# CUSTOMER CHURN PREDICTION

**Name: Balisetty chenchu Nithin**

**[au723921104003]**

**Project: Customer Churn Prediction**

**Phase:3- Development Part 1**

**Topic: In this part you will begin building your project by loading and pre-processing the dataset.**

**Customer churn prediction is a process of identifying and forecasting which customers are likely to stop using a product or service in the near future. Churn, also known as customer attrition, occurs when customers discontinue their relationship with a company, and it's a crucial concern for businesses in various industries, such as telecommunications, subscription services, e-commerce, and more. Churn prediction involves using data analysis and machine learning techniques to estimate the probability of a customer leaving, typically within a defined time frame.**

**Here's why customer churn prediction is important and how it's used:**

**1)Retaining Customers**

**2)** **Cost Reduction**

**3)** **Revenue Protection**

**4)** **Product and Service Improvement**

**5)** **Marketing and Customer Engagement**

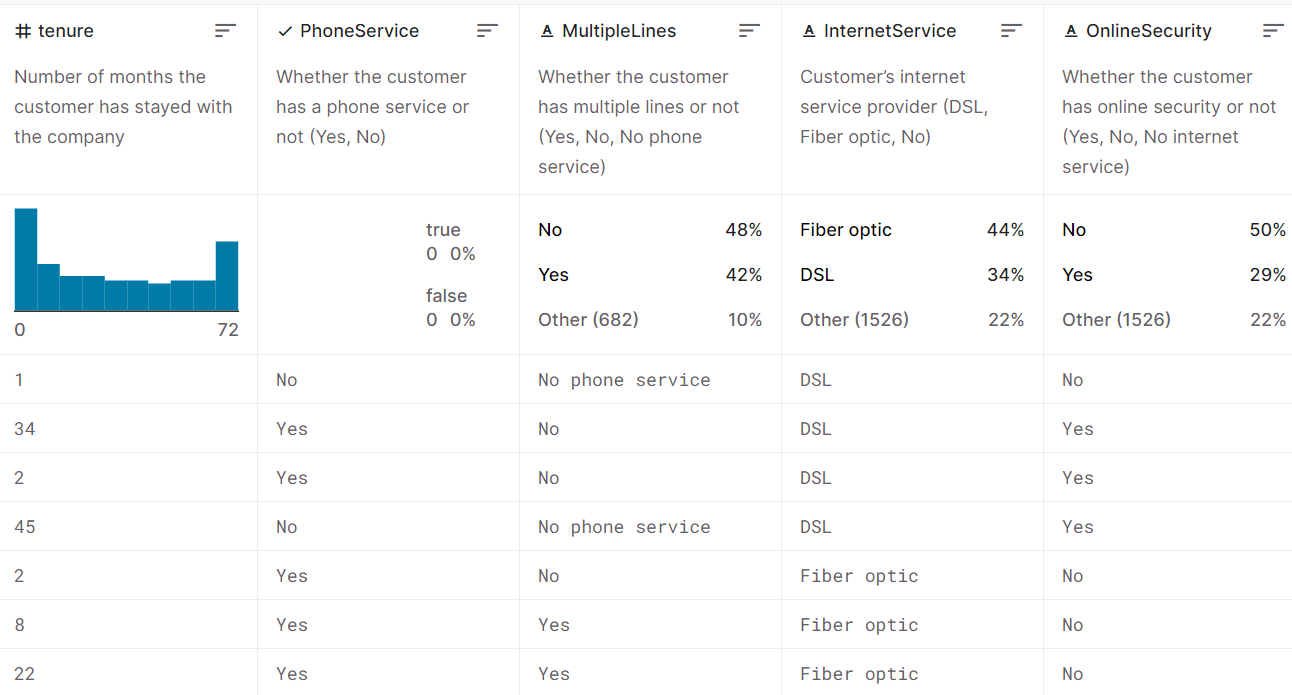
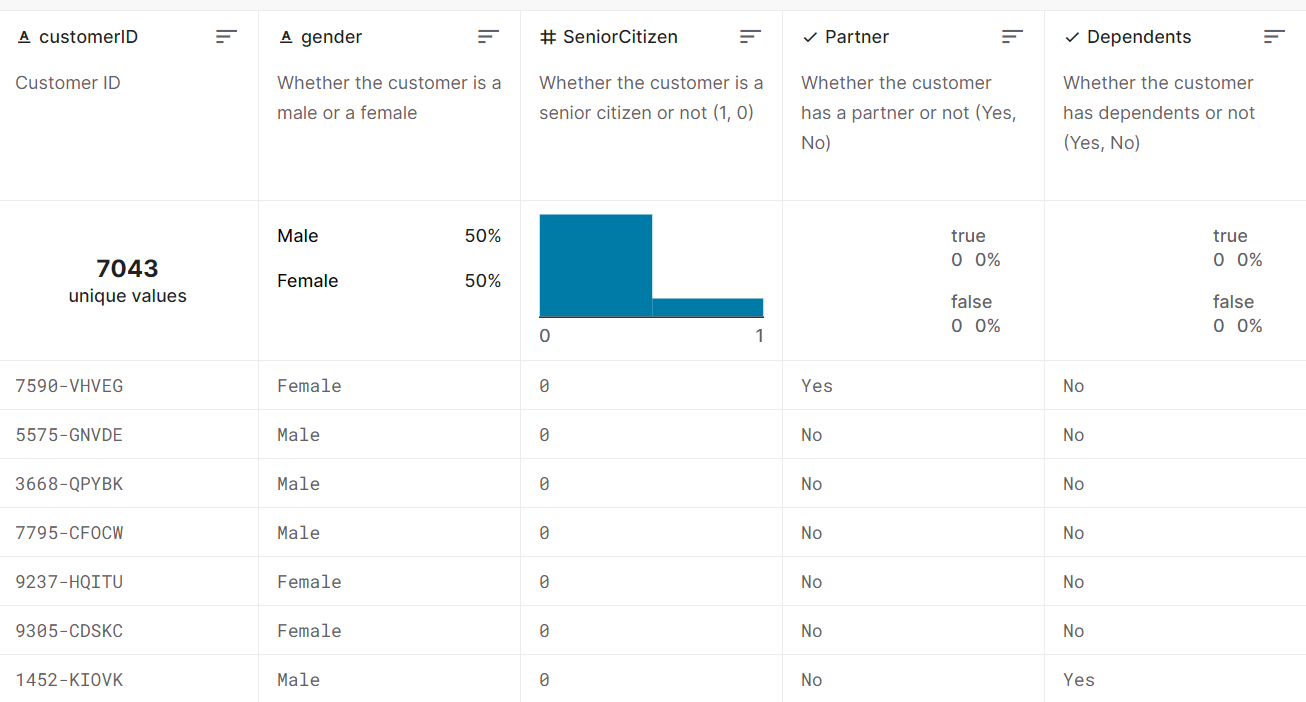
**6)Customer Satisfaction**

**Given Data Set:**

**Each row represents a customer, each column contains customer’s attributes described on the column Metadata.**

**The raw data contains 7043 rows (customers) and 21 columns (features).**

**The “Churn” column is our target.**

****

**Importance of Loading and pre-processing datasets**

Loading and pre-processing datasets is a fundamental and critical step in data analysis, machine learning, and data-driven decision-making. Here are several key reasons for the importance of these tasks:

**Data Quality Assurance:** Loading and pre-processing data allow you to inspect and ensure the quality of the dataset. You can identify and address issues like missing values, outliers, and inconsistencies, which are essential for reliable and accurate analysis.

**Data Understanding:** Loading the data provides an initial understanding of its structure, size, and format. It helps you become familiar with the dataset, which is necessary for planning your analysis, choosing the right techniques, and making informed decisions.

**Data Cleaning:** Pre-processing involves data cleaning, which includes handling missing data, removing duplicates, and addressing outliers. Clean data is vital for obtaining meaningful and reliable insights, as well as for building robust machine learning models.

**Feature Engineering:** Pre-processing often involves feature engineering, where you create, transform, or select features to enhance the predictive power of your models or to gain better insights in data analysis.

**Normalization and Scaling:** Pre-processing ensures that data is on a consistent scale, which is particularly important for many machine learning algorithms. Normalization and scaling help prevent features with larger values from dominating the learning process and ensure the algorithms work effectively.

**Categorical Data Handling:** Many datasets include categorical variables that need to be transformed into numerical format for machine learning models to process. Pre-processing helps in encoding or transforming these variables, making them suitable for analysis.

**Efficiency:** Proper pre-processing can lead to more efficient analysis and modelling. By removing unnecessary or redundant information and optimizing data structures, you can speed up computations and reduce resource requirements.

**Data Privacy and Security:** Pre-processing can also play a role in data privacy and security. Steps like anonymization and data masking can be part of the pre-processing process to protect sensitive information

Python program:

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import missingno as msno

import warnings

warnings.filterwarnings("ignore")

from sklearn.experimental import enable\_iterative\_imputer

from sklearn.impute import IterativeImputer

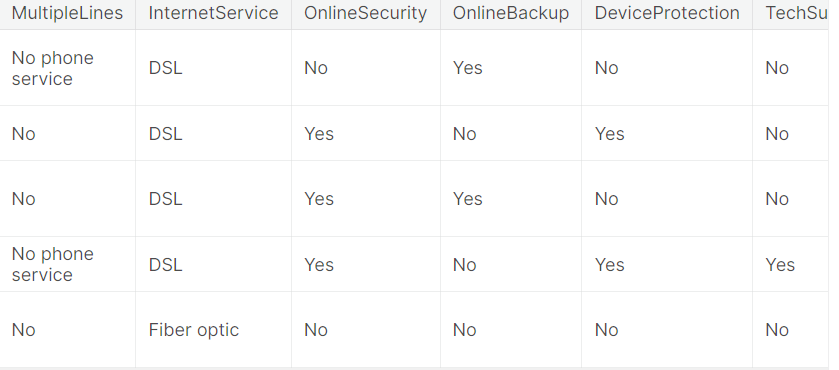
from sklearn.preprocessing import LabelEncoder

pd.options.display.max\_columns = None

# **Import Data**

data = pd.read\_csv("/kaggle/input/telco-customer-churn/WA\_Fn-UseC\_-Telco-Customer-Churn.csv")

**OUTPUT:**



plt.figure(figsize = (15, 4))

for i, col in enumerate(num\_cols):

plt.subplot(1, 3, i+1)

sns.histplot(data, x= col, color = 'red', alpha = 0.2, kde = True)

plt.tight\_layout()

plt.show()

|  | count | unique | top | Freq |
| --- | --- | --- | --- | --- |
| Churn | 7043 | 2 | No | 5174 |
| Contract | 7043 | 3 | Month-to-month | 3875 |
| Dependents | 7043 | 2 | No | 4933 |
| Device protection | 7043 | 3 | No | 3095 |
| Gender | 7043 | 2 | Male | 3555 |
| Internet service | 7043 | 3 | FiberOptic | 3096 |
| Multiple lines | 7043 | 3 | No | 3390 |
| Online backup | 7043 | 3 | No | 3088 |
| Online security | 7043 | 3 | No | 3498 |
| Paperless billing | 7043 | 2 | Yes | 4171 |
| partner | 7043 | 2 | No | 3641 |
| Payment method | 7043 | 4 | Electronic check | 2365 |

plt.figure(figsize = (15, 4))

for i, col in enumerate(num\_cols):

plt.subplot(1, 3, i+1)

sns.histplot(data, x= col, color = 'red', alpha = 0.2, kde = True)

plt.tight\_layout()

plt.show()plt.figure(figsize = (15, 4))

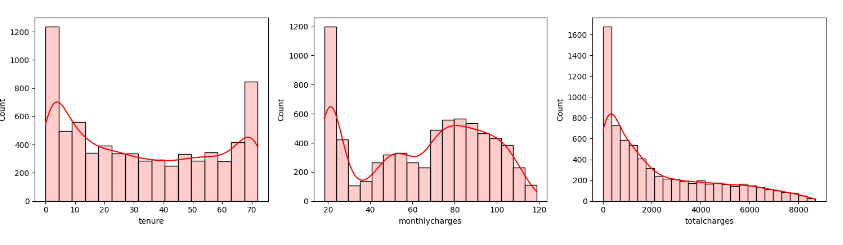
for i, col in enumerate(num\_cols):

plt.subplot(1, 3, i+1)

sns.histplot(data, x= col, color = 'red', alpha = 0.2, kde = True)

plt.tight\_layout()

plt.show()



plt.figure(figsize = (15, 4))

for i, col in enumerate(num\_cols):

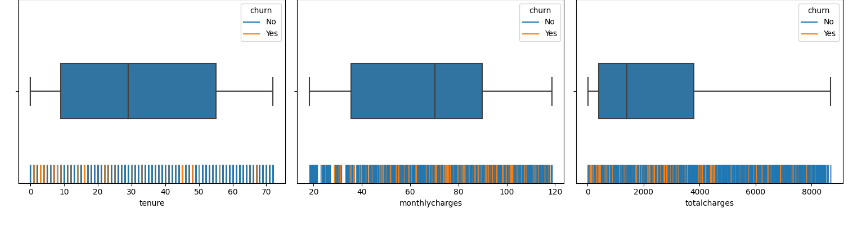
plt.subplot(1, 3, i+1)

sns.rugplot(data, x = col, hue= label, height = 0.1)

sns.boxplot(data, x = col, width = 0.3)

plt.tight\_layout()

plt.show()



plt.figure(figsize = (15, 4))

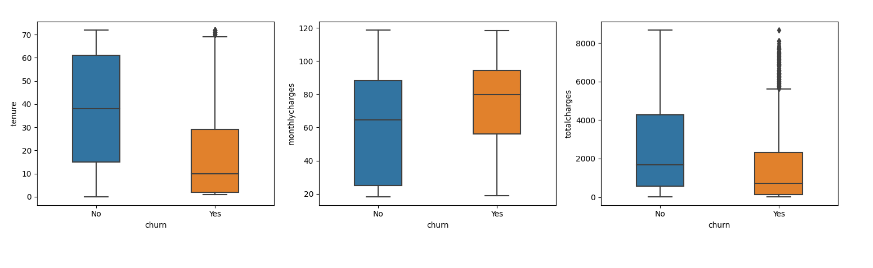
for i, col in enumerate(num\_cols):

plt.subplot(1, 3, i+1)

sns.boxplot(data, x = label, y = col, width = 0.4)

plt.tight\_layout()

plt.show()



plt.figure(figsize = (15, 26))

for i, col in enumerate(data.columns.difference(num\_cols)[1:]):

plt.subplot(6, 3, i+1)

ax = sns.countplot(data, x = col, hue = label)

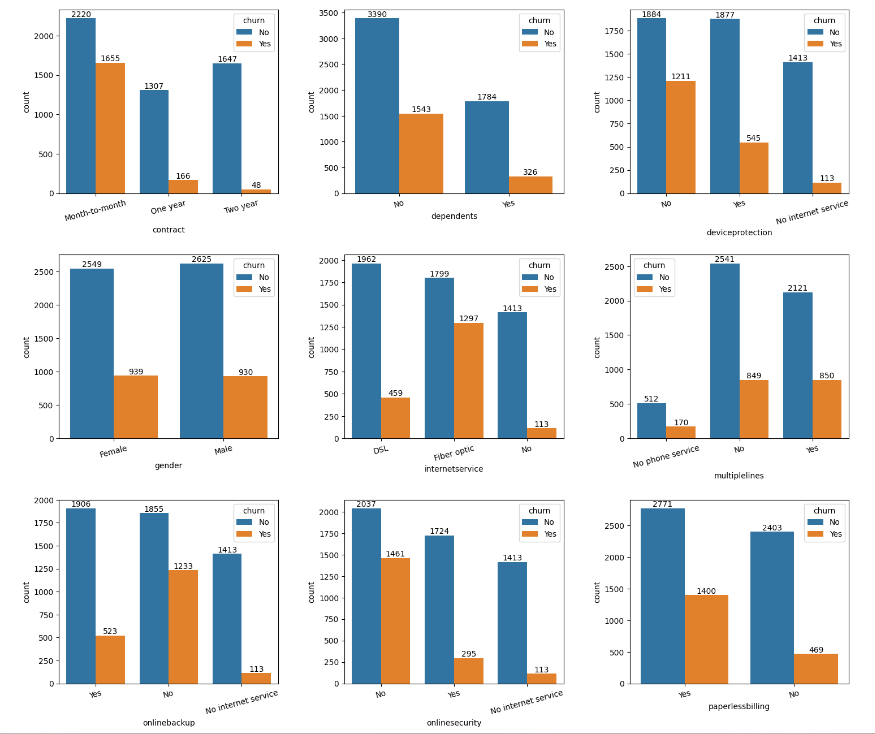
ax.bar\_label(ax.containers[0])

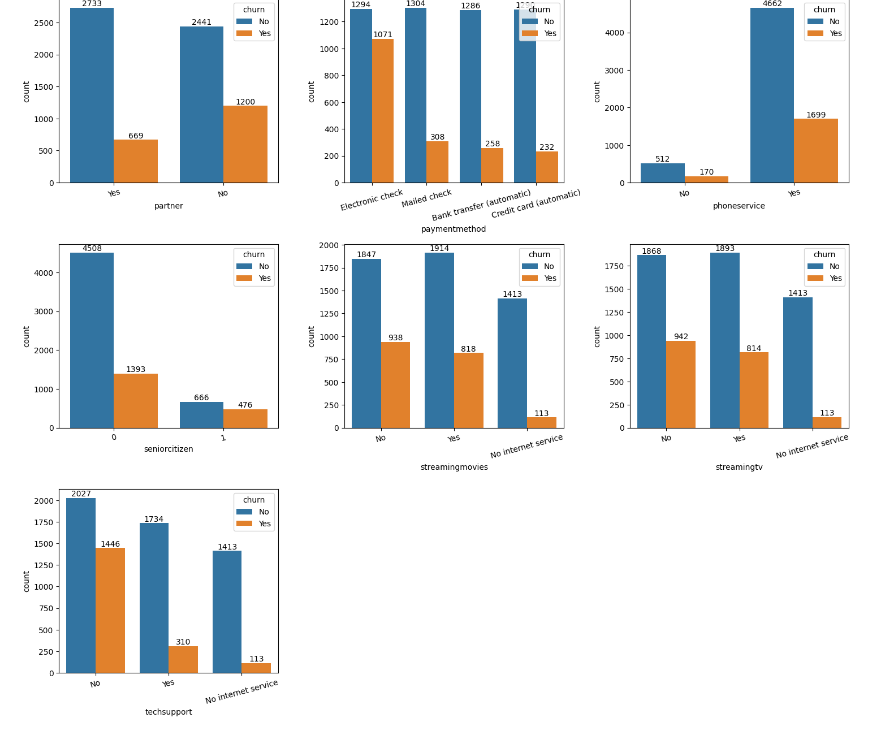
ax.bar\_label(ax.containers[1])

plt.xticks(rotation = 15)

plt.tight\_layout()

plt.show()



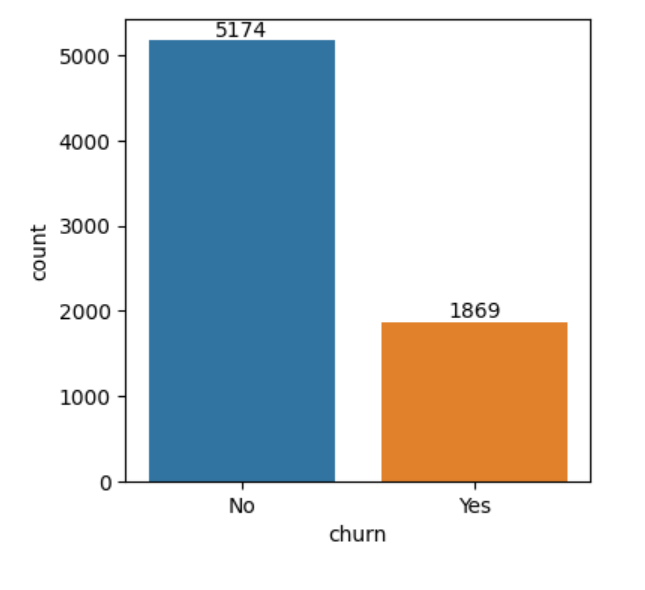


plt.figure(figsize = (4,4))

ax = sns.countplot(data, x = label)

ax.bar\_label(ax.containers[0])

plt.show()



from pycaret.classification import \*

s = setup(data, target = 'churn', session\_id = 42, data\_split\_stratify=True)

|  | Description | Value |
| --- | --- | --- |
| 0 | Session id | 42 |
| 1 | Target | Churn |
| 2 | Target type | Binary |
| 3 | Original data shape | (7043, 41) |
| 4 | Transformed data shape | (7043, 41) |
| 5 | Transformed train set shape | (4930, 41) |
| 6 | Transformed test set shape | (2113, 41) |
| 7 | Numeric features | 40 |
| 8 | Pre process | True |
| 9 | Imputation type | Simple |
| 10 | Numeric imputation | Mean |
| 11 | Categorical imputation | Mode |
| 12 | Fold Generator | Stratified K-Fold |
| 13 | Fold Number | 10 |
| 14 | CPU Jobs | -1 |
| 15 | Use GPU | False |
| 16 | Log Experiment | False |
| 17 | Experiment Name | clf-default-name |
| 18 | USI | 69f2 |

best\_model = compare\_models(sort = 'AUC')



print(best\_model)

GradientBoostingClassifier(ccp\_alpha=0.0, criterion='friedman\_mse', init=None,

learning\_rate=0.1, loss='log\_loss', max\_depth=3,

max\_features=None, max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0, min\_samples\_leaf=1,

min\_samples\_split=2, min\_weight\_fraction\_leaf=0.0,

n\_estimators=100, n\_iter\_no\_change=None,

random\_state=42, subsample=1.0, tol=0.0001,

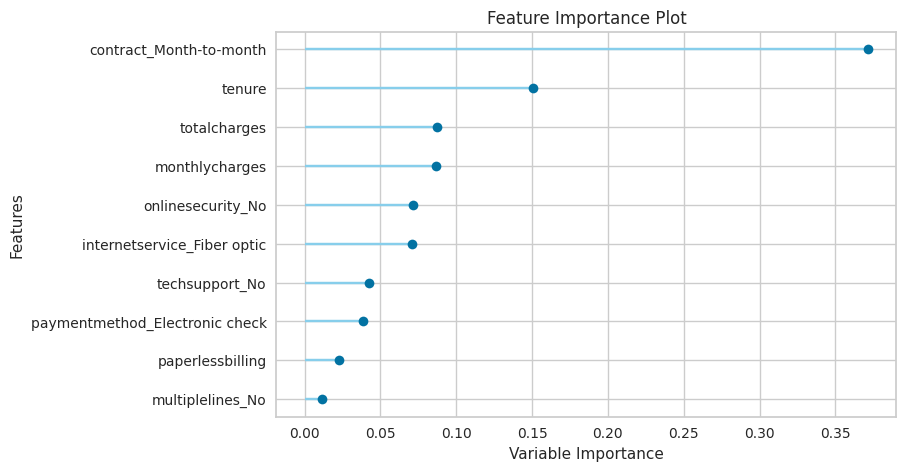
validation\_fraction=0.1, verbose=0,

warm\_start=False)

plt.figure(figsize = (7, 4))

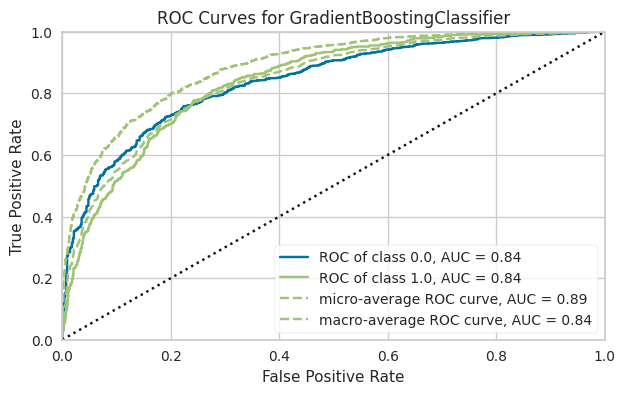
plot\_model(best\_model, plot = 'feature')

<Figure size 700x400 with 0 Axes>



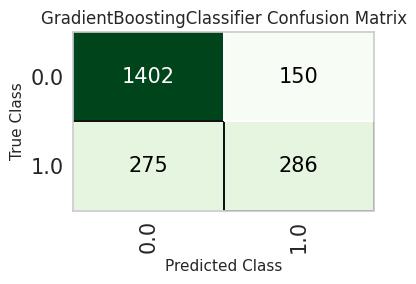
plt.figure(figsize = (7, 4))

plot\_model(best\_model, plot = 'auc')



plt.figure(figsize = (4,3))

plot\_model(best\_model, plot = 'confusion\_matrix')



* We have 286 **True Positives** - these are the customers for which we will be able to extend the lifetime value. If we wouldn't have predicted, then there was no opportunity for intervention
* We also have 150 **False Positives** where we will lose money because the promotion offered to these customers will just be an extra cost
* 1,402 are **True Negatives** (good customers) and 278 are **False Negatives** (this is a missed oppourtunity)

**The model which we have trained above is a good model but the problem is it's not a business smart model. We need to train a model that maximizes the business value. In order to achieve that we have to train, select and optimize models using business metrics instead of any conventional metric like AUC or Accuracy.**

# **Cost Benefit Analysis**

Link code

In a churn model, often the reward of true positives is way different than the cost of false positives. Let's use the following assumptions:

$1000 voucher will be offered to all the customers identified as churn (True Positive + False Positive)

If we are able to stop the churn, we will gain $5000 in customer lifetime value.

Using these assumptions and the confusion matrix above, we can calculate the profit impact of this model:

| Description | Customers | $ Value | Total |
| --- | --- | --- | --- |
| True Positive | 286 | $ 5,000 | $ 1,430,000 |
| True Positive + False Positive | 436 | -$ 1,000 | -$ 436,000 |
|  |  |  | $ 994,000 |

def calculate\_profit(y, y\_pred):

tp = np.where((y\_pred == 1) & (y == 1), 4000, 0)

fp = np.where((y\_pred == 1) & (y == 0), -1000, 0)

return np.sum([tp,fp])

In[45]:

add\_metric('profit', 'Profit''Profit', calculate\_profit)

Out[45]:

Name Profit

Display Name Profit

Score Function <pycaret.internal.metrics.EncodedDecodedLabels...

Scorer make\_scorer(calculate\_profit)

Target pred

Args {}

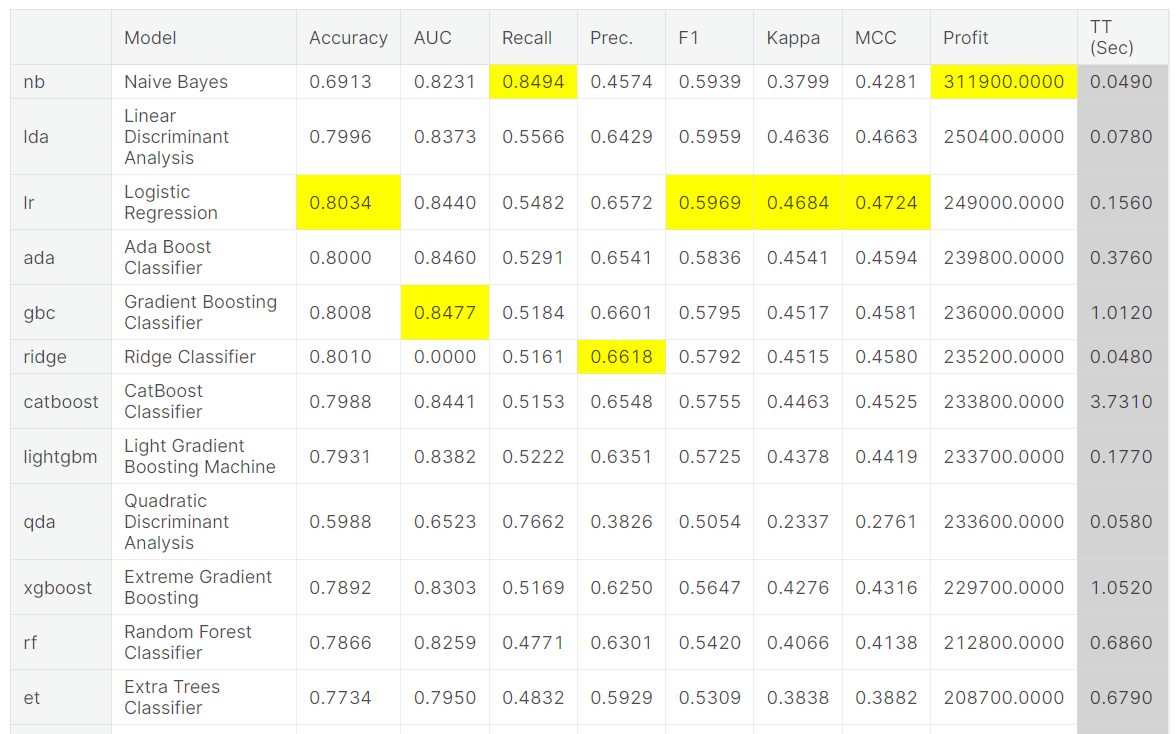
Greater is Better True

Multiclass True

Custom True

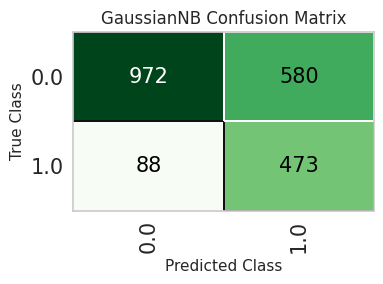
Name: profit, dtype: object

best\_model = compare\_models(sort = 'Profit')



plt.figure(figsize = (4,3))

plot\_model(best\_model, plot = 'confusion\_matrix')



This model is making errors over false positives and false negatives. The profit impact using the same assumptions:

| Description | Customers | $ Value | Total |
| --- | --- | --- | --- |
| True Positive | 473 | $ 5,000 | $ 2,365,000 |
| True Positive + False Positive | 580 | -$ 1,000 | -$ 580,000 |
|  |  |  | $ 1,785,000 |

linkcode

### **We have increased profit by $791,000 with a model that does 2% less in AUC than the best model.**

save\_model(best\_model, 'churn-predict')

Transformation Pipeline and Model Successfully Saved

Out[48]:

(Pipeline(memory=Memory(location=None),

steps=[('numerical\_imputer',

TransformerWrapper(exclude=None,

include=['gender', 'seniorcitizen',

'partner', 'dependents', 'tenure',

'phoneservice', 'paperlessbilling',

'monthlycharges', 'totalcharges',

'multiplelines\_No',

'multiplelines\_No phone service',

'multiplelines\_Yes',

'internetservice\_DSL',

'internetservice\_Fiber optic'...

transformer=SimpleImputer(add\_indicator=False,

copy=True,

fill\_value=None,

keep\_empty\_features=False,

missing\_values=nan,

strategy='most\_frequent',

verbose='deprecated'))),

('clean\_column\_names',

TransformerWrapper(exclude=None, include=None,

transformer=CleanColumnNames(match='[\\]\\[\\,\\{\\}\\"\\:]+'))),

('trained\_model',

GaussianNB(priors=None, var\_smoothing=1e-09))],

verbose=False),

'churn-predict.pkl')

**Submitted by…**

**BALISETTY CHENCHU NITHIN**