Lipigas

January 25, 2022

1 Desafío 1: Determinar probabilidad de churn

El objetivo del desafío es desarrollar un modelo que pueda predecir si un cliente dejará la compañía. Es muy importante justificar la elección del sistema (o modelo), el trabajo previo de la data (EDA) y la documentación de lo que se hizo (no es necesario un informe, pero si comentar porqué se tomaron las decisiones que se tomaron; por ej eliminar una variable o eliminar registros missing, etc).

La variable "Churn" tiene dos niveles, y como el objetivo es predecir si un cliente dejara la compañía, por lo tanto, es un problema de clasificación, en este caso podríamos utilizar una regresión logística binomial, cuyas variables predictoras serian las otras columnas del conjunto de datos, y la variable dependiente "Churn".

[1]:
```
#cargamos los datos
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
df = pd.read_csv(r"C:\Users\Cristopher\Desktop\Desafio_lipigas\Archivo data -␣
↪2022.csv", sep=",")
df.head()
```

[1]:
```
   customerID  gender  SeniorCitizen Partner Dependents  tenure PhoneService  \
0  7590-VHVEG  Female              0     Yes         No      30           No
1  5575-GNVDE    Male              0      No         No    1020          Yes
2  3668-QPYBK    Male              0      No         No      60          Yes
3  7795-CFOCW    Male              0      No         No    1350           No
4  9237-HQITU  Female              0      No         No      60          Yes

      MultipleLines InternetService OnlineSecurity  … DeviceProtection  \
0  No phone service             DSL             No  …                No
1                No             DSL            Yes  …               Yes
2                No             DSL            Yes  …                No
3  No phone service             DSL            Yes  …               Yes
4                No     Fiber optic             No  …                No

  TechSupport StreamingTV StreamingMovies        Contract PaperlessBilling  \
0          No          No              No  Month-to-month              Yes
```

```
1          No          No              No        One year              No
2          No          No              No  Month-to-month             Yes
3         Yes          No              No        One year              No
4          No          No              No  Month-to-month             Yes

               PaymentMethod MonthlyCharges  TotalCharges Churn
0           Electronic check          29.85         29.85    No
1               Mailed check          56.95        1889.5    No
2               Mailed check          53.85        108.15   Yes
3  Bank transfer (automatic)          42.30       1840.75    No
4           Electronic check          70.70        151.65   Yes

[5 rows x 21 columns]
```

Tenemos 7043 filas datos de clientes y 21 variables o columnas.

[270]:
```
df.shape
```

[270]:
```
(7043, 21)
```

Se calcula el porcentaje de hombres y mujeres que abandonaron, y no abandonan.

[271]:
```
tabla_contingencia = pd.crosstab(df["gender"], df["Churn"])
tabla_contingencia
```

[271]:
```
Churn     No  Yes
gender
Female  2549  939
Male    2625  930
```

[272]:
```
tabla_contingencia.astype("float").div(tabla_contingencia.sum(axis=1), axis=0)
```

[272]:
```
Churn         No       Yes
gender
Female  0.730791  0.269209
Male    0.738397  0.261603
```

No parece existir una gran diferencia entre los hombres y las mujeres respecto que abandonan.

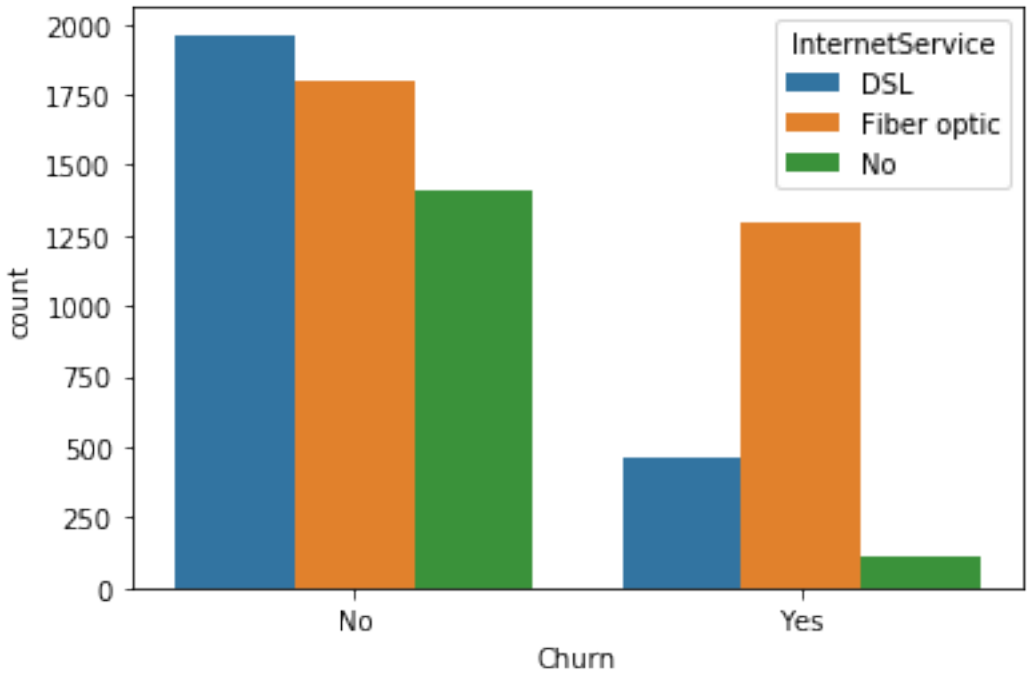
[273]:
```
tabla_contingencia.astype("float").div(tabla_contingencia.sum(axis=1), axis=0)
```

[273]:
```
Churn         No       Yes
gender
Female  0.730791  0.269209
Male    0.738397  0.261603
```

Se pueden realizar gráficos para explorar si existe alguna diferencia de abandono entre quienes utilizan DSL, Fiber Optic y “No”

[274]:
```
sns.countplot(x='Churn',data=df, hue='InternetService')
```

[274]: <matplotlib.axes._subplots.AxesSubplot at 0x220fae07ef0>

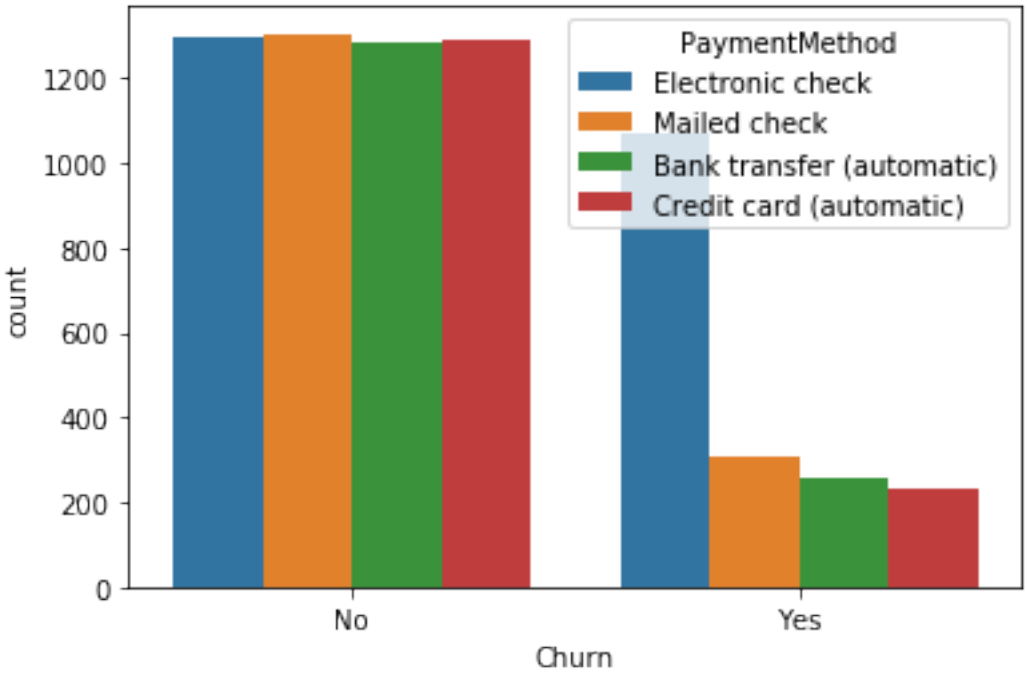


Entre los que abandonan los que utilizan “Fiber Optic” pareciera que tienen una mayor rotación.

Si quisiéramos observar si el tipo de pago tiene alguna influencia en “Churn”

[275]:
```
sns.countplot(x='Churn',data=df, hue='PaymentMethod')
```

[275]: <matplotlib.axes._subplots.AxesSubplot at 0x220fae6f320>

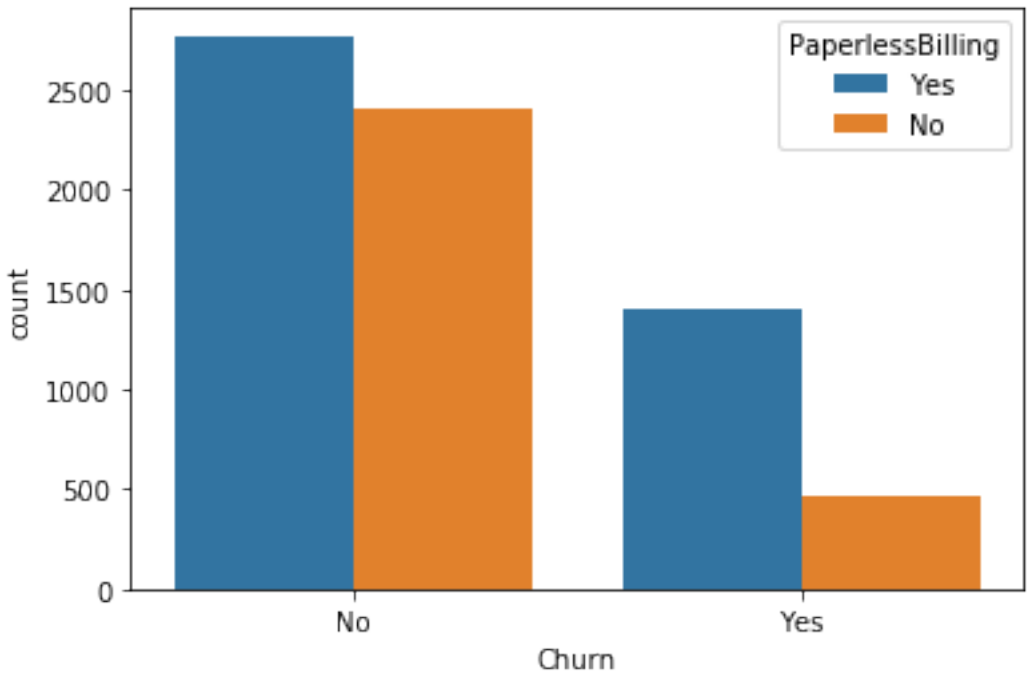


Los que tienen como método de pago "electronic check" tienden a estar sobrerepresentados entre los que abandonan.

Ahora comprobamos si el tener facturación electrónica está vinculado al abandono de la compañía

[276]:
```
sns.countplot(x='Churn',data=df, hue='PaperlessBilling')
```

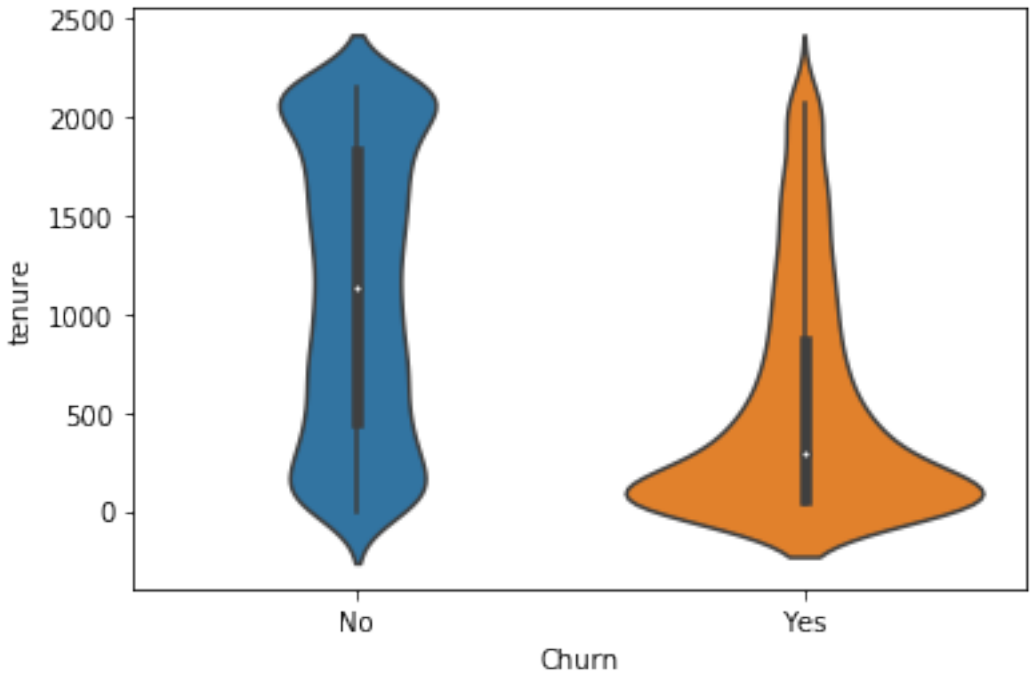
[276]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x220faee2438>
```



Entre los que abandonan, parecen tener una tendencia hacía poseer factura electrónica.

Si quisiéramos saber la relación entre la antigüedad del cliente en días y si el cliente ha dejado la compañía:

[277]:
```
import seaborn as sns

sns.violinplot(x="Churn", y="tenure", data=df);
```



Se observa que los de menor antigüedad tienden a dejar la compañía.

Si quisiéramos saber cuantos clientes abandonaron, primero procedemos a transformar las variables en variables dummies.

[278]:
```
df["Churn"] = (df.Churn=='Yes').astype(int)
df.head()
```

[278]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	…	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling
0	7590-VHVEG	Female	0	Yes	No	30	No	…	No	No	No	No	Month-to-month	Yes
1	5575-GNVDE	Male	0	No	No	1020	Yes	…	Yes	No	No	No	One year	No
2	3668-QPYBK	Male	0	No	No	60	Yes	…	No	No	No	No	Month-to-month	Yes
3	7795-CFOCW	Male	0	No	No	1350	No	…	Yes	No	No	No	One year	No
4	9237-HQITU	Female	0	No	No	60	Yes	…	No	No	No	No	Month-to-month	Yes

```
2          No          No               No  Month-to-month              Yes
3         Yes          No               No        One year               No
4          No          No               No  Month-to-month              Yes

               PaymentMethod MonthlyCharges  TotalCharges Churn
0           Electronic check          29.85         29.85     0
1               Mailed check          56.95        1889.5     0
2               Mailed check          53.85        108.15     1
3  Bank transfer (automatic)          42.30       1840.75     0
4           Electronic check          70.70        151.65     1

[5 rows x 21 columns]
```

[279]:
```
df["Churn"].value_counts()
```

[279]:
```
0    5174
1    1869
Name: Churn, dtype: int64
```

En total existen 1869 clientes que abandonaron la compañía.

Transformamos también la variable género, pre procesamos los datos y creamos una variable con las columnas numéricas y otra para las categorías, además se observa que existen columnas que tienen información redundante, por ejemplo; "No tener servicio de internet" es lo mismo que un "no", asi que se optara por reducir los niveles de las columnas que contengan información redundante, iteramos las "categorías" y reducimos la información.

[281]:
```
df["gender"] = (df.gender=='Male').astype(int)

numericas = list(df.select_dtypes(include=['int64','float64']).keys())


categorias = list(df.select_dtypes(include='O').keys())


for i in categorias:
    df[i].value_counts()

df.MultipleLines = df.MultipleLines.replace('No phone service','No')
df.OnlineSecurity = df.OnlineSecurity.replace('No internet service','No')
df.OnlineBackup = df.OnlineBackup.replace('No internet service','No')
df.DeviceProtection = df.DeviceProtection.replace('No internet service','No')
df.TechSupport = df.TechSupport.replace('No internet service','No')
df.StreamingTV = df.StreamingTV.replace('No internet service','No')
df.StreamingMovies = df.StreamingMovies.replace('No internet service','No')

for i in categorias:
```

```
    df[i] = df[i].replace('Yes',1)
    df[i] = df[i].replace('No',0)
df.head()
```

[281]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	\
0	7590-VHVEG	0	0	1	0	30	
1	5575-GNVDE	1	0	0	0	1020	
2	3668-QPYBK	1	0	0	0	60	
3	7795-CFOCW	1	0	0	0	1350	
4	9237-HQITU	0	0	0	0	60	

	PhoneService	MultipleLines	InternetService	OnlineSecurity	…	\
0	0	0	DSL	0	…	
1	1	0	DSL	1	…	
2	1	0	DSL	1	…	
3	0	0	DSL	1	…	
4	1	0	Fiber optic	0	…	

	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	\
0	0	0	0	0	
1	1	0	0	0	
2	0	0	0	0	
3	1	1	0	0	
4	0	0	0	0	

	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	\
0	Month-to-month	1	Electronic check	29.85	
1	One year	0	Mailed check	56.95	
2	Month-to-month	1	Mailed check	53.85	
3	One year	0	Bank transfer (automatic)	42.30	
4	Month-to-month	1	Electronic check	70.70	

	TotalCharges	Churn
0	29.85	0
1	1889.5	0
2	108.15	1
3	1840.75	0
4	151.65	1

[5 rows x 21 columns]

Tenemos tres variables “interservice”, “contract” y “paymentmethod” que son “object” las transformamos en“category” para posteriormente codificarlas en “labels”.

Eliminamos la variable customerID debido a que cada cliente se encuentra representado por cada fila, mantenerla sería tener una variable redundante.

Convertimos la variable “TotalCharges” en numérica porque es un objeto, nos da un error la conversión debido a la existencia de datos faltantes, así que procedemos a la eliminación de los

datos, ya que todos pertenecen a la misma categoría, y los datos eliminados representan una fracción mínima de los datos totales.

[282]:
```
df["InternetService"] = df["InternetService"].astype('category')
df["Contract"] = df["Contract"].astype('category')
df["PaymentMethod"] = df["PaymentMethod"].astype('category')

#Ahora, podemos convertir las variables en

df["InternetService"] = df["InternetService"].cat.codes
df["Contract"] = df["Contract"].cat.codes
df["PaymentMethod"] = df["PaymentMethod"].cat.codes

del df['customerID']

len(df[df.TotalCharges==' '])
df=df[df.TotalCharges!=' ']

df.TotalCharges=pd.to_numeric(df.TotalCharges)
np.dtype(df.TotalCharges)

df.head()
```

[282]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges	Churn
0	0	0	1	0	30	0	0	1	0	1	0	0	0	0	0	1	2	29.85	29.85	0
1	1	0	0	0	1020	1	0	1	1	0	1	0	0	0	1	0	3	56.95	1889.50	0
2	1	0	0	0	60	1	0	1	1	1	0	0	0	0	0	1	3	53.85	108.15	1
3	1	0	0	0	1350	0	0	1	1	0	1	1	0	0	1	0	0	42.30	1840.75	0
4	0	0	0	0	60	1	0	2	0	0	0	0	0	0	0	1	2	70.70	151.65	1

```
2                1              3           53.85        108.15       1
3                0              0           42.30       1840.75       0
4                1              2           70.70        151.65       1
```

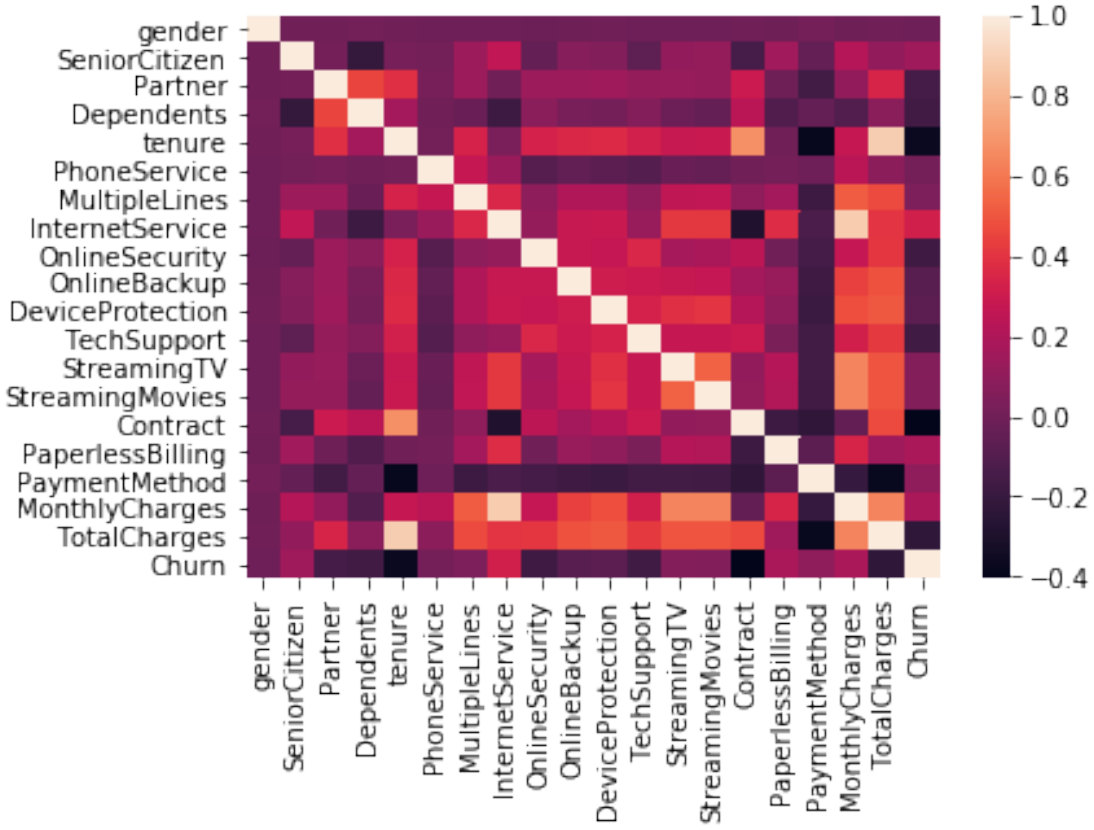
Finalmente, procedemos a normalizar los datos las columnas que sean variables continuas para que estén en escala y sean comparables a las otras variables. Escalamos los datos continuos para que no sean posibles outliers en el modelo.

[283]:
```
import sklearn
from sklearn.model_selection import train_test_split
scale_vars = ['tenure', 'MonthlyCharges','TotalCharges']
scaler = sklearn.preprocessing.StandardScaler()
df[scale_vars] = scaler.fit_transform(df[scale_vars])
df.head()
```

[283]:
```
   gender  SeniorCitizen  Partner  Dependents    tenure  PhoneService  \
0       0              0        1           0 -1.280248             0
1       1              0        0           0  0.064303             1
2       1              0        0           0 -1.239504             1
3       1              0        0           0  0.512486             0
4       0              0        0           0 -1.239504             1

   MultipleLines  InternetService  OnlineSecurity  OnlineBackup  \
0              0                1               0             1
1              0                1               1             0
2              0                1               1             1
3              0                1               1             0
4              0                2               0             0

   DeviceProtection  TechSupport  StreamingTV  StreamingMovies  Contract  \
0                 0            0            0                0         0
1                 1            0            0                0         1
2                 0            0            0                0         0
3                 1            1            0                0         1
4                 0            0            0                0         0

   PaperlessBilling  PaymentMethod  MonthlyCharges  TotalCharges  Churn
0                 1              2       -1.161694     -0.994194      0
1                 0              3       -0.260878     -0.173740      0
2                 1              3       -0.363923     -0.959649      1
3                 0              0       -0.747850     -0.195248      0
4                 1              2        0.196178     -0.940457      1
```

Revisamos la multicolinealidad, como tenemos variables categóricas debemos utilizar la correlación por rangos de Spearman, en este caso un cálculo de VIF no tendría sentido debido a las variables categóricas. Calculamos la multicolinealidad y hacemos un plot.

[287]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
corr = df.corr(method = 'spearman')

sns.heatmap(corr)

plt.show()
```



Tenemos multicolinealidad en algunas de las variables, así lo demuestra el plot, por lo tanto, debemos penalizar la regresión logística, en este caso la penalizaremos con "l1", esto hará que algunos coeficientes tiendan hacia 0, con esto se evitara el sobre ajuste.

[288]:
```
df_vars = df.columns.values.tolist()
Y = ['Churn']
X = [v for v in df_vars if v not in Y]
```

Seleccionamos la información para el modelo e inicialmente se utilizan todas las variables que son 19.

[289]:
```
from sklearn import datasets
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression


n = 19


lr = LogisticRegression(penalty='l1', solver='liblinear',␣
 ↪class_weight='balanced', max_iter=10000)
rfe = RFE(lr, n_features_to_select=19)
rfe = rfe.fit(df[X], df[Y].values.ravel())
rfe
```

[289]:
```
RFE(estimator=LogisticRegression(C=1.0, class_weight='balanced', dual=False,
                                 fit_intercept=True, intercept_scaling=1,
                                 l1_ratio=None, max_iter=10000,
                                 multi_class='auto', n_jobs=None, penalty='l1',
                                 random_state=None, solver='liblinear',
                                 tol=0.0001, verbose=0, warm_start=False),
    n_features_to_select=19, step=1, verbose=0)
```

Nos quedamos con las variables que aparezcan con “TRUE” y eliminamos las “FALSE”

[290]:
```
z=zip(df_vars,rfe.support_, rfe.ranking_)
list(z)
```

[290]:
```
[('gender', True, 1),
 ('SeniorCitizen', True, 1),
 ('Partner', True, 1),
 ('Dependents', True, 1),
 ('tenure', True, 1),
 ('PhoneService', True, 1),
 ('MultipleLines', True, 1),
 ('InternetService', True, 1),
 ('OnlineSecurity', True, 1),
 ('OnlineBackup', True, 1),
 ('DeviceProtection', True, 1),
 ('TechSupport', True, 1),
 ('StreamingTV', True, 1),
 ('StreamingMovies', True, 1),
 ('Contract', True, 1),
 ('PaperlessBilling', True, 1),
 ('PaymentMethod', True, 1),
 ('MonthlyCharges', True, 1),
 ('TotalCharges', True, 1)]
```

Seleccionamos las variables con las que nos quedaremos, en este caso nos quedamos con todas las

columnas, tampoco existe gran problema en un modelo robusto porque es un modelo penalizado.

[291]:
```
columnas =␣
↪['gender','SeniorCitizen','Partner','Dependents','tenure','PhoneService','MultipleLines','I
          ␣
↪'OnlineSecurity','OnlineBackup','DeviceProtection','TechSupport','StreamingTV','StreamingMo
          'PaperlessBilling','PaymentMethod','MonthlyCharges','TotalCharges']

X = df[columnas]
Y = df["Churn"]
```

Implementamos el modelo en Python con statsmodel.api

[124]:
```
import statsmodels.api as sm
logit_model = sm.Logit(Y, X)
result = logit_model.fit()
```

```
Optimization terminated successfully.
         Current function value: 0.416343
         Iterations 8
```

[292]:
```
result.summary2()
```

[292]:
```
<class 'statsmodels.iolib.summary2.Summary'>
"""
                          Results: Logit
=================================================================
Model:              Logit            Pseudo R-squared: 0.281
Dependent Variable: Churn            AIC:              5893.4460
Date:               2022-01-25 04:01 BIC:              6023.7523
No. Observations:   7032             Log-Likelihood:   -2927.7
Df Model:           18               LL-Null:          -4071.7
Df Residuals:       7013             LLR p-value:      0.0000
Converged:          1.0000           Scale:            1.0000
No. Iterations:     8.0000
-----------------------------------------------------------------
                 Coef.  Std.Err.    z     P>|z|   [0.025  0.975]
-----------------------------------------------------------------
gender           -0.0292   0.0645  -0.4522 0.6512 -0.1557  0.0973
SeniorCitizen     0.2360   0.0841   2.8059 0.0050  0.0712  0.4009
Partner           0.0144   0.0775   0.1856 0.8527 -0.1374  0.1662
Dependents       -0.1645   0.0894  -1.8408 0.0657 -0.3397  0.0107
tenure           -1.4448   0.1506  -9.5905 0.0000 -1.7401 -1.1495
PhoneService     -1.2602   0.1017 -12.3964 0.0000 -1.4595 -1.0610
MultipleLines     0.1117   0.0801   1.3935 0.1635 -0.0454  0.2687
InternetService   0.1064   0.0735   1.4474 0.1478 -0.0377  0.2505
OnlineSecurity   -0.5675   0.0819  -6.9274 0.0000 -0.7281 -0.4069
OnlineBackup     -0.3174   0.0755  -4.2051 0.0000 -0.4654 -0.1695
```

```
DeviceProtection -0.2061   0.0783  -2.6338 0.0084 -0.3595 -0.0527
TechSupport      -0.5519   0.0837  -6.5942 0.0000 -0.7159 -0.3878
StreamingTV      -0.0698   0.0826  -0.8453 0.3980 -0.2318  0.0921
StreamingMovies  -0.0639   0.0817  -0.7826 0.4339 -0.2241  0.0962
Contract         -0.7053   0.0770  -9.1555 0.0000 -0.8563 -0.5543
PaperlessBilling  0.3638   0.0741   4.9104 0.0000  0.2186  0.5090
PaymentMethod     0.0455   0.0353   1.2876 0.1979 -0.0237  0.1147
MonthlyCharges    0.8766   0.0828  10.5813 0.0000  0.7142  1.0389
TotalCharges      0.6884   0.1565   4.3978 0.0000  0.3816  0.9952
==============================================================

"""
```

EL modelo tiene un pseudo R2 de 0.281 que representa un buen ajuste del modelo, el índice de McFadden cuyo R2 se utiliza para evaluar el ajuste del modelo, en este caso debemos saber que McFadden (1977, p. 35) se refirió a que los valores que van desde 0.2 a 0.4 representan un buen ajuste del modelo, mientras que 1.0 implica un modelo perfecto.[1]

2 Validamos el modelo

Para validar el modelo, nos quedamos con 0.3 de test y trabajaremos con un threshold de "0.5".

[126]:
```
import sklearn
from sklearn import linear_model
from sklearn.model_selection import train_test_split

#from sklearn import linear_model
#from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size = 0.3,␣
↪random_state=12345)

lm = linear_model.LogisticRegression(penalty='l1', solver='liblinear')
lm.fit(X_train, Y_train)
```

[126]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l1',
                   random_state=None, solver='liblinear', tol=0.0001, verbose=0,
                   warm_start=False)
```

[127]:
```
probs = lm.predict_proba(X_test)
prediction = lm.predict(X_test)
prob = probs[:,1]
prob_df = pd.DataFrame(prob)
threshold = 0.5
prob_df["prediction"] = np.where(prob_df[0]>threshold, 1, 0)
#prob_df.head()
```

```
pd.crosstab(prob_df.prediction, columns="count")
```

[127]:
```
col_0       count
prediction
0            1644
1             466
```

Tenemos un 22% de clientes que son marcados como que abandonaran

[128]:
```
466/len(prob_df)*100
```

[128]:
```
22.085308056872037
```

Si queremos saber cuanto acierta el modelo, en este caso obtenemos que este modelo acierta en un 80% de los casos en sí un cliente abandonara o no.

[129]:
```
from sklearn import metrics
metrics.accuracy_score(Y_test, prediction)
```

[129]:
```
0.8061611374407583
```

Se realiza una validación cruzada para asegurar que nuestro modelo no tenga overfitting, y veremos que tan bien funciona el modelo o que tan bien discrimina en general.

[130]:
```
from sklearn.model_selection import cross_val_score
scores = cross_val_score(linear_model.LogisticRegression(penalty='l1',␣
 ↪solver='liblinear'), X, Y, scoring="accuracy", cv=10)
scores
```

[130]:
```
array([0.796875  , 0.80539773, 0.80227596, 0.83214794, 0.78236131,
       0.78662873, 0.80796586, 0.79943101, 0.80369844, 0.80227596])
```

Calculamos el promedio de la eficacia del modelo con 10 iteraciones obteniendo un 80%.

[131]:
```
scores.mean()
```

[131]:
```
0.8019057933531618
```

Una vez que se ha validado el modelo, para asegurarnos que clasifica correctamente entre categorías, hacemos una curva ROC.

[132]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0.3,␣
 ↪random_state=12345)
lm = linear_model.LogisticRegression(penalty='l1', solver='liblinear')
lm.fit(X_train, Y_train)
probs = lm.predict_proba(X_test)
prob=probs[:,1]
prob_df = pd.DataFrame(prob)
threshold = 0.50
prob_df["prediction"] = np.where(prob_df[0]>=threshold, 1, 0)
```

```
prob_df["actual"] = list(Y_test)
prob_df.head()
```

[132]:

	0	prediction	actual
0	0.135618	0	0
1	0.556369	1	1
2	0.044020	0	0
3	0.173034	0	0
4	0.118723	0	0

Calculamos la sensibilidad

[133]:
```
confusion_matrix = pd.crosstab(prob_df.prediction, prob_df.actual)
print(confusion_matrix)
TN=confusion_matrix[0][0]
TP=confusion_matrix[1][1]
FN=confusion_matrix[0][1]
FP=confusion_matrix[1][0]

sens = TP/(TP+FN)
sens
```

```
actual          0    1
prediction
0            1384  260
1             149  317
```

[133]:
```
0.6802575107296137
```

Obtenemos que el 68% de los clientes que el modelo predijo que abandonaría realmente abandonaron.

Se calcula la especificidad:

[134]:
```
espc_1 = TN/(TN+FP)
espc_1
```

[134]:
```
0.8418491484184915
```

Obtenemos una especificidad del 84%, esto significa que el % de clientes que se clasificaron como que "no" abandonarían y se clasificaron como tales.

Probamos la especificidad y la sensibilidad con varios "thresholds"

[135]:
```
thresholds = [0.04, 0.05, 0.07, 0.10, 0.12, 0.15, 0.18, 0.20, 0.25, 0.3, 0.4, 0.
↪5]
sensitivities = [1]
especifities_1 = [1]

for t in thresholds:
```

```
    prob_df["prediction"] = np.where(prob_df[0]>=t, 1, 0)
    prob_df["actual"] = list(Y_test)
    prob_df.head()

    confusion_matrix = pd.crosstab(prob_df.prediction, prob_df.actual)
    TN=confusion_matrix[0][0]
    TP=confusion_matrix[1][1]
    FP=confusion_matrix[0][1]
    FN=confusion_matrix[1][0]

    sens = TP/(TP+FN)
    sensitivities.append(sens)
    espc_1 = 1-TN/(TN+FP)
    especifities_1.append(espc_1)

sensitivities.append(0)
especifities_1.append(0)

print(sensitivities)
```

```
[1, 0.9809358752166378, 0.9740034662045061, 0.9549393414211439,
0.9341421143847487, 0.9272097053726169, 0.9081455805892548, 0.8908145580589255,
0.8752166377816292, 0.8232235701906413, 0.7677642980935875, 0.6689774696707106,
0.5493934142114385, 0]
```

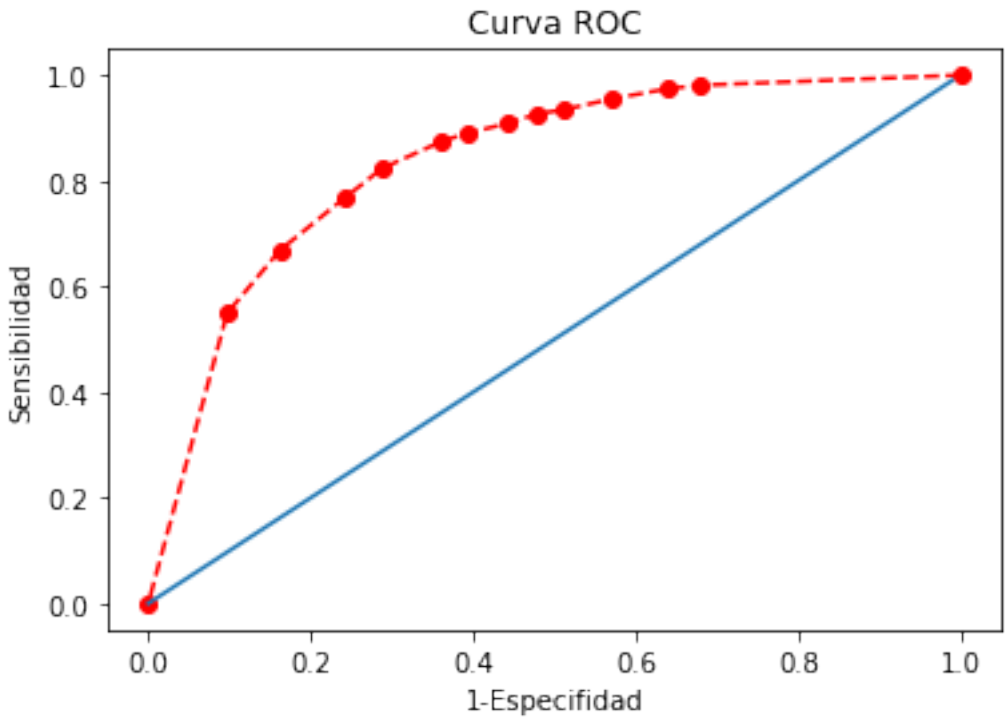
[136]:
```
print(especifities_1)
```

```
[1, 0.680365296803653, 0.6386170906718852, 0.5701239399869538,
0.5107632093933464, 0.4801043705153294, 0.4409654272667971, 0.3946510110893673,
0.3620352250489237, 0.28767123287671237, 0.24200913242009137,
0.16242661448140905, 0.0971950424005219, 0]
```

Comprobamos que tan bueno es el modelo y su capacidad para distinguir entre clases, para lo cual se utilizara la curva ROC y el Área bajo la curva.

[137]:
```
import matplotlib.pyplot as plt
%matplotlib inline
plt.plot(especifities_1, sensitivities, marker="o", linestyle="--", color="r")
x=[i*0.01 for i in range(100)]
y=[i*0.01 for i in range(100)]
plt.plot(x,y)
plt.xlabel("1-Especifidad")
plt.ylabel("Sensibilidad")
plt.title("Curva ROC")
```

[137]:
```
Text(0.5, 1.0, 'Curva ROC')
```



[138]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
auc = metrics.roc_auc_score(Y_test, prob)
print(auc)
```

```
0.8455583178168111
```

El área bajo la curva nos grafica la precisión del modelo, mientras más alto sea el AUC mejor será el modelo, en este caso obtuvimos 0.84, dicho número cae en el rango de "bueno".

Finalmente, calculamos las razones de probabilidad y los intervalos de confianza para cada variable.

[139]:
```
params = result.params
conf = result.conf_int()
conf['Odds Ratio'] = params
conf.columns = ['5%', '95%', 'Odds Ratio']
print(np.exp(conf))
```

```
                    5%       95%  Odds Ratio
gender        0.855827  1.102212    0.971238
SeniorCitizen 1.073755  1.493217    1.266234
Partner       0.871585  1.180807    1.014482
```

```
Dependents        0.711951  1.010709    0.848278
tenure            0.175510  0.316787    0.235795
PhoneService      0.232362  0.346123    0.283594
MultipleLines     0.955624  1.308282    1.118135
InternetService   0.963017  1.284656    1.112271
OnlineSecurity    0.482840  0.665684    0.566938
OnlineBackup      0.627914  0.844113    0.728032
DeviceProtection  0.698021  0.948635    0.813737
TechSupport       0.488756  0.678524    0.575875
StreamingTV       0.793102  1.096486    0.932537
StreamingMovies   0.799241  1.100982    0.938056
Contract          0.424725  0.574458    0.493950
PaperlessBilling  1.244329  1.663661    1.438799
PaymentMethod     0.976531  1.121557    1.046535
MonthlyCharges    2.042549  2.826191    2.402631
TotalCharges      1.464627  2.705268    1.990530
```

Obtenemos la fuerza de asociación con para de una de las variables en relación a la variable dependiente, se podría destacar "MonthlyCharges" que aumenta la probabilidad de abandonar de manera considerable.

3 El modelo final teniendo en consideración los p-value inferiores a 0.05, el modelo final seria:

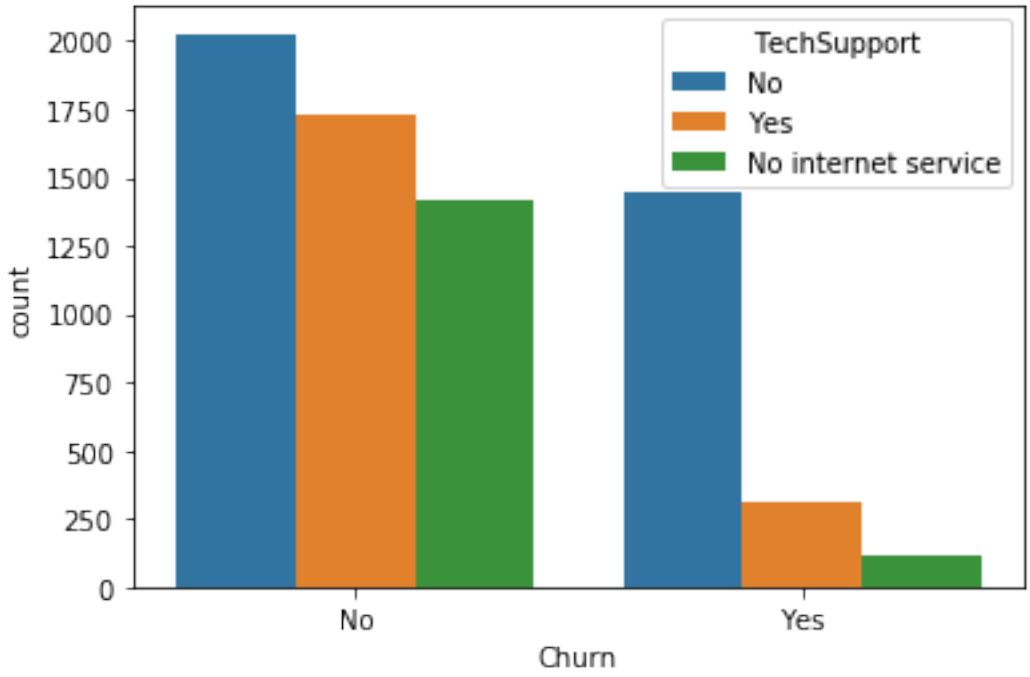
0.2378 * SeniorCitizen - 1.4426 * tenure - 1.2589 * PhoneService - 0.5670 * OnlineSecurity - 0.3172 * OnlineBackup - 0.2054 *DeviceProtection - 0.5517* TechSupport - 0.7054 * Contract + 0.3638 * PaperlessBilling + 0.8746 * MonthlyCharges + 0.6883 * TotalCharges

4 Desafío 2: Pregunta abierta para responder en base a tu conocimiento

Dado que las siguientes variables pueden ser controladas/manipuladas por el cliente, qué acciones podrían llevarse a cabo para minimizar la probabilidad de churn? TechSupport: el cliente posee soporte técnico (Yes/No/No internet service) Contract: vigencia del contrato con el cliente (month-to-month, one year, two year) MonthlyCharges: cantidad cobrada mensualmente al cliente

[210]:
```
sns.countplot(x='Churn',data=df, hue='TechSupport')
```

[210]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x220fa78c908>
```

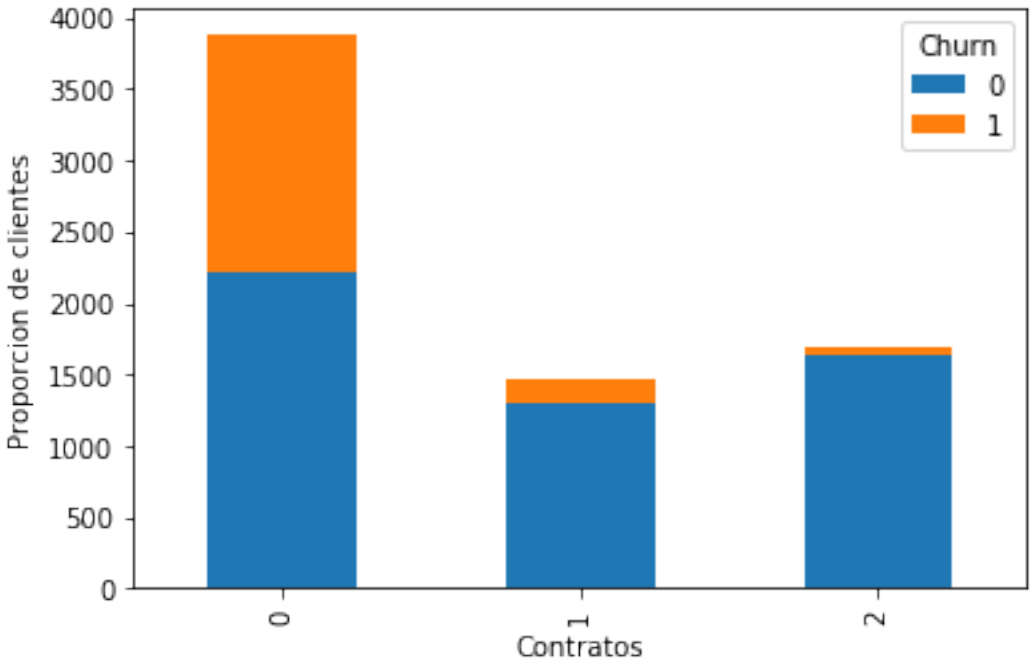


Según el modelo calculado en el apartado anterior y los Odds Ratio obtenemos que la variable TechSupport tiene una relación negativa con el término del vínculo con la compañía, el poseer un soporte en línea se asocia con una disminución en el riesgo de abandonar la compañía de un 42% aproximadamente, esto concuerda con algunos estudios que han vinculado una rápida ayuda en la resolución de problemas de los clientes con la fidelización hacia la compañía, teniendo cuenta datos provenientes del informe de tendencias de Zendesk que indican que el 80% de los clientes se iría hacia un competidor si obtiene malas experiencias[2]. Una manera eficaz de solucionar este problema es implementando varias posibles soluciones como crear una estrategia de gestión de contenidos donde el cliente pueda obtener información que le permita resolver problemas, ademas, se podrían implementar "bots" de atención inmediata que pudieran dar información sobre como resolver problemas comunes, los asistentes virtuales podrían resolver y ayudar a los clientes en tareas rápidas, sin embargo, debe existir personal capacitado que sea capaz de dar respuesta a los clientes una vez que los "bots" no pueden.

Para la variable Contract, mientras más corto sea el tipo de contrato, mayor probabilidad de abandonar.

[153]:
```
from matplotlib import pyplot as plt
import numpy as np

pd.crosstab(df['Contract'],df['Churn']).plot(kind="bar",stacked=True)
plt.xlabel("Contratos")
plt.ylabel("Proporcion de clientes")
```

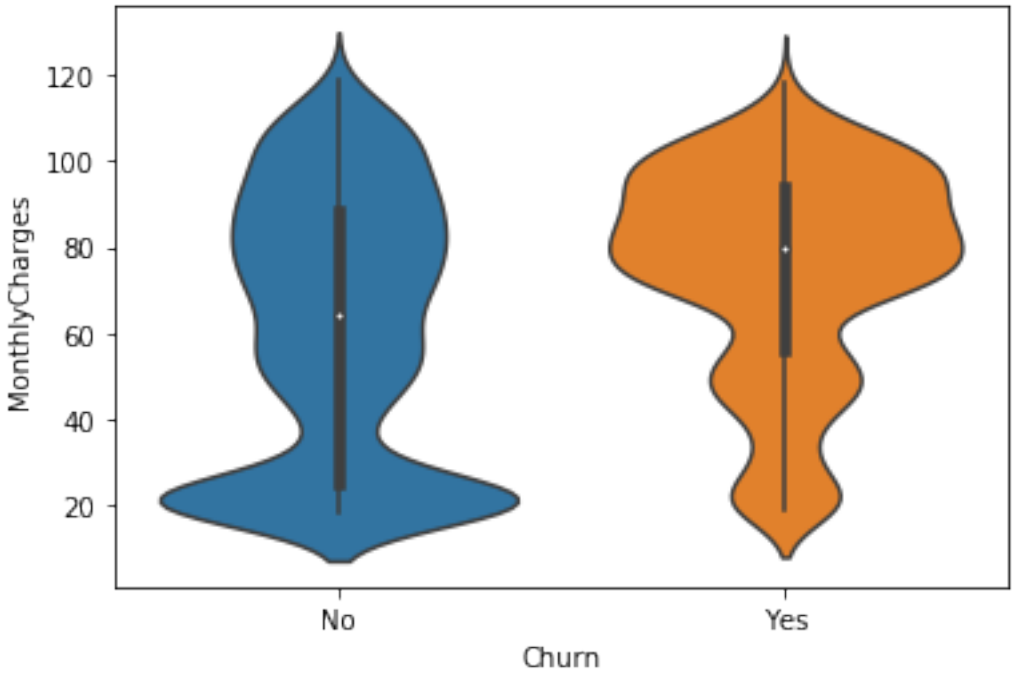
[153]:
```
Text(0, 0.5, 'Proporcion de clientes')
```



En este caso obtenemos que la probabilidad de abandonar cuando el cliente tiene contrato de 2 año o más se reduce, específicamente en un 50% a medida que el contrato es de 2 años, mientras que para el contrato de "mes a mes" la probabilidad de abandonar aumenta en un 48%.

Existe evidencia que el ofrecer duraciones mínimas de contrato ayuda a las empresas en la retención de clientes, si obtiene un alto abandono en clientes con duraciones mínimas de contrato debería enfocarse en dar incentivos extras en los primeros meses, ya que una vez que un cliente toma un servicio, el siguiente paso es convencerlo en que se comprometa con ese contrato, sin embargo, los incentivos son armas de doble filo debido a que muchos de los clientes que reciben los incentivos desertarían si los incentivos de otras empresas son mejores[3], en este caso habría que generar una política de incentivos monetarios enfocada principalmente en los clientes cuyo tipo de contrato es de "mes a mes".

[244]:
```
import seaborn as sns

sns.violinplot(x="Churn", y="MonthlyCharges", data=df);
```



Obtenemos que mientras más alto fue el monto a pagar mayor fue la probabilidad de abandonar la compañía, algo que es intuitivo, mientras más caro mayor abandono, si bien esto podría indicar que simplemente bajando los precios existirá menor rotación, esto puede ser engañoso, existe cierta evidencia que las políticas de precios solo influyen parcialmente en "churn", una reducción del precio muy severo, podría tener un efecto de atraer personas que potencialmente tiendan a abandonar, aumentando en periodos de tiempo la tasa de rotación. En este caso se podría hablar sobre rangos de precio, en este aspecto autores como Vastani y Monroe (2019)[4] exponen que los clientes aceptaran variaciones de un precio dentro de un rango, lo cual no tendrá ningún efecto en los comportamientos de los clientes, por lo tanto, los abandonos no se deben únicamente al precio de referencia puntual, sino porque el producto está fuera de un rango esperado. En este sentido sería adecuado establecer un rango de precios que actúe como indicador de la política de precios de la empresa.

5 Bibliografia

1. McFadden,D. (1974) "Conditional logit analysis of qualitative choice behavior
2. Zendesk's Customer Experience Trends Report (2020)
3. Jan U. Becker Implications of minimum contract durations on customer retention (2014)
4. Saloni Firasta Vastani, Kent Bourdon Monroe Role of customer attributes on absolute price thresholds (2019)