#Predicción de bajas de clientes para empresas de telecomunicaciones indias.

<“Churn - Prediction”/>

Grupo 1

Integrantes

Fabrizio Sgro
Arquitecto de Datos Jr

Rodriguez Natacha
Estudiante avanzado de Ing. Química | Tester Jr

<¿Qué es Churn?/>

El término "churn" se utiliza en el ámbito empresarial, especialmente en la industria de servicios, para referirse a la tasa de pérdida de clientes o usuarios durante un período de tiempo determinado. Es una métrica clave para evaluar la retención de clientes y la salud general de un negocio.

<¿Por qué?/>

1. Los clientes insatisfechos afectan negativamente a su marca.

La empresa podría recibir malas críticas y perjuicios que pueden afectar al valor de su marca.

2. La rotación de clientes le sale más cara.

A menudo se dice que mantener un cliente existente cuesta menos y ofrece más valor que adquirir uno nuevo.

3. La rotación de clientes puede impactar a su crecimiento.

Si la empresa está considerando traer nuevos productos y servicios al mercado, es probable que la mejor audiencia sean sus clientes existentes que ya conocen su marca. Sin embargo, si el valor de vida de sus clientes es bajo, su nuevo proyecto podría verse afectado

<¿Cómo?/>

> Modelos de Machine Learning

Modelos de ensamble: Para capturar relaciones no lineales que mejoren la precisión.

> Procesamiento de datos

One Hot Encoding y Label Encoding para variables categóricas.

Estandarización, normalización y scaling para variables numéricas.

Análisis de ‘churn’ a lo largo del tiempo gracias variables de tiempo.

> Estadística aplicada

Pruebas estadísticas para la aceptación de hipótesis (p-valor).

EDA

FASE 1

Análisis descriptivo

se analizaron los dataframes y se extrajo la información mas relevante, no hay valores vacíos ni duplicados

FASE 2

Re-ajuste de los tipos de variables

se corrobora que los datos estén bien codificados y las variables estén bien definidas.

FASE 3

Detección y tratamiento de datos ausentes

se observa que el dataframe tiene datos hasta el mes 5 del 2023, para el analisis de churn por año se elimina el año 2023

FASE 4

Detección y tratamiento de datos atipicos

Aparecen números negativos en el minimo de consumos (llamadas, sms y datos).

FASE 5

Análisis de correlación de variables

No hay variables que aporten información redundante.

<¿Cómo?/>

Comenzamos uniendo dos dataframes con los datos demográficos y de uso de cada uno de los clientes. Revisamos que el dataset no tenga ni valores nulos ni repetidos.

```
#Lectura de los archivos .csv con pandas
df1 = pd.read_csv('telecom_demographics.csv')
df2 = pd.read_csv('telecom_usage.csv')

print(df1.shape, df2.shape)
```

```
(6500, 10) (6500, 5)
```

```
df_full.isna().sum()
```

```
customer_id          0
telecom_partner      0
gender               0
age                  0
state                0
city                 0
pincode              0
registration_event   0
num_dependents       0
estimated_salary     0
calls_made           0
sms_sent             0
data_used            0
churn                0
dtype: int64
```

<¿Cómo?/>

Variables categóricas

	count	unique	top	freq
telecom_partner	6500	4	Reliance Jio	1658
gender	6500	2	M	3909
state	6500	28	Karnataka	260
city	6500	6	Delhi	1128
registration_event	6500	1216	2021-04-04	14

Variables numéricas

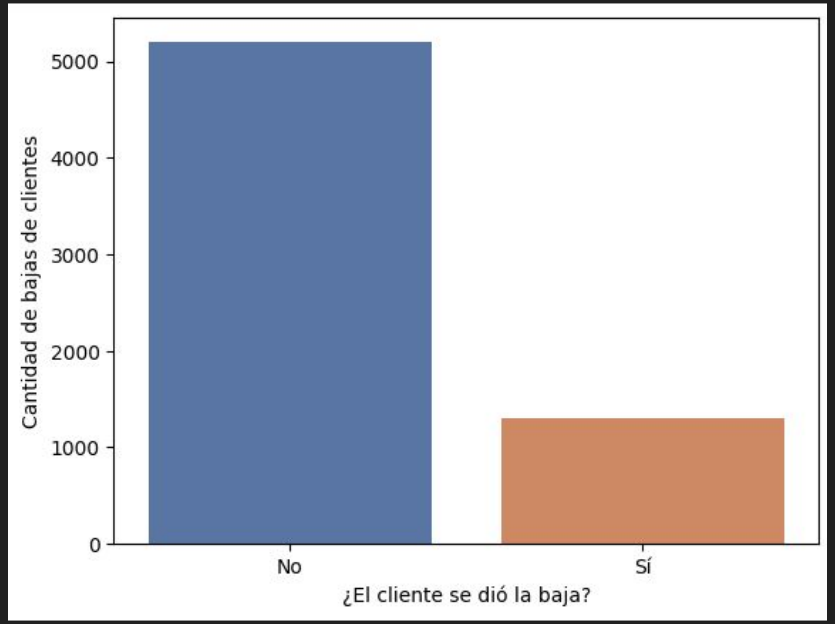
	age	num_dependents	estimated_salary	calls_made	sms_sent	data_used
count	6500.000000	6500.000000	6500.000000	6500.000000	6500.000000	6500.000000
mean	46.108615	1.982308	85529.193385	49.789538	24.257846	5000.956308
std	16.443712	1.404659	37545.639180	29.799221	14.650736	2940.611928
min	18.000000	0.000000	20001.000000	-10.000000	-5.000000	-969.000000
25%	32.000000	1.000000	52905.000000	25.000000	12.000000	2493.750000
50%	46.000000	2.000000	85286.500000	50.000000	25.000000	4975.500000
75%	60.000000	3.000000	118817.500000	75.000000	37.000000	7504.250000
max	74.000000	4.000000	149977.000000	108.000000	53.000000	10919.000000

<¿Cómo?/>

Nos encontramos con un desbalance de clases que tratamos con la metodología de undersampling

```
from imblearn.under_sampling import RandomUnderSampler
from collections import Counter

undersampler = RandomUnderSampler(sampling_strategy=0.28, random_state=42)
X_train, y_train = undersampler.fit_resample(X, y)

print ("Distribution before resampling {}".format(Counter(y)))

print ("Distribution labels after resampling {}".format(Counter(y_train)))
```

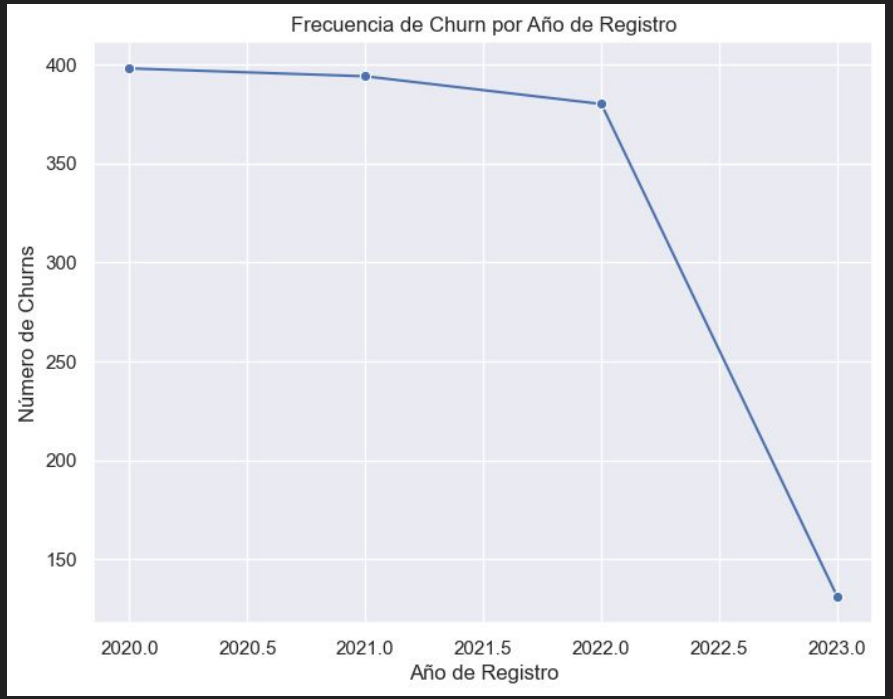
```
Distribution before resampling Counter({0: 5197, 1: 1303})
Distribution labels after resampling Counter({0: 4653, 1: 1303})
```



<¿Cómo?/>

#Continuamos con nuestro Análisis exploratorio de las variables

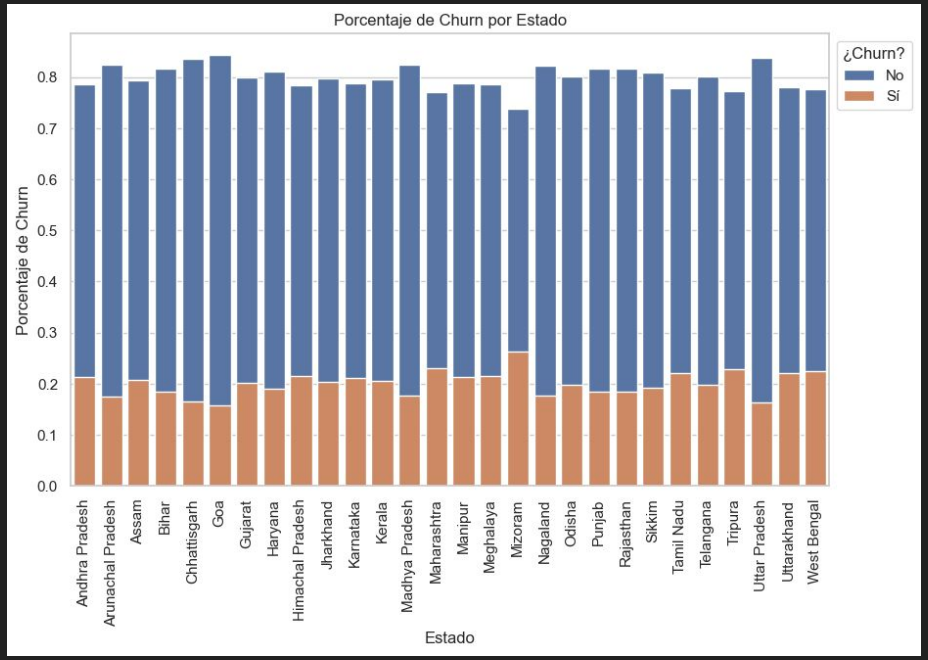
Es muy interesante ver que el año en el que llega la pandemia (2020) se produce la mayor cantidad de bajas y esto puede ser debido a que mucha gente tuvo que dejar de trabajar debido al confinamiento y decidió darse de baja del servicio para reducir costos.



<¿Cómo?/>

Investigamos cómo influye el estado al churn

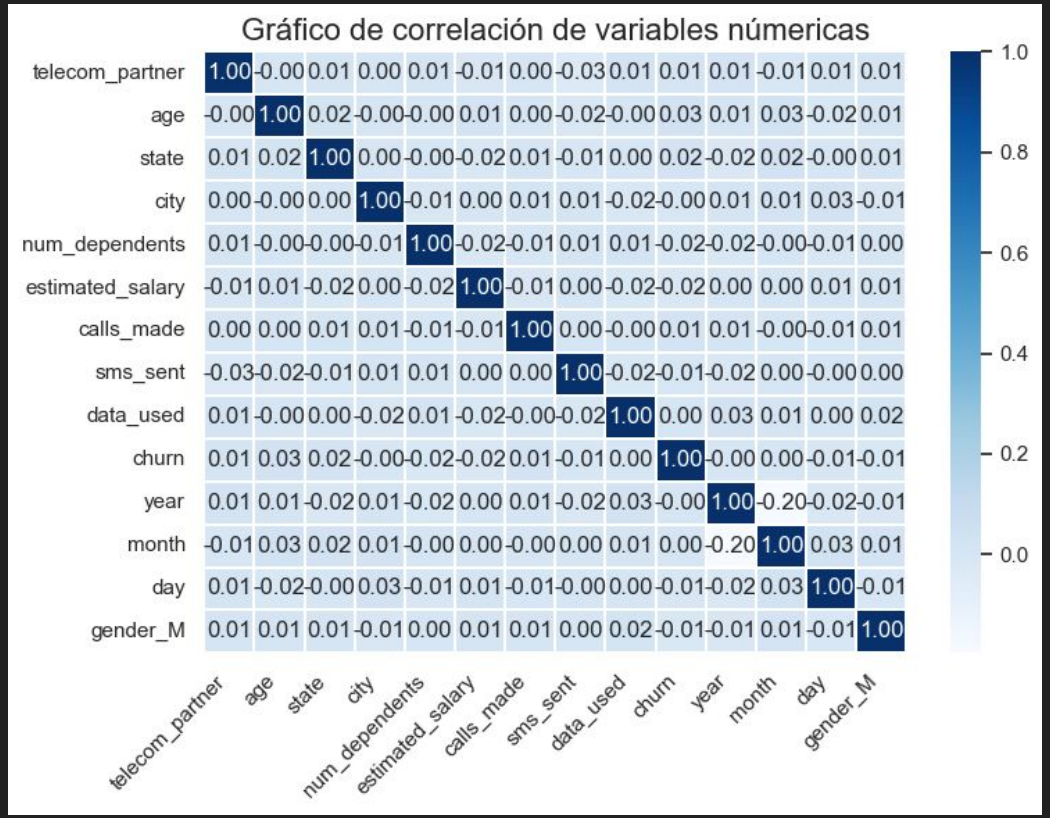
Vemos que hay estados con mayores probabilidades de que sus habitantes renuncien, sin embargo, el porcentaje o probabilidad no difiere mucho de estado a estado



<¿Cómo?/>

Pre-procesamiento

Realizamos tareas de pre-procesamiento, para el entrenamiento de los modelos y revisamos con todas las variables del dataset que correlación, mediante spearman podía tener la variable churn, pero vemos que no existe una relación fuerte (negativa o positiva) con el resto de variables.



<¿Cómo?/>

Disminución de clases mayoritarias en train set

```
X= df_pre_processed.drop(axis=1, columns='churn')
y= df_pre_processed['churn']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.20, stratify=y, random_state=42)
```

✓ 0.0s

```
from imblearn.under_sampling import RandomUnderSampler
from collections import Counter

undersampler = RandomUnderSampler(sampling_strategy=0.28, random_state=42)
X_train, y_train = undersampler.fit_resample(X, y)

print ("Distribution before resampling {}".format(Counter(y)))

print ("Distribution labels after resampling {}".format(Counter(y_train)))
```

✓ 0.0s

```
Distribution before resampling Counter({0: 5197, 1: 1303})
Distribution labels after resampling Counter({0: 4653, 1: 1303})
```

<¿Cómo?/>

Función de entrenamiento + ploteo

Creamos un .py con funciones a importar en nuestro notebook que nos permite hacer predicciones (make_prediction) y luego verificar los resultados mediante una matriz de confusión (verify_results)

machine_learning_utils.py

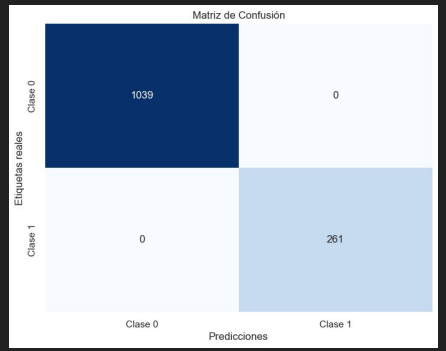
```python
import seaborn as sns
import matplotlib.pyplot as plt

def make_prediction(model, X_train, y_train, X_test):
    """
    Realiza la predicción utilizando un modelo dado.

    Parámetros:
    - model: El modelo de clasificación entrenado.
    - X_train: Datos de entrenamiento.
    - y_train: Etiquetas de entrenamiento.
    - X_test: Datos de prueba.

    Devuelve:
    - predictions: Las predicciones del modelo en los datos de prueba.
    """
    # Entrenar el modelo con los datos de entrenamiento
    model.fit(X_train, y_train)

    # Realizar predicciones en los datos de prueba
    predictions = model.predict(X_test)

    return predictions

def verify_results(y_true, y_pred):
    """
    Muestra un mapa de calor de la matriz de confusión y otros resultados de clasificación.

    Parámetros:
    - y_true: Etiquetas reales.
    - y_pred: Predicciones del modelo.
    """
    # Calcular la matriz de confusión
    confusion_mat = confusion_matrix(y_true, y_pred)

    # Mostrar otros resultados de clasificación
    print("Reporte de Clasificación:")
    print(classification_report(y_true, y_pred))

    # Crear un mapa de calor de la matriz de confusión utilizando Seaborn
    plt.figure(figsize=(8, 6))
    sns.heatmap(confusion_mat, annot=True, fmt="d", cmap="Blues", cbar=False,
                xticklabels=["Clase {}".format(i) for i in range(confusion_mat.shape[0])],
                yticklabels=["Clase {}".format(i) for i in range(confusion_mat.shape[0])])
    plt.xlabel("Predicciones")
    plt.ylabel("Etiquetas reales")
    plt.title("Matriz de Confusión")
    plt.show()
```

<¿Cómo?/>

Modelos seleccionados

Elegimos modelos ensamble (Bagging y boosting) ya que vemos que la clasificación requería de una resolución no-lineal.

Randomforest los modelos sin hyper-param tunning y sin cross-validation presentaron altos niveles de resultados (overfitting)




```
model = RandomForestClassifier()
y_pred = make_prediction(model, X_train, y_train, X_test)
verify_results(y_test, y_pred)
```

✓ 2.1s

```
xgbc = XGBClassifier()

y_pred = make_prediction(xgbc, X_train, y_train, X_test)
verify_results(y_test, y_pred)
```

✓ 0.5s

```
gb_classifier = GradientBoostingClassifier(loss='exponential', learning_rate=0.01)

y_pred = make_prediction(gb_classifier, X_train, y_train, X_test)
verify_results(y_test, y_pred)
```

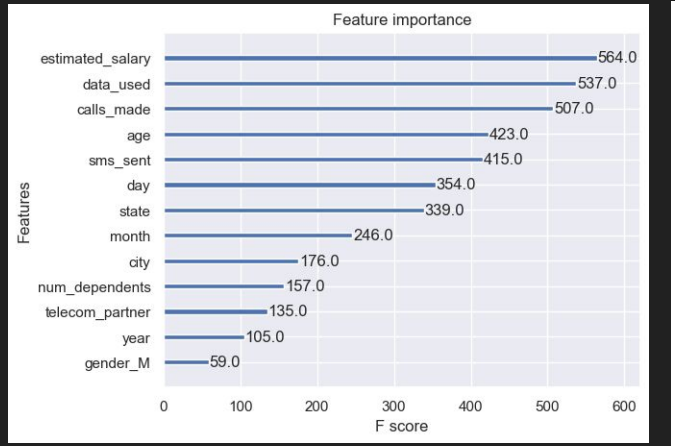
✓ 1.9s

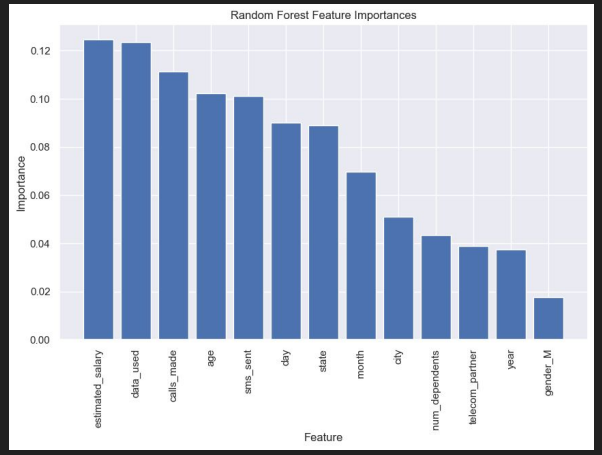
```
adaboost_classifier = AdaBoostClassifier()

y_pred = make_prediction(adaboost_classifier, X_train, y_train, X_test)
verify_results(y_test, y_pred)
```

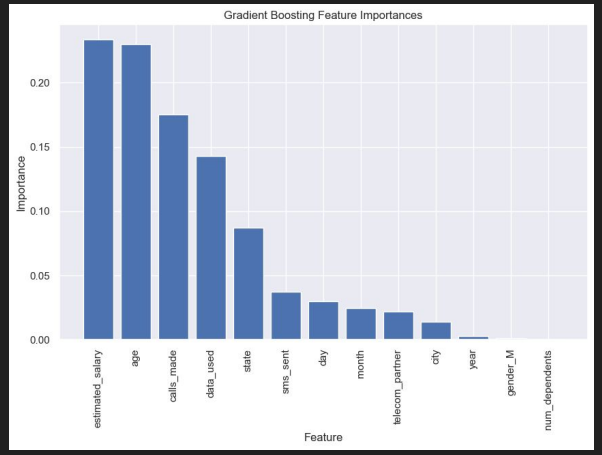
✓ 0.6s

<¿Cómo?/>

Features más importantes para clasificación







Para los modelos **estimated_salary**, **data_used**, **calls_made**, **age** y **sms_sent** son las variables con mayor importancia para definir la clase (churn sí o churn no)

<Conclusiones/>

Pudimos comprobar que predecir el churn de los clientes de empresas de telecomunicaciones sí es posible.

A su vez, logramos detectar qué variables son imprescindible que las empresas de telecomunicación tengan siempre actualizada para poder tener en tiempo real, respuestas concretas sobre un cliente que pueda estar en riesgo de darse de baja, basándonos principalmente en su salario, edad y los consumos que ha realizado en su línea móvil (llamadas, sms y paquetes de datos).

```
print("Muchas gracias")
```