

JULIO DE 2021



CUSTOMER CHURN PREDICTION

MÁSTER DATA SCIENCE (ED. 26)

MEMORIA TFM

ALBERTO DÍAZ RIBÓN

albertodiazr@hotmail.com
https://github.com/albertodiazr/TFM_churn

Contenido

1. Objetivo
2. Herramientas utilizadas
3. Estructura y organización
 - 3.1. Carpetas y archivos
 - 3.2. Descripción del dataset
 - 3.3. Flujo de trabajo
 - 3.4. Organización de los notebooks
4. Análisis exploratorio
5. Metodología y modelos aplicados
 - 5.1. Métricas utilizadas
 - 5.2. Feature engineering
 - 5.3. Machine Learning
 - 5.4. Cross-validation
 - 5.5. Feature importances
6. Resultados y conclusiones
7. Frontend

1. Objetivo

Proyecto de Machine Learning cuya finalidad es predecir la probabilidad de baja de cada cliente de una empresa que instala calderas en inmuebles para la producción energética, vendiendo posteriormente también el combustible necesario para su funcionamiento.

La importancia de la retención proactiva de clientes ha cobrado protagonismo en las últimas décadas en las empresas debido a que ese coste de retención suele ser significativamente menor que el coste de adquisición de un nuevo cliente, que soporta costes de marketing y todo lo asociado a la estructura comercial (salarios, comisiones, vehículos, combustible...), mientras que una retención se puede solventar mediante una simple llamada del servicio de atención al cliente.

2. Herramientas

Previamente se han obtenido los datasets utilizados en el proyecto con consultas al *data warehouse* de la compañía mediante **SAP BusinessObjects**, empleando aproximadamente un tercio del tiempo dedicado al mismo.

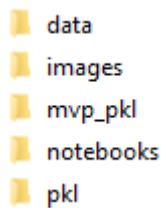
Librerías de **Python** utilizadas en el desarrollo del proyecto:

- [pandas](#)
- [numpy](#)
- [matplotlib](#)
- [seaborn](#)
- [altair](#)
- [itertools](#)
- [datetime](#)
- [scikit-learn](#)
- [category_encoders](#)
- [xgboost](#)
- [pickle](#)
- [plotly](#)
- [Python Imaging Library \(PIL\)](#)
- [streamlit](#)

3. Estructura y organización

3.1. Carpetas y archivos

Clonando el repositorio de GitHub y descargando los archivos del enlace de Google Drive enviado por email, el directorio del proyecto debe quedar organizado en las siguientes carpetas:



3.2. Descripción del dataset

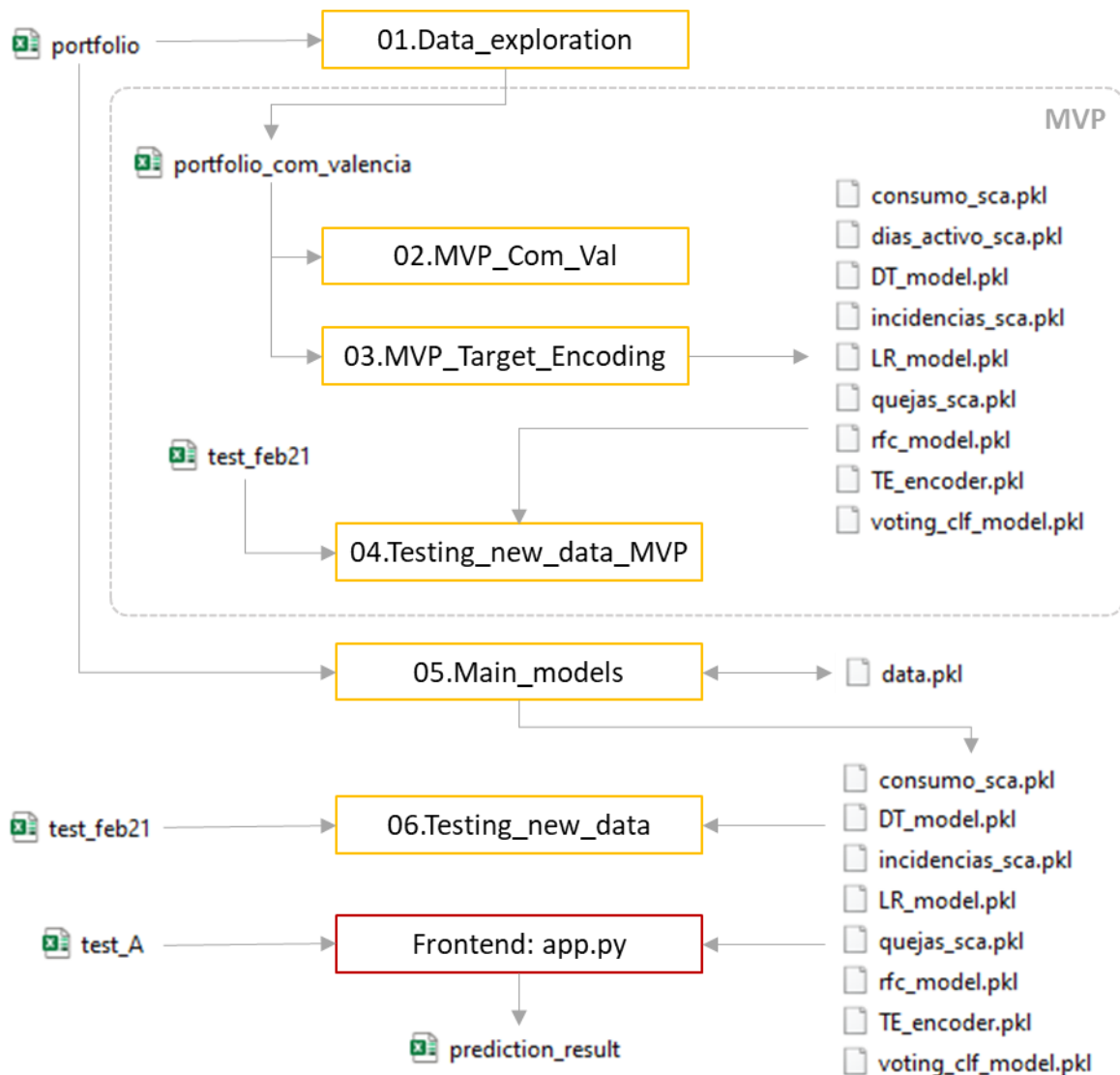
Descripción de cada uno de los campos:

- *Fecha Alta*: Fecha de alta del cliente.
- *Cliente*: Número de cliente.
- *Modelo Caldera*: Modelo de la caldera instalada (Biasi, Domusa).
- *Provincia*: Región donde se encuentra el domicilio donde se instala la caldera.
- *Gender*: Hombre / Mujer.
- *País*: Nacionalidad del cliente.
- *Born Date*: Fecha de nacimiento del cliente.
- *Tipo Propiedad* del inmueble donde se instala la caldera.
- *Situación Laboral* del cliente en el momento de la instalación de la caldera.
- *Estado Civil* del cliente en el momento de la instalación de la caldera.
- *Ingresos* estimados del cliente.
- *Pagos Anuales*: Número de cuotas anuales que se cargan al cliente.
- *Fecha Estado*: Última fecha de cambio de estado del contrato. Será igual a *Fecha Alta*, si sigue activo, o la fecha de la baja si el *Estado* es Baja.
- *Estado*: Activo / Baja
- *Tipo Inmueble* donde se realiza la instalación de la caldera.
- *Financiado*: Si el cliente realiza el pago de la caldera al contado o con una parte o el total del coste financiado.

- *Precio Contado*: Parte del precio (en €) de la caldera que el cliente abona al contado al inicio del contrato.
- *Precio Total* de la caldera en €.
- *Quejas*: Número de quejas registradas en los canales de atención al cliente.
- *Incidencias*: Incidencias técnicas registradas en los canales de atención al cliente.
- *Consumo Mes*: Número de litros de combustible medios consumidos por el cliente al mes.

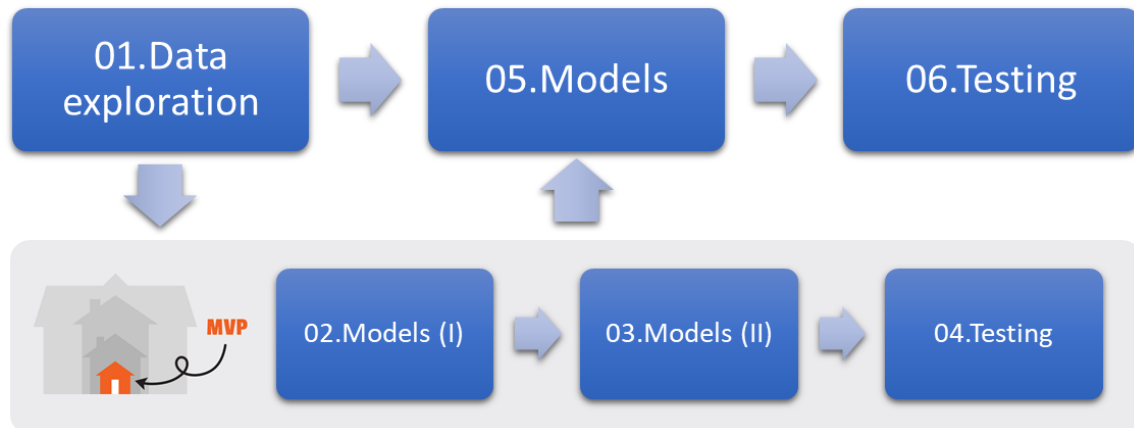
3.3. Flujo de trabajo

Esquema indicando qué inputs necesita cada notebook y qué outputs generan:



3.4. Organización de los notebooks

Los notebooks están numerados en orden de ejecución. No obstante, es posible ejecutar la secuencia 01, 05 y 06 ya que los notebooks 02, 03 y 04 se han creado a modo de MVP con una parte de los datos:



- 01.Data_exploration:

Exploración general de todo el data y obtención del subconjunto de datos para comenzar trabajando únicamente con los clientes de la Comunidad Valenciana, acortando de esta manera las iteraciones necesarias para crear los primeros modelos.

- 02.MVP_Com_Val:

Exploración del subconjunto de datos, *one-hot encoding* de las variables categóricas, escalado de datos numéricos, definición de los *benchmarks* y creación y medición de los primeros modelos.

- 03.MVP_Target_Encoding:

En este notebook se utiliza *target encoding* para transformar las variables categóricas, se vuelven a testar los modelos y se crean archivos *pickle* para utilizarlos en el siguiente notebook.

- 04.Testing_new_data_MVP

Con nuevos clientes de la Comunidad Valenciana, con los que no han sido entrenados los modelos, se testan los modelos del notebook anterior.

- 05.Main_models

Se replica el notebook 03 para todo el data.

- 06.Testing_new_data

Se testan los modelos del notebook anterior con nuevos clientes.

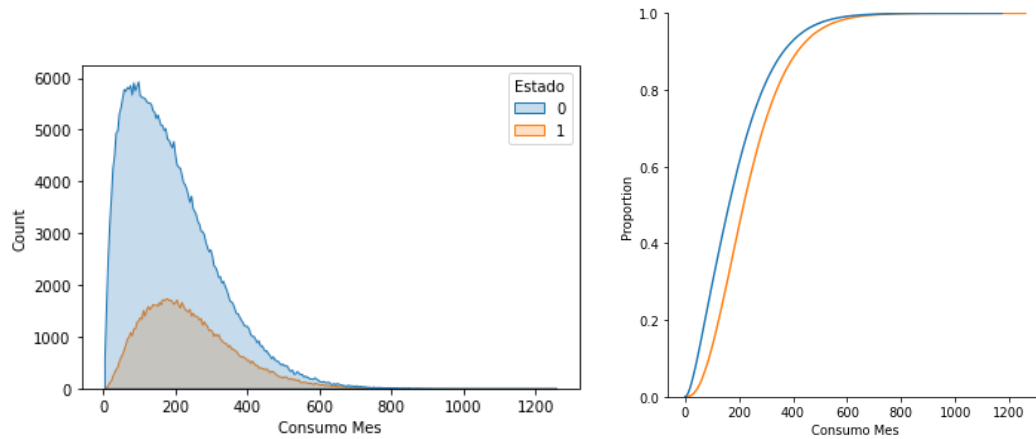
4. Análisis exploratorio

Se exponen las conclusiones más destacadas al realizar el análisis exploratorio de los datos.

Como era de esperar, las *Quejas* y las *Incidencias* tienen una correlación lineal directa con las *Bajas*, motivo por el que se utilizarán en los *benchmarks* como modelos a batir:

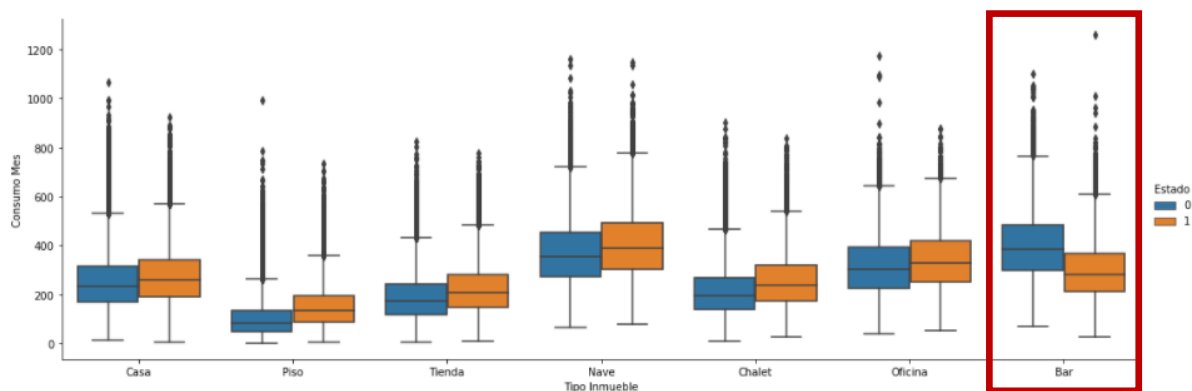


Otra variable en la que se observa una correlación lineal directa es en *Consumo Mes*, con la distribución de la variable en las bajas más desplazada a la derecha que en los clientes activos:



Esta particularidad puede deberse a que los clientes terminan buscando fuentes energéticas que se adapten mejor a sus necesidades de alto consumo.

Sin embargo, al segmentar la información por tipo de inmueble, se mantiene dicha tendencia en la distribución excepto en los bares, en los que las bajas suelen tener menos consumo, lo que puede indicar que muchas de las bajas se deben al cese del negocio por baja actividad:



5. Metodología y modelos aplicados

5.1. Métricas utilizadas

El objetivo principal es la detección temprana de bajas, por lo que se decide utilizar como métrica principal a maximizar el Recall de los casos positivos, controlando a su vez que la precisión (*accuracy*) global de los modelos se mantenga en niveles no demasiado bajos provocados por un aumento en la predicción de falsos positivos, por lo que se utilizarán como métricas de apoyo para facilitar su visualización y control:

- *Classification_report* de *scikit-learn*.
- Curvas *ROC* y *Precision-Recall*.
- *Confusion matrix*.

No se utiliza la precisión como métrica principal debido al desbalanceo en las clases, ya que sólo 1 de cada 4 observaciones es Baja.

5.2. Feature engineering

- Variables numéricas:
 - Se crean las variables *Edad* y *Días_Activo*. En el tercer punto de este apartado se analizan los motivos de por qué ésta última no se utiliza como input en los modelos finales.
 - Las variables numéricas *Income* y *Edad* se transforman en categóricas creando segmentos que ayuden a identificar los estratos más propensos a la baja para que, posteriormente, se puedan definir estrategias de negocio por segmento tras obtener los resultados.
 - Las variables restantes (*Quejas*, *Incidencias* y *Consumo Mes*) se escalan con *MinMaxScaler*, al ofrecer mejores resultados que *StandardScaler* o *RobustScaler*.
- Variables categóricas:

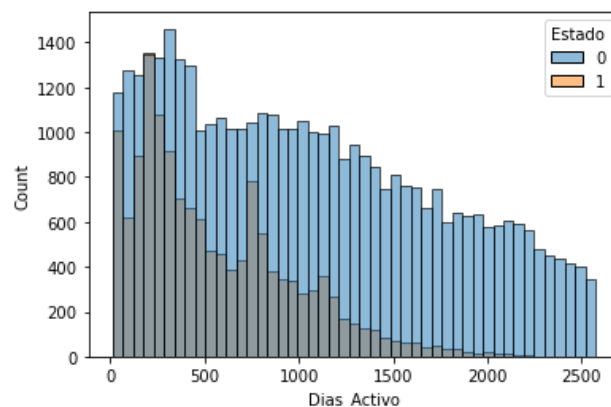
En los primeros modelos del MVP se aplica *one-hot encoding* con la función *get_dummies* de Pandas para codificar este tipo de variables. Posteriormente, sobre el mismo dataset, se repite el proceso con *TargetEncoder* que, al obtener resultados muy

similares, finalmente es el método elegido por su simplicidad a la hora de aplicarlo a nuevos datos.

Se comprueba en los notebook de test que este método no produce *overfitting*, uno de los principales riesgos al aplicar esta técnica por utilizar la variable objetivo para codificar el resto.

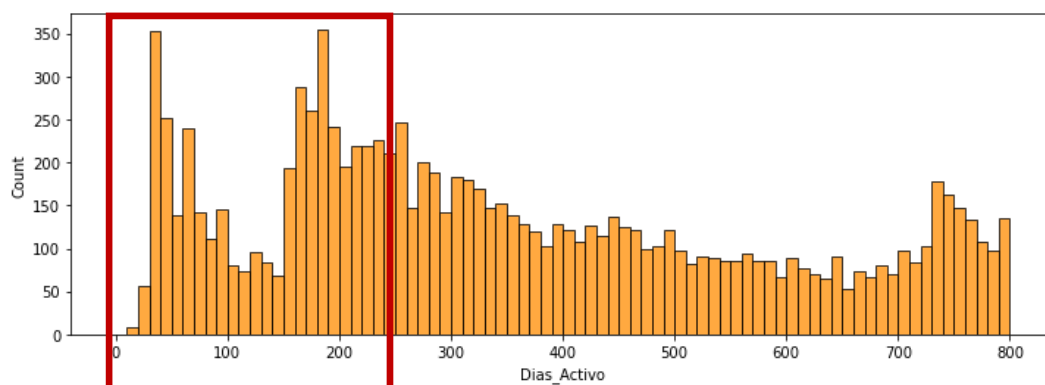
- Variable *Dias_Activo*:

Al crear la variable *Dias_Activo* para utilizarla en los modelos, se advierte que las bajas tienen lugar principalmente en los primeros 2 años:



Al realizar zoom sobre ese período únicamente en las bajas, se comprueba que se concentran esencialmente alrededor de los días 30 y 180, por los siguientes motivos:

- El cliente tiene 30 días de período de desistimiento.
- A los 6 meses, si el cliente adeuda 2 o más cuotas mensuales, se le “facilita” la salida de la compañía llegando a un acuerdo para recuperar la caldera y la mayor parte posible de la cantidad adeudada.

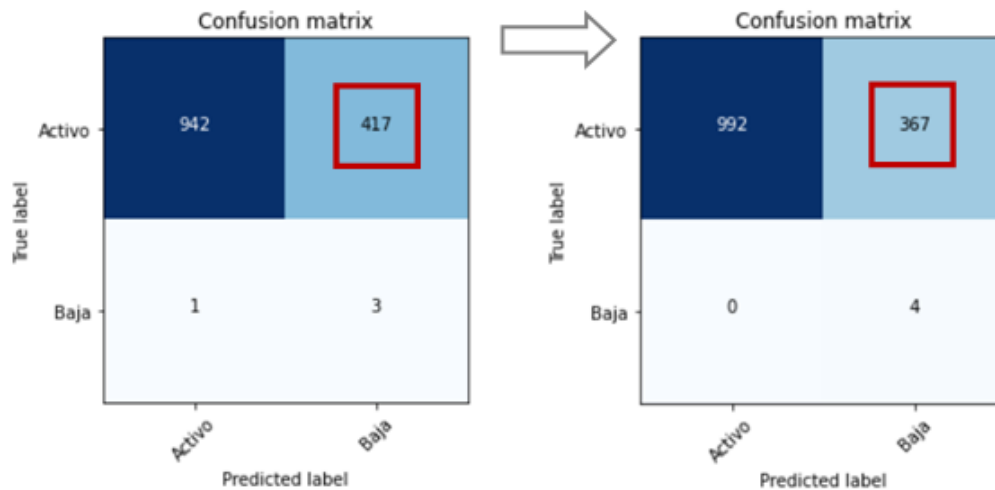


En consecuencia, esta variable acaba influyendo notablemente en el target, como se aprecia al mostrar un *heat map* de correlación directa tras la creación de la nueva variable:



Es al testar con clientes nuevos en el MVP cuando se decide eliminar la variable de los modelos finales, por obtenerse un número elevado de falsos positivos al tratarse en parte de altas en los últimos 30-60 días en el momento del test, período en el que los algoritmos detectan que existe una alta propensión a la baja.

Por ejemplo, en los resultados obtenidos por el *Decision Tree Classifier* se muestra la mejora en las predicciones, al reducir los falsos positivos, además de capturar las 4 bajas del set de test:



5.3. Machine Learning

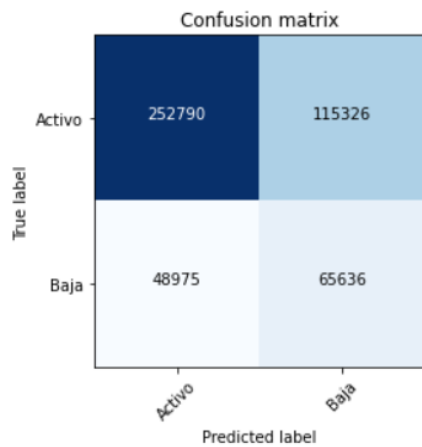
Se comentan brevemente las particularidades de cada algoritmo/modelo implementado con los resultados de la matriz de confusión resultante con todo el data en el notebook 05.

- Baseline:

En el notebook 03, con los datos del MVP, se definen 3 modelos para ser utilizados como *benchmarks*:

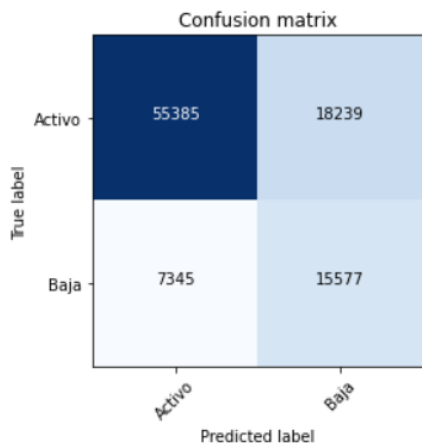
- Naive Models:
 - Modelo en el que una Queja implica Baja.
 - Modelo en el que una Queja o una Incidencia implica Baja.
- *Logistic Regression* sin variables categóricas y sin escalado de variables numéricas.

Como el segundo de los modelos Naive es el que mejor resulta, es el único que se vuelve a replicar como baseline con todos los datos en el notebook 05:



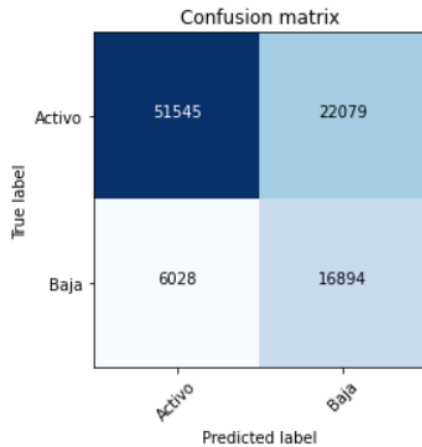
- Logistic Regression:

Se aprovecha el parámetro `class_weight="balanced"` para intentar equilibrar a la clase minoritaria penalizando a la clase mayoritaria durante el entrenamiento y obtener mejores resultados.



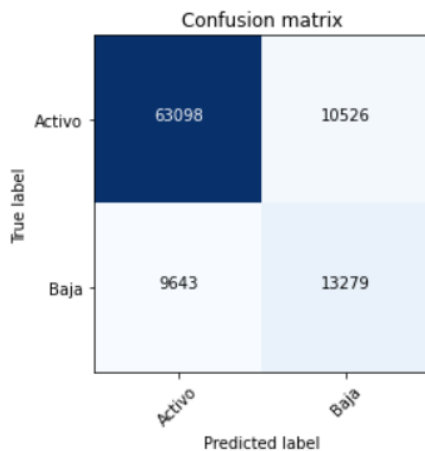
- Decision Tree Classifier:

Al igual que en la regresión logística, se utiliza el parámetro `class_weight="balanced"`, siendo en ambos modelos donde mayor recall de los casos positivos se obtiene (sin tener en cuenta el *voting classifier*, que es la combinación de los dos).



- Random Forest Classifier:

Tuneando los parámetros en el notebook 03 con los datos del MVP se mejora el resultado, pero el aumento en el tiempo de entrenamiento y el mayor espacio que ocupa el *.pkl* del modelo no compensan esa pequeña mejora en el rendimiento, por lo que se decide no implementarlo con todos los datos en el notebook 05.

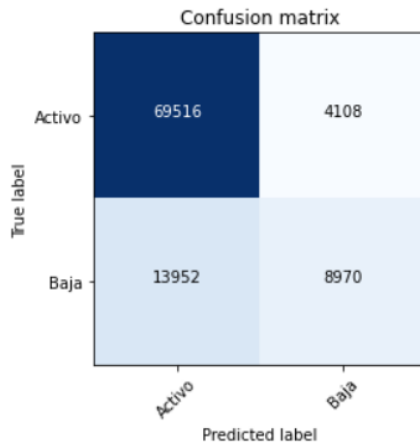


- KNN (K-nearest Neighbours):

El algoritmo que peor funciona en el MVP pese a que se intentan “tunear” los hiperparámetros mediante *GridSearchCV*, motivo por el que no se implementa con todo el dataset en el notebook 05.

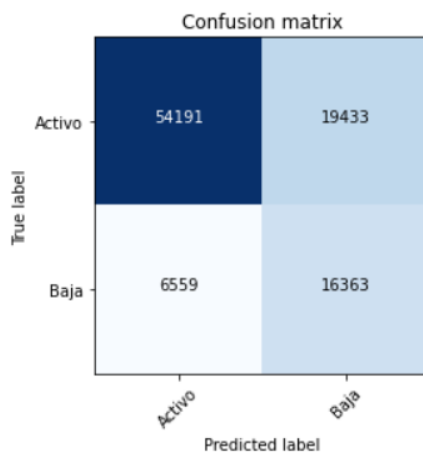
- XGBoost:

Este modelo es el que mayor precisión consigue a costa de detectar en menor medida las bajas. Pese a ser entrenado y testado variando diversos hiperparámetros, no se alcanzan resultados óptimos.



- Voting Classifier:

En el MVP los resultados son muy equilibrados incluyendo los 3 modelos con mejor comportamiento (*Logistic Regression*, *Decision Tree* y *Random Forest*), mientras que en el notebook 05 se deja fuera de “la votación” el *Random Forest* debido a que aumenta significativamente el tamaño del archivo *pickle* al empaquetar el modelo entrenado. En ambos casos, se instancia con el parámetro `voting='soft'` para que *sklearn* prediga la clase con la probabilidad de clase más alta, promediada sobre todos los clasificadores individuales dando más peso a los votos que son más seguros.

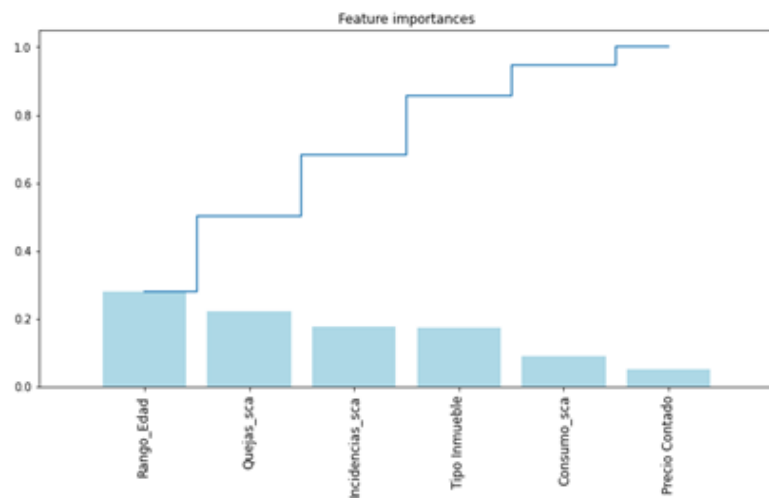
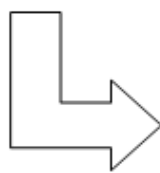
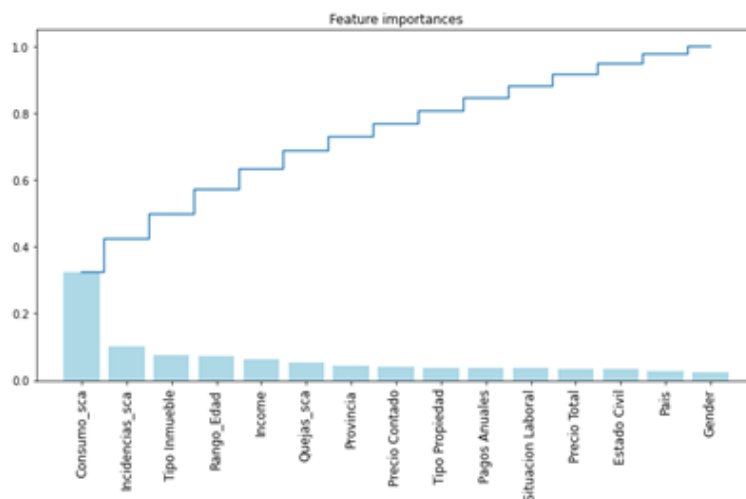


5.4. Cross-validation

Como las clases están desbalanceadas (sólo un 23% son clientes de Baja), a la hora de entrenar y validar los modelos se utiliza *StratifiedKFold* al implementar la *cross-validation* para comprobar que los resultados son robustos en distintas particiones del dataset conservando la proporción de ambas clases (Activo y Baja).

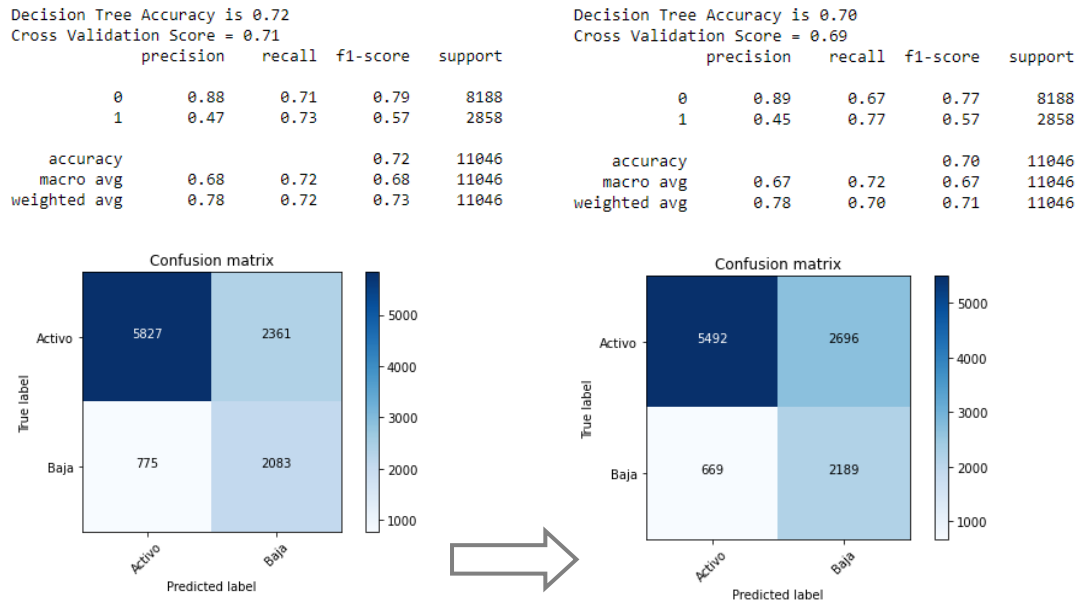
5.5. Feature importances

Ploteando la importancia relativa de las variables mediante el método *feature_importances_* de *sklearn* utilizando el algoritmo con mejor resultado, *Decision Tree*, en el MVP, se eliminan las variables menos significativas para volver a entrenar tanto el mismo *Decision Tree* como la *Logistic Regression*:

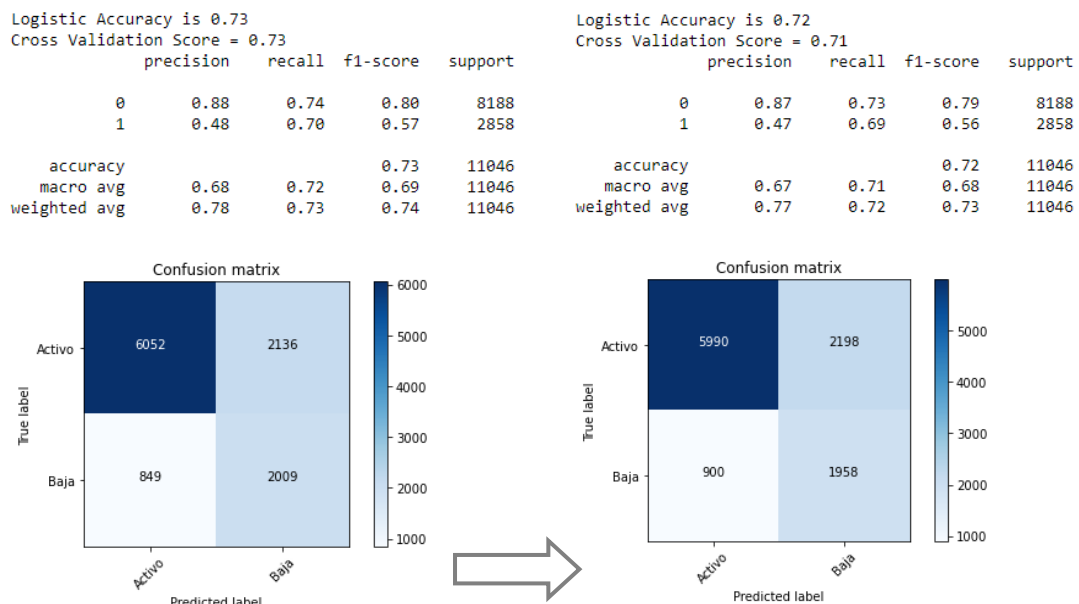


Se alcanzan los siguientes resultados:

- *Decision Tree*. Se aprecia mejora del Recall a costa de reducir la precisión global del modelo:



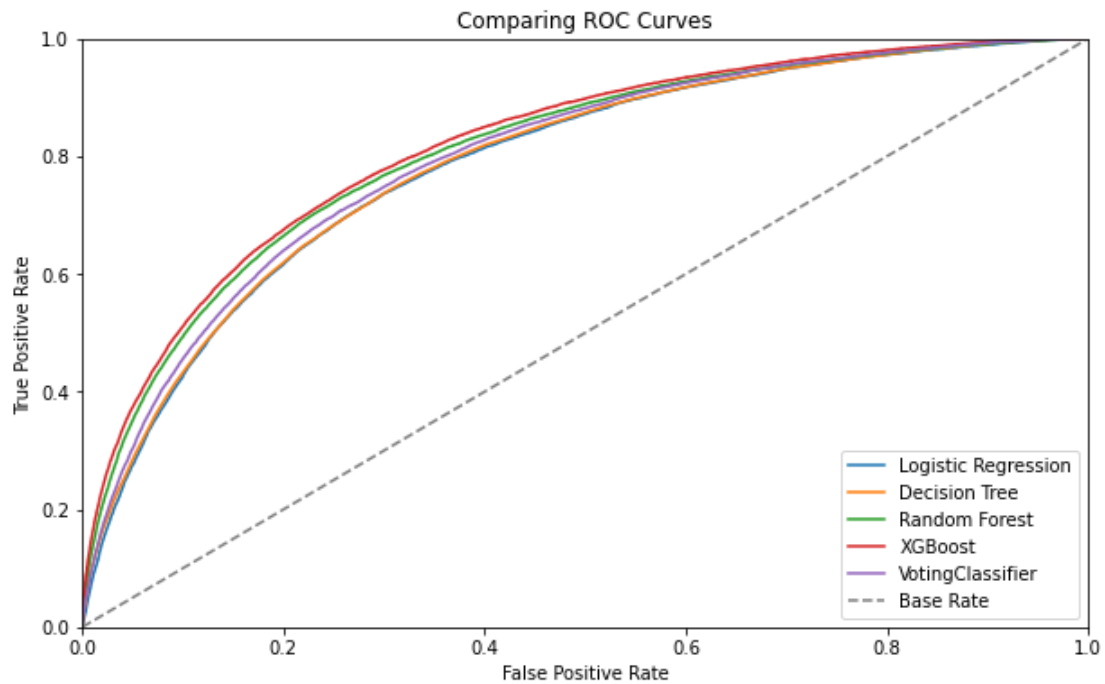
- *Logistic Regression*. En este caso se reduce tanto el Recall como la precisión:



Es por ello que se decide utilizar todas las variables para entrenar los modelos con todo el portfolio.

6. Resultados y conclusiones

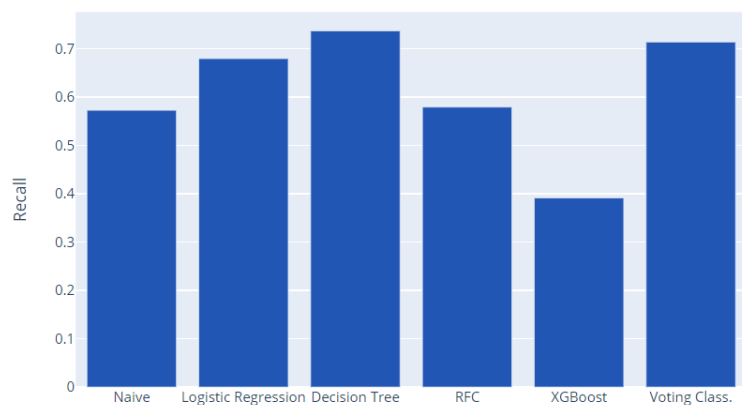
Los modelos de *XGBoost* y *Random Forest Classifier* logran mayor *ROC AUC score* al predecir menos falsos positivos, sin diferencias significativas con el resto de algoritmos empleados:



Si atendemos al Recall de los casos positivos (bajas), estos son los resultados obtenidos en los modelos aplicados:

Recall de las Bajas de cada modelo

Model	Recall 1
Naive	0.5727
Logistic Regression	0.6796
Decision Tree	0.737
RFC	0.5793
XGBoost	0.3913
Voting Class.	0.7139



Siendo el *Decision Tree Classifier* el modelo con mejores resultados en la fase de entrenamiento y test.

Tabla-resumen con los resultados tanto en el MVP como con todo el dataset:

	Model	MVP				All data	
		One Hot Encoding		Target Encoding		Target Encoding	
		Recall 1's	Accuracy	Recall 1's	Accuracy	Recall 1's	Accuracy
Benchmarks	Naive Model 1	▼ 0,33	■ 0,73				
	Naive Model 2	■ 0,57	▼ 0,65			■ 0,57	▼ 0,66
	Logistic Regression 1	■ 0,55	▼ 0,69				
Models	Logistic Regression 2	▲ 0,70	■ 0,74	▲ 0,70	■ 0,73	▲ 0,68	■ 0,74
	KNN	▼ 0,33	■ 0,74	▼ 0,42	▲ 0,76		
	Decision Tree Class.	▲ 0,72	■ 0,71	▲ 0,73	■ 0,72	▲ 0,74	■ 0,71
	Random Forest Class. 1	▲ 0,61	▲ 0,78	▲ 0,61	▲ 0,79	■ 0,58	▲ 0,79
	Random Forest Class. 2			▲ 0,64	▲ 0,77		
	XGBoost	▼ 0,42	▲ 0,80	▼ 0,44	▲ 0,80	▼ 0,39	▲ 0,81
	Voting Classifier			▲ 0,71	■ 0,75	▲ 0,71	■ 0,73

7. Frontend


El frontend para el usuario final ha sido creado con [streamlit](#) y para ejecutar la app es necesario navegar mediante la terminal hasta la carpeta **notebooks** y ejecutar el siguiente comando:

```
streamlit run app.py
```

A continuación, solo hay que seguir los pasos marcados en la app:

1. Subir un archivo con datos de clientes nuevos no vistos en las fases de train-test. Para ello, se ha preparado una muestra de 5.000 clientes recientes en el archivo *test_A.xlsx*:

Upload file .xlsx



Drag and drop file here
 Limit 200MB per file • XLSX

Browse files

2. Exploración de datos: se puede marcar el *check* para un vistazo rápido del archivo que se acaba de subir:

☐ Show Source Data

3. Visualización de datos con un *countplot*: mediante los 2 desplegados se pueden combinar diferentes variables:

Select feature for X axis:

Tipo Inmueble

Select feature to segment:

Precio Contado

4. *Machine Learning*: seleccionar con el desplegable uno de los algoritmos implementados para realizar la predicción de probabilidad de baja de cada cliente del archivo subido a la app:

Select algorithm:

Please choose an algorithm for prediction

5. Resultados generales de la predicción: tabla con % de churn combinando las variables de Tipo Inmueble y Precio Contado.
6. Por último, se facilita que el usuario pueda elegir el límite mínimo de probabilidad de baja de la predicción:

Show customers with churn probability greater than (%):

0.00

100.00

Selected treshold (%): 70.0

Para, posteriormente, poder extraer en .xlsx en la carpeta **data** los clientes que se encuentren por encima de ese umbral:

Extract to .xlsx