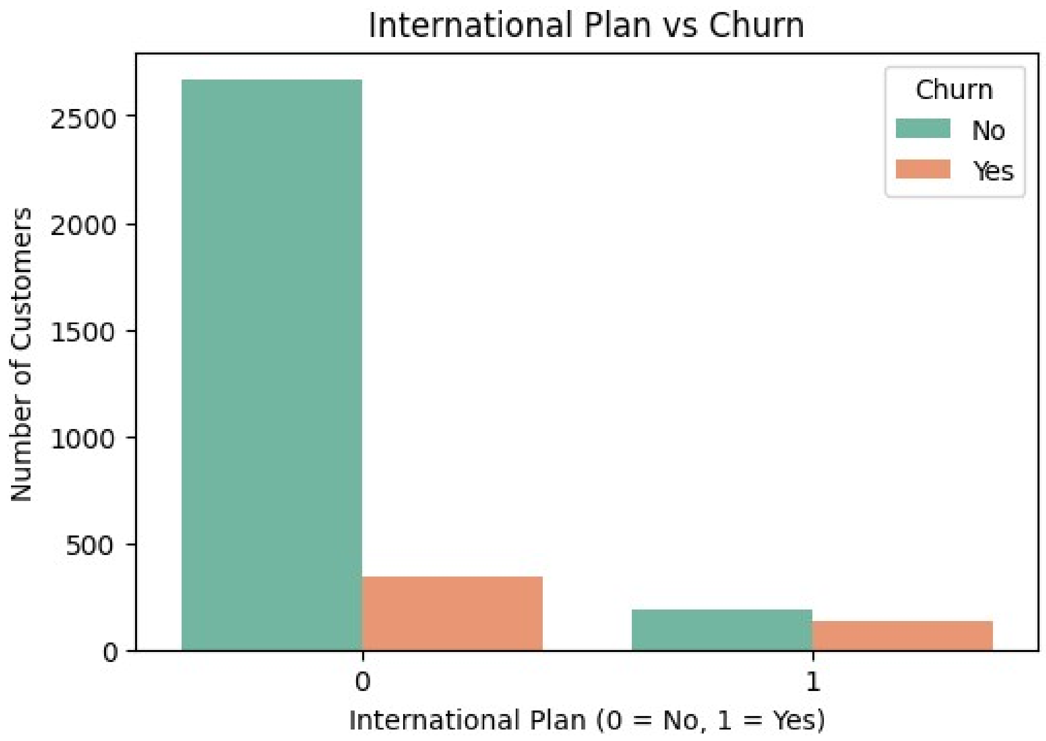


```
## Visualize relationship between customer service calls and churn
plt.figure(figsize=(8,5))
sns.countplot(x='customer service calls', hue='churn',
data=churn_data_df, palette='Set2',)
plt.title('Customer Service Calls vs Churn')
plt.xlabel('Number of Customer Service Calls')
plt.ylabel('Number of Customers')
plt.legend(title='Churn', labels=['No', 'Yes'])
plt.show()

# Boxplot for distribution
plt.figure(figsize=(7,4))
sns.boxplot(x='churn', y='customer service calls', data=churn_data_df,
palette='Set2',hue= 'churn')
plt.title('Distribution of Customer Service Calls by Churn')
plt.xlabel('Churn (0 = No, 1 = Yes)')
plt.ylabel('Customer Service Calls')
legend= False
plt.show()
```





```
# Visualize relationship between international plan and churn
plt.figure(figsize=(6,4))
sns.countplot(x='international plan', hue='churn', data=churn_data_df,
palette='Set2')
plt.title('International Plan vs Churn')
plt.xlabel('International Plan (0 = No, 1 = Yes)')
plt.ylabel('Number of Customers')
plt.legend(title='Churn', labels=['No', 'Yes'])
plt.show()
```



Split the Dataset

- Split the data into training (80%) and testing (20%) sets, using stratification to maintain the churn class distribution.

```
# Split the data into training (80%) and testing (20%) sets, using
stratification to maintain the churn class distribution.
X = churn_data_df.drop('churn', axis=1)
y = churn_data_df['churn']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42, stratify=y)
```

Feature Scaling

```
# feature scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

4. **Modeling**

The modeling process is guided by a combination of:

- **Business goals**- minimize churn-related losses.
- **Data behavior**- imbalance and mixed data types.
- **Interpretability**- so stakeholders can trust and act on predictions.

- **Statistical performance metrics**- recall, F1, ROC-AUC.

Model Selection

In this project, two primary models were selected based on the nature of the problem and business needs:

1. **Logistic Regression** Shows how each feature influences churn.It is simple and interpretable and best for Understanding relationships between variables and churn

2. **Decision Tree Classifier** Handles non-linear relationships and categorical variables well and provides clear decision rules.It is best for Discovering churn patterns and rule-based insights

- These models strike a balance between predictive power and interpretability, both critical for enabling SyriaTel to take informed, proactive actions to reduce churn.

Hyperparameter Tuning

Both models were fine-tuned using RandomizedSearchCV. Randomized search is often faster and can explore a wider range of values, especially useful if you have many parameters or a large dataset.

For **Logistic Regression**, we will adjust the regularization parameter `C` to control model complexity and prevent overfitting.

- For **Decision Tree**, we will tune `max_depth` and `min_samples_split` to balance model complexity and generalization, avoiding overfitting.

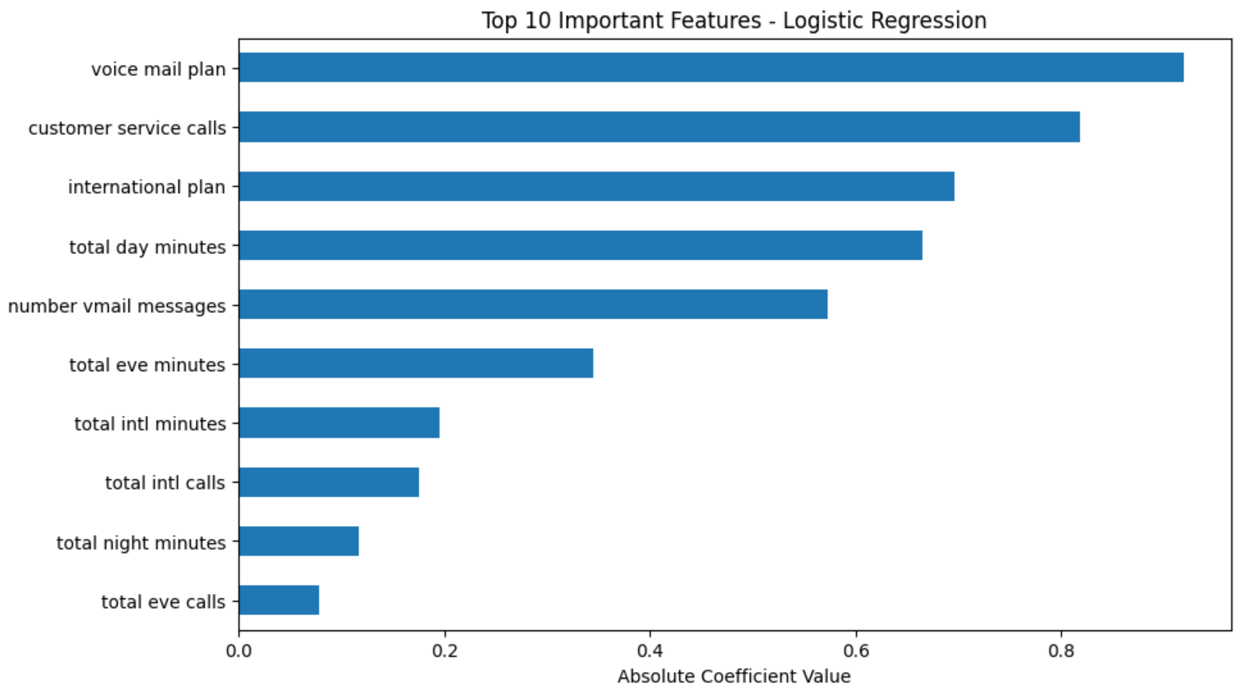
Handling Class Imbalance and Training

- The dataset may have an imbalanced `churn` variable (e.g., ~15% True, ~85% False, based on typical telecom datasets). We will use class weights (`class_weight='balanced'`) in both models to give more importance to the minority class (churners) during training.

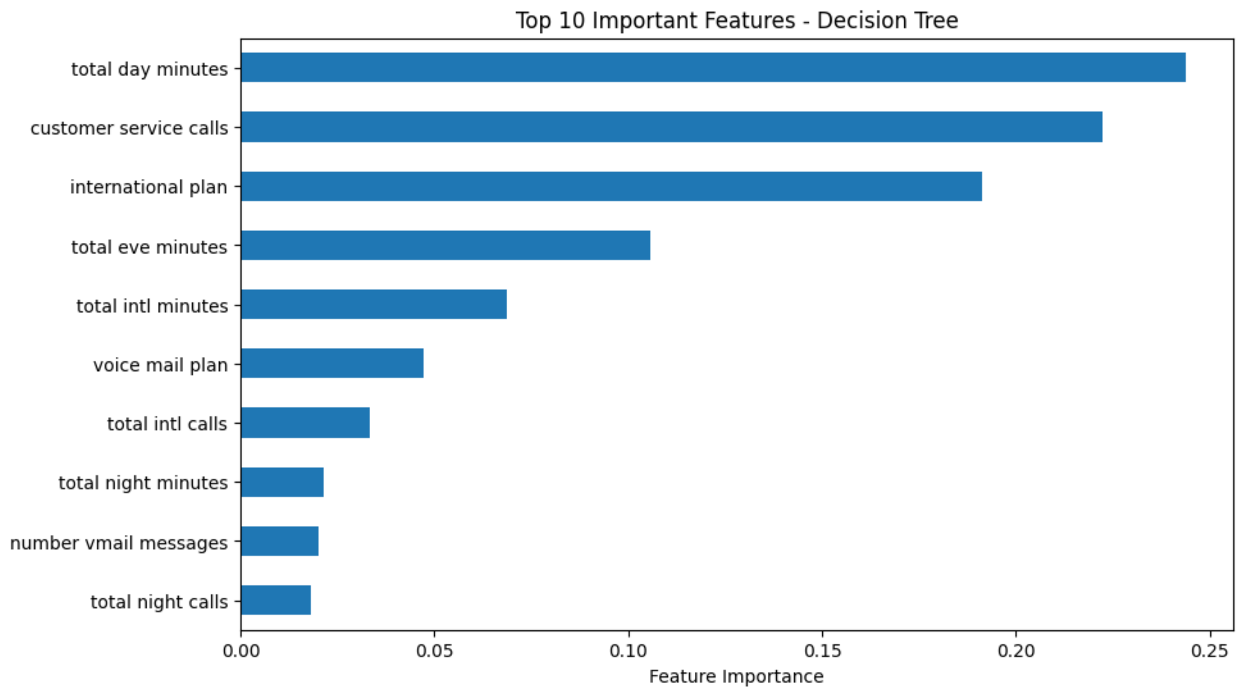
```
# Logistic Regression: trying wider range for C and different solvers
log_reg = LogisticRegression(class_weight='balanced', random_state=42,
max_iter=1000)
log_reg_param_dist = {
    'C': np.logspace(-3, 2, 20),  # 0.001 to 100
    'solver': ['liblinear', 'lbfgs', 'saga']
}
log_reg_random = RandomizedSearchCV(
    log_reg, log_reg_param_dist, n_iter=10, cv=5, scoring='roc_auc',
n_jobs=-1, random_state=42
)
log_reg_random.fit(X_train_scaled, y_train)
best_log_reg_random = log_reg_random.best_estimator_
print("Best Logistic Regression Params (Randomized):",
log_reg_random.best_params_)
print("Best Logistic Regression ROC-AUC (Randomized):",
log_reg_random.best_score_)

# Decision Tree: trying more values for max_depth, min_samples_split,
and add min_samples_leaf
dec_tree = DecisionTreeClassifier(class_weight='balanced',
random_state=42)
dec_tree_param_dist = {
    'max_depth': np.arange(3, 15),
    'min_samples_split': np.arange(2, 20),
    'min_samples_leaf': np.arange(1, 10)
}

dec_tree_random = RandomizedSearchCV(
    dec_tree, dec_tree_param_dist, n_iter=15, cv=5, scoring='roc_auc',
n_jobs=-1, random_state=42
)
dec_tree_random.fit(X_train, y_train)
best_dec_tree_random = dec_tree_random.best_estimator_
print("Best Decision Tree Params (Randomized):",
dec_tree_random.best_params_)
print("Best Decision Tree ROC-AUC (Randomized):",
dec_tree_random.best_score_)
```

```
Best Logistic Regression Params (Randomized): {'solver': 'liblinear',
'C': 0.7847599703514607}
Best Logistic Regression ROC-AUC (Randomized): 0.8175237043658097
Best Decision Tree Params (Randomized): {'min_samples_split': 17,
'min_samples_leaf': 7, 'max_depth': 9}
Best Decision Tree ROC-AUC (Randomized): 0.8901629437813646
```

```
# For Logistic Regression (use absolute value of coefficients)
log_reg_coef = pd.Series(
    abs(best_log_reg_random.coef_[0]),
    index=X.columns
).sort_values(ascending=False)

plt.figure(figsize=(10, 6))
log_reg_coef.head(10).plot(kind='barh')
plt.title('Top 10 Important Features - Logistic Regression')
plt.xlabel('Absolute Coefficient Value')
plt.gca().invert_yaxis()
plt.show()
```



```python
# Visualize important features for Decision Tree
dec_tree_importance = pd.Series(
    best_dec_tree_random.feature_importances_,
    index=X.columns
).sort_values(ascending=False)

plt.figure(figsize=(10, 6))
dec_tree_importance.head(10).plot(kind='barh')
plt.title('Top 10 Important Features - Decision Tree')
plt.xlabel('Feature Importance')
plt.gca().invert_yaxis()
plt.show()
```



```
# Make predictions on the test set using the best models
y_pred_log_reg = best_log_reg_random.predict(X_test_scaled)
y_pred_dec_tree = best_dec_tree_random.predict(X_test)
```

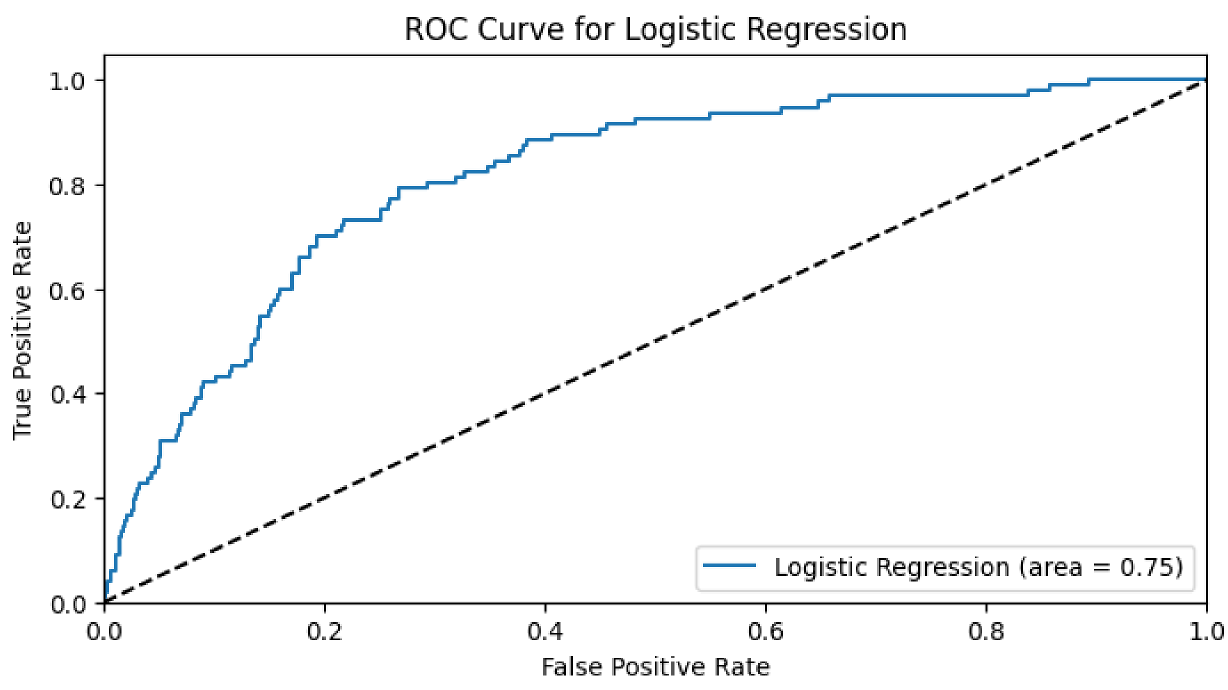
5. **Model Evaluation**

1. **Logistic Regression**

```
# Calculate and print evaluation metrics for Logistic Regression

print("Logistic Regression Evaluation Metrics:")
print("Accuracy:", accuracy_score(y_test, y_pred_log_reg))
print("R2 Score:", r2_score(y_test, y_pred_log_reg))
print("Recall:", recall_score(y_test, y_pred_log_reg))
print("Precision:", precision_score(y_test, y_pred_log_reg))
print("Mean Absolute Error:", mean_absolute_error(y_test,
y_pred_log_reg))
print("Mean Squared Error:", mean_squared_error(y_test,
y_pred_log_reg))
print("ROC AUC:", roc_auc_score(y_test, y_pred_log_reg))
print("F1 Score:", f1_score(y_test, y_pred_log_reg))
print("Classification Report:\n", classification_report(y_test,
y_pred_log_reg))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_log_reg))
```

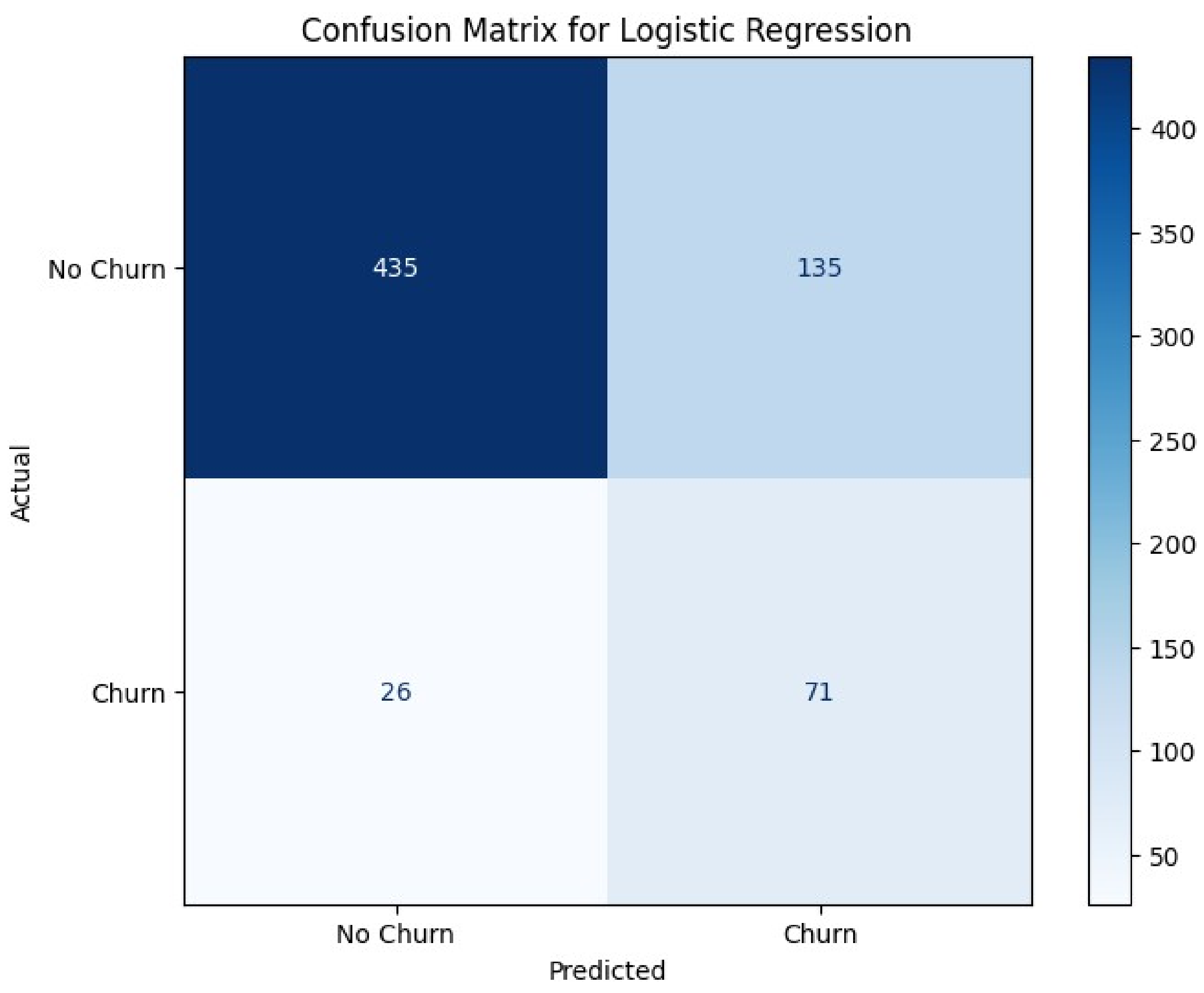
```
Logistic Regression Evaluation Metrics:
Accuracy: 0.7586206896551724
R2 Score: -0.9422499547838667
```

```
Recall: 0.7319587628865979
Precision: 0.3446601941747573
Mean Absolute Error: 0.2413793103448276
Mean Squared Error: 0.2413793103448276
ROC AUC: 0.7475583288117201
F1 Score: 0.46864686468646866
Classification Report:
               precision    recall  f1-score   support

           0       0.94      0.76      0.84       570
           1       0.34      0.73      0.47        97

    accuracy                           0.76       667
   macro avg       0.64      0.75      0.66       667
weighted avg       0.86      0.76      0.79       667

Confusion Matrix:
 [[435 135]
 [ 26  71]]

#Logistic Regression ROC Curve
fpr_log_reg, tpr_log_reg, thresholds_log_reg = roc_curve(y_test,
best_log_reg_random.predict_proba(X_test_scaled)[:, 1])
plt.figure(figsize=(8, 4))
plt.plot(fpr_log_reg, tpr_log_reg, label='Logistic Regression (area =
{:.2f})'.format(roc_auc_score(y_test, y_pred_log_reg)))
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve for Logistic Regression')
plt.legend(loc='lower right')
plt.show()
```



From the above code, we get an accuracy of 0.75 and a ROC AUC of 0.74 for Logistic Regression

```
cm = confusion_matrix(y_test, y_pred_log_reg)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['No
Churn', 'Churn'])
fig, ax = plt.subplots(figsize=(8, 6))
disp.plot(ax=ax, cmap='Blues', values_format='d')
plt.title('Confusion Matrix for Logistic Regression')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```



Explaining the confusion matrix

-**True Negative (TN)= 435:** The number of customers who did not churn and were correctly predicted as non-churners.

-**False Positive (FP)= 135:** The number of customers who did not churn but were incorrectly predicted as churners (Type I error).

-**False Negative (FN)=26:** The number of customers who churned but were incorrectly predicted as non-churners (Type II error).

-**True Positive (TP)=71:** The number of customers who churned and were correctly predicted as churners.

In summary;

- High TN and TP mean this model is making correct predictions.
- High FP means the model is flagging too many loyal customers as churners, which could waste retention resources.
- High FN means the model is missing actual churners, which could lead to revenue loss.

Strengths

Recall (73.20%): The model correctly identifies 73.20% of actual churners (71 out of 97). This is a key strength because, for SyriaTel, catching as many customers at risk of leaving as possible is critical. High recall means the model is effective at flagging most customers who are likely to churn, allowing the business to take proactive retention actions.

Accuracy (75.86%): The model correctly predicts the outcome for about 76% of all customers. This shows the model performs well overall. However, because the dataset is imbalanced having more non-churners than churners, accuracy alone can be misleading. Still, a high accuracy indicates that the model is making correct predictions for the majority of cases

Weaknesses

-**Precision (34.63%):** Only about one-third of customers predicted to churn actually do. This means that many customers who are flagged as likely to leave are actually loyal and would have stayed. As a result, SyriaTel may waste resources (such as special offers or retention campaigns) on customers who are not at risk.

-**High False Positive Rate (134 customers):** The model incorrectly predicts that 134 non-churners will churn. This increases operational costs because the company might target too many customers with unnecessary retention efforts.

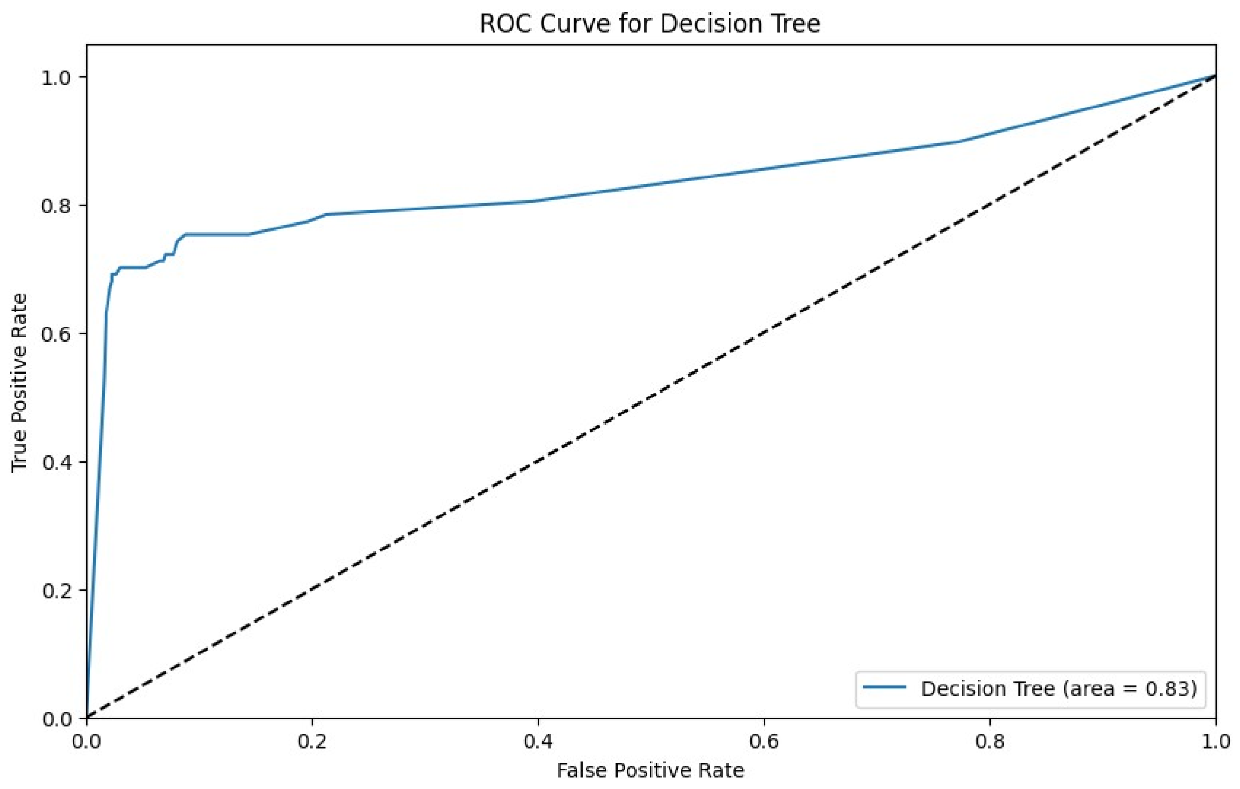
-**F1-Score (47.02%):** The F1-score is a balance between precision and recall. A moderate F1-score indicates that the model does not perform exceptionally well in both catching churners and avoiding false alarms. This means there is a trade-off: while the model catches most churners (high recall), it also mislabels many loyal customers as churners (low precision).

2. **Decision Tree**

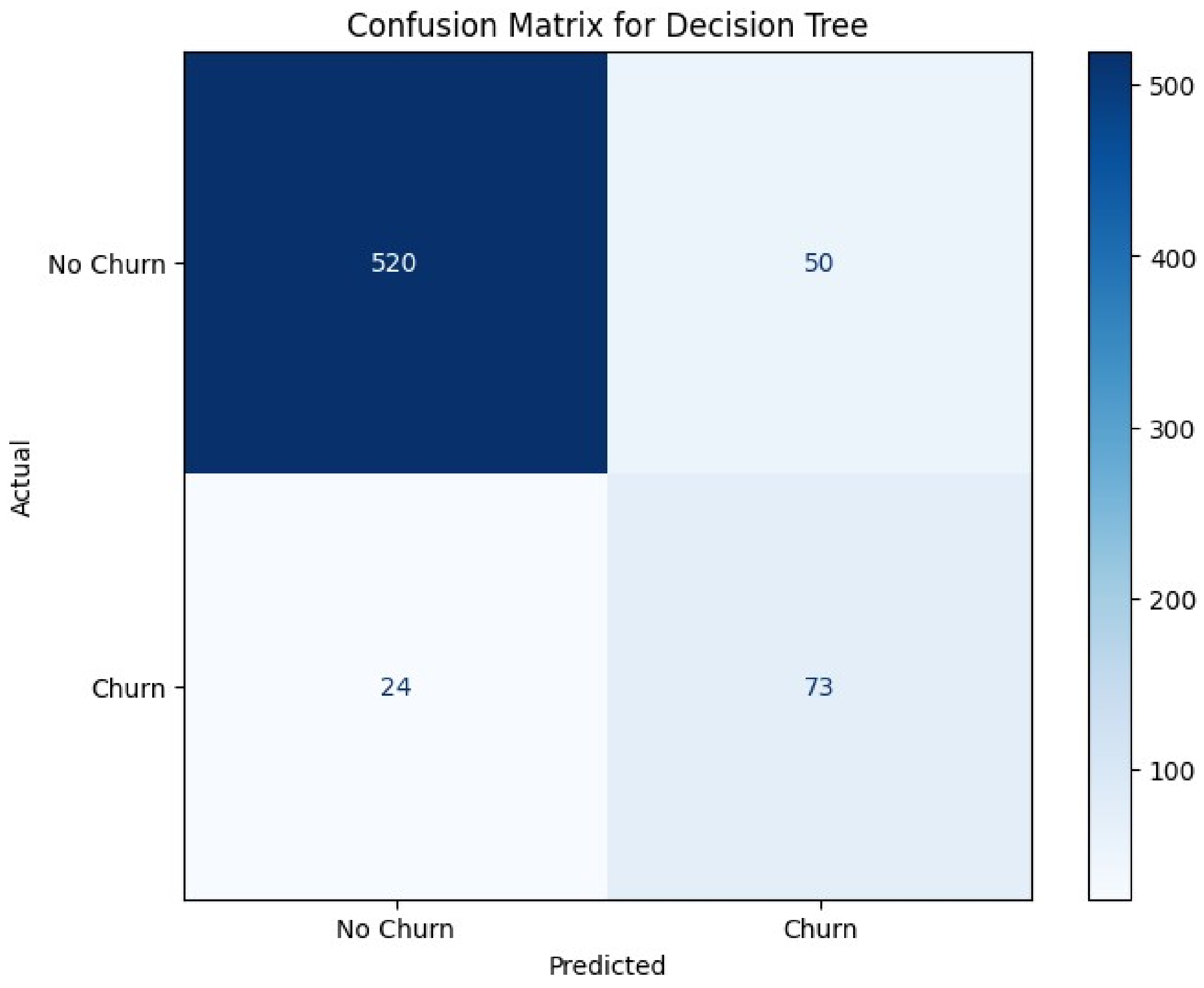
```
# Calculate and print evaluation metrics for Decision Tree
print("Decision Tree Evaluation Metrics:")
print("Accuracy:", accuracy_score(y_test, y_pred_dec_tree))
print("R2 Score:", r2_score(y_test, y_pred_dec_tree))
print("Recall:", recall_score(y_test, y_pred_dec_tree))
print("Precision:", precision_score(y_test, y_pred_dec_tree))
print("Mean Absolute Error:", mean_absolute_error(y_test,
y_pred_dec_tree))
print("Mean Squared Error:", mean_squared_error(y_test,
y_pred_dec_tree))
print("ROC AUC:", roc_auc_score(y_test, y_pred_dec_tree))
print("F1 Score:", f1_score(y_test, y_pred_dec_tree))
print("Classification Report:\n", classification_report(y_test,
y_pred_dec_tree))
print("Confusion Matrix:\n", confusion_matrix(y_test,
y_pred_dec_tree))
```

```
Decision Tree Evaluation Metrics:
Accuracy: 0.889055472263868
R2 Score: 0.10728884065834698
Recall: 0.7525773195876289
```

```
Precision: 0.5934959349593496
Mean Absolute Error: 0.11094452773613193
Mean Squared Error: 0.11094452773613193
ROC AUC: 0.8324290106710074
F1 Score: 0.6636363636363637
Classification Report:
               precision    recall  f1-score   support

           0       0.96      0.91      0.93       570
           1       0.59      0.75      0.66        97

    accuracy                           0.89       667
   macro avg       0.77      0.83      0.80       667
weighted avg       0.90      0.89      0.89       667

Confusion Matrix:
 [[520  50]
 [ 24  73]]

# Plot ROC curve for Decision Tree
fpr_dec_tree, tpr_dec_tree, thresholds_dec_tree = roc_curve(y_test,
best_dec_tree_random.predict_proba(X_test)[:, 1])
plt.figure(figsize=(10, 6))
plt.plot(fpr_dec_tree, tpr_dec_tree, label='Decision Tree (area =
{:.2f})'.format(roc_auc_score(y_test, y_pred_dec_tree)))
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve for Decision Tree')
plt.legend(loc='lower right')
plt.show()
```



```
# Decision Tree Confusion Matrix
cm = confusion_matrix(y_test, y_pred_dec_tree)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['No
Churn', 'Churn'])
fig, ax = plt.subplots(figsize=(8, 6))
disp.plot(ax=ax, cmap='Blues', values_format='d')
plt.title('Confusion Matrix for Decision Tree')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```



Summary of Decision Tree Prediction

-Accuracy: 88.90% — The model correctly predicts 88.90% of all customer cases.

-Recall: 75.26%— It successfully identifies 75.26% of actual churners (73 out of 97), which is crucial for targeting at-risk customers.

-Precision: 59.35% — Of all customers predicted to churn, 59.35% actually do, reducing wasted retention efforts compared to Logistic Regression.

-F1-Score: 66.36% — Shows a good balance between precision and recall.

-ROC-AUC: 83.24%— Indicates strong ability to distinguish between churners and non-churners.

Confusion Matrix:

True Positives (TP): 73 — Correctly predicted churners.

True Negatives (TN): 520 — Correctly predicted non-churners.

False Positives (FP): 50 — Non-churners incorrectly flagged as churners.

False Negatives (FN): 24 — Churners missed by the model.

Business Implications:

- The Decision Tree model captures most churners, enabling proactive retention.
- It reduces false positives compared to Logistic Regression, optimizing resource allocation. Some churners are still missed, but overall, the model is efficient and reliable for deployment.

Comparison with Logistic Regression

Precision:

Decision Tree: 59.35% **Logistic Regression:** 34.46%. The Decision Tree is much better at ensuring that customers flagged as churners are actually at risk, reducing wasted retention efforts

Recall:

Decision Tree: 75.26% **Logistic Regression:** 73.20%. The Decision Tree is still better than logistic Regression while measuring the proportion of actual churners correctly identified by the model.

Accuracy:

Decision Tree: 88.90% **Logistic Regression:** 75.86%. The Decision Tree makes more correct predictions overall, especially for non-churners.

False Positives:

Decision Tree: 50 **Logistic Regression:** 135. The Decision Tree flags far fewer loyal customers as churners, saving resources.

F1-Score:

Decision Tree: 66.36 % **Logistic Regression:** 47.02%. The Decision Tree achieves a better balance between precision and recall.

ROC-AUC:

Decision Tree: 83.24% **Logistic Regression:** 74.76% The Decision Tree is better at distinguishing churners from non-churners

Recommendations for SyriaTel

Deploy the Decision Tree Model: Use the Decision Tree classifier in your CRM system to identify customers at high risk of churning. This model offers higher precision and accuracy, reducing wasted retention efforts.

Target At-Risk Customers: Focus retention campaigns such as special offers, discounts, or improved service on the customers flagged as likely to churn. This will help reduce revenue loss and improve customer loyalty.

Monitor and Adjust: Regularly monitor the model's performance and update it with new data to maintain accuracy. Analyze the cases where churners are missed (false negatives) and adjust the model or intervention strategies as needed.

Address Key Churn Drivers: Pay special attention to customers with frequent customer service calls, as this is a strong churn indicator. Consider reviewing and improving international plan features and voice mail plans, as these are also important predictors.

Optimize Resource Allocation: By reducing false positives, the Decision Tree model helps SyriaTel allocate retention resources more efficiently, focusing on customers who are truly at risk.

Continuous Improvement: Investigate the reasons behind customer churn and use insights from the model to inform business decisions, product improvements, and customer service enhancements.

Conclusion:

The Decision Tree outperforms Logistic Regression in almost every metric, especially in precision and overall accuracy. It is the preferred model for deployment, as it reduces unnecessary retention actions and provides more reliable predictions for SyriaTel.