Análise de churn - Definição do problema de negócio

In [1]:
```
# Big Data na Prática 4 - Customer Churn Analytics

# A rotatividade (churn) de clientes ocorre quando clientes ou assinantes param de f
# com uma empresa ou serviço. Também é conhecido como perda de clientes ou taxa de c

# Um setor no qual saber e prever as taxas de cancelamento é particularmente útil é
# porque a maioria dos clientes tem várias opções de escolha dentro de uma localizaç

# Neste projeto, vamos prever a rotatividade (churn) de clientes usando um conjunto
# Usaremos a regressão logística, a árvore de decisão e a floresta aleatória como mo

# Usaremos um dataset oferecido gratuitamente no portal IBM Sample Data Sets.
# Cada linha representa um cliente e cada coluna contém os atributos desse cliente.

# https://www.ibm.com/communities/analytics/watson-analytics-blog/guide-to-sample-da
```

In [2]:
```
import pycaret
import pandas as pd
```

In [3]:
```
import os
os.getcwd()
```

Out[3]:
```
'C:\\Users\\mayco\\caret\\Scripts'
```

Obter dados

In [4]:
```
data = pd.read_csv('Telco-Customer-Churn.csv')
```

In [5]:
```
data.head()
```

Out[5]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	I
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	
1	5575-GNVDE	Male	0	No	No	34	Yes	No	
2	3668-QPYBK	Male	0	No	No	2	Yes	No	
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	
4	9237-HQITU	Female	0	No	No	2	Yes	No	

5 rows × 21 columns

In [6]:
```
data.drop(['customerID'], axis=1, inplace = True)
```

In [7]:
```
data.head()
```

Out[7]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService
0	Female	0	Yes	No	1	No	No phone service	DSL
1	Male	0	No	No	34	Yes	No	DSL
2	Male	0	No	No	2	Yes	No	DSL
3	Male	0	No	No	45	No	No phone service	DSL
4	Female	0	No	No	2	Yes	No	Fiber optic

In [8]:
```
from pycaret.classification import *
```

Setup

In [9]:
```
clf1 = setup(data, target = 'Churn')
```

	Description	Value
0	session_id	8874
1	Target	Churn
2	Target Type	Binary
3	Label Encoded	No: 0, Yes: 1
4	Original Data	(7043, 20)
5	Missing Values	False
6	Numeric Features	2
7	Categorical Features	17
8	Ordinal Features	False
9	High Cardinality Features	False
10	High Cardinality Method	None
11	Transformed Train Set	(4930, 4691)
12	Transformed Test Set	(2113, 4691)
13	Shuffle Train-Test	True

	Description	Value
14	Stratify Train-Test	False
15	Fold Generator	StratifiedKFold
16	Fold Number	10
17	CPU Jobs	-1
18	Use GPU	False
19	Log Experiment	False
20	Experiment Name	clf-default-name
21	USI	7fc1
22	Imputation Type	simple
23	Iterative Imputation Iteration	None
24	Numeric Imputer	mean
25	Iterative Imputation Numeric Model	None
26	Categorical Imputer	constant
27	Iterative Imputation Categorical Model	None
28	Unknown Categoricals Handling	least_frequent
29	Normalize	False
30	Normalize Method	None
31	Transformation	False
32	Transformation Method	None
33	PCA	False
34	PCA Method	None
35	PCA Components	None
36	Ignore Low Variance	False
37	Combine Rare Levels	False
38	Rare Level Threshold	None
39	Numeric Binning	False
40	Remove Outliers	False
41	Outliers Threshold	None
42	Remove Multicollinearity	False
43	Multicollinearity Threshold	None
44	Remove Perfect Collinearity	True
45	Clustering	False
46	Clustering Iteration	None
47	Polynomial Features	False
48	Polynomial Degree	None
49	Trignometry Features	False

	Description	Value
50	Polynomial Threshold	None
51	Group Features	False
52	Feature Selection	False
53	Feature Selection Method	classic
54	Features Selection Threshold	None
55	Feature Interaction	False
56	Feature Ratio	False
57	Interaction Threshold	None
58	Fix Imbalance	False
59	Fix Imbalance Method	SMOTE

Compare Baseline

In [10]:
```
best_model = compare_models()
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
gbc	Gradient Boosting Classifier	0.7994	0.8421	0.4989	0.6630	0.5685	0.4413	0.4494	15.7250
lr	Logistic Regression	0.7963	0.8401	0.5226	0.6433	0.5759	0.4439	0.4486	31.2920
ada	Ada Boost Classifier	0.7957	0.8424	0.5195	0.6422	0.5738	0.4416	0.4462	4.7260
ridge	Ridge Classifier	0.7939	0.0000	0.5027	0.6425	0.5632	0.4312	0.4372	2.3230
lightgbm	Light Gradient Boosting Machine	0.7884	0.8265	0.5188	0.6210	0.5646	0.4265	0.4299	0.7860
rf	Random Forest Classifier	0.7850	0.8172	0.4529	0.6333	0.5266	0.3928	0.4028	5.1100
et	Extra Trees Classifier	0.7769	0.8050	0.4552	0.6057	0.5184	0.3774	0.3846	8.5330
xgboost	Extreme Gradient Boosting	0.7748	0.8120	0.4997	0.5893	0.5402	0.3926	0.3953	43.3720
svm	SVM - Linear Kernel	0.7635	0.0000	0.3072	0.3870	0.3229	0.2438	0.2589	2.6400
dt	Decision Tree Classifier	0.7586	0.6724	0.4866	0.5522	0.5159	0.3563	0.3585	0.6310
knn	K Neighbors Classifier	0.7548	0.7646	0.4858	0.5419	0.5121	0.3490	0.3501	5.5970
catboost	CatBoost Classifier	0.6410	0.6774	0.4008	0.5337	0.4571	0.3565	0.3630	12.1530
lda	Linear Discriminant Analysis	0.5491	0.5459	0.3664	0.3330	0.3480	0.1753	0.1761	71.5350
nb	Naive Bayes	0.2777	0.4895	0.9403	0.2608	0.4084	-0.0115	-0.0453	0.7290

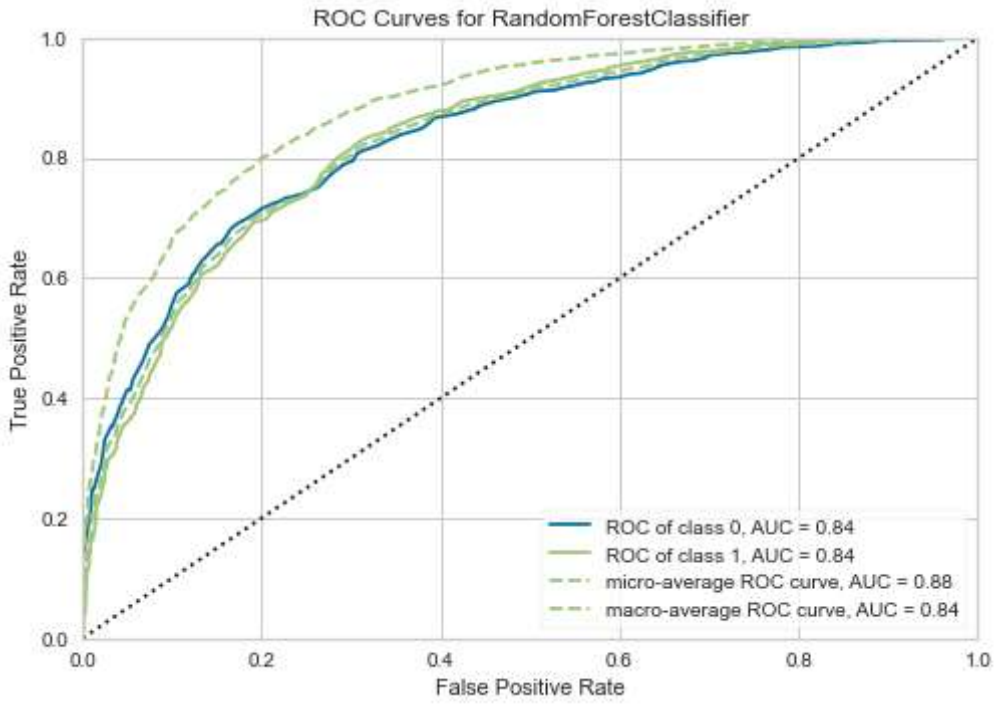
	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
qda	Quadratic Discriminant Analysis	0.2361	0.1582	0.7740	0.2165	0.3378	0.0107	0.0127	45.9770

In [11]:
```
model = create_model('rf')
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.7830	0.8209	0.4615	0.6186	0.5286	0.3915	0.3986
1	0.7972	0.8326	0.4154	0.6923	0.5192	0.4007	0.4217
2	0.7992	0.8441	0.4538	0.6782	0.5438	0.4215	0.4354
3	0.7688	0.8004	0.4351	0.5876	0.5000	0.3539	0.3607
4	0.7769	0.8175	0.4198	0.6180	0.5000	0.3631	0.3743
5	0.7911	0.8077	0.5115	0.6321	0.5654	0.4299	0.4341
6	0.8174	0.8338	0.5725	0.6881	0.6250	0.5057	0.5094
7	0.7586	0.7997	0.4427	0.5577	0.4936	0.3379	0.3418
8	0.7769	0.8036	0.4275	0.6154	0.5045	0.3665	0.3766
9	0.7809	0.8117	0.3893	0.6456	0.4857	0.3572	0.3756
Mean	0.7850	0.8172	0.4529	0.6333	0.5266	0.3928	0.4028
SD	0.0160	0.0146	0.0504	0.0415	0.0403	0.0473	0.0464

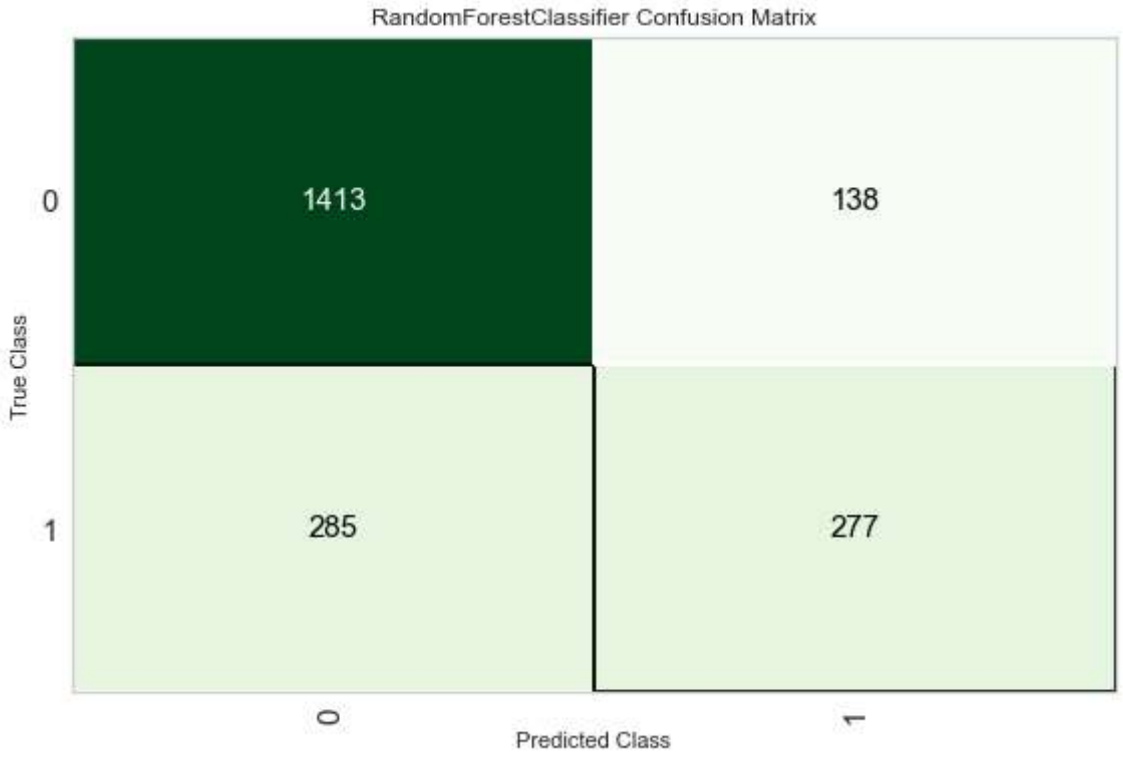
Analyze Model

In [12]:
```
plot_model(model)
```



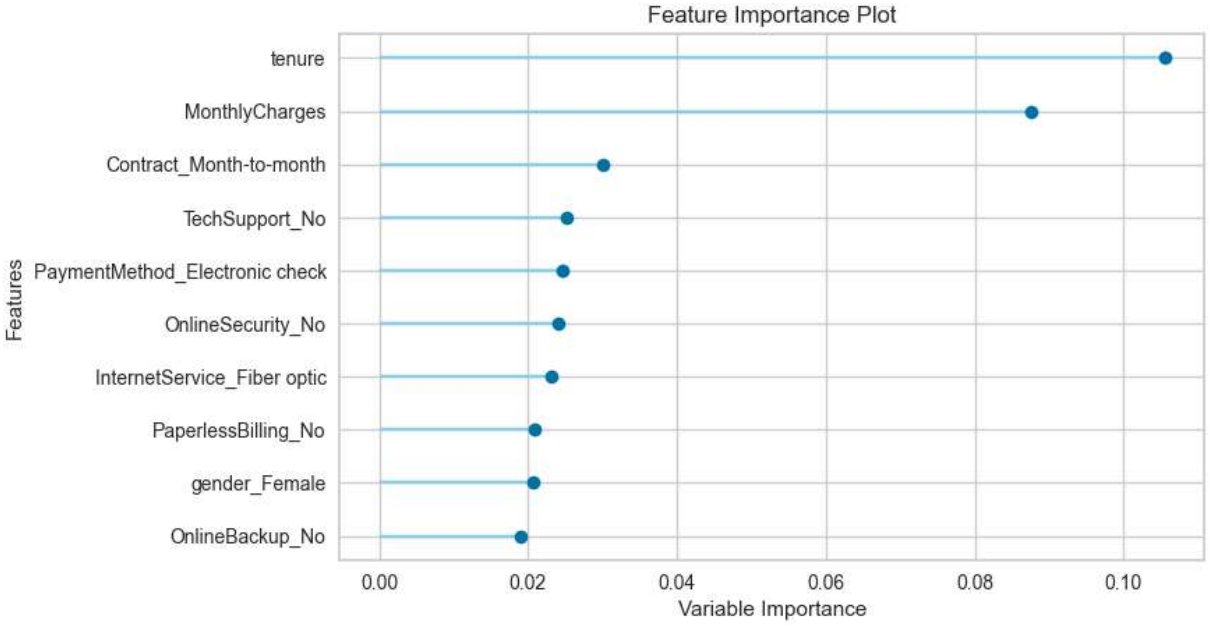
In [13]:

```
plot_model(model, plot = 'confusion_matrix')
```



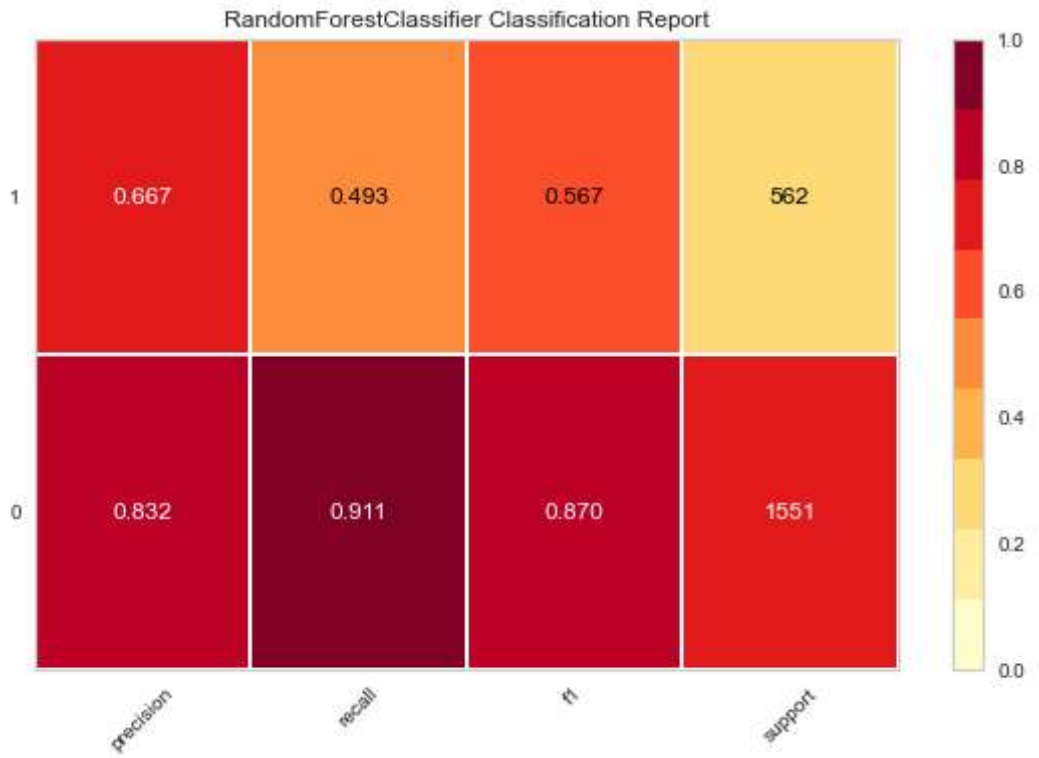
In [14]:

```
plot_model(model, plot = 'feature')
```



In [15]:

```
plot_model(model, plot = 'class_report')
```



In [ ]:
```
#evaluate_model(model)
```

In [18]:
```
#interpret_model(model)
```

In [ ]:
```

```