Telco Retencion Clientes

January 12, 2023

1 Telco Customer Churn

Los datos fueron tomados de https://www.kaggle.com/datasets/blastchar/telco-customer-churn?resource=download

- CustomerID: la identificación del cliente
- Gender: Masculino Femenino
- SeniorCitizen: si el cliente es una persona mayor (0/1)
- Partner: si vive con pareja (sí/no)
- Dependants: si tienen dependientes (sí/no)
- Tenure: número de meses desde el inicio del contrato
- PhoneService: si tienen servicio telefónico (sí/no)
- MultipleLines: si tienen varias líneas telefónicas (sí/no/no servicio telefónico)
- InternetService: el tipo de servicio de internet (no/fibra/óptica)
- OnlineSecurity: si la seguridad en línea está habilitada (si/no/no internet)
- OnlineBackup: si el servicio de copia de seguridad en línea está habilitado (si/no/no internet)
- DeviceProtection: si el servicio de protección de dispositivos está habilitado (si/no/no internet)
- TechSupport: si el cliente tiene soporte técnico (si/no/no internet)
- StreamingTV: si el servicio de streaming de TV está habilitado (si/no/no internet)
- StreamingMovies: si el servicio de streaming de películas está habilitado (si/no/no internet)
- Contract: el tipo de contrato (mensual/anual/bianual)
- PaperlessBilling: si la facturación es sin papel (sí/no)
- PaymentMethod: método de pago (cheque electrónico, cheque enviado por correo, transferencia bancaria, tarjeta de crédito)
- MonthlyCharges: el monto cobrado mensualmente (numérico)
- TotalCharges: el monto total cobrado (numérico)
- Churn: si el cliente ha cancelado el contrato (sí/no)

[1]:
```
import pandas as pd
import numpy as np

import seaborn as sns
from matplotlib import pyplot as plt
%matplotlib inline
```

1.1 1-Importando los datos

[3]:
```
df = pd.read_csv('../data/TelcoCustomerChurn.csv')
df.head()
```

[3]:
```
   customerID  gender  SeniorCitizen Partner Dependents  tenure PhoneService  \
0  7590-VHVEG  Female              0     Yes         No       1           No
1  5575-GNVDE    Male              0      No         No      34          Yes
2  3668-QPYBK    Male              0      No         No       2          Yes
3  7795-CFOCW    Male              0      No         No      45           No
4  9237-HQITU  Female              0      No         No       2          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]
```

[4]:
```
df.shape
```

[4]:
```
(7043, 21)
```

1.1.1 1.1.-Cambio los datos de manera vertical para explorarlos mucho mejor

[5]:
```
df.head().T
```

[5]:
```
                              0           1           2  \
customerID           7590-VHVEG  5575-GNVDE  3668-QPYBK
gender                   Female        Male        Male
SeniorCitizen                 0           0           0
Partner                     Yes          No          No
```

```
Dependents                     No            No              No
tenure                          1            34               2
PhoneService                   No           Yes             Yes
MultipleLines     No phone service            No              No
InternetService                DSL           DSL             DSL
OnlineSecurity                  No           Yes             Yes
OnlineBackup                   Yes            No             Yes
DeviceProtection                No           Yes              No
TechSupport                     No            No              No
StreamingTV                     No            No              No
StreamingMovies                 No            No              No
Contract           Month-to-month      One year  Month-to-month
PaperlessBilling               Yes            No             Yes
PaymentMethod     Electronic check  Mailed check    Mailed check
MonthlyCharges               29.85         56.95           53.85
TotalCharges                 29.85        1889.5          108.15
Churn                          No            No             Yes

                                  3               4
customerID                7795-CFOCW      9237-HQITU
gender                          Male          Female
SeniorCitizen                      0               0
Partner                           No              No
Dependents                        No              No
tenure                            45               2
PhoneService                      No             Yes
MultipleLines       No phone service              No
InternetService                  DSL     Fiber optic
OnlineSecurity                   Yes              No
OnlineBackup                      No              No
DeviceProtection                 Yes              No
TechSupport                      Yes              No
StreamingTV                       No              No
StreamingMovies                   No              No
Contract                  One year    Month-to-month
PaperlessBilling                  No             Yes
PaymentMethod     Bank transfer (automatic)  Electronic check
MonthlyCharges                 42.3            70.7
TotalCharges                1840.75          151.65
Churn                             No             Yes
```

1.1.2 1.2.-Verificando los tipos

[6]:
```
df.dtypes
```

[6]:
```
customerID          object
gender              object
```

```
SeniorCitizen         int64
Partner              object
Dependents           object
tenure                int64
PhoneService         object
MultipleLines        object
InternetService      object
OnlineSecurity       object
OnlineBackup         object
DeviceProtection     object
TechSupport          object
StreamingTV          object
StreamingMovies      object
Contract             object
PaperlessBilling     object
PaymentMethod        object
MonthlyCharges      float64
TotalCharges         object
Churn                object
dtype: object
```

1.2 2.- Explorando los datos

Para este apartado generé una clase ayudante que se llama Explorator, para utilizarla es necesario agregar la ruta del proyecto hacia la lista de búsqueda de python.

[7]:
```
import os
import sys
module_path = os.path.abspath(os.path.join('..'))
if module_path not in sys.path:
    sys.path.append(module_path)
```

[8]:
```
from helpers import Explorator
```

[9]:
```
df_expl = Explorator(df)
```

[10]:
```
df_expl.totals()
```

[10]:

	variable	qty_nan	perc_nan	qty_zeros	perc_zeros	unique	\
0	customerID	0	0.0	0	0.000000	7043	
1	gender	0	0.0	0	0.000000	2	
2	SeniorCitizen	0	0.0	5901	0.837853	2	
3	Partner	0	0.0	0	0.000000	2	
4	Dependents	0	0.0	0	0.000000	2	
5	tenure	0	0.0	11	0.001562	73	
6	PhoneService	0	0.0	0	0.000000	2	
7	MultipleLines	0	0.0	0	0.000000	3	
8	InternetService	0	0.0	0	0.000000	3	

```
9     OnlineSecurity        0       0.0          0    0.000000       3
10      OnlineBackup        0       0.0          0    0.000000       3
11  DeviceProtection        0       0.0          0    0.000000       3
12       TechSupport        0       0.0          0    0.000000       3
13       StreamingTV        0       0.0          0    0.000000       3
14   StreamingMovies        0       0.0          0    0.000000       3
15          Contract        0       0.0          0    0.000000       3
16  PaperlessBilling        0       0.0          0    0.000000       2
17     PaymentMethod        0       0.0          0    0.000000       4
18    MonthlyCharges        0       0.0          0    0.000000    1585
19      TotalCharges        0       0.0          0    0.000000    6531
20             Churn        0       0.0          0    0.000000       2
```

```
       type
0    object
1    object
2     int64
3    object
4    object
5     int64
6    object
7    object
8    object
9    object
10   object
11   object
12   object
13   object
14   object
15   object
16   object
17   object
18  float64
19   object
20   object
```

1.2.1 2.1 Transformado los datos para que tengan consistencia

Hacemos una copia para evitar perder los datos originales

[11]:
```
df_work = df.copy()
```

TotalCharges esta como string y es numerico

[12]:
```
df_work['TotalCharges'] = pd.to_numeric(df_work['TotalCharges'],␣
 ↪errors="coerce")
df_work['TotalCharges'] = df_work['TotalCharges'].fillna(0)
```

1.2.2 2.2 Haciendo consistente el nombre de las columnas

[13]:
```
from helpers import convertir_a_snake_case
```

[14]:
```
def test_convertir_a_snake_case():
    assert convertir_a_snake_case("EsteEsUnEjemplo") == "este_es_un_ejemplo"
    assert convertir_a_snake_case("OtroEjemplo") == "otro_ejemplo"
    assert convertir_a_snake_case("Ejemplo") == "ejemplo"
    assert convertir_a_snake_case("ejemplo") == "ejemplo"
    assert convertir_a_snake_case("EJEMPLO") == "ejemplo"


test_convertir_a_snake_case()
```

[15]:
```
df_work.columns = df_work.columns.to_series().apply(lambda s:␣
 ↪convertir_a_snake_case(s))
df_work.columns
```

[15]:
```
Index(['customer_id', 'gender', 'senior_citizen', 'partner', 'dependents',
       'tenure', 'phone_service', 'multiple_lines', 'internet_service',
       'online_security', 'online_backup', 'device_protection', 'tech_support',
       'streaming_tv', 'streaming_movies', 'contract', 'paperless_billing',
       'payment_method', 'monthly_charges', 'total_charges', 'churn'],
      dtype='object')
```

1.2.3 2.3 Localizando variables categoricas y haciendo consistentes los valores

[16]:
```
df_work_expl = Explorator(df_work)
categoricas = df_work_expl.categorical_vars()

for col in categoricas:
    df_work[col] = df_work[col].str.lower().str.replace(' ', '_')

df_work.head(n=10)
```

[16]:

	customer_id	gender	senior_citizen	partner	dependents	tenure	\
0	7590-vhveg	female	0	yes	no	1	
1	5575-gnvde	male	0	no	no	34	
2	3668-qpybk	male	0	no	no	2	
3	7795-cfocw	male	0	no	no	45	
4	9237-hqitu	female	0	no	no	2	
5	9305-cdskc	female	0	no	no	8	
6	1452-kiovk	male	0	no	yes	22	
7	6713-okomc	female	0	no	no	10	
8	7892-pookp	female	0	yes	no	28	
9	6388-tabgu	male	0	no	yes	62	

phone_service	multiple_lines	internet_service	online_security	…	\

```
0            no  no_phone_service             dsl             no  …
1           yes                no             dsl            yes  …
2           yes                no             dsl            yes  …
3            no  no_phone_service             dsl            yes  …
4           yes                no     fiber_optic             no  …
5           yes               yes     fiber_optic             no  …
6           yes               yes     fiber_optic             no  …
7            no  no_phone_service             dsl            yes  …
8           yes               yes     fiber_optic             no  …
9           yes                no             dsl            yes  …

  device_protection tech_support streaming_tv streaming_movies  \
0                no           no           no               no
1               yes           no           no               no
2                no           no           no               no
3               yes          yes           no               no
4                no           no           no               no
5               yes           no          yes              yes
6                no           no          yes               no
7                no           no           no               no
8               yes          yes          yes              yes
9                no           no           no               no

        contract paperless_billing             payment_method  \
0  month-to-month               yes           electronic_check
1        one_year                no               mailed_check
2  month-to-month               yes               mailed_check
3        one_year                no  bank_transfer_(automatic)
4  month-to-month               yes           electronic_check
5  month-to-month               yes           electronic_check
6  month-to-month               yes    credit_card_(automatic)
7  month-to-month                no               mailed_check
8  month-to-month               yes           electronic_check
9        one_year                no  bank_transfer_(automatic)

  monthly_charges  total_charges  churn
0           29.85          29.85     no
1           56.95        1889.50     no
2           53.85         108.15    yes
3           42.30        1840.75     no
4           70.70         151.65    yes
5           99.65         820.50    yes
6           89.10        1949.40     no
7           29.75         301.90     no
8          104.80        3046.05    yes
9           56.15        3487.95     no
```

[10 rows x 21 columns]

[17]:
```
df_work_expl.frequency()
```

```
     customer_id  frequency  percentage  cumulative_perc
0     7590-vhveg          1    0.000142         0.000142
1     3791-lgqcy          1    0.000142         0.000284
2     6008-naixk          1    0.000142         0.000426
3     5956-yhhrx          1    0.000142         0.000568
4     5365-llfyv          1    0.000142         0.000710
…          …        …          …               …
7038  9796-mvyxx          1    0.000142         0.999432
7039  2637-fkfsy          1    0.000142         0.999574
7040  1552-aagrx          1    0.000142         0.999716
7041  4304-tspvk          1    0.000142         0.999858
7042  3186-ajiek          1    0.000142         1.000000

[7043 rows x 4 columns]

------------------------------------------------------------------

   gender  frequency  percentage  cumulative_perc
0    male       3555    0.504756         0.504756
1  female       3488    0.495244         1.000000

------------------------------------------------------------------

  partner  frequency  percentage  cumulative_perc
0      no       3641    0.516967         0.516967
1     yes       3402    0.483033         1.000000

------------------------------------------------------------------

  dependents  frequency  percentage  cumulative_perc
0         no       4933    0.700412         0.700412
1        yes       2110    0.299588         1.000000

------------------------------------------------------------------

  phone_service  frequency  percentage  cumulative_perc
0           yes       6361    0.903166         0.903166
1            no        682    0.096834         1.000000

------------------------------------------------------------------

     multiple_lines  frequency  percentage  cumulative_perc
0                no       3390    0.481329         0.481329
1               yes       2971    0.421837         0.903166
```

```
2  no_phone_service        682    0.096834         1.000000

------------------------------------------------------------------

  internet_service  frequency  percentage  cumulative_perc
0      fiber_optic       3096    0.439585         0.439585
1              dsl       2421    0.343746         0.783331
2               no       1526    0.216669         1.000000

------------------------------------------------------------------

       online_security  frequency  percentage  cumulative_perc
0                   no       3498    0.496663         0.496663
1                  yes       2019    0.286668         0.783331
2  no_internet_service       1526    0.216669         1.000000

------------------------------------------------------------------

         online_backup  frequency  percentage  cumulative_perc
0                   no       3088    0.438450         0.438450
1                  yes       2429    0.344881         0.783331
2  no_internet_service       1526    0.216669         1.000000

------------------------------------------------------------------

     device_protection  frequency  percentage  cumulative_perc
0                   no       3095    0.439443         0.439443
1                  yes       2422    0.343888         0.783331
2  no_internet_service       1526    0.216669         1.000000

------------------------------------------------------------------

          tech_support  frequency  percentage  cumulative_perc
0                   no       3473    0.493114         0.493114
1                  yes       2044    0.290217         0.783331
2  no_internet_service       1526    0.216669         1.000000

------------------------------------------------------------------

          streaming_tv  frequency  percentage  cumulative_perc
0                   no       2810    0.398978         0.398978
1                  yes       2707    0.384353         0.783331
2  no_internet_service       1526    0.216669         1.000000

------------------------------------------------------------------

      streaming_movies  frequency  percentage  cumulative_perc
0                   no       2785    0.395428         0.395428
```

```
1                  yes       2732    0.387903         0.783331
2  no_internet_service       1526    0.216669         1.000000

----------------------------------------------------------------

         contract  frequency  percentage  cumulative_perc
0  month-to-month       3875    0.550192         0.550192
1        two_year       1695    0.240664         0.790856
2        one_year       1473    0.209144         1.000000

----------------------------------------------------------------

  paperless_billing  frequency  percentage  cumulative_perc
0               yes       4171    0.592219         0.592219
1                no       2872    0.407781         1.000000

----------------------------------------------------------------

              payment_method  frequency  percentage  cumulative_perc
0           electronic_check       2365    0.335794         0.335794
1               mailed_check       1612    0.228880         0.564674
2  bank_transfer_(automatic)       1544    0.219225         0.783899
3    credit_card_(automatic)       1522    0.216101         1.000000

----------------------------------------------------------------

  churn  frequency  percentage  cumulative_perc
0    no       5174     0.73463          0.73463
1   yes       1869     0.26537          1.00000

----------------------------------------------------------------
```

1.2.4 2.4 Cambiando la variable Churn de Categorica a booleana

[18]:
```
df_work['churn'] = (df_work['churn'] == 'yes').astype(int)
```

[19]:
```
df_work.head()[['customer_id','churn']]
```

[19]:
```
   customer_id  churn
0  7590-vhveg      0
1  5575-gnvde      0
2  3668-qpybk      1
3  7795-cfocw      0
4  9237-hqitu      1
```

1.2.5 2.5 Creando un pandas profiling

[20]:
```
from pandas_profiling import ProfileReport

profile = ProfileReport(df_work, title="Telco Retención Clientes")
profile
```

```
Summarize dataset:   0%|          | 0/5 [00:00<?, ?it/s]

Generate report structure:   0%|          | 0/1 [00:00<?, ?it/s]

Render HTML:   0%|          | 0/1 [00:00<?, ?it/s]

<IPython.core.display.HTML object>
```

[20]:

1.2.6 2.6 Guardando el profiling como html

[21]:
```
profile.to_file("../reports/telco_customer_profile.html")
```

```
Export report to file:   0%|          | 0/1 [00:00<?, ?it/s]
```

1.3 3.0.- Dividir el dataset para entrenar el modelo

[22]:
```
from sklearn.model_selection import train_test_split
```

[23]:
```
df_train_full, df_test = train_test_split(df_work, test_size=0.2,␣
 ↪random_state=1)
```

1.3.1 3.1 Dividiendo el dataset de entrenamiento para realizar validaciones

[24]:
```
df_train, df_val = train_test_split(df_train_full, test_size=0.33,␣
 ↪random_state=11)

y_train = df_train.churn.values
y_val = df_val.churn.values

del df_train['churn']
del df_val['churn']
```

[25]:
```
df_train.head()
```

[25]:

	customer_id	gender	senior_citizen	partner	dependents	tenure	\
2935	9435-jmlsx	male	0	yes	no	71	
3639	0512-flfdw	female	1	yes	no	60	
2356	3450-wxoat	male	0	no	no	46	
6660	1447-giqmr	male	0	yes	no	1	
755	6683-vlctz	male	1	no	no	20	

```
     phone_service multiple_lines internet_service online_security  \
2935           yes             no              dsl             yes
3639           yes            yes      fiber_optic              no
2356           yes             no              dsl              no
6660           yes             no      fiber_optic              no
755            yes            yes      fiber_optic              no

     online_backup device_protection tech_support streaming_tv  \
2935           yes               yes          yes          yes
3639            no               yes           no          yes
2356            no                no           no           no
6660            no                no           no           no
755             no               yes           no          yes

     streaming_movies        contract paperless_billing  \
2935              yes        two_year               yes
3639              yes        one_year               yes
2356               no  month-to-month               yes
6660               no  month-to-month               yes
755               yes  month-to-month               yes

                payment_method  monthly_charges  total_charges
2935  bank_transfer_(automatic)            86.10        6045.90
3639    credit_card_(automatic)           100.50        6029.00
2356    credit_card_(automatic)            45.20        2065.15
6660           electronic_check            69.15          69.15
755            electronic_check            98.55        1842.80
```

Creo un explorador y creo la lista de variables categoricas

[26]:
```
expl_train = Explorator(df_train_full)
categorical = expl_train.categorical_vars(exclude_var=['customer_id'])
expl_train.totals()
```

[26]:
```
             variable  qty_nan  perc_nan  qty_zeros  perc_zeros  unique  \
0         customer_id        0       0.0          0    0.000000    5634
1              gender        0       0.0          0    0.000000       2
2      senior_citizen        0       0.0       4722    0.838126       2
3             partner        0       0.0          0    0.000000       2
4          dependents        0       0.0          0    0.000000       2
5              tenure        0       0.0          8    0.001420      73
6       phone_service        0       0.0          0    0.000000       2
7      multiple_lines        0       0.0          0    0.000000       3
8    internet_service        0       0.0          0    0.000000       3
9     online_security        0       0.0          0    0.000000       3
10      online_backup        0       0.0          0    0.000000       3
11  device_protection        0       0.0          0    0.000000       3
12       tech_support        0       0.0          0    0.000000       3
```

```
13       streaming_tv        0       0.0          0    0.000000       3
14   streaming_movies        0       0.0          0    0.000000       3
15           contract        0       0.0          0    0.000000       3
16  paperless_billing        0       0.0          0    0.000000       2
17     payment_method        0       0.0          0    0.000000       4
18    monthly_charges        0       0.0          0    0.000000    1494
19      total_charges        0       0.0          8    0.001420    5291
20              churn        0       0.0       4113    0.730032       2
```

```
       type
0    object
1    object
2     int64
3    object
4    object
5     int64
6    object
7    object
8    object
9    object
10   object
11   object
12   object
13   object
14   object
15   object
16   object
17   object
18  float64
19  float64
20    int64
```

1.3.2 3.2.-Medir las tasas de riesgo de abandono por feature

Verificamos el promedio de abandono global y despues lo comparamos contra cada feature, verificamos si supera el promedio y listamos los valores que son mas probables que detonen una desercion de un cliente, esto lo usaremos para modelar casos hipoteticos mas adelante

[27]:
```
global_churn_mean = df_train_full['churn'].mean()
global_churn_mean
```

[27]:
```
0.26996805111821087
```

[28]:
```
list_best_churn = []

for col in categorical:
    df_group = df_train_full.groupby(by=col).churn.agg(['mean'])
    df_group['diff'] = df_group['mean'] - global_churn_mean
```

```
    df_group['rate'] = df_group['mean'] / global_churn_mean
    df_great_rate = df_group[df_group['rate'] > 1]

    for best_rate in df_great_rate.index:
        list_best_churn.append([col,best_rate,df_group.loc[best_rate,['rate']].
 ↪values[0]])


best_churn_features = pd.DataFrame(list_best_churn,␣
 ↪columns=['feature','value','rate'])
```

[29]:
```
best_churn_features
```

[29]:

	feature	value	rate
0	gender	female	1.025396
1	partner	no	1.221659
2	dependents	no	1.162212
3	phone_service	yes	1.011412
4	multiple_lines	yes	1.076948
5	internet_service	fiber_optic	1.574895
6	online_security	no	1.559152
7	online_backup	no	1.497672
8	device_protection	no	1.466379
9	tech_support	no	1.551717
10	streaming_tv	no	1.269897
11	streaming_tv	yes	1.121328
12	streaming_movies	no	1.255358
13	streaming_movies	yes	1.138182
14	contract	month-to-month	1.599082
15	paperless_billing	yes	1.252560
16	payment_method	electronic_check	1.688682

1.3.3 3.3.-Medir informacion mutua por feature para determinar si nos sera util y no para predecir el churn

[30]:
```
from sklearn.metrics import mutual_info_score

def calculate_mutual_info(series):
    return mutual_info_score(series, df_train_full.churn)

df_mi = df_train_full[categorical].apply(calculate_mutual_info)
df_mi = df_mi.sort_values(ascending=False).to_frame(name='MUTUAL_INFO')
df_mi
```

[30]:

	MUTUAL_INFO
contract	0.098320
online_security	0.063085

```
tech_support          0.061032
internet_service      0.055868
online_backup         0.046923
device_protection     0.043453
payment_method        0.043210
streaming_tv          0.031853
streaming_movies      0.031581
paperless_billing     0.017589
dependents            0.012346
partner               0.009968
multiple_lines        0.000857
phone_service         0.000229
gender                0.000117
```

Esta información indica que "contract" es el feature mas revelante para predecir "churn" y "gender" es el menos relevante.

1.3.4 3.4.-Verificando correlacion entre variables numericas

[31]:
```
numerical = expl_train.numerical_vars(exclude_var=['churn'])
df_train_full[numerical].corrwith(df_train_full.churn)
```

[31]:
```
senior_citizen     0.141966
tenure            -0.351885
monthly_charges    0.196805
total_charges     -0.196353
dtype: float64
```

Se concluye que entre mas tiempo (ternure) este el cliente y pague mas (total_charges) es mas dificil que nos abandone. Mientras que si su pago mensual es mas grande es probable que nos abandone.

1.3.5 3.5.- One-Hot encoding

[32]:
```
cols = categorical.to_list() + numerical.to_list()
train_dict = df_train[cols].to_dict(orient='records')
train_dict[0]
```

[32]:
```
{'gender': 'male',
 'partner': 'yes',
 'dependents': 'no',
 'phone_service': 'yes',
 'multiple_lines': 'no',
 'internet_service': 'dsl',
 'online_security': 'yes',
 'online_backup': 'yes',
 'device_protection': 'yes',
 'tech_support': 'yes',
```

```
 'streaming_tv': 'yes',
 'streaming_movies': 'yes',
 'contract': 'two_year',
 'paperless_billing': 'yes',
 'payment_method': 'bank_transfer_(automatic)',
 'senior_citizen': 0,
 'tenure': 71,
 'monthly_charges': 86.1,
 'total_charges': 6045.9}
```

Vectorizar y entrenar

[33]:
```
from sklearn.feature_extraction import DictVectorizer

dv = DictVectorizer(sparse=False)
dv.fit(train_dict)
X_train = dv.transform(train_dict)
X_train[0]
```

[33]:
```
array([0.0000e+00, 0.0000e+00, 1.0000e+00, 1.0000e+00, 0.0000e+00,
       0.0000e+00, 0.0000e+00, 1.0000e+00, 0.0000e+00, 1.0000e+00,
       1.0000e+00, 0.0000e+00, 0.0000e+00, 8.6100e+01, 1.0000e+00,
       0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00, 1.0000e+00,
       0.0000e+00, 0.0000e+00, 1.0000e+00, 0.0000e+00, 1.0000e+00,
       0.0000e+00, 1.0000e+00, 1.0000e+00, 0.0000e+00, 0.0000e+00,
       0.0000e+00, 0.0000e+00, 1.0000e+00, 0.0000e+00, 0.0000e+00,
       0.0000e+00, 1.0000e+00, 0.0000e+00, 0.0000e+00, 1.0000e+00,
       0.0000e+00, 0.0000e+00, 1.0000e+00, 7.1000e+01, 6.0459e+03])
```

[34]:
```
dv.get_feature_names_out()
```

[34]:
```
array(['contract=month-to-month', 'contract=one_year',
       'contract=two_year', 'dependents=no', 'dependents=yes',
       'device_protection=no', 'device_protection=no_internet_service',
       'device_protection=yes', 'gender=female', 'gender=male',
       'internet_service=dsl', 'internet_service=fiber_optic',
       'internet_service=no', 'monthly_charges', 'multiple_lines=no',
       'multiple_lines=no_phone_service', 'multiple_lines=yes',
       'online_backup=no', 'online_backup=no_internet_service',
       'online_backup=yes', 'online_security=no',
       'online_security=no_internet_service', 'online_security=yes',
       'paperless_billing=no', 'paperless_billing=yes', 'partner=no',
       'partner=yes', 'payment_method=bank_transfer_(automatic)',
       'payment_method=credit_card_(automatic)',
       'payment_method=electronic_check', 'payment_method=mailed_check',
       'phone_service=no', 'phone_service=yes', 'senior_citizen',
       'streaming_movies=no', 'streaming_movies=no_internet_service',
       'streaming_movies=yes', 'streaming_tv=no',
```

```
       'streaming_tv=no_internet_service', 'streaming_tv=yes',
       'tech_support=no', 'tech_support=no_internet_service',
       'tech_support=yes', 'tenure', 'total_charges'], dtype=object)
```

1.4 4.-Entrenar el modelo

[35]:
```
from sklearn.linear_model import LogisticRegression
```

[36]:
```
model = LogisticRegression(solver='liblinear', random_state=1)
model.fit(X_train, y_train)
```

[36]:
```
LogisticRegression(random_state=1, solver='liblinear')
```

1.5 5.-Validar nuestro modelo

[37]:
```
val_dict = df_val[cols].to_dict(orient='records')
print(cols)
X_val = dv.transform(val_dict)
```

```
['gender', 'partner', 'dependents', 'phone_service', 'multiple_lines',
'internet_service', 'online_security', 'online_backup', 'device_protection',
'tech_support', 'streaming_tv', 'streaming_movies', 'contract',
'paperless_billing', 'payment_method', 'senior_citizen', 'tenure',
'monthly_charges', 'total_charges']
```

[38]:
```
y_pred = model.predict_proba(X_val)[:, 1]
```

[39]:
```
y_pred
```

[39]:
```
array([0.23490456, 0.26884788, 0.319449  , …, 0.05724806, 0.61522716,
       0.06127278])
```

[40]:
```
churn = y_pred >= 0.5
```

[41]:
```
accuracy = (y_val == churn).mean()
accuracy
```

[41]:
```
0.8016129032258065
```

1.6 6.- Guardar el vector y el modelo

[42]:
```
import pickle

pickle.dump(dv, open('../models/dv.pkl', 'wb'))
pickle.dump(model, open('../models/model.pkl', 'wb'))
```

1.7 7.- Testeando el modelo

[43]:
```
y_test = df_test.churn.values
del df_test['churn']
```

[44]:
```
dv_pickle = pickle.load(open('../models/dv.pkl', 'rb'))

test_dict = df_test[cols].to_dict(orient='records')
X_test = dv_pickle.transform(test_dict)
```

[45]:
```
loaded_model = pickle.load(open('../models/model.pkl', 'rb'))
result = loaded_model.score(X_test, y_test)
print(result)
```

```
0.8069552874378992
```

1.8 8- Usando el modelo con un consumidor real

[46]:
```
customer1 = {
    'customerid': '8879-zkjof',
    'gender': 'female',
    'seniorcitizen': 0,
    'partner': 'no',
    'dependents': 'no',
    'tenure': 41,
    'phoneservice': 'yes',
    'multiplelines': 'no',
    'internetservice': 'dsl',
    'onlinesecurity': 'yes',
    'onlinebackup': 'no',
    'deviceprotection': 'yes',
    'techsupport': 'yes',
    'streamingtv': 'yes',
    'streamingmovies': 'yes',
    'contract': 'one_year',
    'paperlessbilling': 'yes',
    'paymentmethod': 'bank_transfer_(automatic)',
    'monthlycharges': 79.85,
    'totalcharges': 3320.75,
}

X_customer1 = dv_pickle.transform([customer1])
churn_proba_c1 = loaded_model.predict_proba(X_customer1)[0,1] * 100
churn_proba_c1
```

[46]:
```
4.060124210839723
```

[47]:
```
customer2 = {
    'gender': 'female',
    'seniorcitizen': 1,
    'partner': 'no',
    'dependents': 'no',
    'phoneservice': 'yes',
    'multiplelines': 'yes',
    'internetservice': 'fiber_optic',
    'onlinesecurity': 'no',
    'onlinebackup': 'no',
    'deviceprotection': 'no',
    'techsupport': 'no',
    'streamingtv': 'yes',
    'streamingmovies': 'no',
    'contract': 'month-to-month',
    'paperlessbilling': 'yes',
    'paymentmethod': 'electronic_check',
    'tenure': 1,
    'monthlycharges': 300,
    'totalcharges': 300
}

X_customer2 = dv_pickle.transform([customer2])
churn_proba_c2 = loaded_model.predict_proba(X_customer2)[0,1] * 100
churn_proba_c2
```

[47]:
```
56.617789065278615
```

1.9 9.- Usando el enfoque de Ensamble

[ ]:
```
```

[60]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC


log_clf = LogisticRegression(max_iter=500)
rnd_clf = RandomForestClassifier()
svm_clf = SVC(probability=True, max_iter=460)

voting_clf = VotingClassifier(
   estimators=[('lr',log_clf), ('rf', rnd_clf), ('svc', svm_clf)],
   voting="soft"
)
```

```
for clf in (log_clf, rnd_clf, svm_clf,voting_clf):
    clf.fit(X_train, y_train)
    y_pred = clf.predict_proba(X_test)[:,1]
    churn =  y_pred >= 0.5
    print(clf.__class__.__name__, (y_test == churn).mean())
```

```
LogisticRegression 0.8062455642299503
RandomForestClassifier 0.7934705464868701

/home/paco/py/cf-bootcamp-project/venv/lib/python3.10/site-
packages/sklearn/svm/_base.py:299: ConvergenceWarning: Solver terminated early
(max_iter=460).  Consider pre-processing your data with StandardScaler or
MinMaxScaler.
  warnings.warn(

SVC 0.7530163236337828
VotingClassifier 0.8119233498935415

/home/paco/py/cf-bootcamp-project/venv/lib/python3.10/site-
packages/sklearn/svm/_base.py:299: ConvergenceWarning: Solver terminated early
(max_iter=460).  Consider pre-processing your data with StandardScaler or
MinMaxScaler.
  warnings.warn(
```

[61]:
```
pickle.dump(voting_clf, open('../models/voting_model.pkl', 'wb'))
```