Roticha / Phase_3_Project

Code Issues Pull requests Actions Projects Security Insights Settings

0 stars 0 forks 1 watching 1 Branch 0 Tags Activity

Private repository

1 Branch 0 Tags Go to file t Go to file Add file Code

Roticha Powerpoint presentations completed 8c8ed32 · 2 minutes ago

.ipynb_checkpoints	Powerpoint presentations completed	2 minutes ago
.gitignore	Done with the introduction	2 days ago
Presentation.pdf	Powerpoint presentations completed	2 minutes ago
README.md	Done with README.md	2 hours ago
bigml_59c28831336c6604c8...	starting off phase 3 project	2 days ago
index.ipynb	Powerpoint presentations completed	2 minutes ago

PREDICTING CUSTOMER CHURN FOR SYRIATEL

Name: Amos Kipngetich Rotich

Student Pace: Part Time

Scheduled Project Review Date/Time: September 1, 2024

Instructor Name: Winnie Anyoso

1. Introduction

Background:

Customer churn/turnover is a significant concern for telecommunications companies like SyriaTel. Churn occurs when customers discontinue their service, leading to revenue loss and increased costs for acquiring new customers. Understanding and predicting churn enables companies like SyriaTel to take proactive measures to retain customers at risk of leaving, thereby improving customer satisfaction and reducing financial losses.

Objective:

The primary objective of this project is to develop a predictive model that identifies SyriaTel customers likely to end their relationship with the company. By accurately predicting the turnover, SyriaTel can focus its retention efforts on at-risk customers, ultimately reducing churn rates and enhancing the company's profitability.

Dataset Overview:

The dataset used in this project contains records of 3,333 SyriaTel customers. The dataset includes various attributes related to customer demographics, account details, service usage, and customer service interactions. The target variable is churn, a binary indicator of whether a customer has stopped using SyriaTel's services.

Business Problem:

For SyriaTel, retaining customers is crucial to maintaining a stable revenue stream. By analyzing data on customer behavior and service usage, the company can identify patterns that suggest a higher likelihood of turnover. The predictive model developed in this project will assist SyriaTel in implementing targeted retention strategies, such as personalized offers or improved customer support, to reduce churn.

2. Data Processing

We start by handling any necessary preprocessing steps, including converting categorical variables, handling missing values (if any), and scaling numerical features.

Let's start with the preprocessing:

A. Data Understanding

1. Import Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split
```

2. Import Dataset

```
data = 'bigml_59c28831336c6604c800002a.csv'
df = pd.read_csv(data)
```

3. Data Overview

```
df.shape
```

Comments

The dataset consists of 3,333 customer records, each containing 21 attributes.

```
df.head()
```

Comments

The dataset includes features such as:

State: The U.S. state in which the customer resides. Account length: The duration of the customer's account. Area code: The area code of the customer's phone number. Phone number: The customer's phone number. International plan: Whether the customer has an international calling plan. Voice mail plan: Whether the customer has a voice mail plan. Churn: A binary column indicating if the customer has churned (True) or not (False).

B. Data Preprocessing

1. Converting Categorical Variables:

We'll convert categorical variables such as state, international plan, and voice mail plan into numerical values using one-hot encoding.

2. Scaling Numerical Features:

For models sensitive to feature scales, such as logistic regression, we will standardize the numerical features.

```
# Separate features and target variable
X = df.drop(columns=['churn', 'phone number'])
y = df['churn']

# Define the preprocessing pipeline
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), ['account length', 'number vmail messages',
                                   'total day minutes', 'total day calls', 'total day
charge',
```

```
                                    'total eve minutes', 'total eve calls', 'total eve
charge',
                                    'total night minutes', 'total night calls', 'total night
charge',
                                    'total intl minutes', 'total intl calls', 'total intl
charge',
                                    'customer service calls']),
        ('cat', OneHotEncoder(), ['state', 'area code', 'international plan', 'voice mail
plan'])
    ])

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Apply preprocessing
X_train = preprocessor.fit_transform(X_train)
X_test = preprocessor.transform(X_test)
```

C. Exploratory Data Analysis

We will visualize the dataset to understand the relationships between features and the target variable (churn). This step helps in feature selection and in gaining insights that might influence business recommendations.

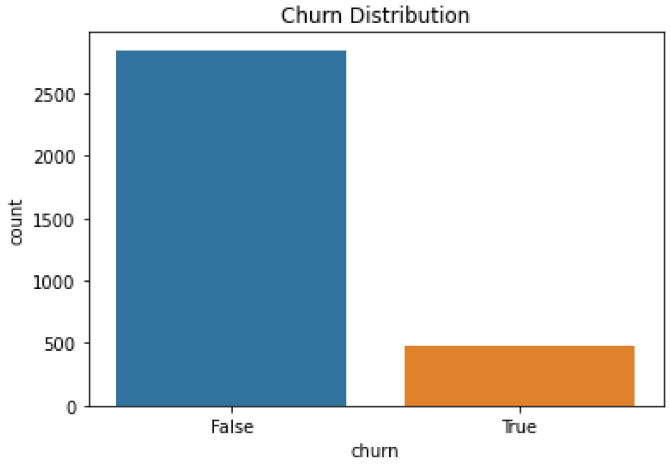
Key Analyses:

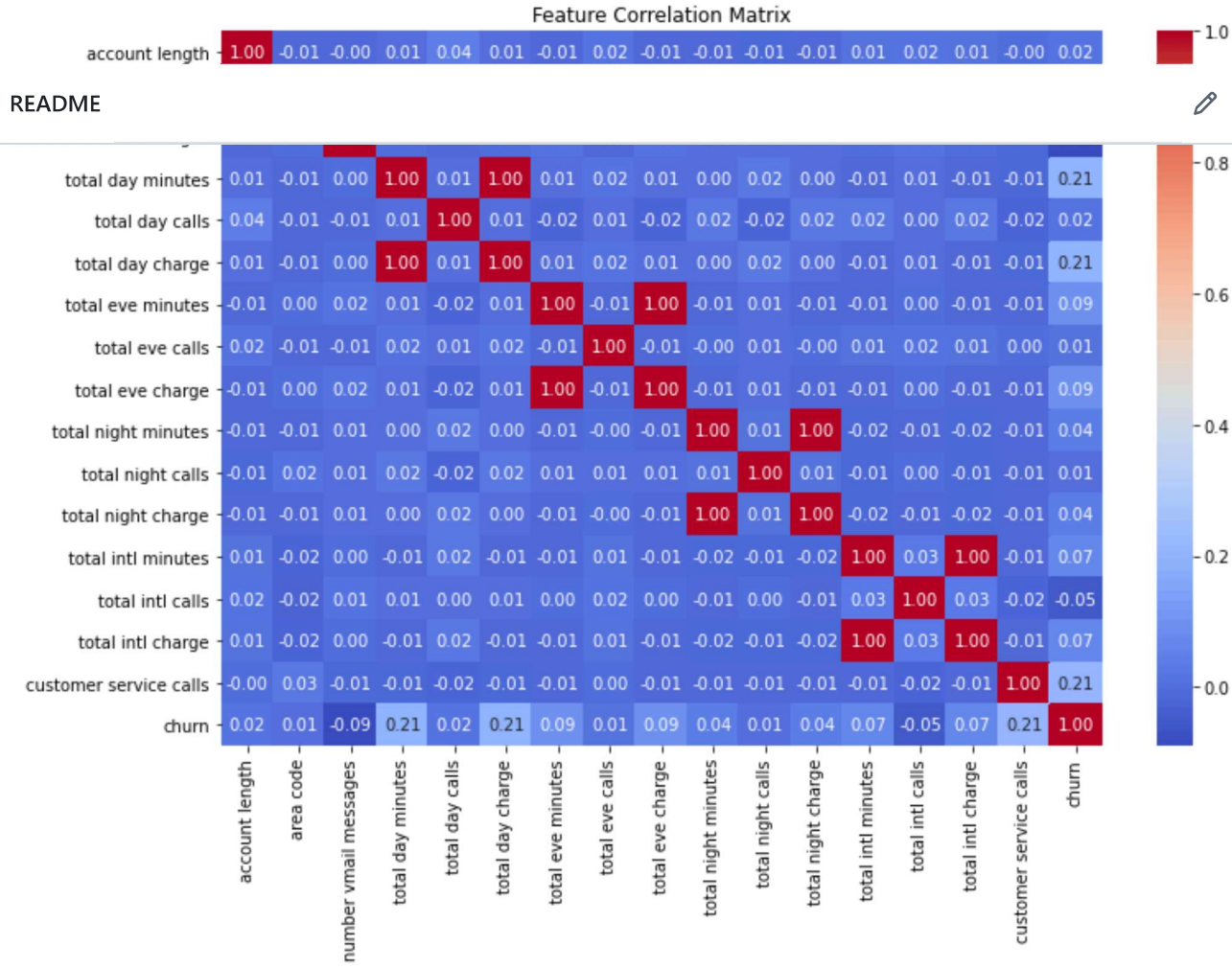
i. Distribution of numerical features like total day minutes, customer service calls, etc.

ii. Churn rates across different states and plans.

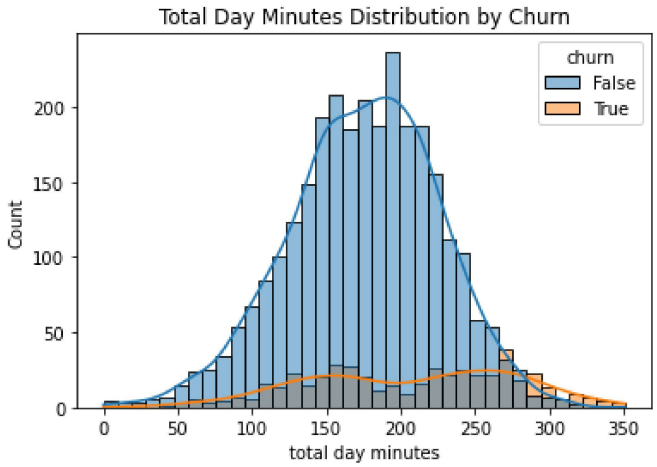
iii. Correlation matrix to identify highly correlated features.

```
# Visualize churn rate
sns.countplot(x='churn', data=df)
plt.title('Churn Distribution')
plt.show()

# Correlation matrix
plt.figure(figsize=(12, 8))
sns.heatmap(df.corr(), annot=True, fmt=".2f", cmap='coolwarm')
plt.title('Feature Correlation Matrix')
plt.show()

# Distribution of a key feature
sns.histplot(data=df, x='total day minutes', kde=True, hue='churn')
plt.title('Total Day Minutes Distribution by Churn')
plt.show()
```


Churn Distribution
count
2500
2000
1500
1000
500
0
False
True
churn


Feature Correlation Matrix
account length
total day minutes
total day calls
total day charge
total eve minutes
total eve calls
total eve charge
total night minutes
total night calls
total night charge
total intl minutes
total intl calls
total intl charge
customer service calls
churn
1.0
0.8
0.6
0.4
0.2
0.0

README



Interpreting the Plots

Churn Distribution Bar Plot

Majority of SyriaTel customers did not churn, as indicated by the tall blue bars(False) while only a small proportion of its customers churned, represented by the short orange bars(True).

This indicates an imbalanced dataset which could greatly impact model performance.

Correlation Heatmap

Most of the elements show weak correlations (close to 0). This indicates that most features are not strongly correlated with each other.

The weak correlation indicates that each feature contributes independently to predicting SyriaTel's customer turnover. This therefore favours Decision Tree Model which handles uncorrelated features very well.

Distribution Plot of 'Total Day Minutes' by Churn

The distribution of non-churned customers is fairly normal, dorminates the plot and peaks around the middle while churned customers' distribution is spread out with a lower peak.

The plot suggests that higher usage might be associated with an increased likelihood of churn.

3. Model Building and Selection

We will train Logistic Regression and Decision Tree models and compare their performance to identify the best model for this task.

```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor
from sklearn.metrics import classification_report, roc_auc_score
from sklearn.model_selection import cross_val_score
```

```
# Initialize models
logistic_model = LogisticRegression(max_iter=1000, random_state=42)
```

```
decision_tree_model = DecisionTreeClassifier(random_state=42)

# Train the models
logistic_model.fit(X_train, y_train)
decision_tree_model.fit(X_train, y_train)

# Predict using Logistic Regression
logistic_pred = logistic_model.predict(X_test)
logistic_prob = logistic_model.predict_proba(X_test)[:, 1]

# Predict using Decision Tree
decision_tree_pred = decision_tree_model.predict(X_test)
decision_tree_prob = decision_tree_model.predict_proba(X_test)[:, 1]

# Evaluate Logistic Regression
logistic_report = classification_report(y_test, logistic_pred)
logistic_auc = roc_auc_score(y_test, logistic_prob)

# Evaluate Decision Tree
decision_tree_report = classification_report(y_test, decision_tree_pred)
decision_tree_auc = roc_auc_score(y_test, decision_tree_prob)

print("Logistic Regression Report:")
print(logistic_report)
print(f"Logistic Regression AUC-ROC Score: {logistic_auc}\n")

print("Decision Tree Report:")
print(decision_tree_report)
print(f"Decision Tree AUC-ROC Score: {decision_tree_auc}\n")
```

Comments

The Logistic Regression Model seems to be struggling with the minority class which is the churned customers.

Precision - 0.55

Recall - 0.18

f1score - 0.27

The recall of 0.18 indicates that our model misses a significant number of actual churn cases. This is because the model is biased towards the majority class.

The Decision Tree Model performs better than the Logistic Regression Model.

Precision - 0.77

Recall - 0.74

f1score - 0.76

The recall of 0.74 indicates that the model can identifying churned customers, which suggests that it handles the class imbalance better than Logistic Regression.

We therefore pick to build on the Decision Tree Model

Decision Tree Model Tuning

Fine-tune the Decision Tree Model using cross-validation and hyperparameter tuning.

Cross Validation

```
# Instantiate the model
dt = DecisionTreeRegressor(random_state=42)

# Fit the model to the training data here
dt.fit(X_train, y_train)

# Testing out the model's r2 score on the training data overall
dt_train_score = dt.score(X_train, y_train)
dt_train_score
```

```
# Assign the cross validated score to dt_cv

dt_cv = cross_val_score(dt, X_train, y_train, cv=5)

dt_cv
```

Hyperparameter Tuning

Random_state = 42 Max_depth = 10

```
# Creating a second decision tree model
dt_tuned = DecisionTreeRegressor(random_state=42, max_depth=10)

# Fit the new model on the training data
dt_tuned.fit(X_train, y_train)
```

```
# Testing out the model's r2 score on the training data overall
dt_tuned_train_score = dt_tuned.score(X_train, y_train)
dt_tuned_train_score
```

```
# Checking the Cross-validated coefficient of determination (r2 score) of the predictions of
`dt_tuned` on the training data
dt_tuned_cv = cross_val_score(dt_tuned, X_train, y_train, cv=5)

dt_tuned_cv
```

Comments

The r2 scores obtained varies from 0.31 to 0.56. While this is an improvement from our pre-tuned model (range from 0.1 to 0.46), the variation across folds suggests that the model might be overfitting to certain subsets of the data or that there are still opportunities to improve its generalizability.

We will address this issue using SMOTE (Synthetic Minority Over-sampling Technique) which is a popular method used to handle class imbalance by generating synthetic examples for the minority class

```
from imblearn.over_sampling import SMOTE

# Define the SMOTE object
smote = SMOTE(random_state=42)

# Apply SMOTE to the training data
X_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)
```

```
# Define model
model = DecisionTreeClassifier(random_state=42)

# Fit the pipeline on the SMOTE-applied training data
model.fit(X_train_smote, y_train_smote)
```

Evaluate the model

```
# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model
print(classification_report(y_test, y_pred))
```

Comments

By handling the imbalance using SMOTE, our model's r2 scores are much better.

4. Analysis and Visualization

- Analysis of feature importance

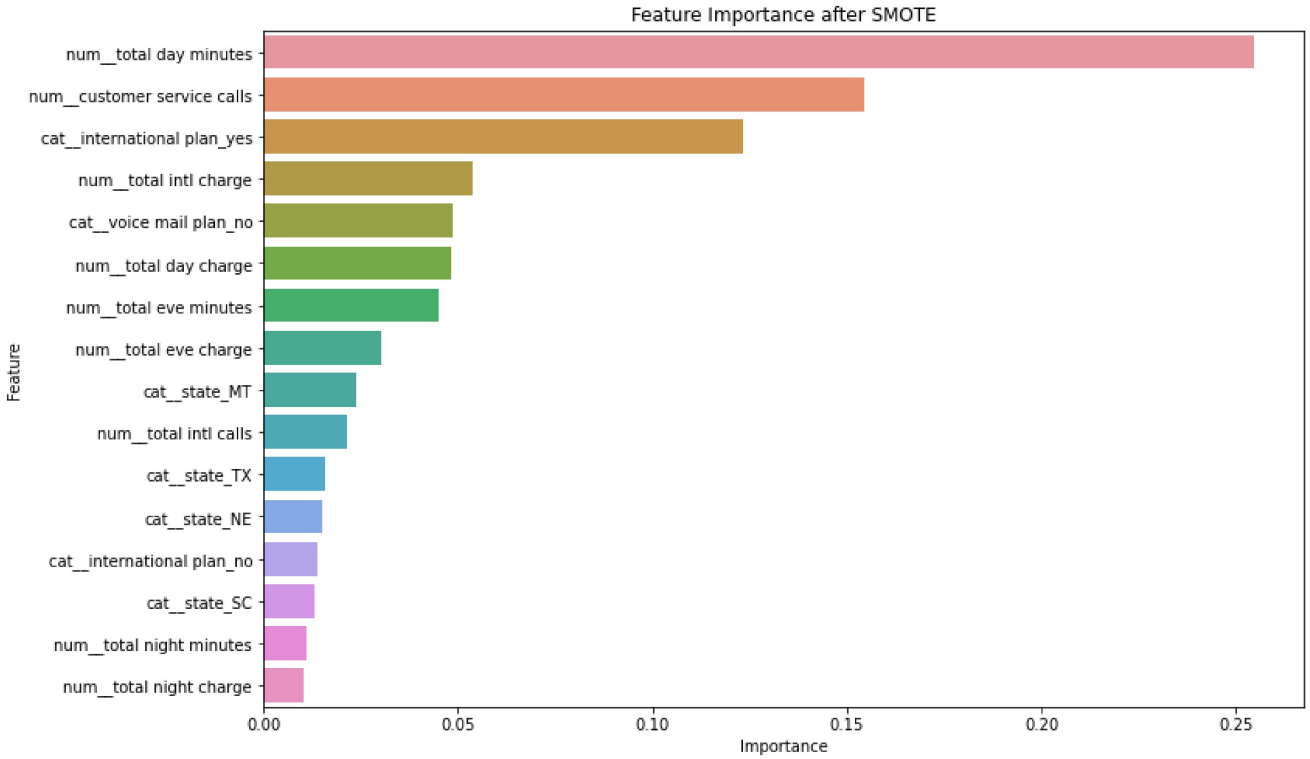
Analyze feature importance to understand which features contribute most to the model's decisions.

```
# Get feature importance
feature_importance = model.feature_importances_

# Get the feature names from the preprocessor
feature_names = preprocessor.get_feature_names_out()
```

```
# Create a DataFrame for feature importance
importance_df = pd.DataFrame({
    'Feature': feature_names,
    'Importance': feature_importance
})
```

```
# Sort the DataFrame by importance
importance_df = importance_df.sort_values(by='Importance', ascending=False)

# Display the feature importance
print(importance_df)
```

```
# Filter out features with zero importance
importance_df_filtered = importance_df[importance_df['Importance'] > 0.01]

# Visualize the feature importance
plt.figure(figsize=(12, 8))
sns.barplot(x='Importance', y='Feature', data=importance_df_filtered)
plt.title('Feature Importance after SMOTE')
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.show()
```



Comments

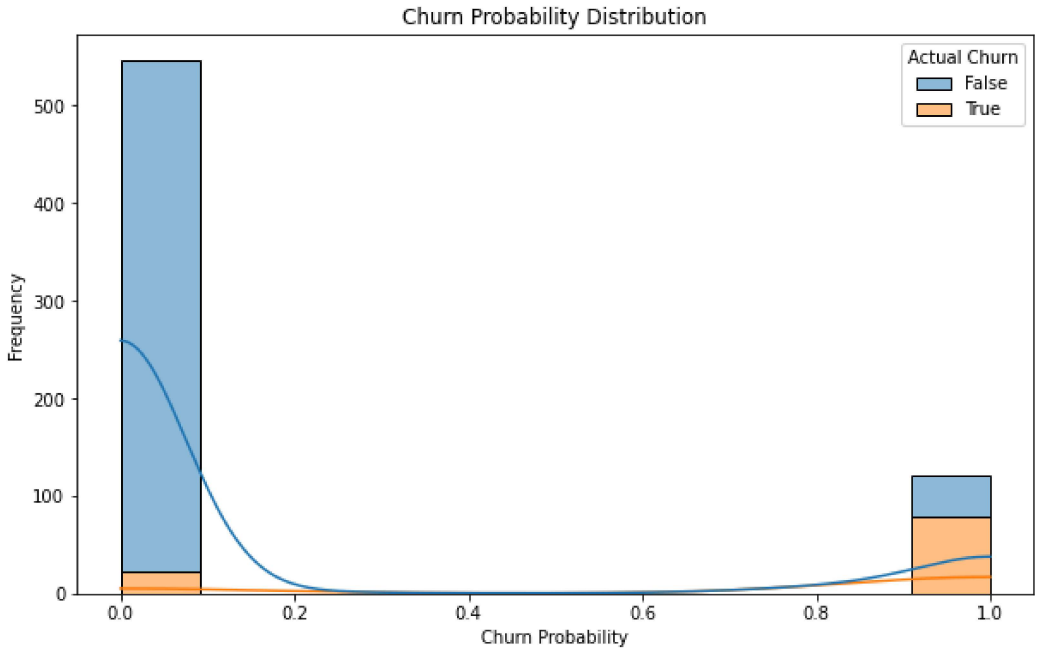
Here are interpretations for each of the top 5-ranked most important features:

- Total Day Minutes: The amount of time customers spend on calls during the day is a significant predictor of churn, suggesting that higher usage is associated with lower likelihood of leaving.
- Customer Service Calls: The frequency of customer service calls is an important factor; customers who make more calls may indicate dissatisfaction or issues that could lead to higher churn rates.
- International Plan: Having an international plan positively influences customer retention, indicating that customers with this plan are less likely to churn, possibly due to the value they perceive from international calling options.
- Total International Charge: The total charges for international calls serve as a predictor of churn; customers with higher international charges may be more likely to churn, possibly due to cost concerns or dissatisfaction with service.
- Voice Mail Plan: Customers without a voice mail plan tend to show higher churn rates, suggesting that having this feature may enhance customer satisfaction and retention.

- Customer Churn Prediction

We will use our model to predict customers who are likely to churn.

```
# Predict churn probabilities
y_proba = model.predict_proba(X_test)[:, 1]

# Predict churn classes
y_pred = model.predict(X_test)

# Create a DataFrame with actual and predicted values
results_df = pd.DataFrame({
    'Actual Churn': y_test,
    'Predicted Churn': y_pred,
    'Churn Probability': y_proba
})

# Ensure that 'Churn Probability' is numeric and 'Actual Churn' is categorical
results_df['Churn Probability'] = results_df['Churn Probability'].astype(float)
results_df['Actual Churn'] = results_df['Actual Churn'].astype(str)

# Visualize the probability distribution
plt.figure(figsize=(10, 6))
sns.histplot(data=results_df, x='Churn Probability', kde=True, hue='Actual Churn',
multiple='stack')
plt.title('Churn Probability Distribution')
plt.xlabel('Churn Probability')
plt.ylabel('Frequency')
plt.show()
```




```
# Drop the target variable and any identifier columns
X_predict = df.drop(columns=['churn', 'phone number'])

# Step 2: Preprocess the Data
X_predict_preprocessed = preprocessor.transform(X_predict)

# Step 3: Use the trained pipeline to make predictions
predictions = model.predict(X_predict_preprocessed)

# Add predictions to the original DataFrame
df['Churn Prediction'] = predictions

# Filter to get rows where Churn Prediction is True
true_churn_predictions = df[df['Churn Prediction'] == True]

# Display the DataFrame with only True churn predictions
print(true_churn_predictions[['phone number', 'Churn Prediction']])
```

Comments

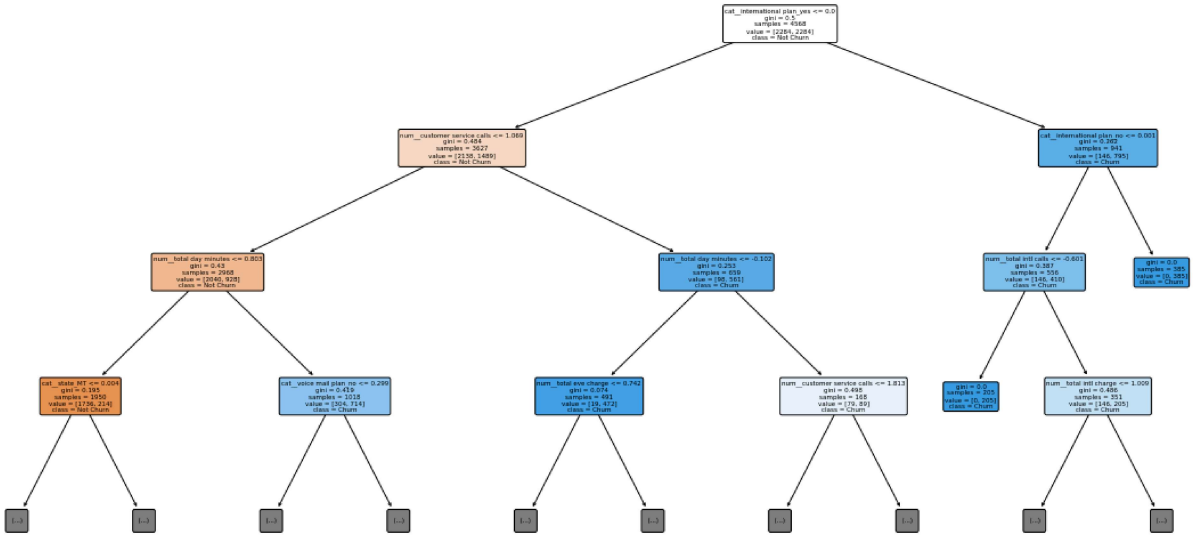
Our model has predicted 503 customers who are likely to churn. This enables the SyriaTel to make data-driven decisions, proactively address potential issues, optimize resource allocation, and ultimately improve customer retention and profitability.

- Decision Tree Visualization

This is a visual representation of how decisions were made in our model.

```
from sklearn.tree import plot_tree

# Visualize the Decision Tree
plt.figure(figsize=(20, 10))
plot_tree(model, feature_names = preprocessor.get_feature_names_out(), class_names=['Not
```

```
Churn', 'Churn'], filled=True, rounded=True, max_depth=3)
plt.show()
```



5. Business Implications and Recommendations

Our model has given us actionable business insights that SyriaTel can use to address customer churn issue.

1. Customer Retention

We identified the most important features driving churn. Our top 5 of these features included:

- Total Day Minutes: The amount of time customers spend on calls during the day is a significant predictor of churn, suggesting that higher usage is associated with lower likelihood of leaving.
- Customer Service Calls: The frequency of customer service calls is an important factor; customers who make more calls may indicate dissatisfaction or issues that could lead to higher churn rates.
- International Plan: Having an international plan positively influences customer retention, indicating that customers with this plan are less likely to churn, possibly due to the value they perceive from international calling options.
- Total International Charge: The total charges for international calls serve as a predictor of churn; customers with higher international charges may be more likely to churn, possibly due to cost concerns or dissatisfaction with service.

Releases

No releases published

Create a new release

Packages

No packages published
Publish your first package

Languages

- **Jupyter Notebook** 100.0%