UNIVERSIDADE FEDERAL DO ABC

ENGENHARIA DE INFORMAÇÃO

Marcelo dos Santos Nascimento Vieira

Comparação de técnicas baseadas em árvores de decisão para o problema de análise de satisfação do cliente

Santo André, SP

2025

Marcelo dos Santos Nascimento Vieira

Comparação de técnicas baseadas em árvores de decisão para o problema de análise de satisfação do cliente

Trabalho apresentado como requisito parcial para a Conclusão do Curso de Engenharia de Informação na Universidade Federal do ABC.

Orientador: Prof. Dr. Kenji Nose Filho

Santo André

2025

RESUMO

Com a crescente utilização de *machine learning* no mundo corporativo, este trabalho aborda o desempenho de algoritmos de *machine learning* baseados em árvores de decisão. O objetivo é comparar o desempenho dos algoritmos utilizados, sendo eles a árvore de decisão, o *random forest* e o XGBoost. Para isto, foi utilizada uma base de dados disponibilizada pelo Banco Santander, que pode ser encontrada na plataforma Kaggle, onde o problema apresentado é classificar os clientes em duas classes, satisfeitos ou insatisfeitos, para posteriormente aplicar uma campanha de retenção. Após a classificação em clientes satisfeitos ou insatisfeitos calcula-se o lucro através dos resultados. Este lucro é dado pelo resultado da classificação, através de um ganho ou uma perda associada à classificação de cada cliente. A ideia é maximizar o lucro, ou seja, ter o maior número de verdadeiros positivos e o menor número de falsos positivos. As principais métricas utilizadas foram a área sob a curva e o lucro, e o algoritmo que teve o melhor desempenho foi o *random forest*, possivelmente pelo fato de ser mais robusto que a árvore de decisão e menos suscetível ao sobreajuste e ser menos complexo do que o XGBoost.

Palavras-chave: Machine Learning, Modelos Baseados em árvores, classificação

ABSTRACT

With the increasing use of machine learning in the corporate world, this work explores the performance of tree-based machine learning algorithms. The objective is to compare the performance of three algorithms: Decision Tree, Random Forest, and XGBoost. For this purpose, a dataset provided by Banco Santander was used, which is available on the Kaggle platform. The presented problem involves classifying customers into two categories—satisfied or dissatisfied—in order to apply a retention campaign afterward. After classifying customers as satisfied or dissatisfied, profit is calculated based on the classification results. This profit is determined by the outcome of the classification, through a gain or loss associated with each customer's classification. The goal is to maximize profit, that is, to achieve the highest number of true positives and the lowest number of false positives. The main metrics used were the Area Under the Curve (AUC) and profit, and the algorithm that showed the best performance was Random Forest, possibly due to its greater robustness compared to the Decision Tree, being less prone to overfitting and less complex than XGBoost.

Keywords: Machine Learning, Tree-Based Models, Classification

SUMÁRIO

1. INTRODUÇÃO

Em um cenário de avanço da Inteligência Artificial (IA) alinhado com o avanço computacional, empresas têm usado cada vez mais de algoritmos de Machine Learning para auxílio de resolução de problemas e soluções em diversas áreas como Crédito, Produtos, Saúde, etc. Um dos temas que a IA tem auxiliado é em relação ao conhecimento dos seus clientes. Para isso, algoritmos de sistemas de recomendação têm sido utilizados com o objetivo de recomendar produtos com mais assertividade, desta forma, é possível muitas vezes reduzir custos e aumentar as vendas. Em um mercado cada vez mais competitivo é fundamental que as empresas conheçam melhor os seus clientes sempre tentando retê-los através de produtos de mais qualidade e necessidade, e através de campanhas atrativas.

Este trabalho tem como objetivo comparar o desempenho de alguns modelos de aprendizado de máquina baseados em árvores de decisão, como a própria árvore de decisão, o *random forest* e o *XGBoost*, aplicados ao problema de análise de satisfação do cliente. Para isso utilizou-se o dataset da competição "*Santander Customer Satisfaction*" disponível pela plataforma Kaggle.Esse dataset consiste em uma base de dados contendo 76020 registros e 371 colunas dividida em base de treino e teste. Este trabalho tem como foco apenas a Parte A do problema apresentado, como descrito abaixo:

Parte A: Um falso positivo ocorre quando classificamos um cliente como insatisfeito, mas ela não se comporta como tal. Neste caso, o custo de preparar e executar uma ação de retenção é um valor fixo de 10,00 por cliente. Nada é ganho pois a ação de retenção não é capaz de mudar o comportamento do cliente. Um falso negativo ocorre quando um cliente é previsto como satisfeito, mas na verdade ele estava insatisfeito. Neste caso, nenhum dinheiro foi gasto e nada foi ganho. Um verdadeiro positivo é um cliente que estava insatisfeito e foi alvo de uma ação de retenção. O benefício neste caso é o lucro da ação (100,00) menos os custos relacionados à ação de retenção (10,00). Por fim, um verdadeiro negativo é um cliente insatisfeito e que não é alvo de nenhuma ação. O benefício neste caso é zero, isto é, nenhum custo, mas nenhum lucro. O objetivo é maximizar o lucro esperado por cliente considerando o contexto descrito, sendo que:

- Falso Positivo (FP): cliente marcado como insatisfeito mas estava satisfeito. custo: -10,00 lucro: 0,00

- Falso negativo (FN): Cliente previsto como satisfeito mas estava insatisfeito. custo: 0,00 lucro: 0,00

- Verdadeiro Positivo (VP): Cliente previsto como insatisfeito e é insatisfeito. custo: -10,00 lucro: 100,00

- Verdadeiro Negativo (VN): Cliente previsto como satisfeito e que não é alvo de nenhuma ação. custo: 0,00 lucro: 0,00

No Capítulo 2 será apresentada uma pequena fundamentação teórica sobre os algoritmos de aprendizado de máquina utilizados e também as métricas de avaliação dos

resultados. No Capítulo 3 apresentamos a base de dados que foi utilizada, descrevemos a etapa de pré-processamento dos dados e ajuste dos hiperparâmetros. Já no Capítulo 4 são apresentados os resultados obtidos e, por fim, no Capítulo 5 são apresentadas as conclusões deste trabalho.

2. FUNDAMENTAÇÃO TEÓRICA

O aprendizado de máquina é uma área da inteligência artificial onde utiliza-se de um conjunto de dados para fazer mensurações de valor, classificações ou agrupamentos. O aprendizado de máquina pode ser dividido principalmente em algoritmos de aprendizado supervisionado e não supervisionado [JAMES et al., 2021].

O aprendizado supervisionado é quando existe a informação do rótulo, por exemplo, numa área de fraude há uma lista de clientes onde sabe-se quem são os clientes fraudulentos e quem não são, desta forma, imagine que se tem uma lista de variáveis (informações) desses clientes como salário, tipo de cartão, produto que mais consome, valor da última compra, horário da última compra, valor médio da conta nos últimos sete dias, etc. A ideia do aprendizado supervisionado é mapear o comportamento através das variáveis de tal forma que o algoritmo consiga aprender esse comportamento e prever se determinado cliente é fraudulento ou não. Nesse exemplo citado o algoritmo supervisionado seria de classificação, mas também existem casos onde o algoritmo supervisionado é de regressão, onde o rótulo nesse caso não seria uma classe (geralmente utilizamos 0 ou 1), mas sim um valor real, como, por exemplo, a estimação do valor (preço) de uma casa.

Já nos algoritmos de aprendizado não supervisionado a diferença é que não existe o rótulo, ou seja, no caso do exemplo de fraude dado acima, não se tem conhecimento de quem são os clientes fraudulentos e quem não são, contudo têm-se as variáveis, e talvez dessas variáveis é possível fazer agrupamentos por comportamentos similares com o objetivo de compreender esses clientes e ver como podem ser classificados. Geralmente algoritmos de aprendizado não supervisionado não costumam ser utilizados em áreas de fraude, mas sim em áreas de CRM (do inglês *Customer Relationship Management*) e/ou Produto, devido ao fato de justamente ter o objetivo de agrupar os clientes por similaridade em relação às variáveis e entender seu comportamento.

Existem outras classes de algoritmos como os semi supervisionados, contudo neste trabalho foi utilizado somente aprendizado supervisionado, devido a isso não será aprofundado sobre os demais. Os algoritmos de aprendizado supervisionado utilizados neste trabalho foram a Árvore de Decisão, o *Random Forest* e o XGboost.

2.1. Árvore de Decisão

A árvore de decisão é um algoritmo supervisionado não paramétrico, ou seja, não existe nenhum pressuposto em relação ao conjunto de dados e pode ser usado tanto para classificação como para regressão [JAMES et al., 2021]. Árvores de decisão são formadas por nós, ramos e folhas. O primeiro nó da árvore é denominado nó raiz (nó pai) e os demais nós são denominados nós filhos. Cada nó tem uma variável, no caso de um modelo de classificação a variável é selecionada pela capacidade de conseguir dividir os dados de forma homogênea em relação ao alvo (rótulo).

Suponha que desejamos prever se um cliente é de alta renda ou não, e a variável que melhor separa o conjunto de dados de forma homogênea em relação ao alvo é a quantidade de imóveis que este cliente possui, desta forma, a variável quantidade de imóveis estará no nó raiz, sendo uma árvore binária o nó raiz terá dois nós filhos, supondo que o corte em relação a variável quantidade de imóveis seja de possuir três imóveis, clientes que tem três imóveis ou mais irão para o nó da direita enquanto que clientes com menos de três imóveis irão para o nó da esquerda, essas ramificações entre nós com variáveis com potencial de dividir os dados de forma homogênea em nós filhos se repete até se acabar a quantidade de variáveis ou atingir um critério de parada, que pode ser a quantidade de variáveis máxima, assim como a quantidade de registros numa subdivisão, um nó que não se subdivide é chamado de folha.

A seleção de variáveis em relação ao nó no caso de um problema de classificação é feita através do índice de Gini ou da Entropia. Gini e Entropia são métricas que mensuram a desigualdade. Quanto maior o seu valor, mais homogênea é a sua distribuição.

$$\text{GINI} = 1 - \sum_{i=1}^{k} p_i^2,$$

onde p_i é a proporção (ou probabilidade) de elementos pertencentes à classe i e k é a quantidade de classes.

$$\text{Entropia} = -\sum_{i=1}^{k} p_i \log_2 p_i,$$

onde p_i é a proporção (ou probabilidade) de elementos pertencentes à classe i e k é a quantidade de classes.

O algoritmo de árvore de decisão tem uma grande vantagem que é a capacidade de ser facilmente interpretável, em um ambiente de negócio onde se utiliza árvore de decisão para predizer, por exemplo, modelos de crédito, com o objetivo de determinar quais clientes terão crédito aprovado, através da árvore de decisão facilmente consegue-se saber quais são as principais variáveis, assim como seus respectivos cortes aplicados, podendo assim entender e justificar porque o algoritmo classificou um cliente como crédito aprovado e outro como crédito negado.

Contudo, também existem desvantagens, árvores de decisão são muito sensíveis ao conjunto de dados, desta forma, qualquer alteração no conjunto de dados pode mudar significativamente o resultado. Por isso, a importância da amostra utilizada ser representativa em relação à população objetivo. Além do ponto citado, sabe-se também que árvores de decisão são suscetíveis ao *overfitting*, que é quando o modelo não generaliza bem para dados não vistos (dados de teste e validação) e tem bom desempenho para dados vistos (dados de treino). Por exemplo, supondo um modelo de crédito, onde os dados para desenvolver/treinar esse modelo são de 2021 e para testar e validar são de 2022, caso o modelo tenha um bom desempenho nos dados de 2021 e não tenha um desempenho similar nos dados de 2022, pode-se dizer que ocorreu *overfitting*. Existe também o

underfitting que é quando o modelo não apresenta bom desempenho nos dados de treino, tampouco nos dados de teste e validação.

Uma forma de lidar com o *overfitting* é através do ajuste dos hiperparâmetros de regularização, conforme ajustados eles podem aumentar e/ou reduzir o viés e/ou a variância do modelo, o que impacta de forma direta no *overfitting*. Segue abaixo alguns dos hiperparâmetros mais utilizados no desenvolvimento de um modelo de árvore de decisão:

- **max_feature:** Número máximo de recursos avaliados para divisão em cada nó.
- **max_leaf_nodes**: Número máximo de nós folha.
- **min_samples_split:** Número mínimo de amostras que um nó deve ter antes de poder ser dividido.
- **min_samples_leaf:** Número mínimo de amostras que um nó folha deve ter para ser criado.

2.2. Random Forest

O *Random Forest* é um algoritmo baseado em árvores de decisão, e assim como a árvore de decisão, pode ser utilizado tanto para regressão como para classificação. Consiste em um conjunto de árvores, onde em cada árvore tem-se um subconjunto da amostra e das variáveis, através de um processo de amostragem com reposição. O processo do treino de cada árvore é feito de forma independente, ou seja, pode ser feito de forma paralela. Em relação à previsão final de cada observação, no caso de um modelo de regressão considera-se a média final de todas as árvores e no caso de classificação considera-se a classe majoritária entre as previsões de cada árvore [GÉRON, 2019].

Devido à aleatoriedade contida na seleção das variáveis e ao fato do resultado final ser resultado de um conjunto de árvores, o *random forest* é mais robusto ao *overfitting* do que a árvore de decisão. Outro ponto vantajoso além do citado acima é a facilidade de implementação e a sua flexibilidade podendo ser usado tanto para regressão como para classificação.

No entanto, pode ser mais lento que algoritmos mais simples, por ter maior complexidade computacional.

O desempenho do *random forest* pode ser ajustado por meio de diversos hiperparâmetros. Alguns dos mais importantes são:

- **n_estimators:** Número de árvores na floresta. Um valor maior pode melhorar a precisão, mas também aumenta o custo computacional.
- **max_depth:** Profundidade máxima das árvores. Controla o quão complexas as árvores podem ser, evitando sobreajuste.
- **min_samples_split:** Número mínimo de amostras necessárias para dividir um nó. Valores mais altos levam a árvores mais simples.

- **min_samples_leaf:** Número mínimo de amostras em uma folha terminal. Um valor maior ajuda a reduzir o *overfitting*.
- **max_features:** Número máximo de atributos considerados para cada divisão. Pode ser *sqrt*, *log2* ou um número inteiro.
- **bootstrap:** Define se o método de amostragem com reposição que será utilizado. Quando *False*, todas as amostras são utilizadas para cada árvore.
- **oob_score:** Se verdadeiro, usa as amostras fora da amostragem (*out-of-bag*) para avaliar o modelo sem necessidade de um conjunto de validação separado.

2.3. XGBoost

O XGBoost (*Extreme Gradient Boosting*) é um algoritmo de aprendizado de máquina baseado em árvores de decisão que utiliza a técnica de *boosting* para otimizar previsões. Desenvolvido para ser eficiente, flexível e altamente performático, o XGBoost é amplamente utilizado em competições de ciência de dados e aplicações do mundo real devido à sua capacidade de lidar com grandes volumes de dados e oferecer alta precisão, esse algoritmo pode ser utilizado tanto para classificação como para regressão [GÉRON, 2019].

O *boosting* é uma técnica usada em *machine learning* com o objetivo de melhorar o desempenho de modelos preditivos combinando modelos fracos de tal forma que através disso seja obtido um modelo forte. No *boosting* isso é feito de forma sequencial, onde cada modelo novo tenta corrigir os erros do modelo anterior, no caso do XGBoost o algoritmo utilizado como modelo fraco é uma árvore de decisão, ou seja, um conjunto de árvore é criado de forma sequencial, onde a predição de cada árvore será baseada nos erros da árvore anterior. A saída final do modelo é uma combinação ponderada dos modelos treinados.

As principais vantagens é a sua capacidade de atingir alta performance, possui regularização interna, onde o uso é feito através de seus parâmetros e lida bem com dados faltantes. No entanto, é muito sensível à combinação e ajuste de parâmetros, e o seu treinamento é mais lento do que algoritmos mais simples, como a árvore de decisão e o *random forest*. Os principais hiperparâmetros do XGBoost são:

- **max_depth:** Define a profundidade máxima de cada árvore. Árvores mais profundas conseguem capturar padrões mais complexos, mas também aumentam o risco de *overfitting*.
- **min_child_weight:** Representa o número mínimo de instâncias (ou soma dos pesos) necessários em um nó filho para que ele seja dividido.
- **gamma:** É o ganho mínimo necessário na função de perda para que uma divisão seja realizada em um nó. Se o ganho for menor que esse valor, a divisão não acontece. Serve como regularização e ajuda a evitar o *overfitting*.
- **subsample:** Define a fração de amostras que será usada para treinar cada árvore.

- **lambda:** É o coeficiente de regularização L2. Ajuda a controlar o tamanho dos coeficientes e a evitar o *overfitting*.
- **alpha:** É a regularização L1, semelhante à usada na regularização L2. Além de controlar o modelo, ela também pode forçar alguns pesos a zero, o que ajuda na seleção de variáveis.
- **learning_rate:** Define o quanto cada nova árvore contribui para o modelo final. Valores menores como 0.01 ou 0.1 deixam o treinamento mais lento, contudo mais robusto.
- **n_estimators:** Define o número de árvores que serão treinadas. Quanto maior, mais potente o modelo, mas pode aumentar o risco de *overfitting* se não for bem regularizado.

2.4. Métricas

Ao desenvolver um modelo de *machine learning* é importante conseguir avaliar a performance deste modelo, para isso, existem algumas métricas, no caso de modelos de classificação são:

- Acurácia: Percentual total de acertos do modelo considerando tanto os casos verdadeiros positivos como os verdadeiros negativos, ou seja, supondo um modelo de fraude, onde é utilizada uma base de clientes onde se tem clientes fraudadores e não fraudadores, a acurácia irá indicar o percentual total de acertos (casos positivos e negativos). O que pode não ser indicado para bases desbalanceadas.
- Precisão: Retorna o percentual dos registros que o modelo classificou como positivos que de fato eram positivos, ou seja, dos registros que o modelo classificou como fraudadores, qual percentual de fato é fraudador.
- Revocação ou Sensibilidade: Retorna o percentual de positivos reais que realmente foram classificados como positivos, ou seja, dos registros que realmente são fraudulentos quantos o modelo acertou.
- F1-Score: Média Harmônica entre Precisão e Sensibilidade
- AUC ROC Curve: Mostra a taxa de verdadeiros positivos (Recall) vs falsos positivos. AUC (Área sob a curva) mede o desempenho geral, sendo 1 excelente, 0.5 aleatório e < 0.5 pior do que aleatório.

3. METODOLOGIA

3.1. Base de dados

Neste trabalho, foi utilizado o dataset da competição "Santander Customer Satisfaction" disponível pela plataforma Kaggle [Jimenez e Cukierski, 2018], que possui um total de 76020 registros e 371 colunas.

Cada linha corresponde a um elemento do banco de dados e cada coluna corresponde a uma variável de entrada (um atributo). As colunas não tem identificação em relação ao conteúdo das variáveis, ou seja, não é possível saber o significado da variável.

Para a etapa da modelagem foi feita a separação em treino e teste, sendo 70% da base destinada para treino e 30% para teste. Para o treino foi usada uma amostra estratificada em relação à *target*.

3.2. Pré-processamento dos dados

Com relação ao pré-processamento dos dados foram feitas verificações como existência de duplicidade, remoção de variáveis com valores constantes e/ou faltantes (*missing values*), conforme detalhado abaixo:

- Duplicidade: Não foram encontradas duplicidades na base.
- Valores Constantes: Foram retiradas da base 34 variáveis devido ao fato de não ajudarem na discrinação do modelo por serem constantes.
- *Missing Values*: Não foram encontrados valores faltantes na base.

Para a seleção de variáveis foi utilizado o *Information Value* (IV). O IV é uma técnica estatística que avalia o poder preditivo de uma variável em relação à target, essa técnica é bem utilizada no mercado de crédito e fraude [SIDDIQI, 2005]. Por exemplo, dado um conjunto de variáveis podemos ver quais variáveis têm maior poder preditivo. Segue cálculo do IV abaixo:

$$\text{IV} = \Sigma(\%\ nao\ eventos\ -\ \%\ eventos)\ *\ WOE,$$

onde WOE é o peso de evidência que é calculado da seguinte forma:

$$\text{WOE} = \text{Ln}(\%\ nao\ eventos / \%\ eventos).$$

Neste trabalho o IV foi utilizado com o objetivo de reduzir a quantidade de variáveis utilizadas no desenvolvimento do modelo. Após o cálculo do information value, foram mantidas apenas as variáveis com IV acima de 0.1. Desta forma, restaram apenas 15

variáveis, as variáveis mantidas e seus respectivos valores de IV podem ser vistos na Tabela 1 abaixo.

Tabela 1. Lista de variáveis com os maiores valores de IV.

Variável	**IV**
var15	0.76
saldo_var30	0.62
saldo_var42	0.54
num_var4	0.54
num_var35	0.54
saldo_medio_var5_hace2	0.53
saldo_var5	0.53
num_meses_var5_ult3	0.53
num_meses_var5_hace3	0.52
num_var42	0.51
num_var5	0.49
num_var13	0.47
saldo_var42	0.45
var36	0.44
var38	0.31

3.3. Ajuste dos hiperparâmetros e descrição dos classificadores utilizados

Foram testados três algoritmos de *machine learning* com o objetivo de se analisar qual teria o melhor desempenho, que será medido através do lucro. Os algoritmos testados foram Árvore de Decisão, *Random Forest* e XGBoost.

O lucro é calculado através da quantidade de Verdadeiros Positivos e Falsos Positivos através da seguinte lógica:

•**Falso Positivo (FP):** cliente marcado como insatisfeito mas estava satisfeito. custo: -10,00 lucro: 0,00

•**Falso negativo (FN):** Cliente previsto como satisfeito mas estava insatisfeito. custo: 0,00 lucro: 0,00

•**Verdadeiro Positivo (VP):** Cliente previsto como insatisfeito e é insatisfeito. custo: -10,00 lucro: 100,00

•**Verdadeiro Negativo (VN):** Cliente previsto como satisfeito e que não é alvo de nenhuma ação. custo: 0,00 lucro: 0,00

A Figura 1 ilustra as linhas de código utilizadas para o cálculo do lucro.

Figura 1. Linhas de código utilizadas para o cálculo do lucro.

```
def funcao_lucro(df):

    lucro = 0
    for ind in df['previsao'].index:
            if df['TARGET'][ind] == 1.0 and df['previsao'][ind] == 1.0: #Verdadeiro Positivo
                lucro += 100 - 10
            elif df['TARGET'][ind] == 0.0 and df['previsao'][ind] == 1.0: #Falso positivo
                lucro += -10

    return lucro
```

Para os três algoritmos foi utilizado o *grid search*, que é uma técnica com o objetivo de selecionar os melhores valores para os hiperparâmetros do modelo, dado um conjunto de valores possíveis. O objetivo é encontrar os valores para os hiperparâmetros que irão corroborar para o melhor desempenho do modelo. O *grid search* testa todas as combinações entre os hiperparâmetros e os respectivos valores escolhidos para teste.

Em relação a árvore de decisão, os parâmetros e valores utilizados podem ser vistos na Tabela 2.:

Tabela 2. Valores dos parâmetros utilizados para o *grid search* com a árvore de decisão. Em negrito, destaca-se os valores dos parâmetros do modelo com melhor desempenho.

Parâmetros	**Valores de Teste**
Criterion	**Gini**, Entropia
max_depth	2, **4**, 6
min_samples_split	2, **4**, 6
min_samples_leaf	2, 4, **6**, 8, 10

Em relação ao *Random Forest*, os parâmetros e valores utilizados podem ser vistos na Tabela 3.

Tabela 3. Valores dos parâmetros utilizados para o *grid search* com o *Random Forest*. Em negrito, destaca-se os valores dos parâmetros do modelo com melhor desempenho.

Parâmetros	**Valores de Teste**
Criterion	**Gini**, Entropia
n_estimators	**70**,100,120
max_depth	5, **7**, 9, 11
max_features	'**sqrt**', 'log2'

Já para o XGBoost, os parâmetros e valores utilizados podem ser vistos na Tabela 4.

Tabela 4. Valores dos parâmetros utilizados para o *grid search* como XGBoost. Em negrito, destaca-se os valores dos parâmetros do modelo com melhor desempenho.

Parâmetros	**Valores de Teste**
learning_rate	**0.1**,0.01,0.001
n_estimators	**50**,70,100,150,200
max_depth	**3**,4,5,7,8
max_features	'**sqrt**', 'log2'

4. RESULTADOS

O objetivo central do estudo é verificar qual algoritmo de *machine learning* obteve o maior lucro, para isso, após o desenvolvimento e treinamento do modelo e apuração de métricas, foi utilizado uma função denominada lucro que mensura o lucro obtido.

Os valores das métricas utilizadas nos modelos, assim como o valor obtido do lucro para cada modelo são apresentados nas Tabelas 5, 6 e 7.

Corte Utilizado: Valor de saída do predict_proba, onde a partir desse valor classifica-se a observação pertencente a classe 1 e em caso contrário a observação é pertencente a classe 0.

Lucro Treino: Valor de Lucro Absoluto obtido no treino do modelo com o respectivo corte utilizado.

Lucro Treino Relativo: Lucro obtido considerando a razão entre o lucro na base de treino e a quantidade de clientes insatisfeitos multiplicada pelo lucro unitário que é 90.

Tabela 5. Resultados obtidos variando o corte utilizado com a árvore de decisão.

Corte Utilizado	Lucro Treino	Lucro Treino Relativo	ROC AUC treino	Lucro Teste	Lucro Teste Relativo	ROC AUC teste
0,44	-102630,00	-0,54	0,68	-42260,00	-0,51	0,68
0,46	**41080,00**	**0,22**	**0,72**	**15890,00**	**0,19**	**0,70**
0,48	41080,00	0,22	0,72	15890,00	0,19	0,70
0,50	41080,00	0,22	0,72	15890,00	0,19	0,70
0,52	41080,00	0,22	0,72	15890,00	0,19	0,70
0,54	41080,00	0,22	0,72	15890,00	0,19	0,70
0,56	41080,00	0,22	0,72	15890,00	0,19	0,70
0,58	41080,00	0,22	0,72	15890,00	0,19	0,70
0,60	41080,00	0,22	0,72	15890,00	0,19	0,70
0,62	41080,00	0,22	0,72	15890,00	0,19	0,70
0,64	41080,00	0,22	0,72	15890,00	0,19	0,70
0,66	41080,00	0,22	0,72	15890,00	0,19	0,70

Pela Tabela 5, percebe-se que a partir do corte de 0,46 feito no predict_proba não houve variação nos valores de lucro e AUC.

Tabela 6. Resultados obtidos através da *Random Forest*.

Corte Utilizado	**Lucro Treino**	**Lucro Treino Relativo**	**ROC AUC treino**	**Lucro Teste**	**Lucro Teste Relativo**	**ROC AUC teste**
0,44	–240,00	0,00	0,75	-2850,00	-0,03	0,73
0,46	7100,00	0,22	0,72	15890,00	0,19	0,72
0,48	31780,00	0,22	0,72	15890,00	0,19	0,73
0,50	41860,00	0,22	0,72	15890,00	0,19	0,74
0,52	46780,00	0,22	0,72	15890,00	0,19	0,73
0,54	49040,00	0,22	0,72	15890,00	0,19	0,72
0,56	48940,00	0,22	0,72	15890,00	0,19	0,72
0,58	48980,00	0,22	0,72	15890,00	0,19	0,71
0,60	49720,00	0,22	0,72	15890,00	0,19	0,70
0,62	49590,00	0,22	0,72	15890,00	0,19	0,70
0,64	49860,00	0,22	0,72	15890,00	0,19	0,70
0,66	**50000,00**	**0,22**	**0,72**	**15890,00**	**0,19**	**0,70**
0,68	49380,00	0,26	0,72	17110,00	0,21	0,69
0,70	48690,00	0,26	0,70	17570,00	0,21	0,69
0,72	46700,00	0,25	0,69	17370,00	0,20	0,67
0,74	45270,00	0,24	0,68	16620,00	0,20	0,67
0,76	45060,00	0,24	0,68	16570,00	0,20	0,66
0,78	42280,00	0,22	0,67	16430,00	0,20	0,66
0,80	0,00	0,00	0,50	0,00	0,00	0,50
0,82	0,00	0,00	0,50	0,00	0,00	0,50

Em relação ao algoritmo *Random Forest*, o corte que obteve em geral o melhor resultado considerando o lucro foi o corte de 0,66.

Tabela 7. Resultados obtidos com o XGBoost.

Corte Utilizado	Lucro Treino	Lucro Treino Relativo	ROC AUC treino	Lucro Teste	Lucro Teste Relativo	ROC AUC teste
0,40	**4490,00**	**0,02**	**0,51**	**440,00**	**0,01**	**0,50**
0,42	4080,00	0,02	0,51	390,00	0,00	0,50
0,44	3640,00	0,02	0,51	410,00	0,00	0,50
0,46	3220,00	0,02	0,51	430,00	0,00	0,50
0,48	3050,00	0,02	0,51	260,00	0,00	0,50
0,50	2600,00	0,01	0,51	260,00	0,00	0,50
0,52	2160,00	0,01	0,51	260,00	0,00	0,50
0,54	1710,00	0,01	0,50	170,00	0,00	0,50
0,56	1440,00	0,01	0,50	0,00	0,00	0,50
0,58	1260,00	0,01	0,50	0,00	0,00	0,50
0,60	990,00	0,01	0,50	0,00	0,00	0,50
0,62	630,00	0,00	0,50	0,00	0,00	0,50
0,64	360,00	0,00	0,50	0,00	0,00	0,50
0,66	0,00	0,00	0,50	0,00	0,00	0,50
0,68	0,00	0,00	0,50	0,00	0,00	0,50
0,70	0,00	0,00	0,50	0,00	0,00	0,50
0,72	0,00	0,00	0,50	0,00	0,00	0,50
0,74	0,00	0,00	0,50	0,00	0,00	0,50
0,76	0,00	0,00	0,50	0,00	0,00	0,50
0,78	0,00	0,00	0,50	0,00	0,00	0,50
0,80	0,00	0,00	0,50	0,00	0,00	0,50

Em relação ao modelo utilizando o XGBoost, os valores de AUC tanto no treino como no teste ficou por volta de 0,50 e os melhores valores de lucro tanto no treino como no teste foram respectivamente de 4490 e 440 com o corte de 0,40 no predict_proba.

É importante mencionar que é visto nos resultados, principalmente tratando da Árvore de Decisão e Random Forest, Tabelas 6 e 7, respectivamente, uma mudança brusca nos resultados do corte 0.44 para o 0.46, seguem abaixo alguns pontos que podem ser responsáveis por essa alteração brusca:

1. Distribuição das probabilidades do modelo

- Os modelos classificatórios (como árvore, XGBoost, regressão logística, etc.) geram **probabilidades**.

- Se muitas observações estão com score próximo de **0,45**, uma pequena alteração no corte (de 0,44 para 0,46) pode fazer várias delas mudarem de classe (positivo ↔ negativo).

- Isso causa impacto direto no cálculo do **lucro**, já que muda o conjunto de casos classificados como positivos.

2. Overfitting localizado

- Note que no corte 0,44:

    - **Lucro treino** é **muito negativo** (-102630), mas no 0,46 já fica positivo (41080).
    - Isso sugere que naquele ponto (0,44) o modelo está capturando exemplos que **no treino pesam muito contra**, indicando sobreajuste ou concentração ruim de casos.

3. Curva ROC e estabilidade

- O **AUC** praticamente não muda (0,68 → 0,72 treino; 0,68 → 0,70 teste).

- Isso mostra que o modelo em si não mudou, só a forma de **escolher o cutoff** alterou os resultados do lucro.

- Ou seja: o desempenho em termos de separação não é afetado, mas o **critério de decisão** faz muita diferença para métricas de negócio.

Ao comparar os resultados, o pior desempenho foi o modelo utilizando o algoritmo XGBoost, talvez pela quantidade de parâmetros disponíveis e complexidade do algoritmo em relação ao problema apresentado, a predição se deu de forma aleatória apresentando um AUC ao redor de 0.50, o que indica que não houve capacidade preditiva da parte do modelo.

Em relação a árvore de decisão, mesmo sendo um algoritmo mais sensível ao *overfitting,* apresentou um desempenho melhor do que o XGBoost, obtendo um valor de lucro no treino e teste respectivamente de 41080 e 15890, AUC de treino e teste de 0,72 e 0,70 no corte de 0,46, esse mesmo desempenho foi apresentado pela Random Forest.

5. CONCLUSÕES

O objetivo desse trabalho era verificar dentre os algoritmos utilizados, sendo eles, a árvore de decisão, o *Random Forest* e o XGBoost, qual tem melhor desempenho para o problema proposto, que é o desenvolvimento de um modelo onde deve-se identificar os clientes que são insatisfeitos e para esse grupo de clientes aplicam uma campanha de retenção mensurando o lucro obtido.

Após o tratamento dos dados e seleção de hiperparâmetros, os modelos foram desenvolvidos com os três algoritmos citados, onde foi avaliado em cada modelo o melhor corte no predict_proba que obtivesse o melhor lucro e AUC.

Diante disso, percebeu-se que o *Random Forest* e a Árvore de Decisão tiveram o mesmo desempenho, sendo o XGBoost o algoritmo que obteve o pior desempenho. O desempenho não satisfatório do XGBoost pode ser justificado pelo fato de ser um algoritmo muito complexo com muitos hiperparâmetros a serem manuseados, o que demanda muito conhecimento do algoritmo.

REFERÊNCIAS BIBLIOGRÁFICAS

GÉRON, Aurélien. **Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems**. 2. ed. Sebastopol: O'Reilly Media, 2019.

JAMES, Gareth; WITTEN, Daniela; HASTIE, Trevor; TIBSHIRANI, Robert. **An Introduction to Statistical Learning: with Applications in R**. New York: Springer, 2021

SIDDIQI, N. **Credit Risk Scorecards: Developing and Implementing Intelligent Credit Scoring**. Hoboken: John Wiley & Sons, 2005

JIMENEZ, Soraya; CUKIERSKI, Will. ***Santander Customer Satisfaction*. 2018**. Disponível em: https://kaggle.com/competitions/santander-customer-satisfaction. Acesso em: 31 jul. 2025.