MBA USP ESALQ

Previsão de Churn de assinantes em serviço de Marketplace

Fabricio Rodrigues Zillig
Rodrigo Rosalis Da Silva

CONTEXTO

O cenário empresarial atual é caracterizado por intensa competição e busca incessante por expandir a base de clientes. Essa realidade de competição acirrada, é mais caro atrair novos clientes do que manter os existentes.

Nessa realidade, de abundância de dados e necessidade de retenção de clientes, uma métrica importante a ser observada é o **Customer Churn**

A capacidade de **antecipar** quais clientes estão mais propensos a encerrar sua relação com uma empresa **possibilita a adoção de estratégias proativas** para evitar esse desfecho

MBA USP ESALQ

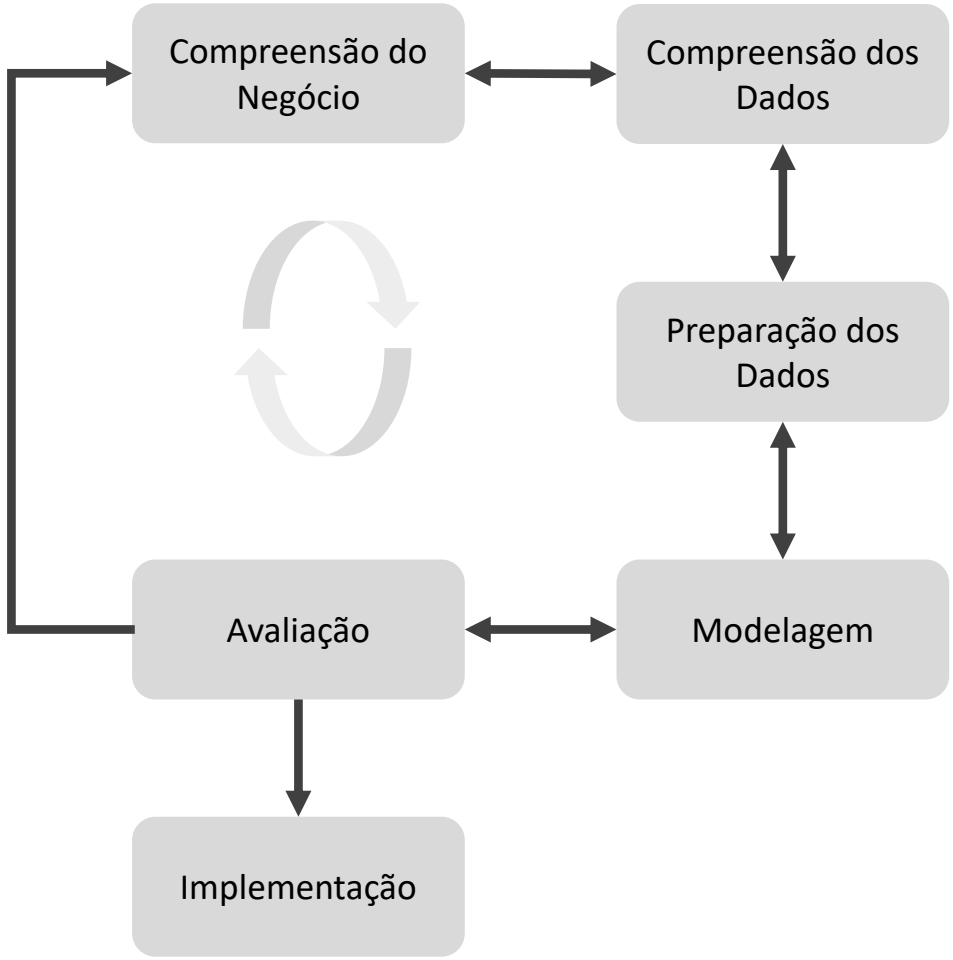
OBJETIVO DO TRABALHO

Este trabalho se propõe a desenvolver um modelo de Machine Learning capaz de prever o churn de assinantes de determinado serviço com base nas características dos clientes e no seu comportamento.

MODELO DE NEGÓCIO

Os dados utilizados pertencem a uma empresa que oferece como serviço a centralização da exibição de produtos de diversos comerciantes em um único local.

Esta estratégia facilita a gestão de presenças online em múltiplos marketplaces, maximizando a visibilidade dos itens ofertados e simplificando operações comerciais.



MÉTODO

O trabalho de pesquisa foi baseado no processo CRISP-DM, um modelo para projetos de Ciência de Dados não proprietário, de uso livre e comum na indústria.

Essa metodologia divide o processo em seis fases: Compreensão do Negócio, Compreensão dos Dados, Preparação dos Dados, Modelagem, Avaliação e Implementação

MBA USP ESALQ

1. COMPREENSÃO DO NEGÓCIO

Duas definições importantes é a definição de Churn e Base Ativa para a empresa. Essas informações geralmente são fornecidas pelas áreas de Negócio ou Marketing. Para este trabalho, usaremos a definição abaixo:

Churn

Considerando que não temos acesso aos dados de assinantes da plataforma, vamos determinar que os clientes que não vendem no período de 3 meses, cancelam a assinatura.

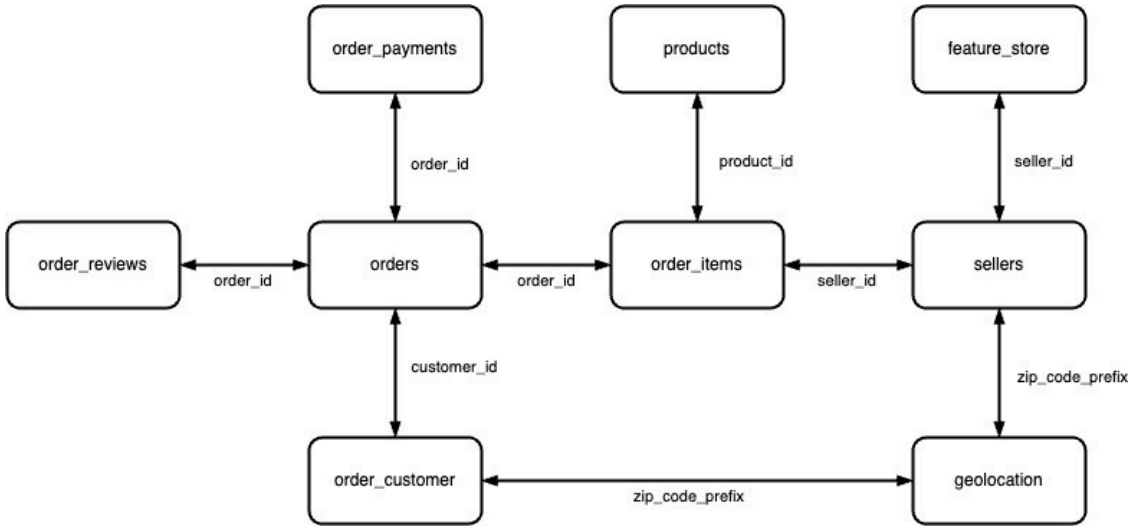
Base Ativa

Nesse contexto, também vamos definir como base ativa (clientes que se relacionam com a empresa) todos os usuários que fizeram alguma venda nos últimos seis meses.

2. COMPREENSÃO DOS DADOS

Os dados foram importados dos arquivos CSV fornecidos pela empresa para um banco de dados relacional SQLite. Essa transformação permitiu uma melhor organização e manipulação das informações durante as etapas subsequentes.

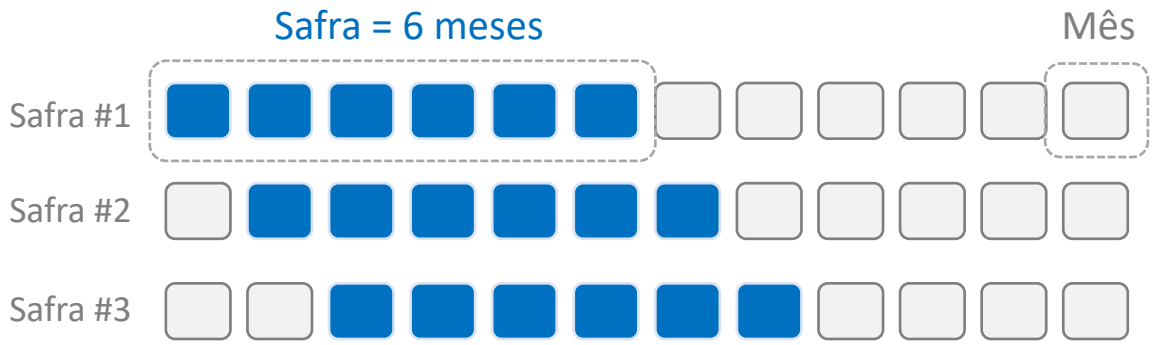
Cada arquivo CSV foi transformado em tabelas dentro desse banco de dados e os registros delas se relacionam através de chaves de valores únicos



3. PREPARAÇÃO DOS DADOS

Uma das etapas mais importantes e trabalhosas no processo de criação de um modelo preditivo. É nesse momento que a criatividade é crucial para transformar os dados brutos em features significativas.

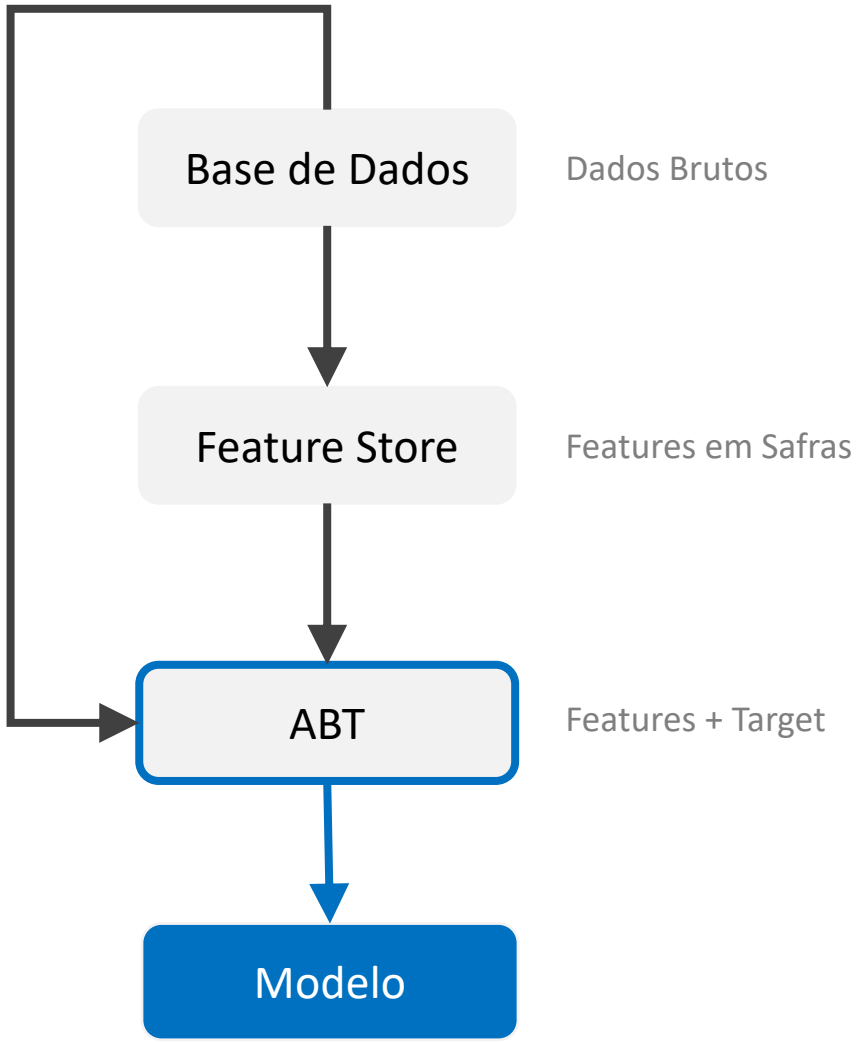
As features criadas incluíram características do cliente, estatísticas relacionadas às vendas, avaliações dos compradores, entre outros e se baseiam em períodos de 6 meses (base ativa). Cada agrupamento chamamos de Safra.

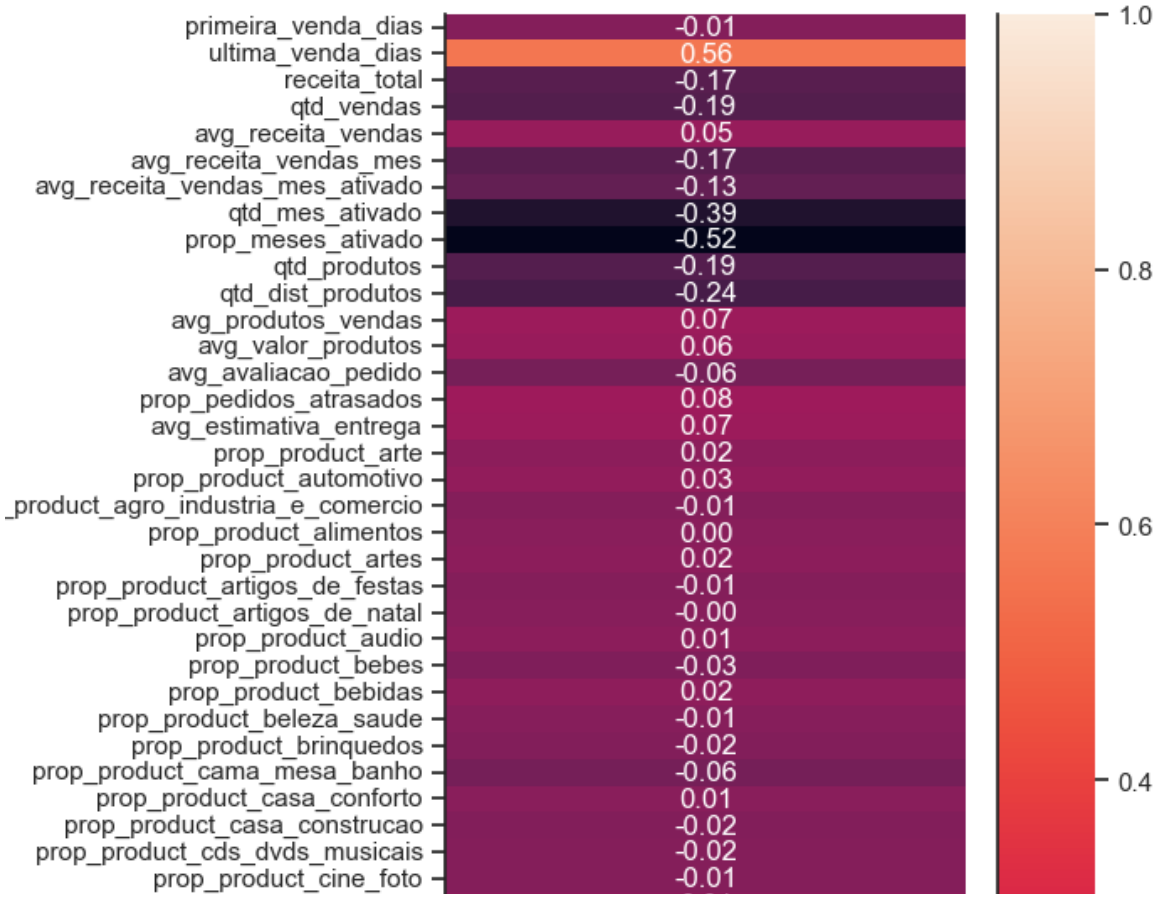


3. PREPARAÇÃO DOS DADOS

A partir da Feature Store e de todas as safras disponíveis construímos a tabela analítica [ABT] que consiste em uma única tabela contendo features pré-definidas e a variável alvo

As covariáveis, ou features, já estavam construídas e coletadas na feature store, já a variável alvo, ou target, construímos através de uma consulta SQL específica.





3.1. EDA

Antes do treinamento, foi realizada uma análise exploratória para entendimento dos comportamentos dos dados usados.

Uma das análises feitas foi a correlação das features com a variável resposta. A variável ultima_venda_dias e prop_meses_ativado aparentam ser variáveis interessantes por apresentarem uma correlação significativa. Isso indica que elas serão importantes para o modelo

4. MODELAGEM

A realizamos o treinamento de um modelo de aprendizado de máquina para problemas de classificação chamado Floresta de Árvores, utilizando Python e a biblioteca scikit-learn

Esse tipo de modelo foi escolhido pelas características do problema que temos: classificar observações com inúmeras features com variâncias e escalas diversas.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.ensemble import RandomForestClassifier
from sklearn.dummy import DummyClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import roc_auc_score, f1_score, recall_score, roc_curve, spe
```

4.1. TREINO E TESTE

A ABT foi dividida em dois dataframes, utilizando a biblioteca Pandas, onde um desses subconjuntos é usado para o treinamento e o outro, chamado Out of Time, utilizado para a avaliação final do modelo.

A base OOT (Out of Time) contém a última safra gerada (mais recente), referente à 01-06-2019 e será usada para avaliar o desempenho do modelo em um cenário próximo do real.

```
[98]
df_oot = df[df['date_ref'] >= '2018-06-01']
df_train = df[df['date_ref'] < '2018-06-01']

print('Train shape:', df_train.shape)
print('OOT shape:', df_oot.shape)
```

```
Train shape: (17015, 77)
OOT shape: (1858, 77)
```

Dataset	Número de Registros
df_train	17.015
df_oot	1.858
Total	**18.873**

4.2. MÉTRICAS

Um passo importante ao treinar um modelo é definir como ele será avaliado para determinar seu desempenho e se ele é melhor que um modelo baseline, por tanto, útil para o uso no mundo real.

Além de um modelo baseline, é importante definir a métrica usada para dizer se um modelo é bom ou não.

Como modelo baseline foi utilizado o DummyClassifier da biblioteca scikit-learn. Esse modelo classifica todos os resultados como 1, ou seja, como 100% de chance de acontecer churn.

4.2. MÉTRICAS

Um método muito comum é o emprego da matriz de confusão e as métricas derivadas dela.

Cada linha é referenciada pelas classes esperadas enquanto as colunas referenciam as classes preditas pelo modelo.

		$f(x)$	
		Classe Predita Positiva	*Classe Predita Negativa*
y	Classe Real Positiva	Verdadeiro Positivo (**VP**)	Falso Negativo (**FN**)
	Classe Real Negativa	Falso Positivo (**FP**)	Verdadeiro Negativo (**VN**)

4.2. MÉTRICAS

$$\text{precisão} = \frac{VP}{VP+FP}$$

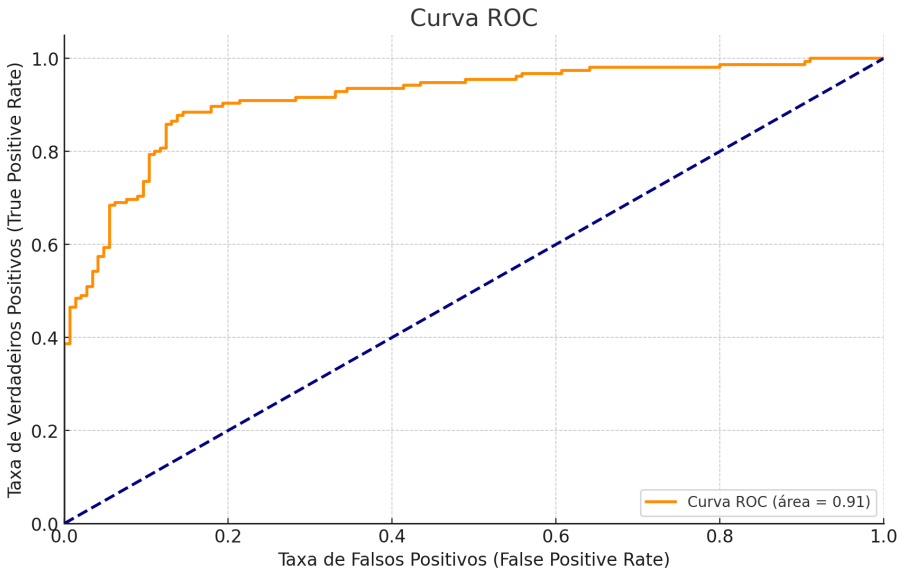
A Precisão corresponde a quantidade de predições positivas corretas sobre a soma de todos os valores classificados como positivos

$$\text{sensibilidade} = \frac{VP}{VP+FN}$$

A Sensibilidade (ou Recall) é a quantidade de predições positivas corretas sobre a soma de todos os valores verdadeiramente positivos

$$\text{F1} = 2 * \frac{precisão * sensibilidade}{precisão + sensibilidade}$$

É a média harmônica entre precisão e sensibilidade. É uma métrica bastante usada por combinar a sensibilidade e precisão em um único indicador



4.2. MÉTRICAS

É comum usarmos um gráfico conhecido como curva ROC (Receiver Operating Characteristic) que apresenta a variação da sensibilidade (Recall ou TPR) em função de (1 – especificidade), conhecido como FPR.

Uma métrica derivada da ROC é a AUC (Area under the ROC curve) que produz valores entre 0 e 1, onde valores próximos de 1 são considerados melhores que valores próximos de 0,5 (equivalente a resultados aleatórios)

4.2. MÉTRICAS ESCOLHIDAS

Por conta da característica do problema, foi escolhido como métrica de desempenho a AUC, que se comporta bem nos casos de datasets desbalanceados, que era o caso.

Como complemento, foi analisado a F1-score, por ser uma métrica que se comporta bem com desbalanceamento, e a Sensibilidade, já que para o problema proposto é importante identificarmos a maior quantidade possível de usuários que cometerão churn e, assim, poder agir de forma preventiva.

AUC

F1 Score

Sensibilidade

4.3. TREINAMENTO

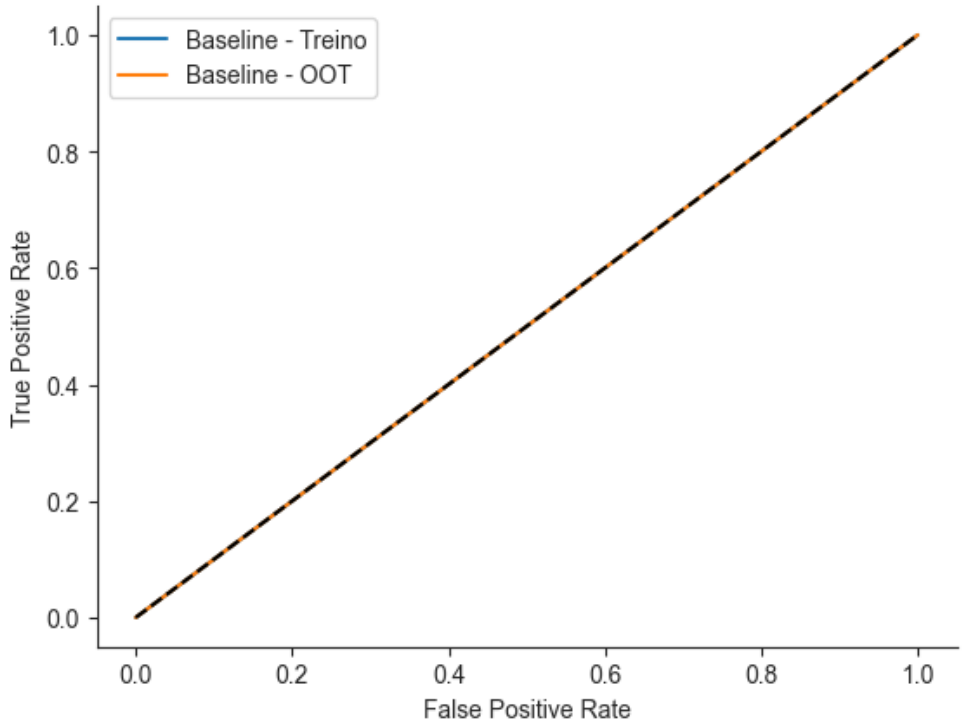
Para o treinamento foi escolhido o uso de duas técnicas em conjunto: GridSearch, para escolha dos hiperparâmetros e K-fold Cross Validation que permite avaliar que determinada performance obtida com o modelo não é específica de uma divisão particular do dataset.

Primeiro treinamos o modelo Baseline, que não tem necessidade de GridSearch ou Cross Validation, em seguida uma Random Forest com hiperparâmetros padrão e por fim uma Random Forest com hiperparâmetros encontrados pelo Grid Search que apresentou melhor resultado de AUC

MODELO BASELINE

O primeiro modelo treinado foi o Baseline. Tanto na base de Treinamento quanto na base de validação (OOT), obtivemos um AUC de 0.5, resultando em uma linha diagonal no gráfico da curva ROC.

Já a Sensibilidade (Recall), obtivemos uma métrica igual a 1, o é esperado dado que todos os registros foram classificados como churn, logo, de todos os churns possíveis, identificamos 100% dos casos.



MODELO BASELINE

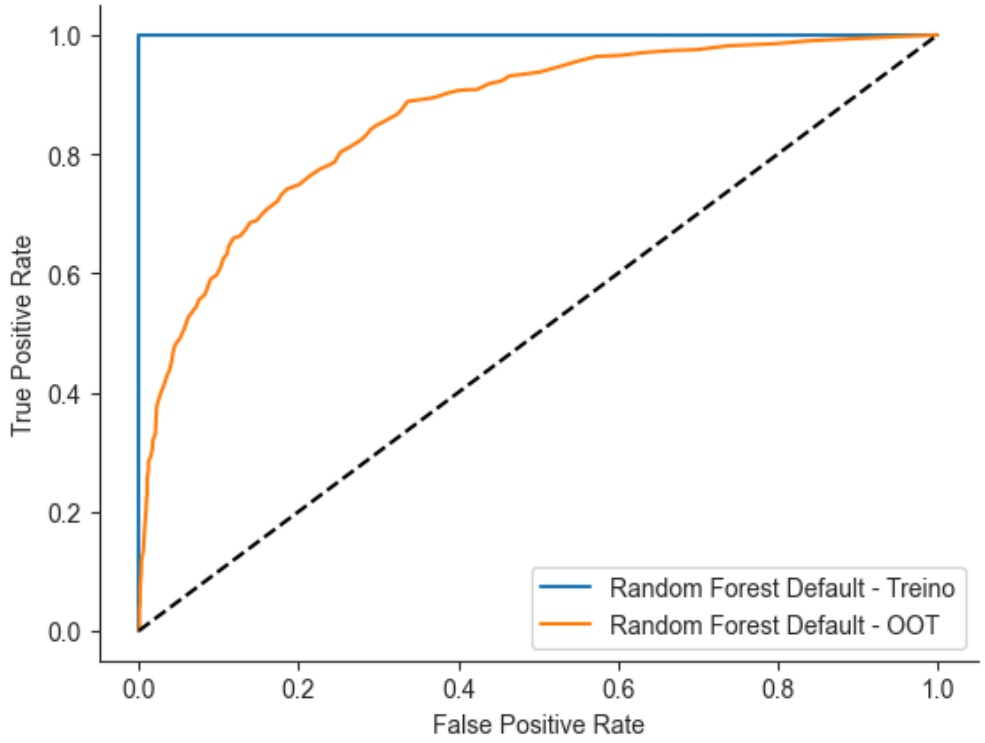
Modelo	Base de Treino			Base de Validação (OOT)		
	AUC	F1	Recall	AUC	F1	Recall
Baseline	0.500	0.480	1.000	0.500	0.494	1.000

4.3. TREINAMENTO

MODELO RANDOM FOREST

Para fins de comparação, treinamos um modelo com os hiperparâmetros padrão definidos pela biblioteca scikit-learn.

Obtivemos uma AUC de 1 no conjunto de Treino e 0.86 no conjunto OOT, indicando que, com os hiperparâmetros padrão, nosso modelo provavelmente está sofrendo de overfit (o modelo se especializa nos dados de treino e por isso não apresenta uma boa capacidade de generalização)



4.3. TREINAMENTO

MODELO RANDOM FOREST

Modelo	Base de Treino			Base de Validação (OOT)		
	AUC	F1	Recall	AUC	F1	Recall
Random Forest (Default)	1.000	1.0000	1.000	0.864	0.681	0.633

4.3. TREINAMENTO

MODELO RANDOM FOREST TUNNED

Após realizar um GridSearch com k-fold Cross Validation variando alguns hiperparâmetros, chegamos aos seguintes valores:

- max_depth igual a 20;
- min_samples_leaf igual a 1;
- min_samples_split igual a 2; e
- n_estimator igual a 500

```
pd.set_option('display.max_colwidth', None)
pd.DataFrame(grid_search.cv_results_)[['rank_test_score','params','mean_test_score', 'mean_fit_time']].sort
```

Python

	rank_test_score	params	mean_test_score	mean_fit_time
113	1	{'max_depth': 20, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 800}	0.877348	16.966870
221	2	{'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 800}	0.877173	23.235194
112	3	{'max_depth': 20, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 500}	0.877003	10.484084
220	4	{'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 500}	0.876794	13.234657
167	5	{'max_depth': 40, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 800}	0.876778	18.123581
...	...	...	...	...
168	266	{'max_depth': 40, 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 10}	0.851242	0.242967
48	267	{'max_depth': 5, 'min_samples_leaf': 4, 'min_samples_split': 10, 'n_estimators': 10}	0.850958	0.084007
6	268	{'max_depth': 5, 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 10}	0.849485	0.102784
216	269	{'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 10}	0.849268	0.221672
162	270	{'max_depth': 40, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 10}	0.846723	0.277226

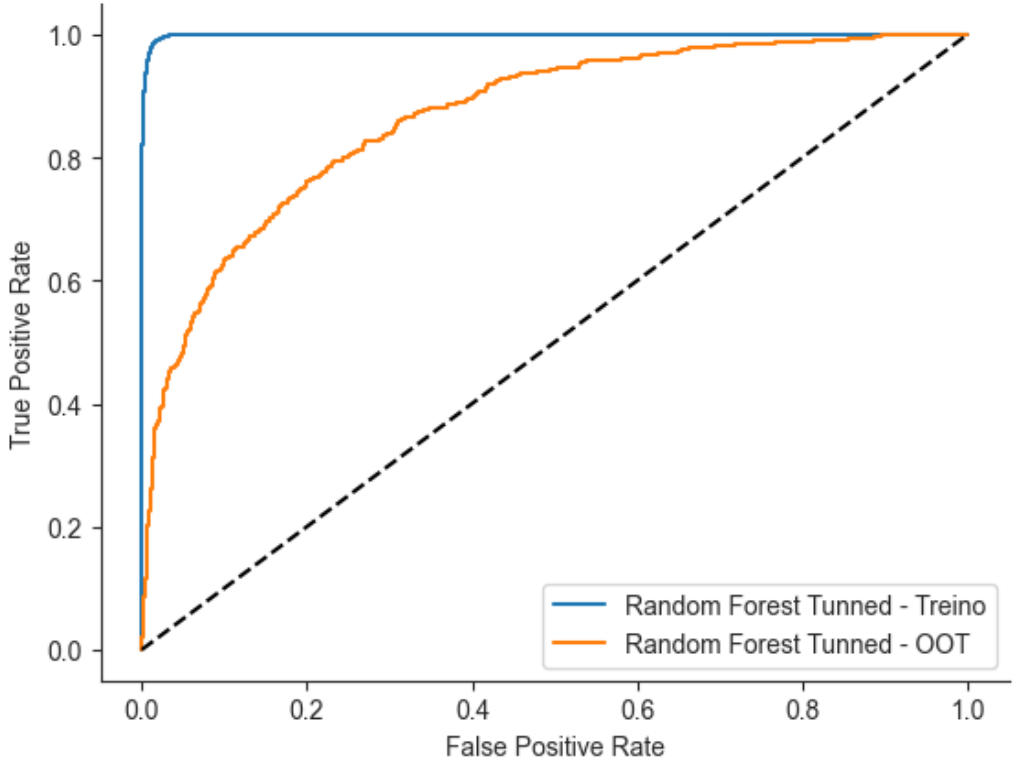
MBA USP ESALQ

4.3. TREINAMENTO

MODELO RANDOM FOREST

Utilizando os hiperparâmetros com melhor custo benefício (AUC vs Tempo de treinamento) conseguimos diminuir um pouco a AUC de Treino e aumentar nossa AUC, chegando a ~ 0.87.

Conseguimos também um ligeiro aumento de F1 score, 0.69 e Sensibilidade, 0.64



MODELO RANDOM FOREST

Modelo	Base de Treino			Base de Validação (OOT)		
	AUC	F1	Recall	AUC	F1	Recall
Random Forest (Tunned)	0.999	0.970	0.956	0.867	0.687	0.638

RESULTADO

Ao final do treinamento e da validação dos três modelos, podemos verificar que o melhor modelo foi a Random Forest Tunned com um AUC de 0.87. Obtivemos com esse modelo o maior F1 Score e Sensibilidade (Recall) sendo 0.69 e 0.64 respectivamente.

Com nosso modelo, conseguimos identificar 64% de todos os usuários que, provavelmente, abandonarão nosso serviço (churn) nos próximos 3 meses

	Base de Treino		
Modelo	AUC	F1	Recall
Baseline	0.500	0.480	1.000
Random Forest (Default)	1.000	1.0000	1.000
Random Forest (Tunned)	0.999	0.970	0.956

	Base de Validação (OOT)		
Modelo	AUC	F1	Recall
Baseline	0.500	0.494	1.000
Random Forest (Default)	0.864	0.681	0.633
Random Forest (Tunned)	0.867	0.687	0.638

RESULTADO

Feature	Feature Importance
ultima_venda_dias	0.182
prop_meses_ativado	0.093
avg_receita_venda_mes	0.080
primeira_venda_dias	0.061
receita_total	0.057
avg_receita_vendas_mes_ativado	0.052
qtd_vendas	0.047
avg_receita_vendas	0.046
avg_valor_produtos	0.046
qtd_produtos	0.042

Outra informação interessante é a importância das features usadas no treinamento, que é determinada observando-se quão efetivo cada feature é na redução do erro nas decisões das árvores individuais. Essa medida é calculada para cada variável em todas as árvores, sendo depois agregada e normalizada.

Features com valores de importância mais altos são consideradas mais influentes para o modelo.

CONCLUSÃO

A solução desenvolvida se mostrou madura e viável de ser utilizada, mas também apresenta espaço para evoluções futuras. A exploração de outros algoritmos de machine learning e a criação de novas features representam oportunidades de refinamento e aumento da acurácia das predições.

Desenvolver um projeto que abrange desde a coleta e o tratamento de dados até a criação de features pertinentes para a resolução do problema de churn em um marketplace representou um desafio interessante e extremamente valioso para o aprendizado sobre problemas reais. O uso da metodologia CRISP-DM mostrou um caminho claro a ser seguido, além de proporcionar uma imersão profunda nos dados, facilitando a compreensão das nuances que caracterizam situações de abandono de clientes, e destacando a importância desse tipo de trabalho para a tomada de decisão