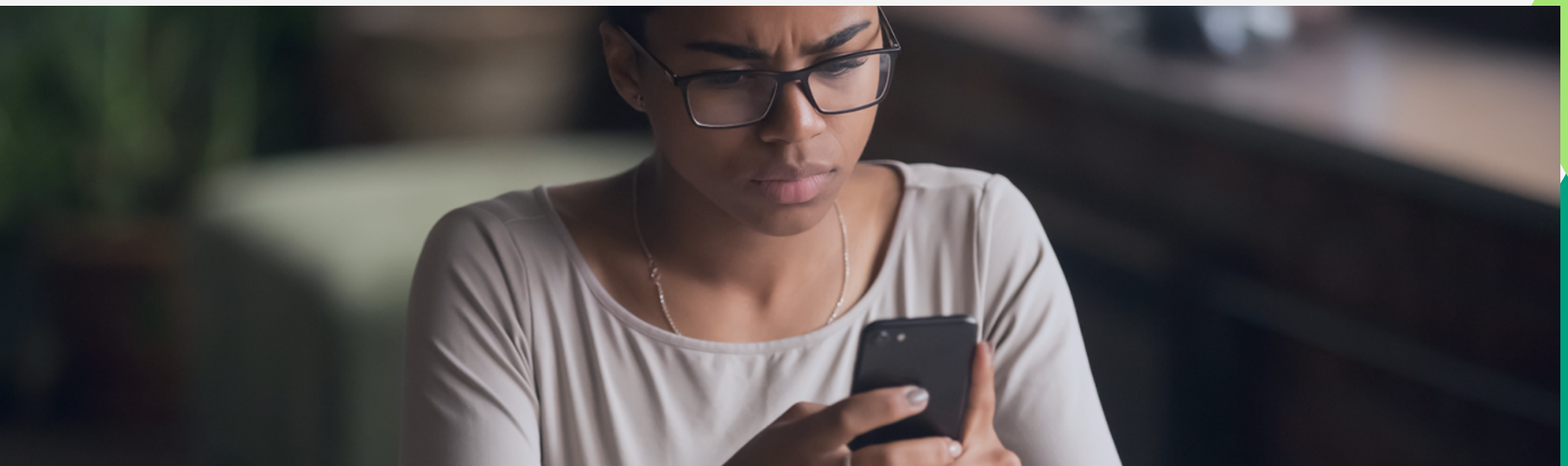
GRUPO 3:

Mirsky & associates

ML para predecir bajas en empresa de telecomunicaciones



GRUPO 3:

Altamirano Marina
Barrio Alejo
Fayó Daniela
Mirsky Alejandro
Siveira Joaquin

Agenda

- Descripción general del dataset
- Antecedentes
- Nuevo enfoque: objetivos
- Desarrollo de las soluciones
  - Análisis de datos
  - Simulación de distintos modelos
  - Análisis de Features
- Construcción del Pipeline
- Deploy del modelo
- Análisis de los resultados y recomendaciones
- Conclusiones

Mirsky & associates

Descripción general del dataset

- Predicción de bajas de clientes en empresa de telecomunicación (Churn)
- Empresa que brinda servicios: telefonía, internet y add-ons

Antecedentes: TP3

Modelado y predicción del Churn

Métrica a privilegiar: Recall

Limpieza y entrenamiento con distintos modelos

Regresión logística
KNN
Naive-Bayes Multinomial
Naive-Bayes Bernoulli

Optimización de los modelos

Variación de hiperparámetros

Varicación del umbral de desición

Mejor modelo obtenido

Regresión logística dado un mayor recall que el resto de los modelos
Recall: 85%

TP Final: Nuevo Enfoque

Objetivos

- Utilizar nuevas técnicas más complejas de predicción: Modelos de ensamble, Feature Selection, y de balanceo de clases.
- Emplear herramientas aprendidas en el curso que faciliten la gestión del modelo (Pipelines, Gridsearch, deploy)
- Comparar los modelos obtenidos para elegir la mejor solución en base a la métrica recall

Desarrollo de las soluciones

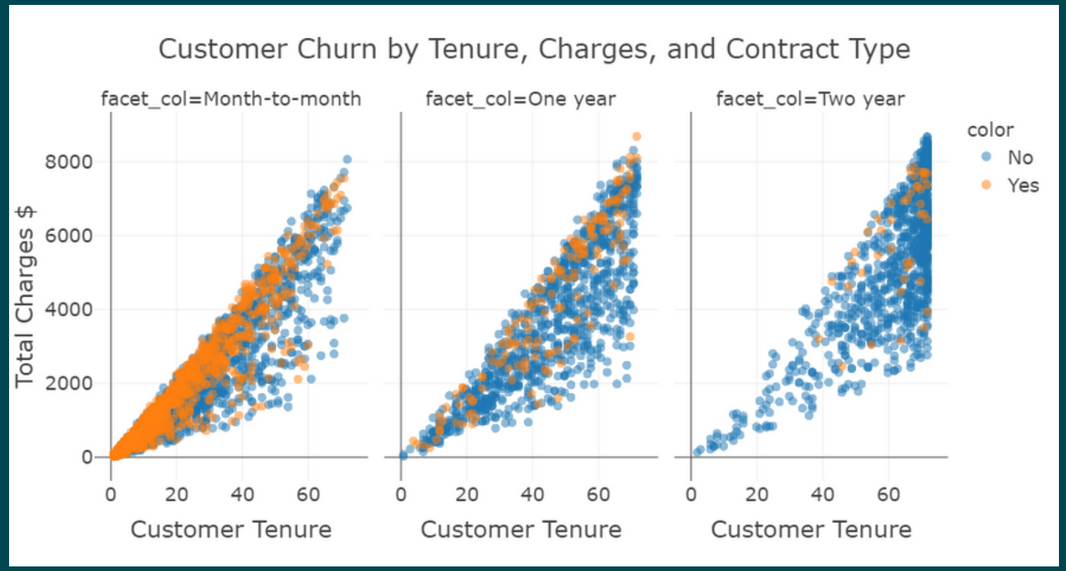
Re-análisis de datos

Simulación de modelos

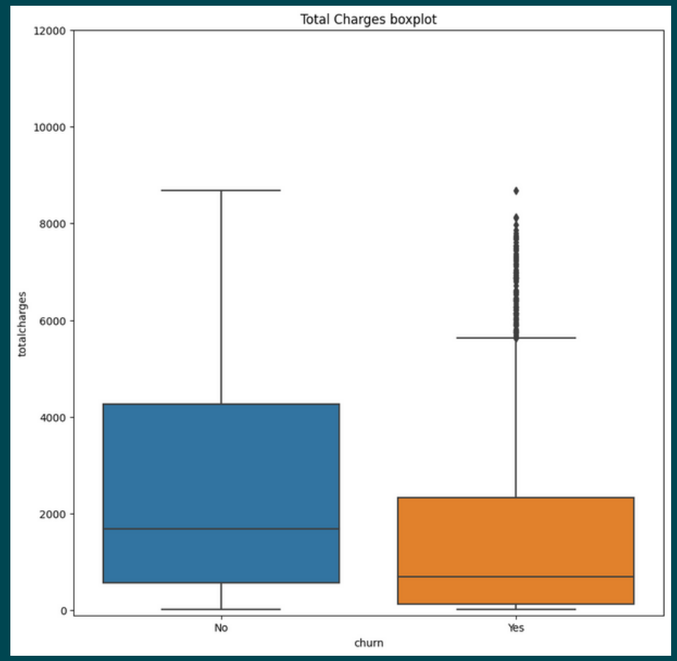
Análisis de features

Desarrollo de las soluciones

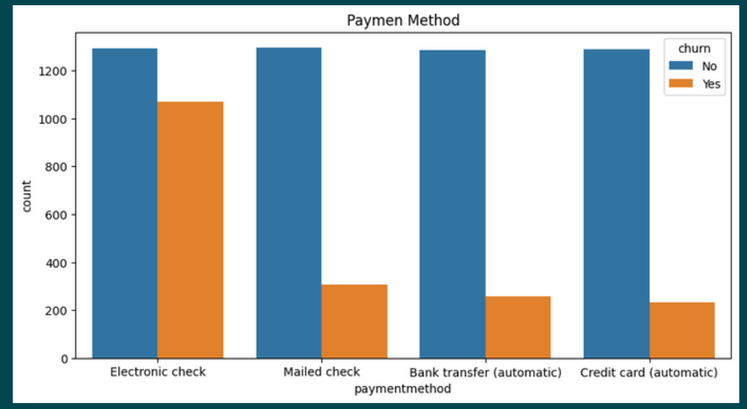
Re-análisis de datos

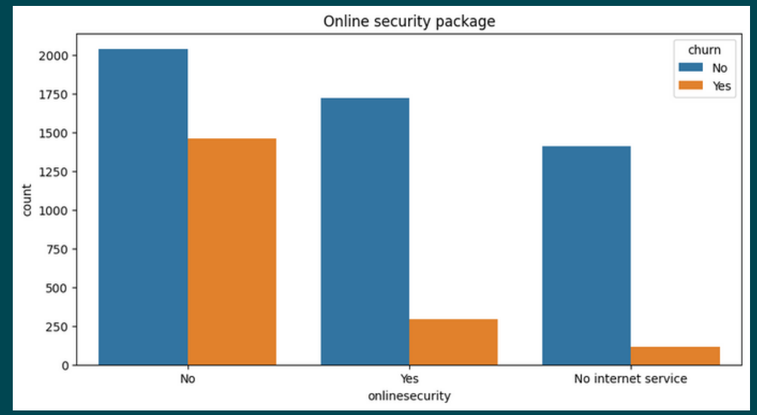


Se puede observar que los clientes con contratos cortos son los que permanecen menos contratando el servicio, pues concentran el mayor porcentaje del churn.



La variable TotalCharges nos muestra que quienes pagan un servicio más alto tienen mayor permanecnia, mientras que quienes contratan un servicio más económico tienen mayor churn





Algunas variables de pago parecieran tener más incidencia en el churn, mientras que la variable del servicio adicional nos muestra que el churn se concentra en quienes no contratan tal paquete. Esto último podría relacionarse con que los clientes que contratan un servicio barato son quienes menos permanencia tienen en la empresa

Desarrollo de las soluciones

Simulación de modelos

Trabajamos sobre los siguientes modelos:

LogisticRegression

GaussianNB

GradientBoostingClassifier

RandomForestClassifier

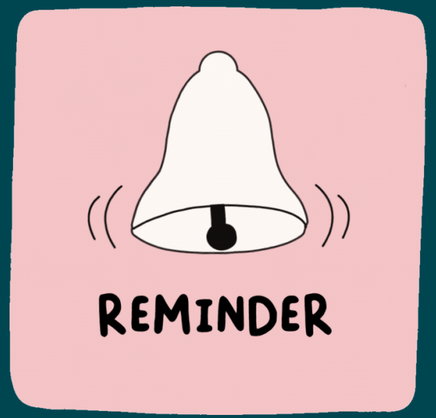
XGBoostClassifier

Desarrollo de las soluciones

Simulación de modelos

Estudiamos mediante diferentes métodos la mejor selección de features

FeatureSelection:
SelectKBest
SelectKpercentil
Recursive Feature Elimination (RFE)



Recordemos que queremos mejorar el recall

Construcción del Pipeline

- Construimos un pipeline.
- Construimos un gridsearch para variar hiperparámetros dentro del pipeline.

Mirsky & associates

Estructura del gridsearch del pipeline

Feature selection:

- SelectKBest(chi2,k=10)
- None

Feature engineering:

- SMOTENC
- RandomOverSampler
- RandomUnderSampler
- None

Preprocesamiento:

- StandardScaler
- MinMaxScaler
- None

Modelos (con sus hiperparámetros):

LogisticRegression
GradientBoosting Classifier
RandomForest Classifier
XGBoost Classifier

Feature selection:

- None

Feature engineering:

- RandomOverSampler

Preprocesamiento:

- MinMaxScaler

Modelos:

GaussianNB

Mirsky & associates

Estructura del gridsearch del pipeline

Feature selection:

- SelectKBest(chi2,k=10)
- None

Feature selection:

- None

```
pipe_full = Pipeline([('feature_selection',SelectKBest(chi2, k = 10)),
    ('feature_engineering',SelectKBest(chi2, k = 10)),
        ('preprocesamiento', StandardScaler()),
    ('switchable', ClassifierWrapper())])


hyperparameters = [

        {
        # Clasificador Logistic regresion
        'feature_selection': [SelectKBest(chi2,k=10),None],
        'feature_engineering':[SelectKBest(chi2, k = 10),smote,oversampler,undersampler, None],
        'preprocesamiento':[StandardScaler(), MinMaxScaler(),None],
        'switchable__estimator': [GaussianNB()],
        'switchable__estimator__var_smoothing': np.logspace(0,-9, num=100),
    },
    {
        # Clasificador Logistic regresion
        'feature_selection': [SelectKBest(chi2,k=10),None],
        'feature_engineering':[SelectKBest(chi2, k = 10),smote,oversampler,undersampler, None],
        'preprocesamiento':[StandardScaler(), MinMaxScaler(),None],
        'switchable__estimator': [LogisticRegression()],
        'switchable__estimator__C': [1, 10, 100, 1000],
        'switchable__estimator__penalty': ['l1', 'l2',],
        'switchable__estimator__solver': ['saga'],
    },
    {
        # GradientBoostClassifier
        'feature_selection': [SelectKBest(chi2,k=10),None],
        'feature_engineering':[SelectKBest(chi2, k = 10),smote,oversampler,undersampler, None],
        'preprocesamiento':[StandardScaler(), MinMaxScaler(),None],
        'switchable__estimator': [GradientBoostingClassifier()],
        'switchable__estimator__n_estimators':[5,50,250,500],
        'switchable__estimator__max_depth':[1,3,5,7,9],
        'switchable__estimator__learning_rate':[0.01,0.1,0.2,0.3,0.4,0.5]
    },
    {
        # RandomforestClassifier
        'feature_selection': [SelectKBest(chi2,k=10),None],
        'feature_engineering':[SelectKBest(chi2, k = 10),smote,oversampler,undersampler, None],
        'preprocesamiento':[StandardScaler(), MinMaxScaler(),None],
        'switchable__estimator': [RandomForestClassifier(random_state = 5)],
        'switchable__estimator__n_estimators': [1,100,200, 500,1000],
        'switchable__estimator__max_features': ['auto', 'sqrt', 'log2'],
        'switchable__estimator__max_depth' : [4,5,6,7,8,9,10,11,12],
        'switchable__estimator__criterion' :['gini', 'entropy']
```

Modelos (con sus hiperparámetros):

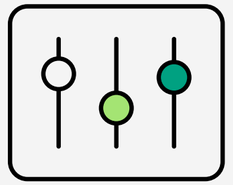
LogisticRegression
GradientBoosting Classifier
RandomForest Classifier
XGBoost Classifier

Modelos:

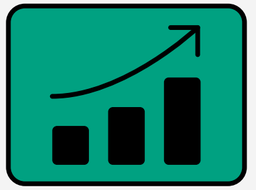
GaussianNB

Deploy del modelo

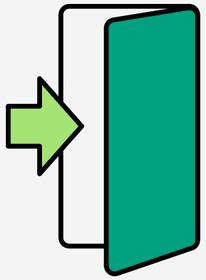
Utilizando Pickle y Flask



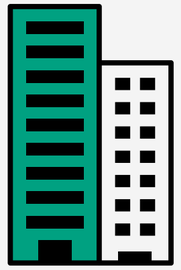
Variamos hiperparámetros de modelos, selección de features, tratamiento de datos para desbalanceo y estandarizaciones



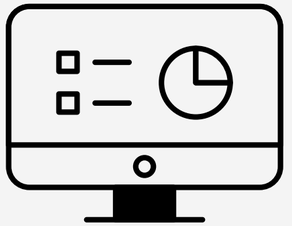
Entrenamos nuestro mejor modelo



Guardamos nuestro mejor modelo



Creamos un servidor con una función para subir una base de datos y otra para predecir el churn



Creamos la notebook del cliente para que pueda predecir si tendrá Churn utilizando el número de cliente

Deploy del modelo

Utilizando Pickle y Flask

In [11]:
```python
def prediccion(numero_ID):
    grilla_usuarios = frozenset(data.customerid)

    if numero_ID in grilla_usuarios:
#         model.fit(X_train,y_train)
        features = data[numero_ID == data.customerid].drop(columns=['customerid','churn_yes']).values.tolist()
        #         print(type({"valores":[feat.tolist()]}))
#         print(numero_ID)
#         r = requests.get("http://10.1.8.31:5020/predict",json={"valores":features.tolist()})
        return requests.get("http://10.1.8.26:5020/predict",json={"valores":features}).json()
#     r.json()
    else:
        return print('no es un usuario registrado')
```

In [36]:
```python
# Cambiando el numero se cambia el ID cliente
numero_ID = data.customerid.iloc[1000]
numero_ID
```

Out[36]: `'9220-ZNKJI'`

In [ ]:
```python

```

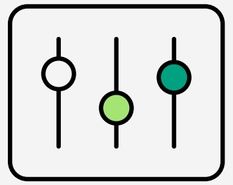
In [38]:
```python
prediccion(numero_ID)
```

Out[38]: `{'El Churn predicho para el cliente es:': 0.0}`

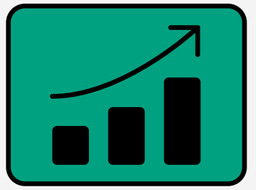
In [24]:
```python
# Si coloco un numero de usuario inexistente pasa lo siguiente
numero_ID_incorrecto='1111-INVENTADO'
```

In [25]:
```python
prediccion(numero_ID_incorrecto)
```

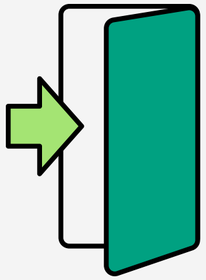
```
no es un usuario registrado
```



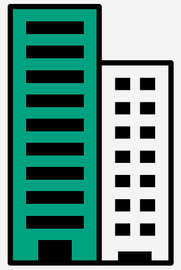
Variamos hiperparámetros de modelos, selección de features, tratamiento de datos para desbalanceo y estandarizaciones



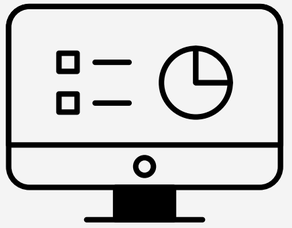
Entrenamos nuestro mejor modelo



Guardamos nuestro mejor modelo



Creamos un servidor con una función para subir una base de datos y otra para predecir el churn



Creamos la notebook del cliente para que pueda predecir si tendrá Churn utilizando el número de cliente

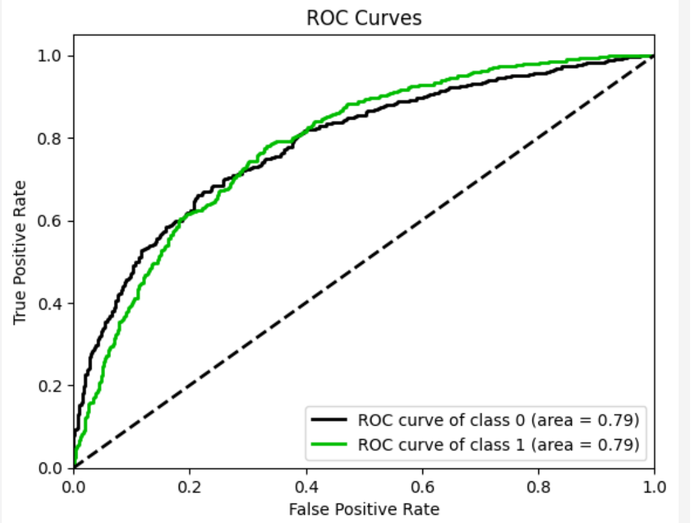
Análisis de los resultados y recomendaciones

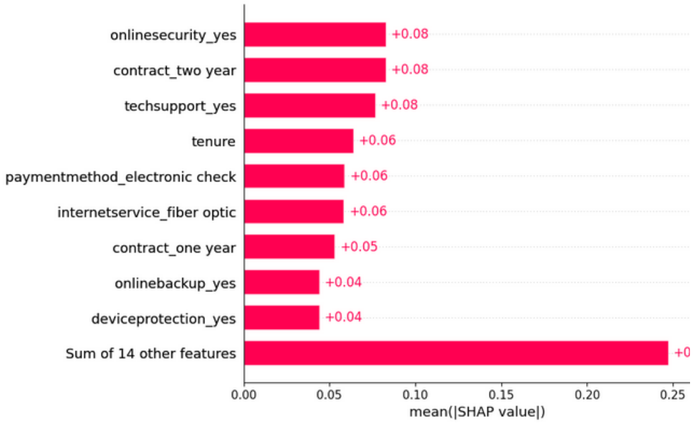
Utilizamos la librería SHAP para ver la importancia de las features según el modelo

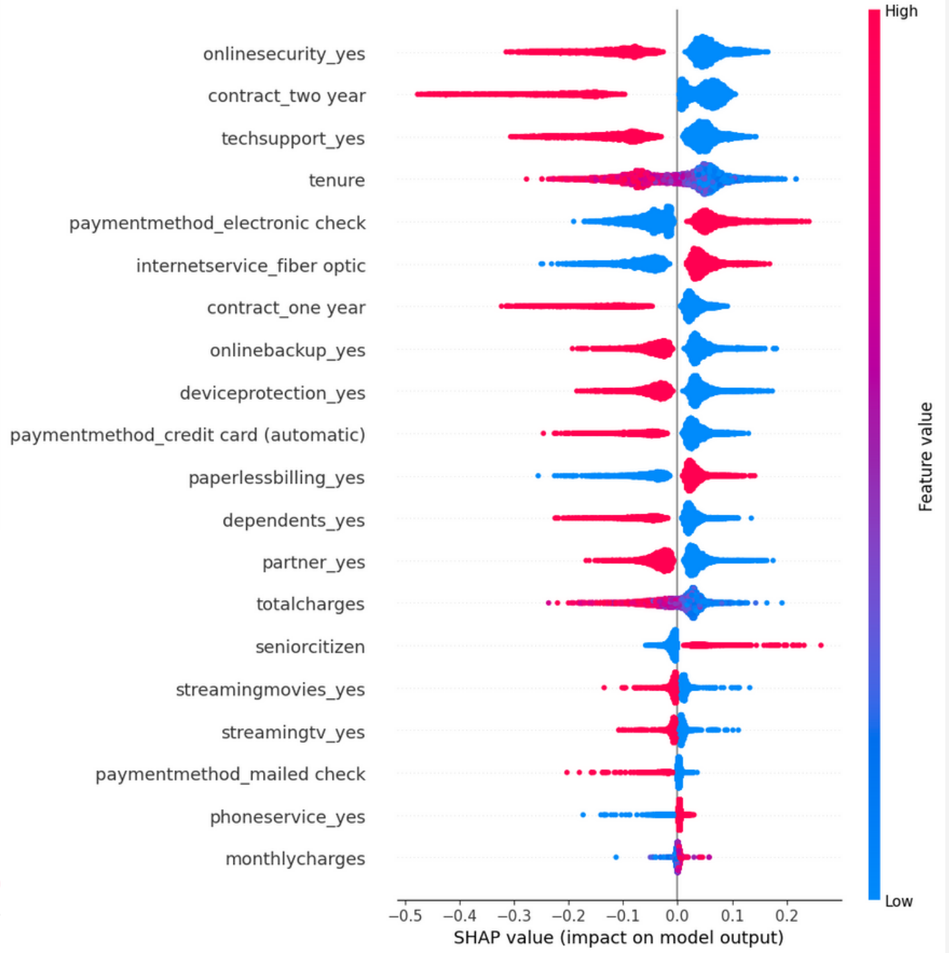
Recall obtenido: 86%

Reales\Predicciones	Churn 0	Churn 1
Churn 0	518	421
Churn 1	63	376

Matriz de confusión para el modelo GaussianNB







Impacto de las variables en el modelo según la librería SHAP

Conclusiones...

- Logramos utilizar nuevas técnicas más complejas de predicción: Modelos de ensamble, Feature Selection, y de balanceo de clases.
- Empleamos herramientas aprendidas en el curso para facilitar la gestión del modelo (Pipelines, Gridsearch, deploy)
- Comparamos los modelos obtenidos para elegir la mejor solución en base a la métrica recall, obteniendo un 86%
- Construimos un modelo mínimo viable (MVP) que predice si el cliente continuará con el servicio o no, para ser operado por un potencial cliente

¡Muchas gracias!