Predicción de Abandono

Contexto

Cell2Cell es una compañía de teléfonos celulares que intenta mitigar el abandono de sus usuarios. Te contratan para 1) Encontrar un modelo que prediga el abandono con acierto y para usar los insights de este modelo para proponer una estrategia de manejo de abandono.

Las preguntas que contestaremos son:

1. Se puede predecir el abandono con los datos que nos compartieron?

2. Cuáles son las variables que explican en mayor medida el abandono?

3. Qué incentivos da Cell2Cell a sus usarios para prevenir el abandono?

4. Cuál es el valor de una estrategia de prevención de abandono focalizada y cómo difiere entre los segmentos de los usuarios? Qué usuarios deberían de recibir incentivos de prevención? Qué montos de incentivos

Nota: Voy a evaluar las tareas con base en la respuesta a cada pregunta. Como hay algunas preguntas que no tienen una respuesta clara, al final ponderaré de acuerdo al poder predictivo de su modelo vs las respuestas sugeridas.

Datos

Los dotos los pueden encontrar en 'Cell2Cell.csv'. En el archivo 'Cell2Cell-Database-Documentation.xlsx' pueden encontrar documentación de la base de datos.

```
import pandas as pd
import math
import numpy as np
import datetime as dt
!pip install matplotlib-inline
from plotnine import ggplot, aes, geom_point, geom_line, geom_col, geom_histogram, geom_sm
```

1. Qué variables tienen missing values? Toma alguna decisión con los missing values. Justifica tu respuesta

```
# Leyendo la base de datos
data = pd.read_csv('Bases input/cell2cell.csv')

# Que me muestre todo
pd.set_option('display.max_columns', None)


# how many missings per variable
missings = data.isna().mean()*100
missings = missings[missings>0]
missings.sort_values(ascending= False)
```

	0
age1	1.750954
age2	1.750954
changem	0.706575
changer	0.706575
revenue	0.304024
mou	0.304024
recchrge	0.304024
directas	0.304024
overage	0.304024
roam	0.304024
phones	0.001408
models	0.001408
eqpdays	0.001408

eqpdays, phones, models, los voy a quitar dando que son menos del 0.001% de las observaciones

El resto las reemplazo por ceros.

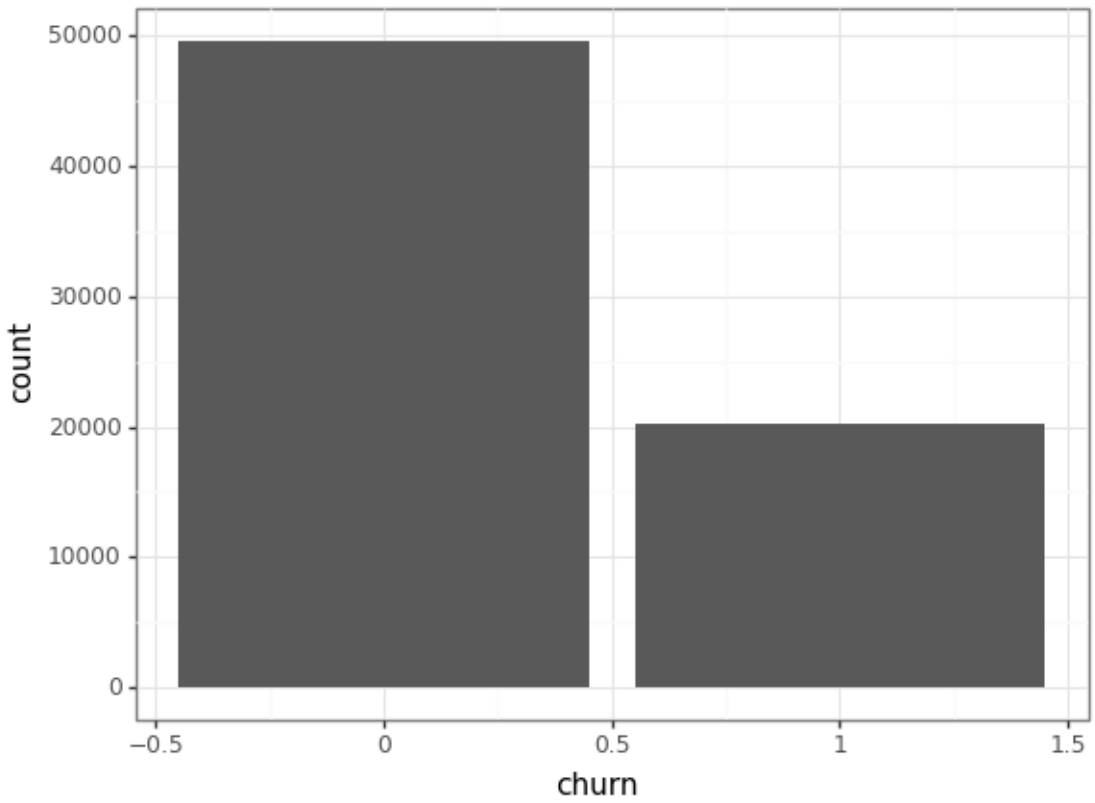
```
# Quitando las que tienen pocas observaciones

data = data[~pd.isnull(data.age1)]
data = data[~pd.isnull(data.age2)]
data = data[~pd.isna(data.eqpdays)]


# El resto las lleno con zero
data = data.fillna(0)
np.shape(data)
```

```
(69802, 68)
```

2. Tabula la distribución de la variable `churn`. Muestra la frecuencia absoluta y relativa. Crees que se debe hacer oversampling/undersamping?

Tiene una distribución 70%/30%. Parece que si habrá que hacer algo de sampling.

```
tab = 100*data.churn.value_counts()/data.shape[0]
%matplotlib inline
ggplot(data)+geom_bar(aes(x = 'churn'))+theme_bw()
```




```
<ggplot: (8778046428697)>
```

3. (2 pts) Divide tu base en entrenamiento y validación (80/20).

Además, considera hacer oversampling (SMOTE) o undersampling.

(Tip: Recuerda que el objetivo final es tener muestra ~balanceada en el traning set. En el validation la distribución debe ser la original).

La distribución esta 70% vs 30%. Si queremos construir un training set = 80%. Dentro del training, hay que hacer el undersampling tal que se balanceen las clases.

```
# Pandas way
data_train = data.sample(frac = 0.8, random_state = 1990)
data_test = data.drop(data_train.index)
```

```
# Definiendo una funcion
def n_distinct(x):
    a = np.shape(np.unique(x))
    return a[0]

n_distinct(data.customer)
```

69802

Undersampling

```
# Filtro la base donde churn == 1
data_train_1 = data_train[data_train.churn ==1]
data_train_0 = data_train[data_train.churn == 0]

# Undersampling
data_train_0 = data_train_0.sample(frac = 0.41, random_state = 1990)

# Base train final
data_train = pd.concat([data_train_1, data_train_0])

# Removing aux tables
del data_train_0, data_train_1
```

Model Estimation

Pondremos a competir 3 modelos:

1. Cross-Validated LASSO-logit
2. Prune Trees
3. Random Forest
4. Gradient Boosting Machine

4 (2 pts). Estima un cross validated LASSO. Muestra el la gráfica de CV Binomial Deviance vs Complejidad

```
from sklearn import linear_model
from sklearn.linear_model import lasso_path
from sklearn.linear_model import  LassoLarsIC

# https://scikit-learn.org/stable/auto_examples/linear_model/plot_lasso_model_selection.ht
```