SyriaTel_Customer_Churn_Prediction / index.ipynb

mbuvimuia Done Conclusion and Recommendation bdd4066 · 41 minutes ago History

Predicting Customer Churn for SyriaTel

Business Understanding

Overview

SyriaTel is a telecommunications company providing mobile services to customers across Syria. In the highly competitive telecom industry, customer retention is crucial for maintaining revenue and growth. Churn, refers to customers leaving the service thus poses a significant threat to the company's profitability and market share. Predicting customer churn allows SyriaTel to proactively address factors leading to customer dissatisfaction and implement strategies to retain valuable customers.

Challenges

1. High Competition: The telecom industry is saturated with multiple service providers, offering similar services at competitive prices making it easy for customers to switch.
2. Customer Behavior Analysis: Understanding the diverse needs and behaviors of customers to identify those at risk of churning.
3. Data Quality: Ensuring the data used for analysis is accurate, complete, and up-to-date to build reliable predictive models.
4. Resource Allocation: Allocating resources effectively to retain customers at risk of churning while minimizing costs.

Problem Statement

SyriaTel wants to predict customer churn based on historical data to identify customers at risk of leaving the service. By accurately predicting churn, SyriaTel can implement targeted retention strategies to reduce customer attrition and improve overall customer satisfaction.

Objectives

1. To gather and pre-process customer data to ensure quality and completeness.
2. To identify and create relevant features that can contribute to customer churn.

3. To build and train various machine learning model to predict churn.
4. To asses the performance of the models using appropriate metrics and select the best-performing model.

Proposed Solution

The proposed solution involves building a predictive model to identify customers at risk of churning based on historical data. By accurately predicting churn, SyriaTel can:

- Target at-risk customers with personalized retention strategies.
- Optimize marketing and promotional efforts.
- Improve overall customer satisfaction and loyalty.

Metrics of the Model

To evaluate the performance of the churn prediction model, the following metrics will be used:

1. **Accuracy**: The proportion of correctly predicted instances(both churn and non-churn) out of the total instances.
2. **Precision**: The proportion of correctly predicted churn instances out of all instances predicted as churn.
3. **Recall**: The proportion of correctly predicted churn instances out of all actual churn instances.
4. **F1 Score**: The harmonic mean of precision and recall, providing a balance between the two metrics.
5. **ROC-AUC**: The area under the receiver operating characteristic curve, which measures the model's ability to distinguish between churn and non-churn instances.
6. **Confusion Matrix**: A table showing the number of true positives, true negatives, false positives, and false negatives, providing insights into the model's performance.
7. **PR Curve**: Precision-Recall curve, which shows the trade-off between precision and recall at different thresholds.
8. **Learning Curve**: A plot showing the model's performance on training and validation data as a function of the training set size, helping identify overfitting or underfitting.

Conclusion

By leveraging machine learning to predict customer churn, SyriaTel can gain valuable insights into customer behavior and proactively address

churn risks. This approach not only helps in retaining customers but also enhances the overall customer experience and strengthens the company's competitive position in the telecom industry.

Source of Data

The dataset used for this analysis is obtained from the Kaggle website. The dataset contains information about customer demographics, usage patterns, and churn status for a telecom company. The dataset consists of 3333 observations and 21 features, including customer attributes such as account length, international plan, voicemail plan, total day minutes, total day calls, total day charge, etc.

Data Understanding

In [42]:
```
# Remove warning messages
import warnings
warnings.filterwarnings("ignore")
import pandas as pd

from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier

from sklearn.ensemble import VotingClassifier
```

In [2]:
```
# Import the necessary classes from data_processing.py
from data_processing import DataProcessor, DataAnalysis, ModelEvaluation

# Load the data
processor = DataProcessor('data/telecom_dataset.xls')
data = processor.load_data()
```

In [3]:
```
# Exploration of the data
# Initial exploration of the data
print("First 5 rows of the data:")
# Print the first 5 rows of the data in pandas dataframe
data.head()
```

```
First 5 rows of the data:
```

Out[3]:

	state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total day charge	...	total eve calls	total eve charge	total night minutes	total night calls	total night charge	mi
0	KS	128	415	382-4657	no	yes	25	265.1	110	45.07	...	99	16.78	244.7	91	11.01	

1	OH	107	415	371-7191	no	yes	26	161.6	123	27.47	...	103	16.62	254.4	103	11.45
2	NJ	137	415	358-1921	no	no	0	243.4	114	41.38	...	110	10.30	162.6	104	7.32
3	OH	84	408	375-9999	yes	no	0	299.4	71	50.90	...	88	5.26	196.9	89	8.86
4	OK	75	415	330-6626	yes	no	0	166.7	113	28.34	...	122	12.61	186.9	121	8.41

5 rows × 21 columns

The dataset contains the following columns:

- State: The state in which the customer resides.
- Account Length: The number of days the customer has been with the company.
- Area Code: The area code of the customer's phone number.
- Phone: The customer's phone number.
- International Plan: Whether the customer has an international plan (yes/no).
- Voice Mail Plan: Whether the customer has a voicemail plan (yes/no).
- Number Vmail Messages: The number of voicemail messages.
- Total Day Minutes: Total number of minutes the customer used during the day.
- Total Day Calls: Total number of calls the customer made during the day.
- Total Day Charge: Total charges for calls made during the day.
- Total Eve Minutes: Total number of minutes the customer used during the evening.
- Total Eve Calls: Total number of calls the customer made during the evening.
- Total Eve Charge: Total charges for calls made during the evening.
- Total Night Minutes: Total number of minutes the customer used during the night.
- Total Night Calls: Total number of calls the customer made during the night.
- Total Night Charge: Total charges for calls made during the night.
- Total Intl Minutes: Total number of international minutes used.

- Total Intl Calls: Total number of international calls made.
- Total Intl Charge: Total charges for international calls.
- Customer Service Calls: Number of customer service calls made.
- Churn: Whether the customer churned or not (yes/no).

In [4]:
```
# Summary of the data
print("Summary of the data:")
data.info()
```

```
Summary of the data:
<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
```

The dataset contains 3333 observations and 21 columns.

- Customer Demographics(state)
- Account details(Account Length, Area Code, Phone)
- Service Plans(International Plan, Voice Mail Plan)

- Voice Mail Usage(Number Vmail Messages)
- Call Minutes and Charges(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 interactions(Customer Service Calls)
- Customer Churn (boolean value indicating whether a customer churned)

In [5]:
```
# Check for missing values
print("Missing values:")
data.isnull().sum()
```

```
Missing values:
```

Out[5]:
```
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 the dataset.

In [6]:
```
# Check for duplicates
print("Duplicates:")
data.duplicated().sum()
```

```
Duplicates:
```

Out[6]:
```
0
```

There are no duplicate rows in the dataset.

Data Preparation

In [7]:
```
# Rename the columns(removing spaces)
data.columns = data.columns.str.replace(' ','_')
data.columns
```

Out[7]:
```
Index(['state', 'account_length', 'area_code', 'phone_number',
       'international_plan', 'voice_mail_plan', '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', 'churn'],
      dtype='object')
```

Data Analysis

In [8]:
```
# Descriptive Statistics
analysis = DataAnalysis(data)
print("Descriptive Statistics:")

analysis.summary_statistics()
```

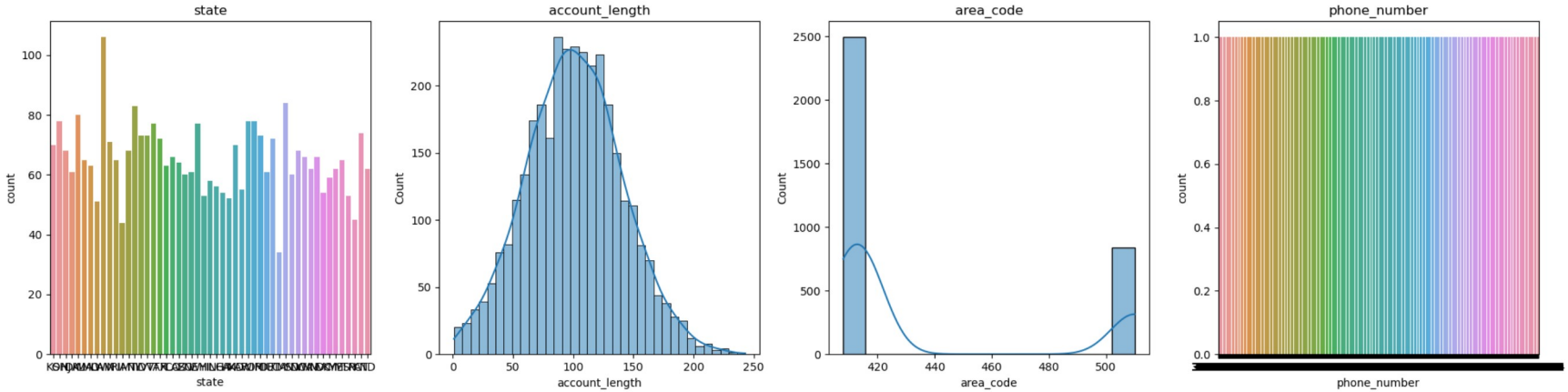
Descriptive Statistics:

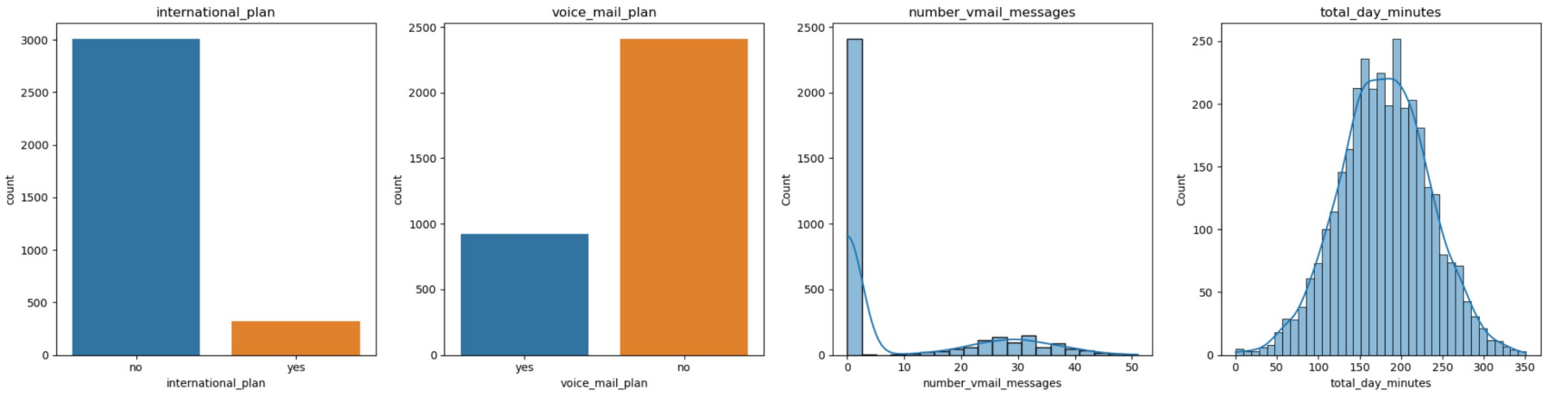
Out[8]:

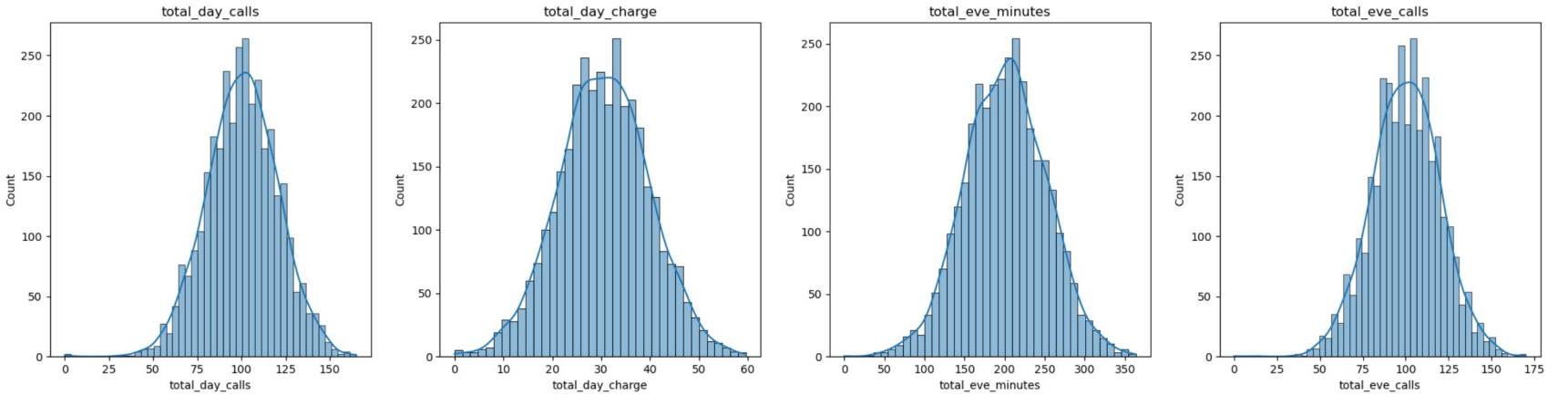
	account_length	area_code	number_vmail_messages	total_day_minutes	total_day_calls	total_day_charge	total_eve_minutes	total_
count	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	33
mean	101.064806	437.182418	8.099010	179.775098	100.435644	30.562307	200.980348	1(
std	39.822106	42.371290	13.688365	54.467389	20.069084	9.259435	50.713844	
min	1.000000	408.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	74.000000	408.000000	0.000000	143.700000	87.000000	24.430000	166.600000	
50%	101.000000	415.000000	0.000000	179.400000	101.000000	30.500000	201.400000	1(
75%	127.000000	510.000000	20.000000	216.400000	114.000000	36.790000	235.300000	1
max	243.000000	510.000000	51.000000	350.800000	165.000000	59.640000	363.700000	1

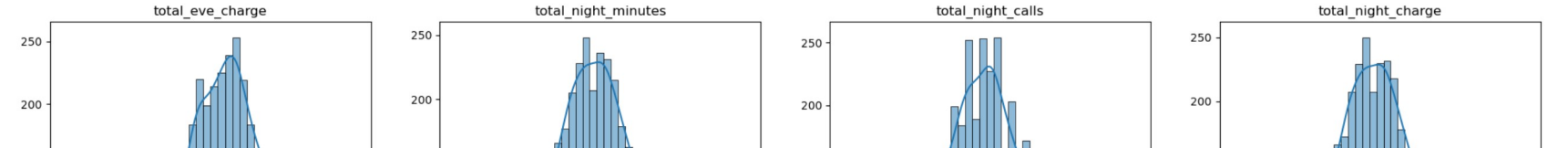
In [9]:

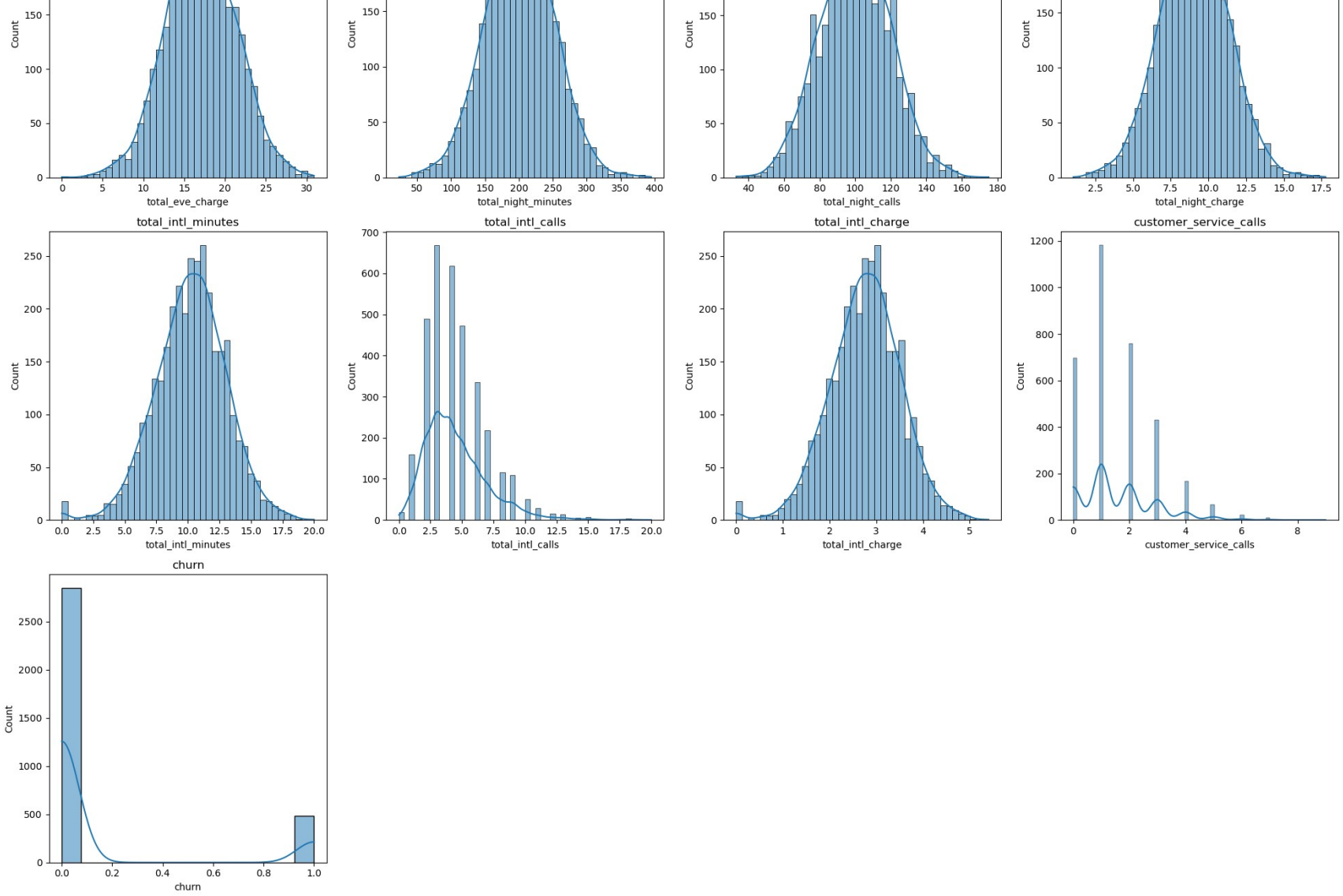
```
# Univariate Analysis
analysis.univariate_analysis(data.columns)
```









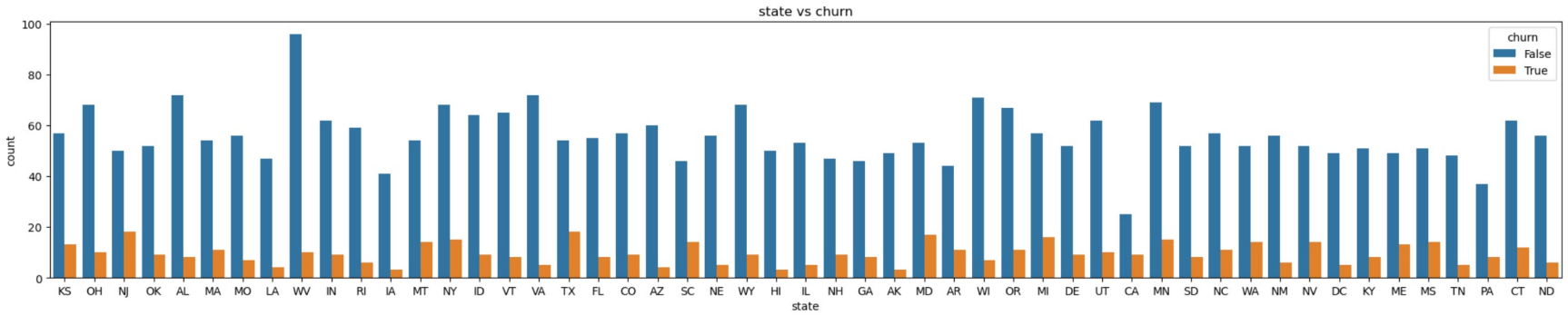


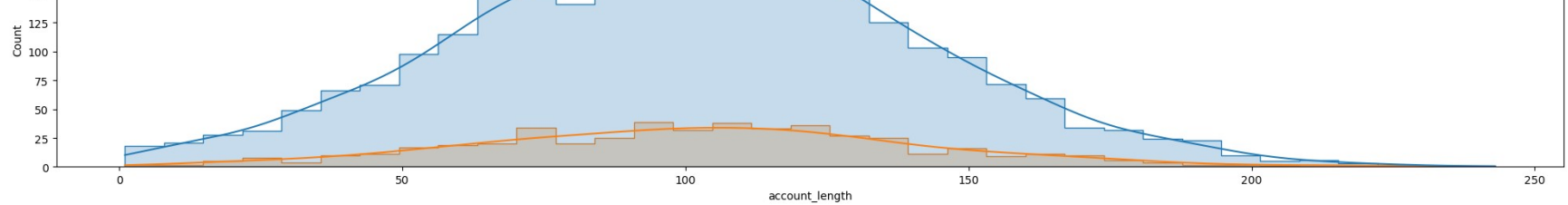
- **State** has a uniform distribution across many categories indicating data from multiple states. Each state appears to have a similar number of customers in the dataset.
- **Account Length** is normally distributed, with most values ranging between 50 to 150 days. This suggests that most customers have been with the company for a moderate period.
- **Area Code** shows three distinct categories, with '415' being the most common, followed by '510' and '408' being the least common.
- **Phone Number** is likely a unique identifier thus the column should be excluded from further analysis.
- **International Plan** is binary with majority of customers not subscribed to the international plan.

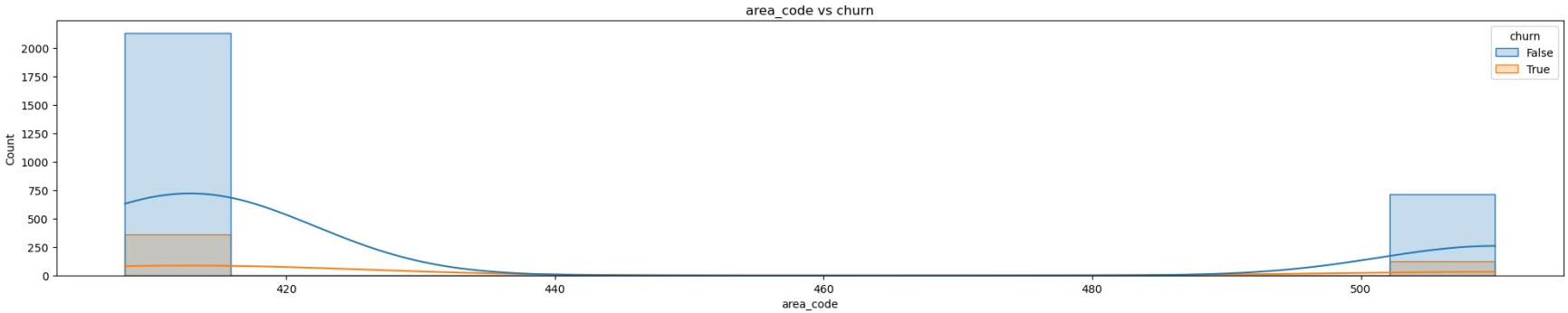
- **Voice Mail Plan** is binary with majority of customers not subscribed to the voicemail plan.
- **Number Vmail Messages** shows that most customers do not use voicemail, with a few customers having a high number of voicemail messages.
- **Total Day Minutes, Total Day Calls, Total Day Charge** are normally distributed, indicating consistent usage patterns during the day.
- **Total Eve Minutes, Total Eve Calls, Total Eve Charge** are normally distributed, indicating consistent usage patterns during the evening.
- **Total Night Minutes, Total Night Calls, Total Night Charge** are normally distributed, indicating consistent usage patterns during the night.
- **Total Intl Minutes, Total Intl Calls, Total Intl Charge** have some skewness, 'total_int_calls' has a right-skewed distribution, indicating that most customers make few international calls. 'total_intl_minutes' and 'total_intl_charge' are somewhat normally distributed but have a tail indicating higher usage by fewer customers.
- **Customer Service Calls** is right-skewed, with most customers making few calls to customer service, but a notable minority making multiple calls.
- **Churn** is binary and shows that the majority of customers have not churned. This indicates an imbalance in the target variable, which should be considered when building predictive models.

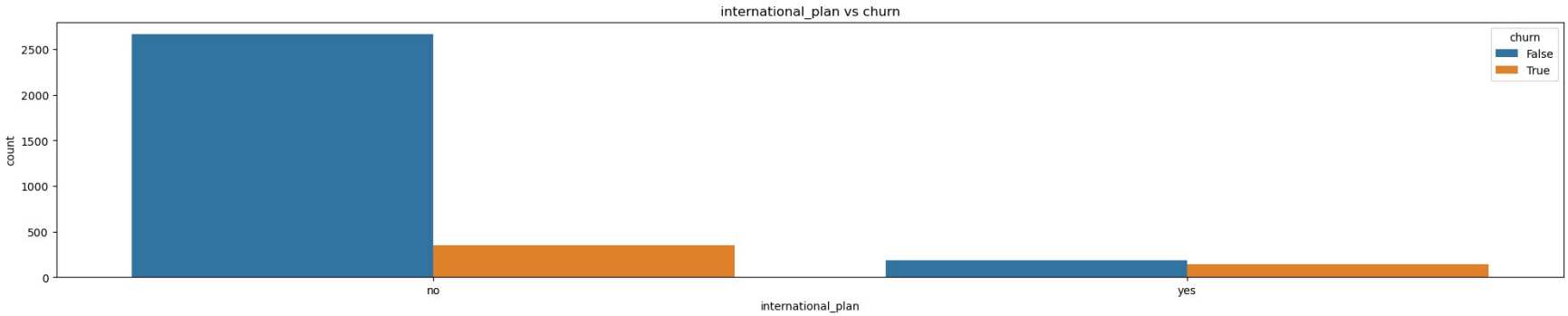
In [10]:
```
# Drop phone number column (not useful for analysis)
data.drop('phone_number', axis=1, inplace=True)
```

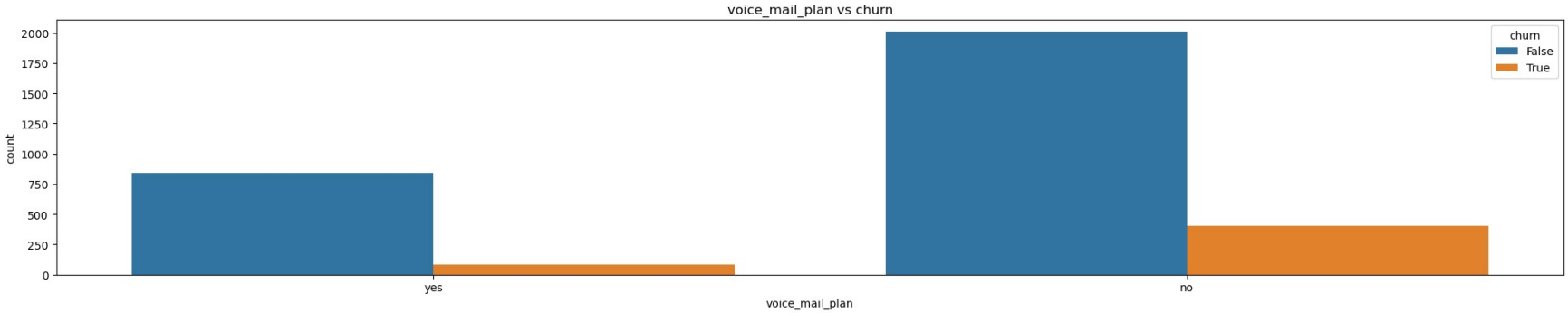
In [11]:
```
# Bivariate Analysis
analysis.bivariate_analysis(target='churn')
```

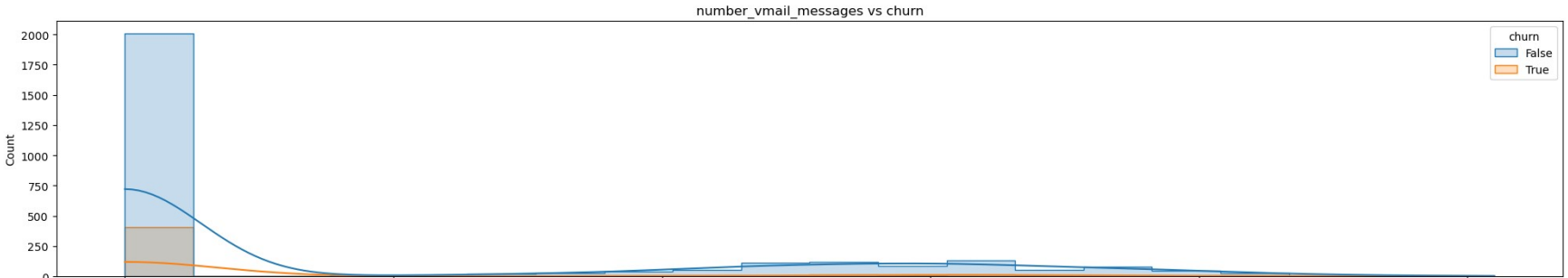


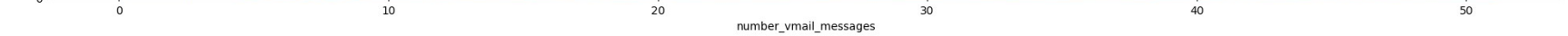

Count
account_length

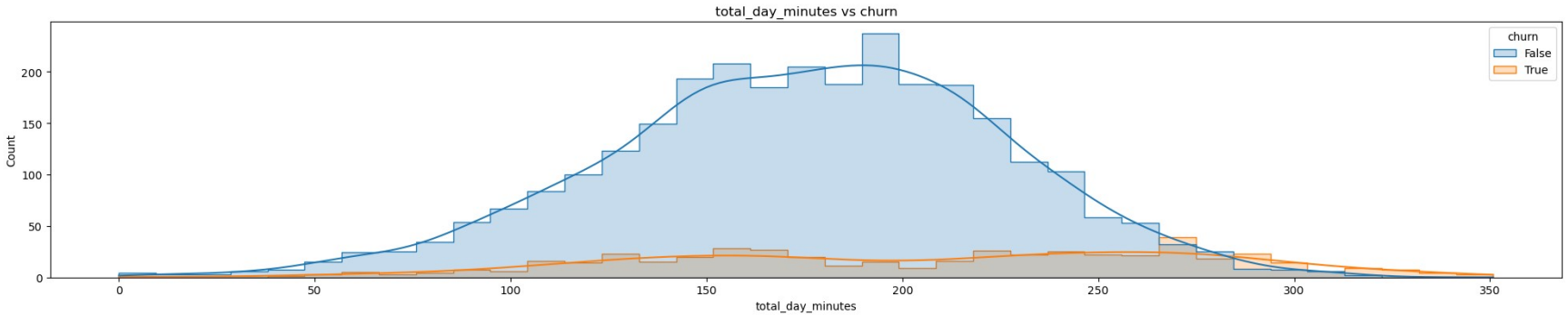

area_code vs churn
churn
False
True
Count
area_code

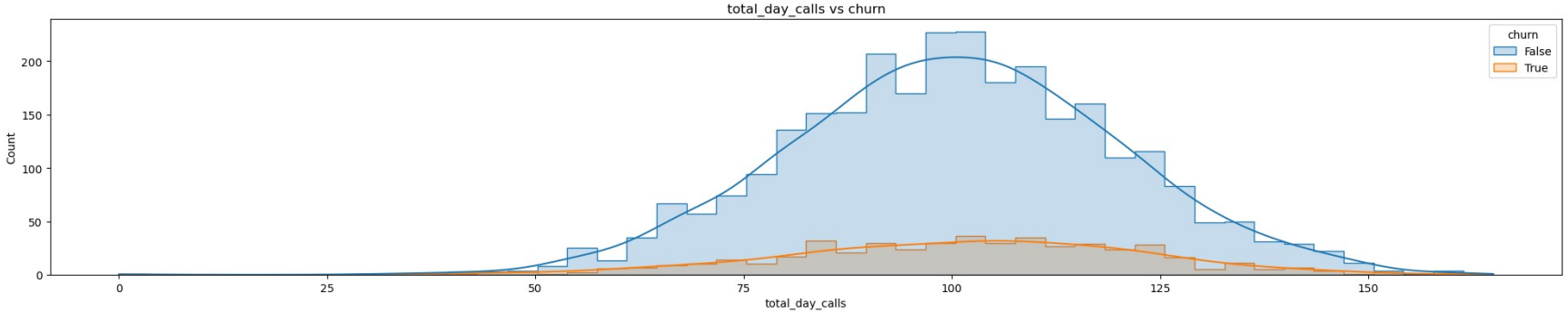

international_plan vs churn
churn
False
True
count
no
yes
international_plan

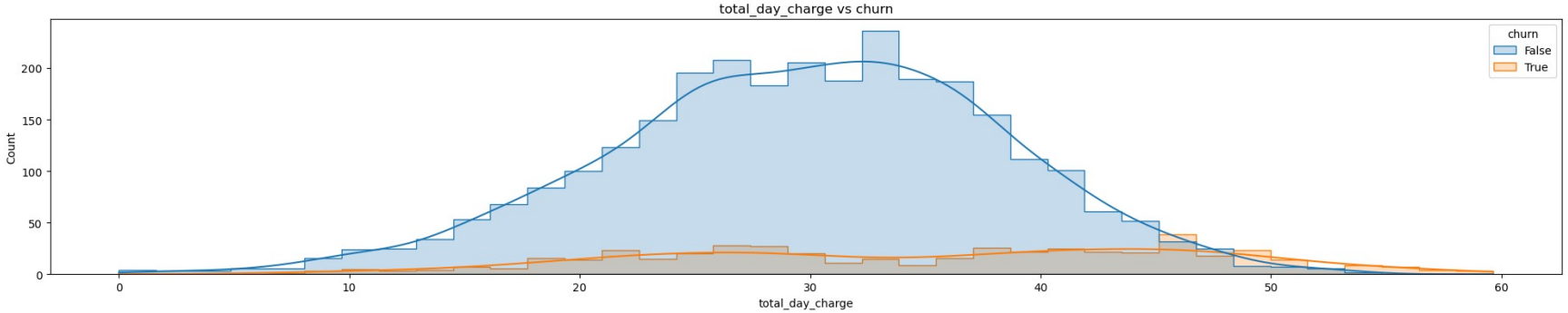

voice_mail_plan vs churn
churn
False
True
count
yes
no
voice_mail_plan

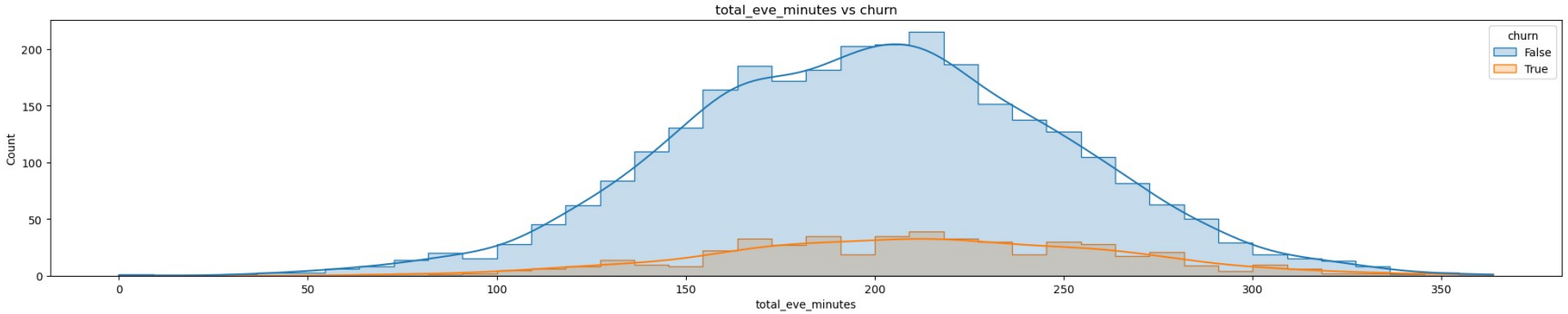

number_vmail_messages vs churn
churn
False
True
Count

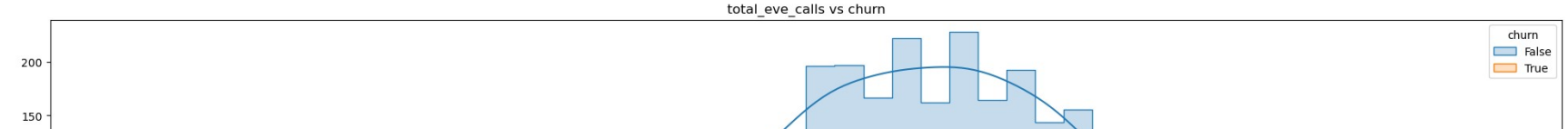

0
10
20
30
40
50
number_vmail_messages

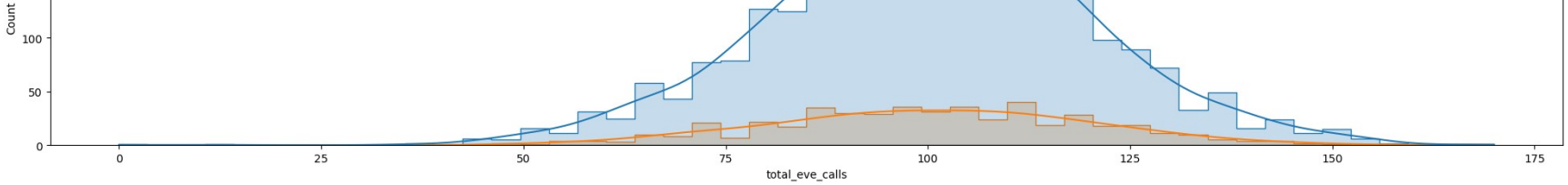

total_day_minutes vs churn
churn
False
True
Count
0
50
100
150
200
250
300
350
total_day_minutes

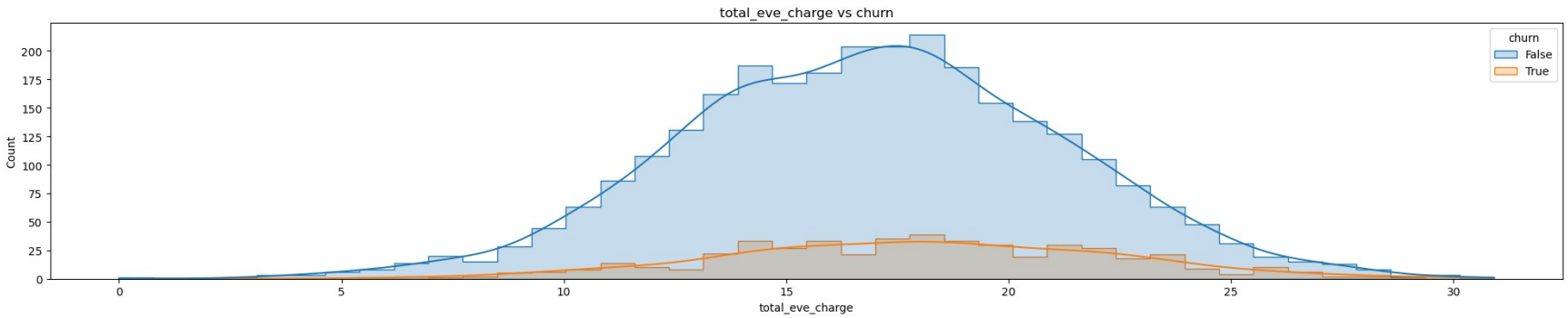

total_day_calls vs churn
churn
False
True
Count
0
25
50
75
100
125
150
total_day_calls

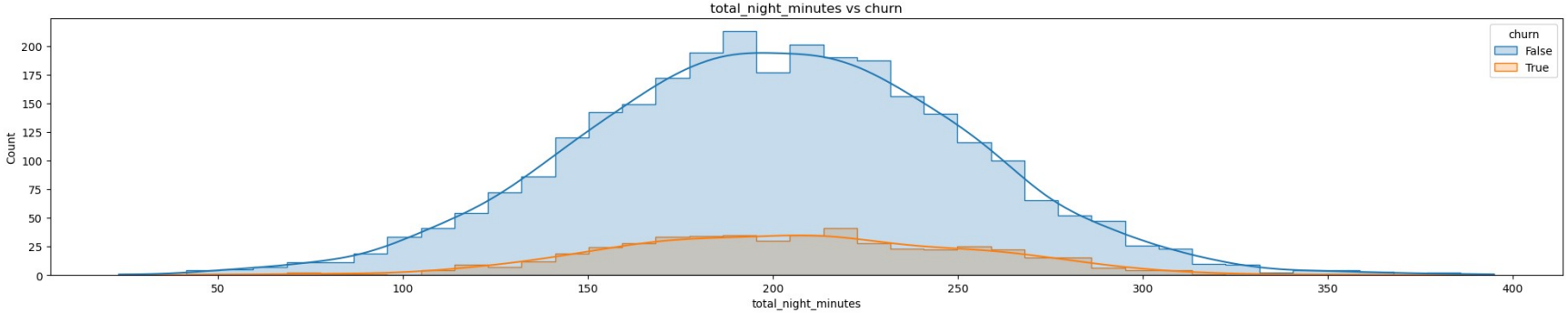

total_day_charge vs churn
churn
False
True
Count
0
10
20
30
40
50
60
total_day_charge

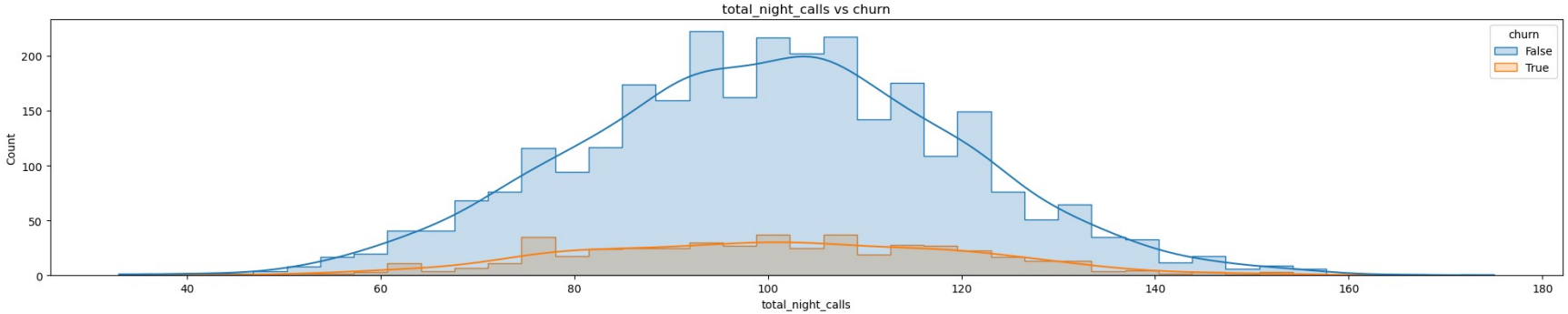

total_eve_minutes vs churn
churn
False
True
Count
0
50
100
150
200
250
300
350
total_eve_minutes

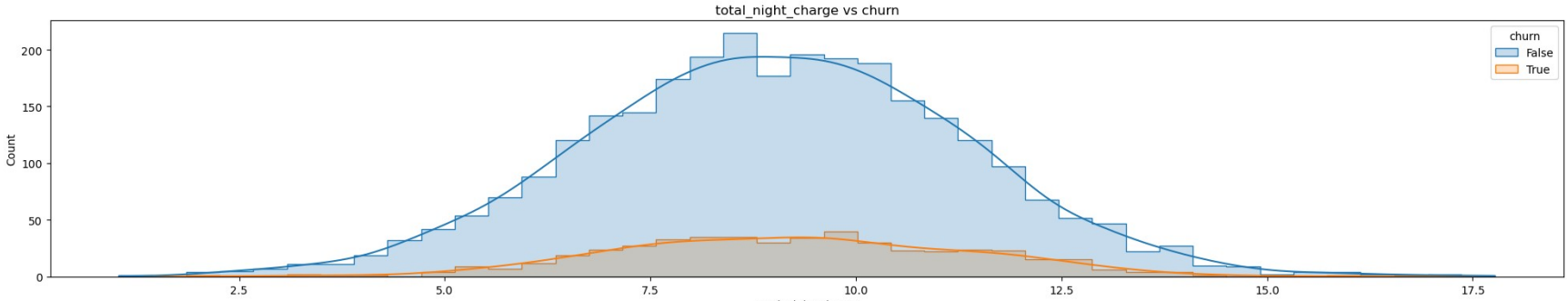

total_eve_calls vs churn
churn
False
True
Count
200
150

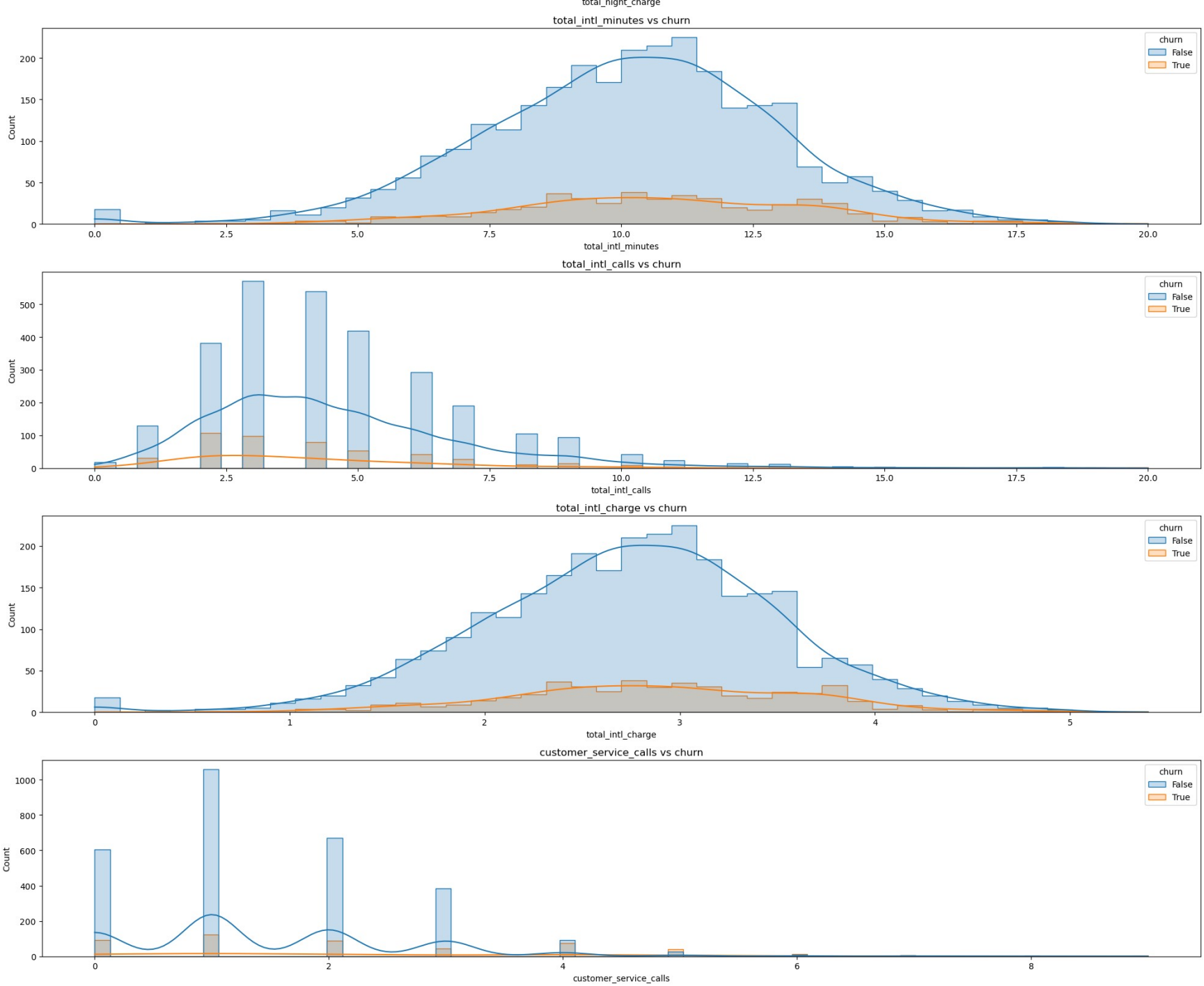

Count
total_eve_calls


total_eve_charge vs churn
churn
False
True
Count
total_eve_charge


total_night_minutes vs churn
churn
False
True
Count
total_night_minutes


total_night_calls vs churn
churn
False
True
Count
total_night_calls


total_night_charge vs churn
churn
False
True
Count



state vs churn the churn rate varies across different states, with some states having higher churn rates than others. This suggests that the state may be a relevant feature for predicting churn.

account_length vs churn there is no clear relationship between account length and churn, indicating that the length of time a customer has

been with the company may not be a strong predictor of churn.

area_code vs churn the churn rate is consistent across different area codes, indicating that the area code may not be a strong predictor of churn.

international_plan vs churn customers with no international plan have a higher churn rate than those with an international plan, suggesting that the international plan may be a relevant feature for predicting churn.

voice_mail_plan vs churn customers with a voice mail plan have a lower churn rate than those without a voice mail plan, suggesting that the voice mail plan may be a relevant feature for predicting churn.

number_vmail_messages vs churn customers with a higher number of voicemail messages have a lower churn rate, suggesting that voicemail usage may be a relevant feature for predicting churn.

total_day_minutes vs churn there is no clear relationship between total day minutes and churn, indicating that the total day minutes may not be a strong predictor of churn.

total_day_calls vs churn there is no clear relationship between total day calls and churn, indicating that the total day calls may not be a strong predictor of churn.

total_day_charge vs churn there is no clear relationship between total day charge and churn, indicating that the total day charge may not be a strong predictor of churn.

total_eve_minutes vs churn there is no clear relationship between total evening minutes and churn, indicating that the total evening minutes may not be a strong predictor of churn.

total_eve_calls vs churn there is no clear relationship between total evening calls and churn, indicating that the total evening calls may not be a strong predictor of churn.

total_eve_charge vs churn there is no clear relationship between total evening charge and churn, indicating that the total evening charge may not be a strong predictor of churn.

total_night_minutes vs churn there is no clear relationship between total night minutes and churn, indicating that the total night minutes may not be a strong predictor of churn.

total_night_calls vs churn there is no clear relationship between total night calls and churn, indicating that the total night calls may not be a strong predictor of churn.

total_night_charge vs churn there is no clear relationship between total night charge and churn, indicating that the total night charge may not be a strong predictor of churn.

total_intl_minutes vs churn there is no clear relationship between total international minutes and churn, indicating that the total

international minutes may not be a strong predictor of churn.

total_intl_calls vs churn there is no clear relationship between total international calls and churn, indicating that the total international calls may not be a strong predictor of churn.

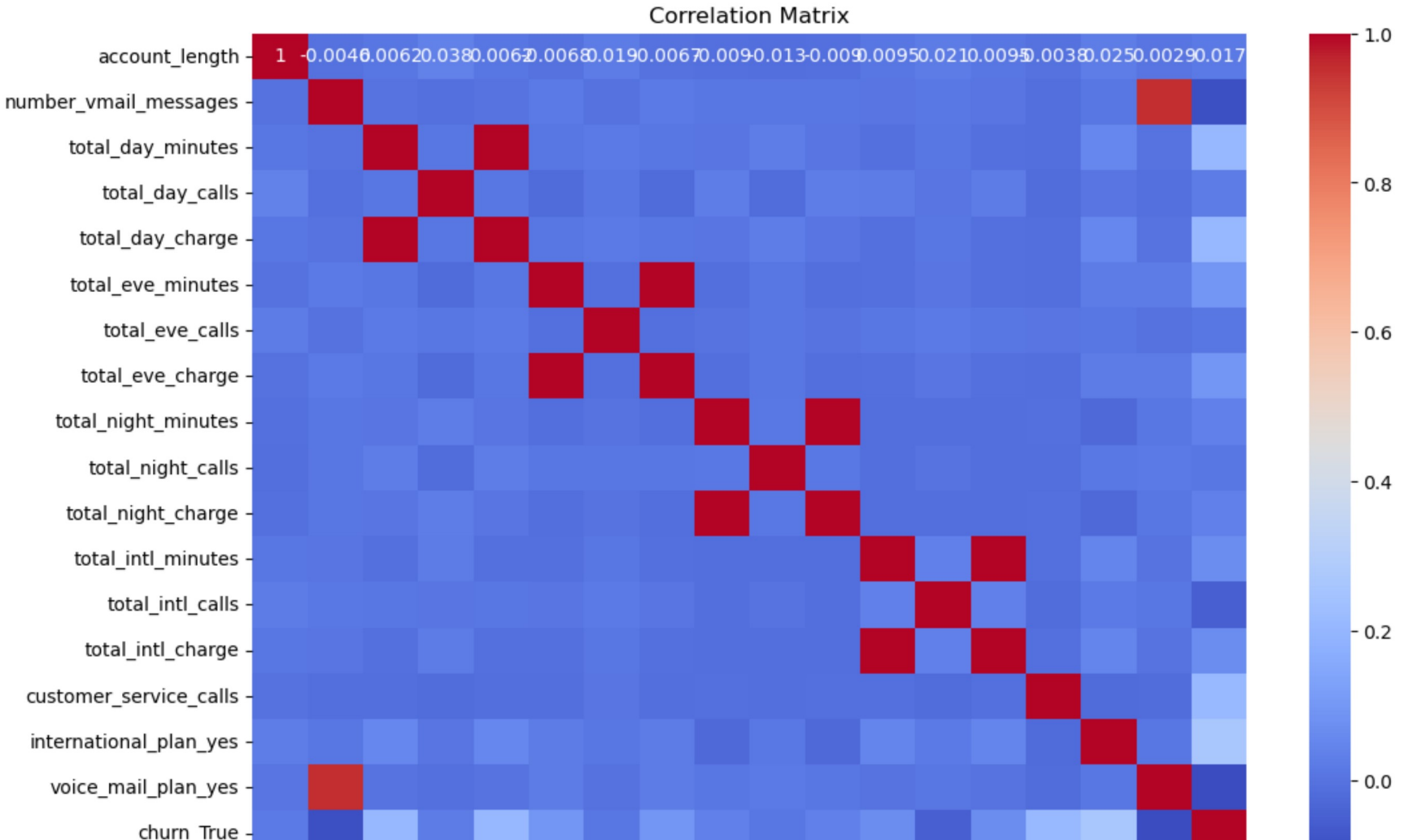
total_intl_charge vs churn there is no clear relationship between total international charge and churn, indicating that the total international charge may not be a strong predictor of churn.

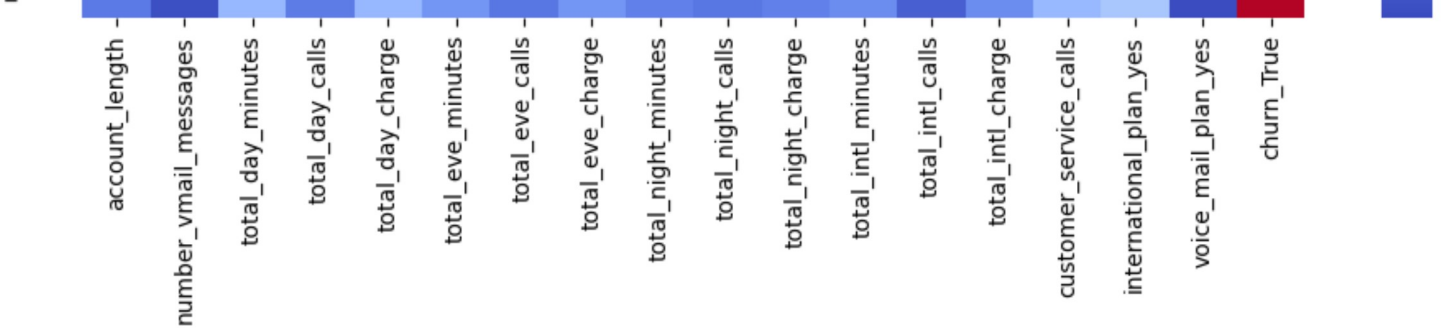
customer_service_calls vs churn customers with a lower number of customer service calls have a lower churn rate, suggesting that the number of customer service calls may be a relevant feature for predicting churn.

In [12]:

```
# Correlation Analysis

analysis.correlation_matrix()
```





Out[12]:

	account_length	number_vmail_messages	total_day_minutes	total_day_calls	total_day_charge	total_eve_minutes
account_length	1.000000	-0.004628	0.006216	0.038470	0.006214	-0.006757
number_vmail_messages	-0.004628	1.000000	0.000778	-0.009548	0.000776	0.017562
total_day_minutes	0.006216	0.000778	1.000000	0.006750	1.000000	0.007043
total_day_calls	0.038470	-0.009548	0.006750	1.000000	0.006753	-0.021451
total_day_charge	0.006214	0.000776	1.000000	0.006753	1.000000	0.007050
total_eve_minutes	-0.006757	0.017562	0.007043	-0.021451	0.007050	1.000000
total_eve_calls	0.019260	-0.005864	0.015769	0.006462	0.015769	-0.011430
total_eve_charge	-0.006745	0.017578	0.007029	-0.021449	0.007036	1.000000
total_night_minutes	-0.008955	0.007681	0.004323	0.022938	0.004324	-0.012584
total_night_calls	-0.013176	0.007123	0.022972	-0.019557	0.022972	0.007586
total_night_charge	-0.008960	0.007663	0.004300	0.022927	0.004301	-0.012593
total_intl_minutes	0.009514	0.002856	-0.010155	0.021565	-0.010157	-0.011035
total_intl_calls	0.020661	0.013957	0.008033	0.004574	0.008032	0.002541
total_intl_charge	0.009546	0.002884	-0.010092	0.021666	-0.010094	-0.011067
customer_service_calls	-0.003796	-0.013263	-0.013423	-0.018942	-0.013427	-0.012985
international_plan_yes	0.024735	0.008745	0.049396	0.003755	0.049398	0.019100
voice_mail_plan_yes	0.002918	0.956927	-0.001684	-0.011086	-0.001686	0.021545
churn_True	0.016541	-0.089728	0.205151	0.018459	0.205151	0.092796

Based on the correlation matrix, the following observations can be made in relation to the target variable 'churn':

- The features 'total day minutes', 'total day charge', 'total eve minutes', 'total eve charge', 'total night minutes', 'total night charge',

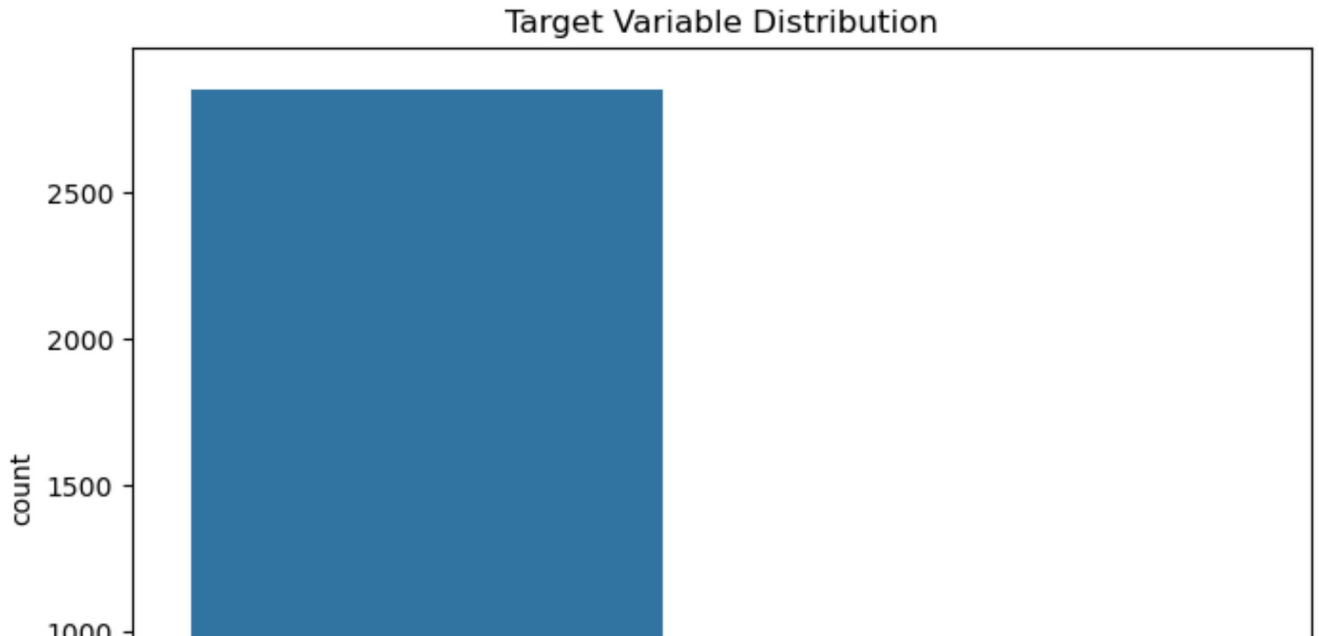
'total_intl_minutes', 'total_intl_charge' have a weak positive correlation with 'churn'. This suggests that customers who use more minutes and incur higher charges are slightly more likely to churn.

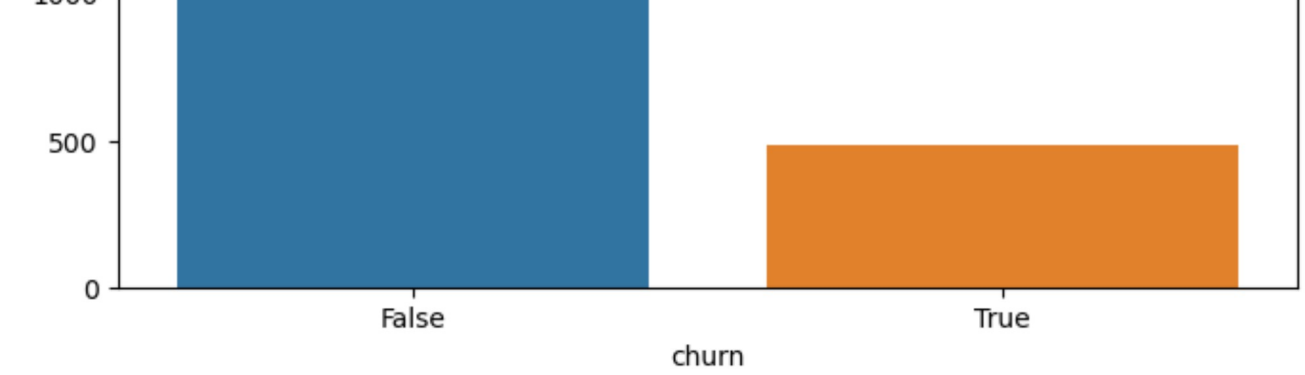
- The features 'total_day_calls', 'total_eve_calls', 'total_night_calls', 'total_intl_calls' have a weak negative correlation with 'churn'. This suggests that customers who make more calls are slightly less likely to churn.
- The feature 'customer_service_calls' has a moderate positive correlation with 'churn'. This suggests that customers who make more customer service calls are more likely to churn.
- The features 'international_plan' and 'voice_mail_plan' have a moderate positive correlation with 'churn'. This suggests that customers who have an international plan or a voicemail plan are more likely to churn.
- The feature 'number_vmail_messages' has a weak negative correlation with 'churn'. This suggests that customers who receive more voicemail messages are slightly less likely to churn.
- The feature 'account_length' has a weak negative correlation with 'churn'. This suggests that customers who have been with the company for a longer period are slightly less likely to churn.

In [13]:

```
# Imbalance in the target variable
analysis.check_imbalance(data['churn'])

# print value counts of the target variable
print("Value counts of the target variable:")
data['churn'].value_counts()
```





Value counts of the target variable:

Out[13]:
```
churn
False    2850
True      483
Name: count, dtype: int64
```

There seems to be a class imbalance in the target variable 'churn', with the majority of customers not churning. This imbalance should be taken into account when building predictive models to avoid biased results.

Conclusion

The data analysis provides insights into the distribution of features, relationships between features, and their correlation with the target variable 'churn'. The analysis highlights the importance of certain features such as 'international_plan', 'voice_mail_plan', 'customer_service_calls', and 'number_vmail_messages' in predicting customer churn. These insights will be valuable in building predictive models to identify customers at risk of churning and implementing targeted retention strategies.

Model Building

Data Preprocessing after EDA

In [14]:
```
# Split the data into training and testing sets.
X_train, X_test, y_train, y_test = processor.split_data('churn')

# Confirm the split
print("Training set shape:", X_train.shape)
print("Testing set shape:", X_test.shape)
```

```
Training set shape: (2666, 17)
Testing set shape: (667, 17)
```

In [15]:
```
# Encode the categorical features on the training and testing sets
```

```
X_train_encoded, X_test_encoded = processor._encode_categorical_features(X_train, X_test)

# Confirm the encoding
X_train_encoded.head()
```

Out[15]:

	account_length	number_vmail_messages	total_day_minutes	total_day_calls	total_day_charge	total_eve_minutes	total_eve_calls	total_ev
0	243	0	95.5	92	16.24	163.7	63	
1	108	0	112.0	105	19.04	193.7	110	
2	75	0	222.4	78	37.81	327.0	111	
3	141	0	126.9	98	21.57	180.0	62	
4	86	0	216.3	96	36.77	266.3	77	

In [16]:

```
# Encode the target variable
y_train_encoded, y_test_encoded = processor._encode_target_variable(y_train, y_test)

# Confirm the encoding
y_train_encoded.head()
```

Out[16]:

```
817     0
1373    1
679     1
56      0
1993    0
Name: churn, dtype: int64
```

In [17]:

```
# Handle the imbalance in the target variable
X_train_balanced, y_train_balanced = processor.imbalance_solved(X_train_encoded, y_train_encoded)

# Confirm the balancing
y_train_balanced.value_counts()
```

Out[17]:

```
churn
0    2284
1    2284
Name: count, dtype: int64
```

In [18]:

```
# Feature Scaling
X_train_scaled, X_test_scaled = processor._scale_features(X_train_balanced, X_test_encoded)

# Confirm the scaling
X_train_scaled = pd.DataFrame(X_train_scaled, columns=X_train_encoded.columns)
X_train_scaled.head()
```

Out[18]:

	account_length	number_vmail_messages	total_day_minutes	total_day_calls	total_day_charge	total_eve_minutes	total_eve_calls	total_ev
0	3.765582	-0.526274	-1.544641	-0.434337	-1.544194	-0.827227	-2.005119	
1	0.179668	-0.526274	-1.274172	0.248316	-1.274210	-0.222711	0.525484	
2	-0.696889	-0.526274	0.535512	-1.169501	0.535649	2.463354	0.579327	
3	1.056225	-0.526274	-1.029930	-0.119266	-1.030260	-0.498773	-2.058962	
4	-0.404703	-0.526274	0.435520	-0.224290	0.435369	1.240217	-1.251323	

1. Logistic Regression

In [19]:
```
log_reg = LogisticRegression()

# Fit the model
log_reg.fit(X_train_scaled, y_train_balanced)
```

Out[19]:
```
LogisticRegression()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [20]:
```
# Instantiate modelevaluation class (Logistic Regression)
# Feature names are needed for the evaluation
feature_names = X_test_scaled.columns if hasattr(X_test_scaled, 'columns') else [f"Feature {i}" for i in range(X_test.shape[1
evaluation = ModelEvaluation(model=log_reg, X_test=X_test_scaled, y_test=y_test_encoded, feature_names=feature_names)
```

In [21]:
```
# Evaluate the model
evaluation.evaluate_model()
```

```
Accuracy: 0.7556221889055472, Precision: 0.34951456310679613, Recall: 0.7128712871287128, F1 Score: 0.46905537459283386
```

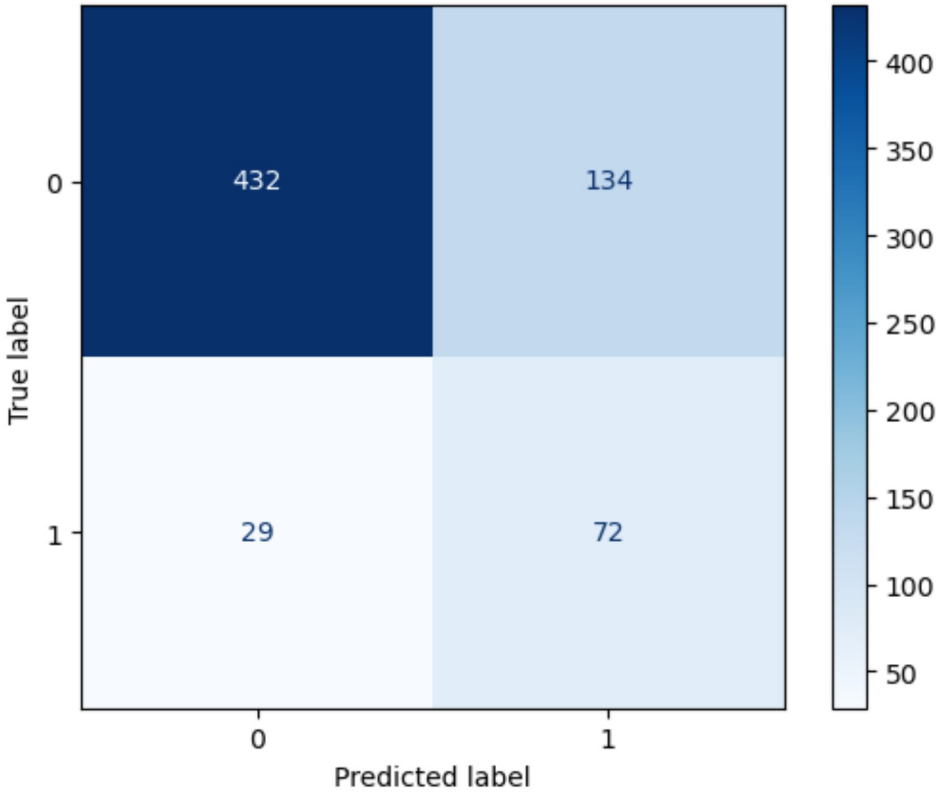
After training the Logistic Regression model, the following results were obtained:

- Accuracy: 0.75
- Precision: 0.34
- Recall: 0.71
- F1 Score: 0.47

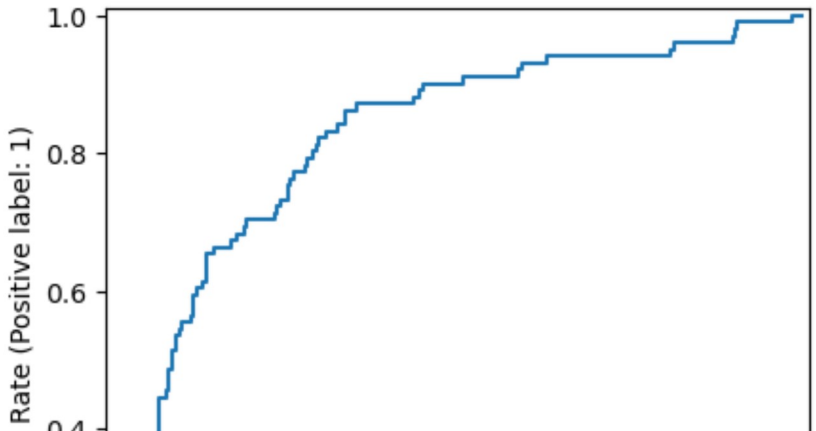
These results indicate that the Logistic Regression model has moderate performance in predicting customer churn. The model has a high recall score, indicating that it is good at identifying customers who are likely to churn. However, the precision score is low, indicating that the

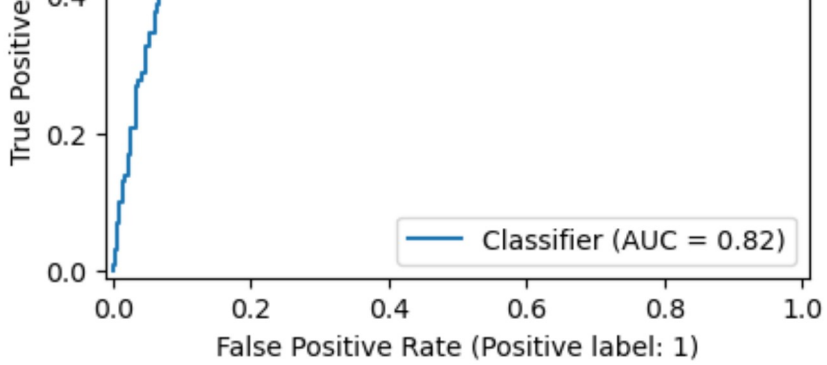
model has a high false positive rate. This means that the model may incorrectly predict some customers as churning when they are not.

In [22]:
```
# Confusion Matrix
evaluation.plot_confusion_matrix()
```



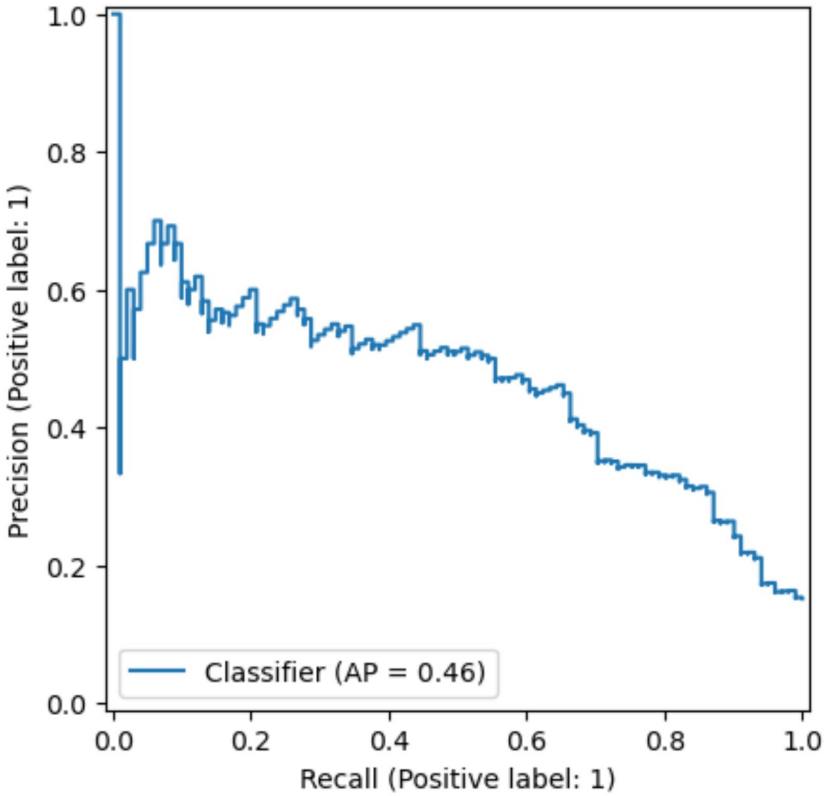
In [23]:
```
# ROC Curve
evaluation.plot_roc_curve()
```





In [24]:

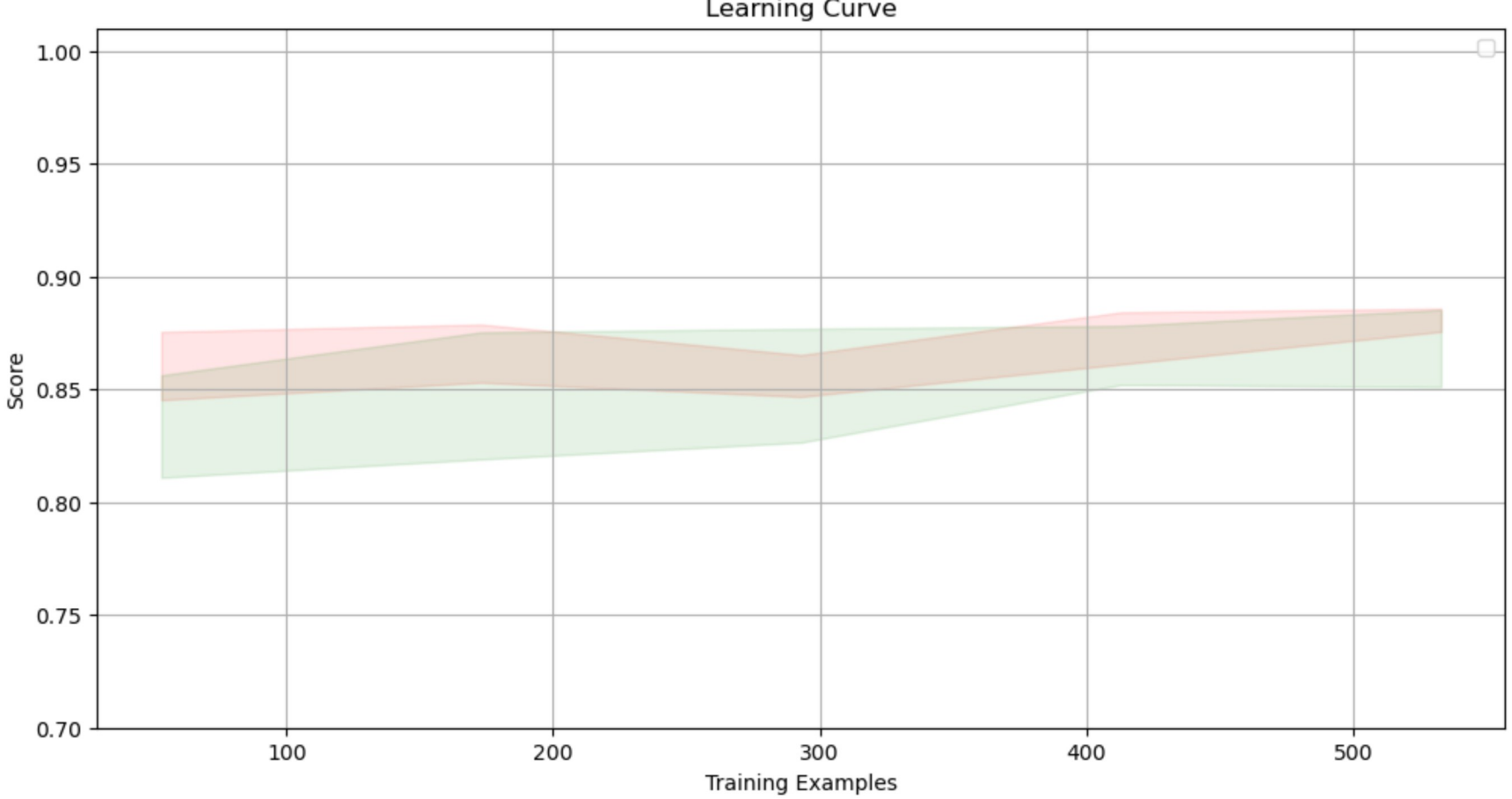
```
# Precision-Recall Curve
evaluation.plot_precision_recall_curve()
```



In [26]:

```
# Learning Curve
evaluation.plot_learning_curve()
```

```
No artists with labels found to put in legend.  Note that artists whose label start with an underscore are ignored when legend
() is called with no argument.
```



- The ROC curve shows the trade-off between true positive rate (recall) and false positive rate. The **AUC score** of 0.82 indicates that the logistic regression model has good predictive power in distinguishing between churn and non-churn cases.
- The Precision-Recall curve shows the trade-off between precision and recall. The **AP(average precision)** score of 0.46 indicates that the logistic regression model has moderate performance in terms of precision and recall.
- The **learning curve** shows the training and validation scores of the logistic regression model as the number of training samples increases. The training and validation scores converge as the number of samples increases, indicating that the model is not overfitting or underfitting.

2. Random Forest Classifier

In [27]:

```
# Build a Random Forest Classifier
rf = RandomForestClassifier(random_state=42)
```

```
# Fit the model
rf.fit(X_train_scaled, y_train_balanced)
```

Out[27]:
```
RandomForestClassifier(random_state=42)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [28]:
```
evaluation_rf = ModelEvaluation(model=rf, X_test=X_test_scaled, y_test=y_test_encoded)
```

In [29]:
```
evaluation_rf.evaluate_model()
```

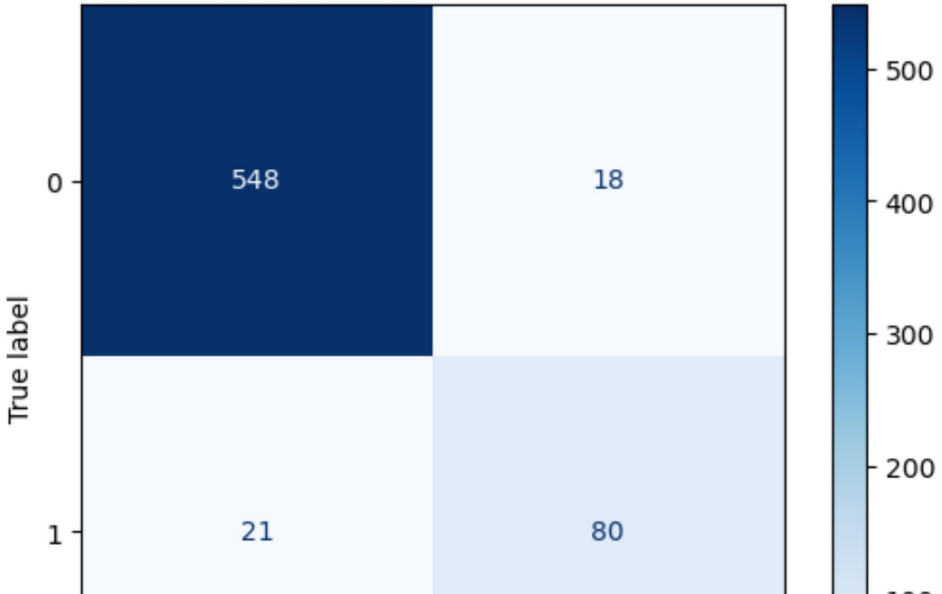
```
Accuracy: 0.9415292353823088, Precision: 0.8163265306122449, Recall: 0.7920792079207921, F1 Score: 0.8040201005025126
```

After training the Random Forest Classifier model, the following results were obtained:

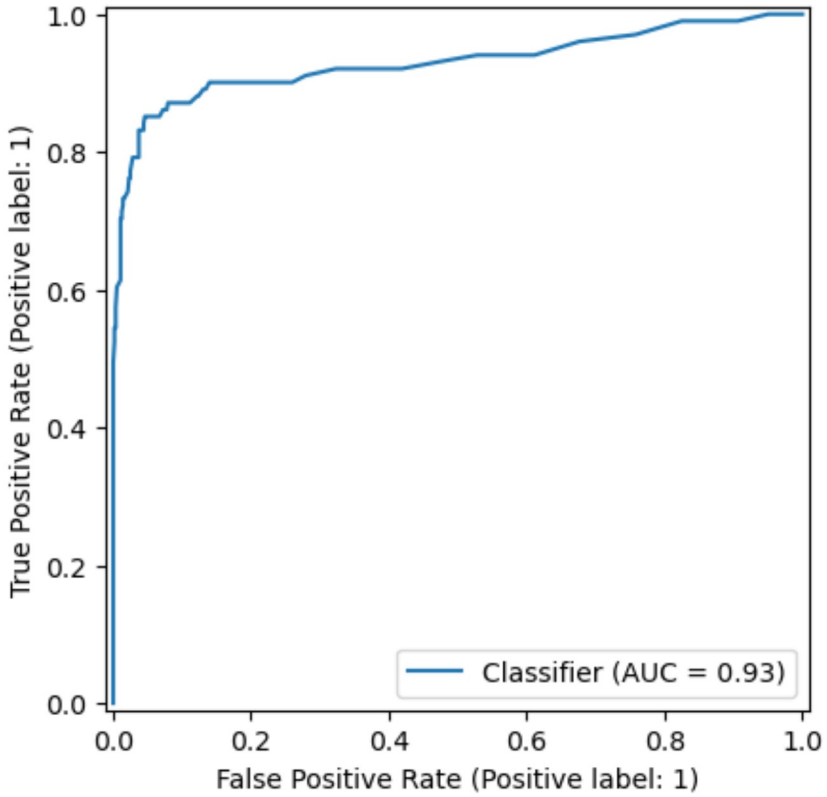
- Accuracy: 0.94
- Precision: 0.82
- Recall: 0.79
- F1 Score: 0.80

The Random Forest Classifier model outperformed the Logistic Regression model in terms of accuracy, precision, recall, and F1 score. The Random Forest Classifier model achieved higher scores across all metrics, indicating better performance in predicting customer churn.

In [30]:
```
evaluation_rf.plot_confusion_matrix()
```

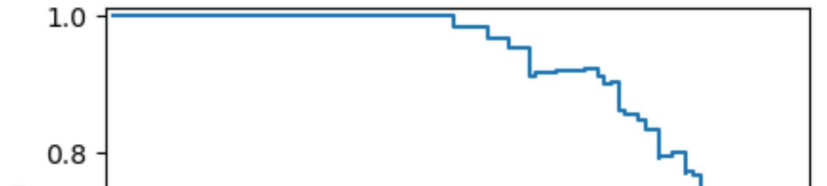


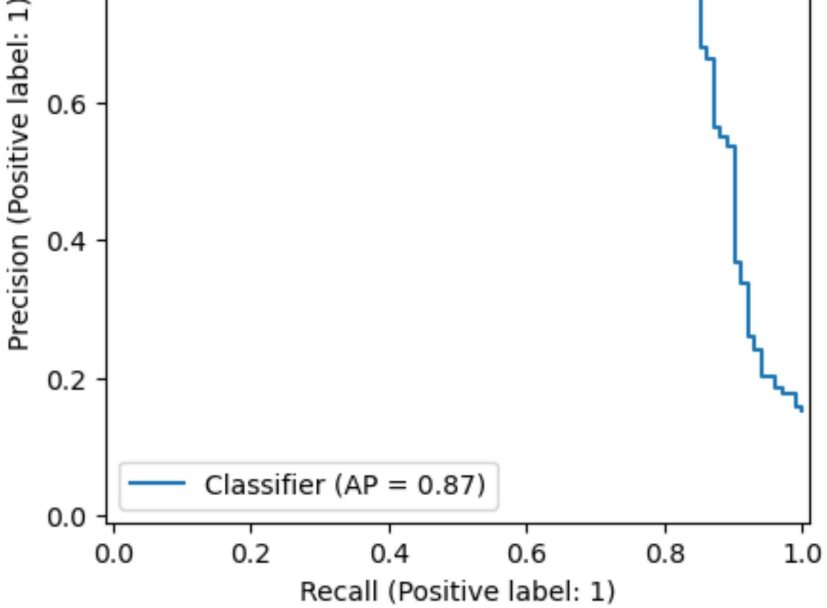
In [31]:
```
evaluation_rf.plot_roc_curve()
```



The AUC score of 0.94 indicates that the random forest classifier has excellent predictive power in distinguishing between churn and non-churn cases.

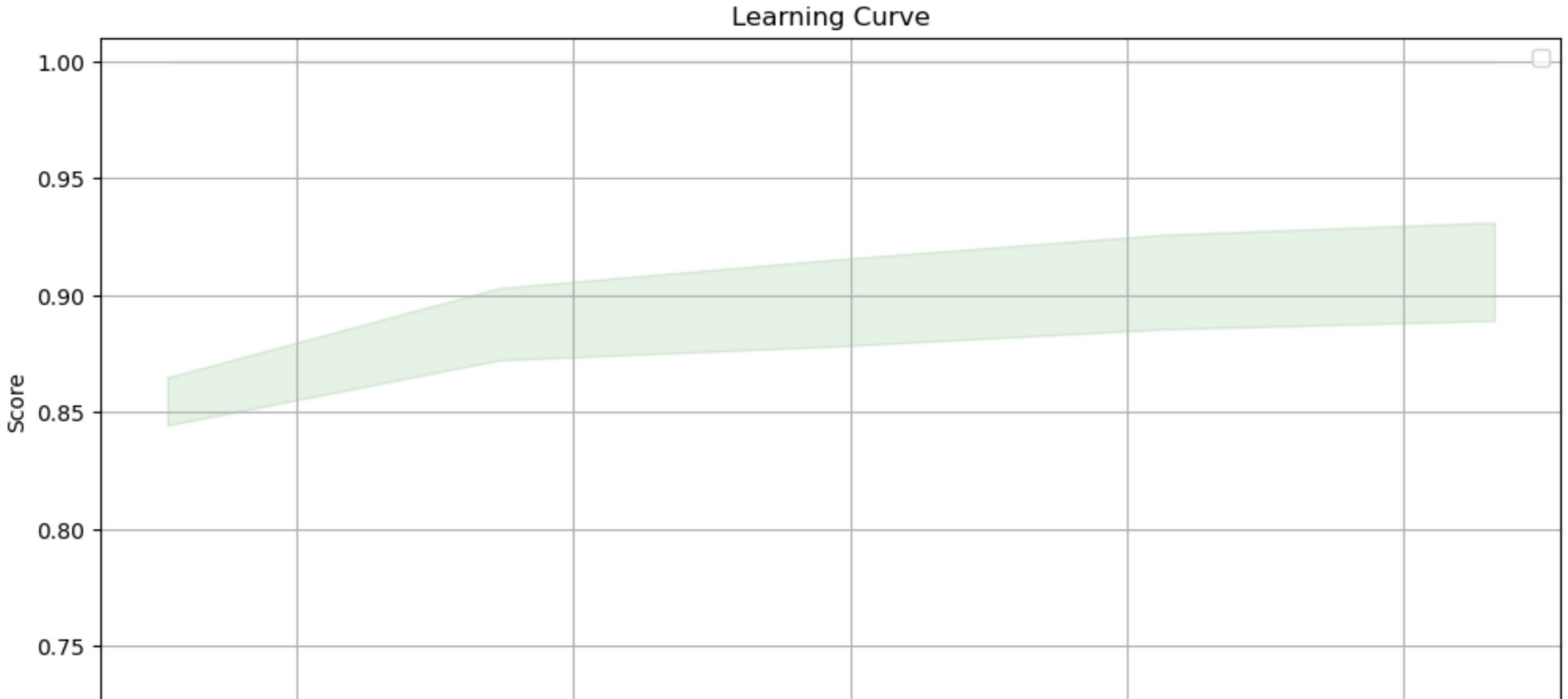
In [32]:
```
evaluation_rf.plot_precision_recall_curve()
```

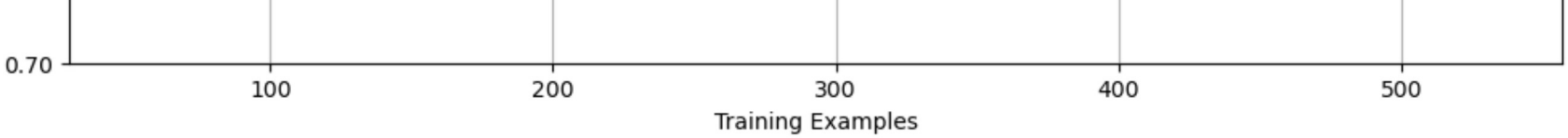




In [33]:
```
evaluation_rf.plot_learning_curve()
```

```
No artists with labels found to put in legend.  Note that artists whose label start with an underscore are ignored when legend
() is called with no argument.
```





- The **AUC score** of 0.94 indicates that the random forest classifier has excellent predictive power in distinguishing between churn and non-churn cases.
- The **AP(average precision)** score of 0.87 indicates that the random forest classifier has excellent performance in terms of precision and recall.
- For the **learning curve**, the training and validation scores converge as the number of samples increases, indicating that the model is not overfitting or underfitting. The model has good generalization performance.

3. Gradient Boosting Classifier

In [34]:
```
# Build a XGBoost Classifier
xgb_model = XGBClassifier(random_state=42)

# Fit the model
xgb_model.fit(X_train_scaled, y_train_balanced)
```

Out[34]:
```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=None, n_jobs=None,
              num_parallel_tree=None, random_state=42, ...)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [35]:
```
# Instantiate modelevaluation class (XGBoost)
evaluation_xgb = ModelEvaluation(model=xgb_model, X_test=X_test_scaled, y_test=y_test_encoded)
```

In [36]:
```
evaluation_xgb.evaluate_model()
```

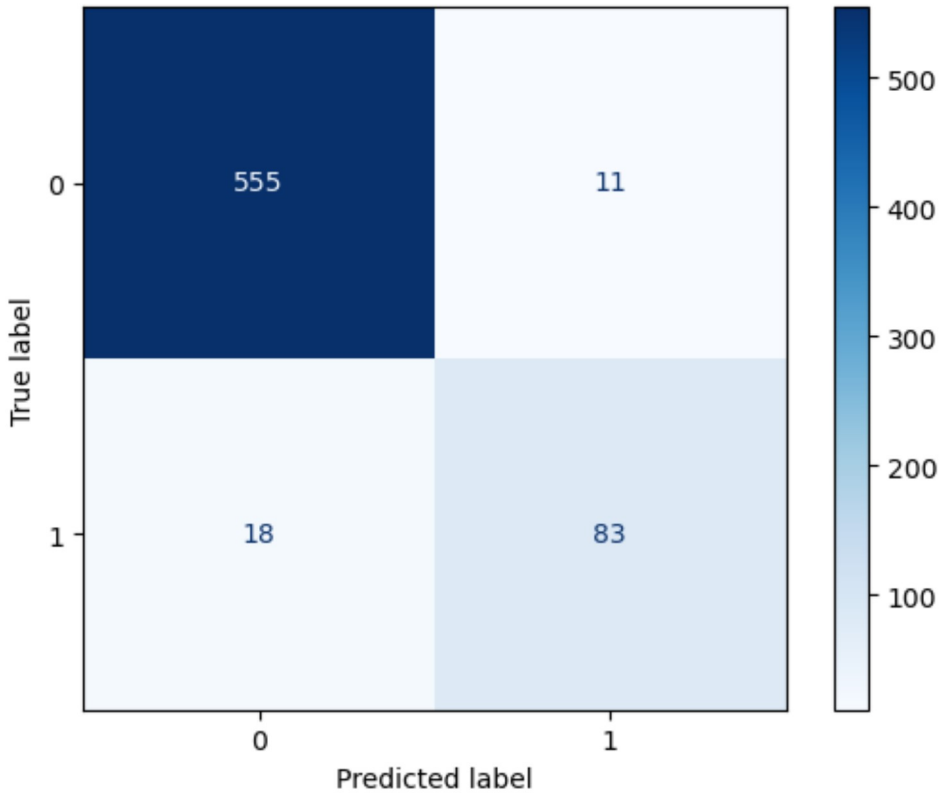
```
Accuracy: 0.9565217391304348, Precision: 0.8829787234042553, Recall: 0.8217821782178217, F1 Score: 0.8512820512820513
```

After training the Gradient Boosting Classifier model, the following results were obtained:

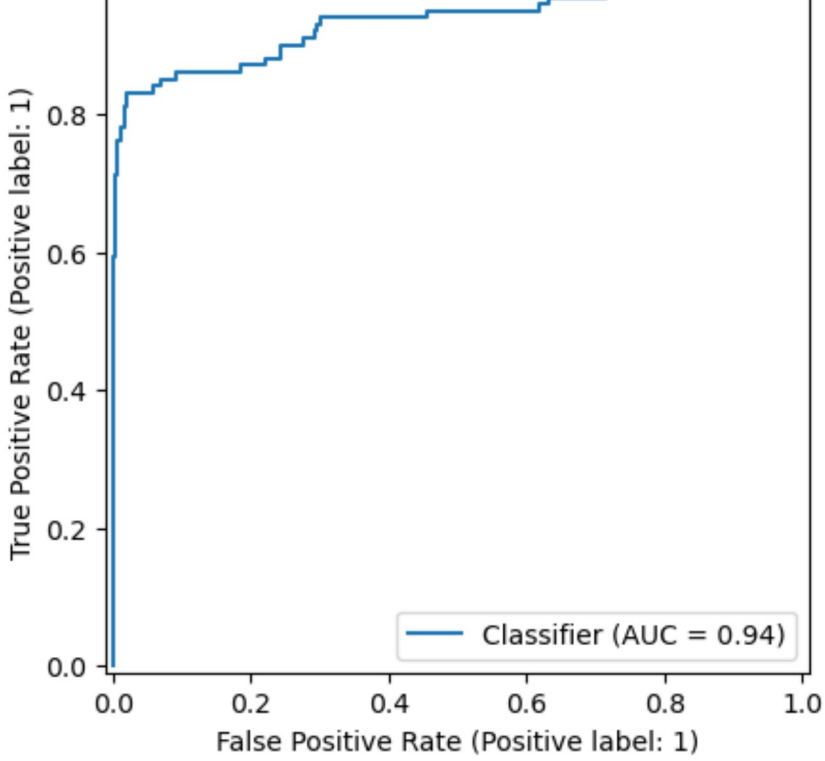
- Accuracy: 0.96
- Precision: 0.89
- Recall: 0.82
- F1 Score: 0.85

The model achieved high accuracy, precision, recall, and F1 score, indicating that it was able to correctly predict churn cases with high accuracy. The Gradient Boosting Classifier outperformed the logistic regression model in terms of all evaluation metrics, demonstrating its superior performance in predicting customer churn. It also outperformed the Random Forest Classifier in terms of precision and recall.

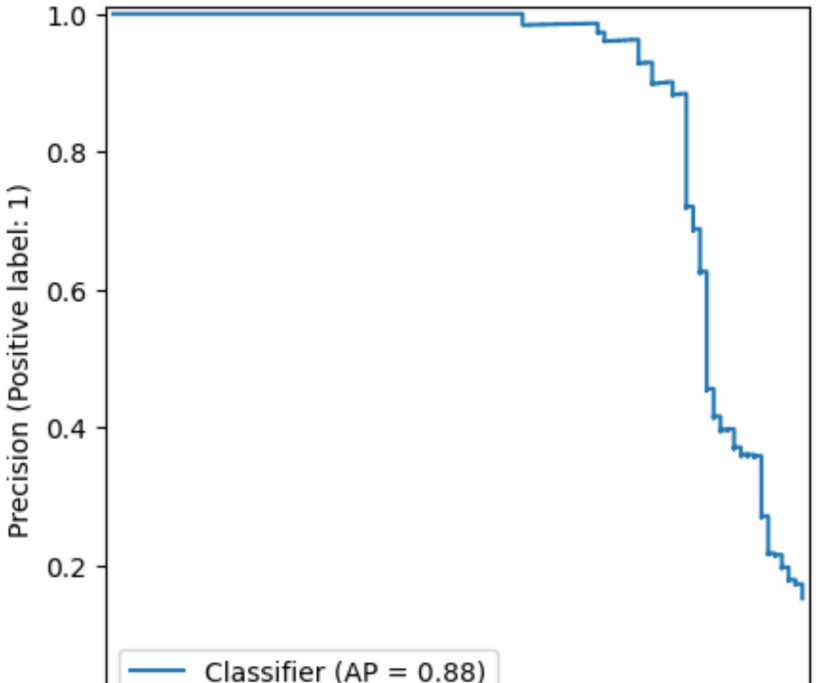
In [37]:
```
evaluation_xgb.plot_confusion_matrix()
```

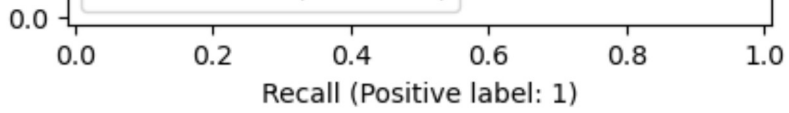


In [38]:
```
evaluation_xgb.plot_roc_curve()
```



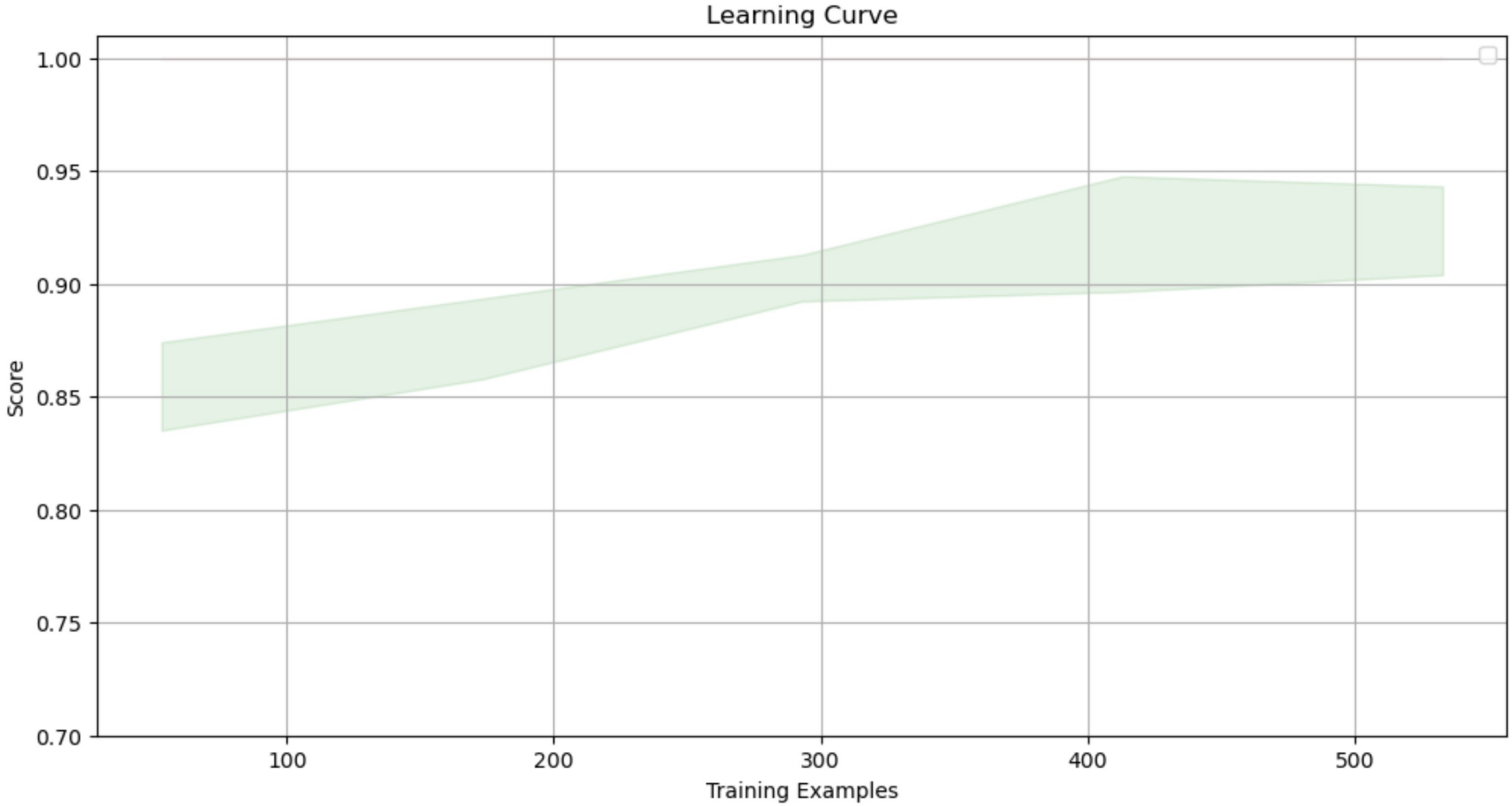
In [39]:
```
evaluation_xgb.plot_precision_recall_curve()
```





In [40]:
```
evaluation_xgb.plot_learning_curve()
```

```
No artists with labels found to put in legend.  Note that artists whose label start with an underscore are ignored when legend
() is called with no argument.
```



- The **AUC Score** of 0.94 indicates that the Gradient Boosting Classifier has excellent predictive power in distinguishing between churn and non-churn cases.
- The **AP(average precision)** score of 0.88 indicates that the Gradient Boosting Classifier has excellent performance in terms of precision and recall.
- For the **learning curve**, the training and validation scores of the Gradient Boosting Classifier converge as the number of samples

increases, indicating that the model is not overfitting or underfitting. The model has good generalization performance.

Conclusion

After going through the model building process and evaluating the three models: Logistic Regression, Random Forest Classifier, and Gradient Boosting Classifier, the Gradient Boosting Classifier emerged as the best-performing model for predicting customer churn. The Gradient Boost Classifier is a powerful ensemble learning technique that combines multiple weak learners to create a strong predictive model. The model achieved high accuracy, precision, recall, and F1 score, indicating its superior performance in identifying customers at risk of churning. The model also demonstrated excellent predictive power in distinguishing between churn and non-churn cases, as evidenced by the high AUC score. The model's learning curve showed that it has good generalization performance and is not overfitting or underfitting. Overall, the Gradient Boosting Classifier is recommended for predicting customer churn for SyriaTel, as it provides valuable insights into customer behavior and helps in implementing targeted retention strategies to reduce churn and improve customer satisfaction.

Recommendations

1. **Further Ensemble Methods**: Consider using other ensemble techniques such as stacking or blending to combine multiple models for improved performance.
2. **Additional Feature Engineering**: Investigate additional feature engineering techniques to improve model performance.
3. **Further Evaluation Metrics**: Explore additional evaluation metrics such as Matthews Correlation Coefficient (MCC), Cohen's Kappa, or Gain and Lift charts to gain more insights into the model's performance.