

**Inteligência Artificial para Detecção de Fraudes Bancárias**

**Integrantes:**

**Diogo Tayjen Dagnino**  **RA 823130326**

**Henrique Vieira Dias** **RA 823129452**

**Letícia Maria Gonçalves Santos** **RA 823130667**

**Victor de Almeida Martins** **RA 823117661**

**2025**

**ÍNDICE DETALHADO**

Sumário

[1. Introdução 3](#_Toc200142895)

[2. Desenvolvimento do projeto 4](#_Toc200142896)

[2.1 Importação de bibliotecas 4](#_Toc200142897)

[2.2 Tratamento dos dados 5](#_Toc200142898)

[2.2.1 Leitura e Tratamento da Base 5](#_Toc200142899)

[2.2.2 Análise Exploratória Inicial 6](#_Toc200142900)

[2.2.3 Tratamento de Valores Ausentes e Categóricos e Separação das Variáveis Independentes e Alvo 6](#_Toc200142901)

[2.2.4 Balanceamento com SMOTE 7](#_Toc200142902)

[2.2.5 Padronização e Divisão dos Dados 8](#_Toc200142903)

[2.2.6 Padronização e Divisão dos Dados 9](#_Toc200142904)

[2.3 Aplicação dos Modelos 10](#_Toc200142905)

[2.3.1 Random Forest 10](#_Toc200142906)

[2.3.2 Regressão Logística 11](#_Toc200142907)

[2.3.3 XGBoost 11](#_Toc200142908)

[2.3.4 Árvore de Decisão 12](#_Toc200142909)

[2.3.5 Rede Neural (MLP) 13](#_Toc200142910)

[2.4 Comparação – Curva ROC 15](#_Toc200142911)

[3. Análise Final e Conclusão 17](#_Toc200142912)

[3.1 Resultado dos modelos 17](#_Toc200142913)

[3.2 Resultado dos modelos 19](#_Toc200142914)

# Introdução

Com o avanço das tecnologias digitais e o aumento expressivo do número de transações financeiras realizadas por meios eletrônicos, também se intensificaram os casos de fraudes bancárias. Esse fenômeno impõe desafios às instituições financeiras, que buscam soluções automatizadas e eficientes para detectar comportamentos suspeitos de forma rápida e precisa.

Neste cenário, a Inteligência Artificial (IA) vem ganhando destaque como uma ferramenta poderosa na análise de grandes volumes de dados e na identificação de padrões anômalos. Técnicas de aprendizado de máquina (machine learning) permitem construir modelos preditivos capazes de distinguir transações legítimas de possíveis fraudes com base em características extraídas dos dados.

O presente trabalho tem como objetivo investigar a aplicação de algoritmos de aprendizado de máquina na detecção de fraudes bancárias, avaliando diferentes modelos quanto à sua eficiência, robustez e capacidade de generalização. Para isso, foi utilizada uma base de dados simulada contendo registros de transações rotuladas como legítimas ou fraudulentas. Além disso, foram aplicadas técnicas de pré-processamento e redução de dimensionalidade, como a Análise de Componentes Principais (PCA), visando compreender seus impactos sobre o desempenho dos modelos.

# Desenvolvimento do projeto

## Importação de bibliotecas

Para a implementação do projeto, foram utilizadas diversas bibliotecas da linguagem Python, que oferecem recursos especializados para análise de dados, construção de modelos e visualização de resultados. As principais utilizadas foram:

Interface gráfica do usuário, Texto, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

* **pandas**: Usada para carregar, organizar e manipular os dados em formato de tabelas (DataFrames). Facilitou tarefas como leitura da base de dados, seleção de colunas e análise inicial dos dados.
* **numpy**: Suporte para operações matemáticas e manipulação de arrays numéricos. Foi utilizada de forma complementar, pois serve de base para outras bibliotecas como pandas e sklearn.
* **matplotlib**: Ferramenta para criação de gráficos e visualizações, como a curva ROC, que permite avaliar o desempenho dos modelos de classificação.
* **scikit-learn (sklearn)**: Biblioteca central para a construção e avaliação de modelos de machine learning. Foi utilizada para treinar algoritmos como redes neurais, regressão e árvore de decisão, além de fornecer ferramentas para pré-processamento, divisão de dados e cálculo de métricas como F1-Score e AUC.
* **seaborn**: Biblioteca de visualização de dados baseada no matplotlib. Foi usada para gerar gráficos estatísticos com uma estética aprimorada, facilitando a análise visual dos dados.
* **xgboost**: Algoritmo poderoso baseado em árvores de decisão com foco em desempenho e precisão. Foi utilizado como modelo complementar para comparar sua eficácia com as outras abordagens testadas.
* **imblearn**: Biblioteca voltada para o tratamento de bases de dados desbalanceadas. Foi utilizada para aplicar técnicas de balanceamento como o SMOTE, aumentando a representatividade da classe minoritária (fraudes) no treinamento dos modelos.

## Tratamento dos dados

### 2.2.1 Leitura e Tratamento da Base

A base de dados foi carregada a partir de um arquivo .csv denominado "Base.csv", utilizando a função read\_csv() da biblioteca pandas. Essa base, proveniente da plataforma Kaggle, contém informações detalhadas sobre transações bancárias, sendo que cada linha representa uma operação rotulada como legítima (0) ou fraude (1), por meio da coluna fraud\_bool.

Esse processo inicial permite visualizar o conjunto completo e identificar a estrutura das colunas. Foi realizado um display(df.head()) para inspecionar as primeiras linhas, e também a verificação do formato da base (df.shape), garantindo que o carregamento foi bem-sucedido.

Texto

O conteúdo gerado por IA pode estar incorreto.

### 2.2.2 Análise Exploratória Inicial

Antes de aplicar qualquer modelo, foi feita uma análise da distribuição da variável-alvo (fraud\_bool). Com o uso do seaborn, foi gerado um gráfico de contagem (countplot) para evidenciar o desbalanceamento entre as transações legítimas e as fraudulentas.

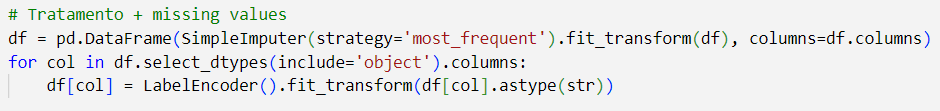
Como esperado em problemas reais de detecção de fraudes, verificou-se que a base de dados era fortemente desbalanceada, com um número muito maior de transações legítimas em relação às fraudulentas.



### 2.2.3 Tratamento de Valores Ausentes e Categóricos e Separação das Variáveis Independentes e Alvo

Foi aplicado um tratamento padronizado para valores ausentes (missing values) utilizando a estratégia "most\_frequent", ou seja, preenchendo com o valor mais frequente em cada coluna. Esse preenchimento foi feito por meio do SimpleImputer.

Além disso, as colunas do tipo object (categóricas) foram transformadas em variáveis numéricas com o uso do LabelEncoder. Esse passo é essencial, pois os modelos de machine learning exigem que todos os dados estejam em formato numérico.



Após o tratamento dos dados, foi realizada a separação entre os atributos preditores (X) e a variável de destino (y). A variável fraud\_bool foi isolada como y, enquanto todas as demais colunas foram mantidas em X.

Texto

O conteúdo gerado por IA pode estar incorreto.

A fim de comparação, criamos um gráfico através da biblioteca matplotlib para traduzir graficamente como nossa base está em relação a variável “fraud\_bool”, trazendo uma quantidade muito diferente.

Gráfico, Gráfico de barras

O conteúdo gerado por IA pode estar incorreto.

### 2.2.4 Balanceamento com SMOTE

Dada a forte desproporção entre as classes, foi aplicado o algoritmo SMOTE (Synthetic Minority Over-sampling Technique), proveniente da biblioteca imblearn. O SMOTE cria novas amostras sintéticas da classe minoritária (fraudes), equilibrando as proporções para que o modelo possa aprender melhor os padrões de ambos os grupos.

Uma imagem contendo Texto

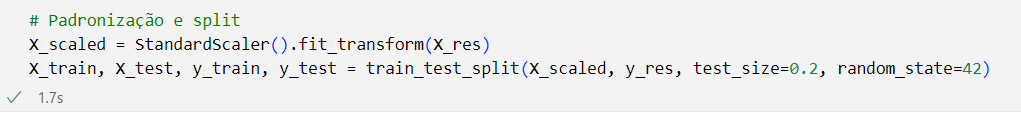
O conteúdo gerado por IA pode estar incorreto.

Agora pode-se ver graficamente como a base se encontra com os dados balanceados:

Gráfico, Gráfico de barras

O conteúdo gerado por IA pode estar incorreto.

### 2.2.5 Padronização e Divisão dos Dados



Após o balanceamento da base, os dados foram padronizados com o uso da classe “StandardScaler”, da biblioteca “sklearn”. Esse procedimento transforma os dados para que tenham média zero e desvio padrão igual a um, garantindo que todas as variáveis numéricas estejam na mesma escala. Isso é especialmente importante para algoritmos sensíveis à magnitude dos dados, como redes neurais.

Em seguida, os dados foram divididos em conjuntos de treino e teste, utilizando a função “train\_test\_split”. Foi reservada uma fração de 20% da base para testes, garantindo que a avaliação dos modelos fosse feita sobre dados que não foram vistos durante o treinamento.

### 2.2.6 Padronização e Divisão dos Dados

Para avaliar o impacto da redução de dimensionalidade no desempenho dos modelos, foi aplicada a técnica PCA (Análise de Componentes Principais). O procedimento foi realizado após a separação entre treino e teste, garantindo que a transformação fosse ajustada apenas sobre os dados de treino.

Foram testadas duas versões:

* Uma mantendo 95% da variância total dos dados;
* Outra mantendo 99% da variância total.

Essa etapa teve como objetivo verificar se a simplificação dos dados poderia melhorar a performance dos modelos ou reduzir seu tempo de processamento, sem comprometer a capacidade preditiva.

Texto

O conteúdo gerado por IA pode estar incorreto.

## **Aplicação dos Modelos**

Com os dados devidamente tratados, balanceados e padronizados, foram aplicados diferentes algoritmos de aprendizado de máquina para realizar a detecção de fraudes bancárias. O objetivo foi comparar o desempenho entre os modelos e verificar quais apresentavam melhor capacidade de identificar padrões suspeitos.

Todos os modelos foram treinados utilizando o conjunto de treino e avaliados com base no conjunto de teste. Para garantir a comparabilidade dos resultados, foram utilizadas métricas consistentes como F1-Score, AUC e Recall. Além disso, testou-se cada modelo tanto com os dados originais quanto com dados reduzidos por PCA, permitindo uma análise mais completa do impacto da redução de dimensionalidade.

### Random Forest

O modelo Random Forest é utilizado com sua configuração padrão e uma semente aleatória fixa (random\_state=42) para garantir reprodutibilidade. Ele constrói uma coleção (ou floresta) de árvores de decisão independentes, cada uma treinada com uma amostra diferente dos dados e com subconjuntos aleatórios de variáveis.

A saída final é determinada por votação entre essas árvores, o que torna o modelo altamente robusto contra overfitting e muito eficaz para dados tabulares, como os do nosso caso. Sua capacidade de capturar interações entre variáveis e lidar com dados não linearmente separáveis explica seu desempenho geralmente superior em tarefas de detecção de fraudes.

Desenho de uma pessoa

O conteúdo gerado por IA pode estar incorreto. Tela de celular com texto preto sobre fundo branco

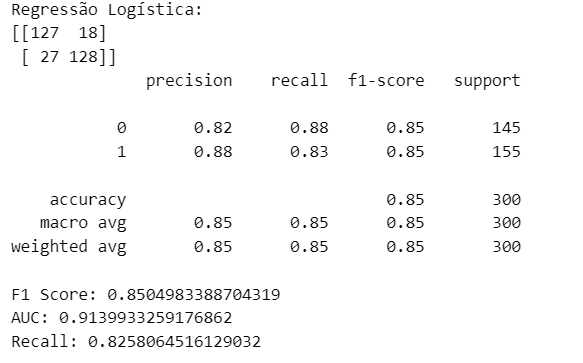
O conteúdo gerado por IA pode estar incorreto.

### 2.3.2 Regressão Logística

O modelo de Regressão Logística utilizado é configurado com até 1000 iterações para garantir que a função de otimização tenha tempo suficiente para convergir, especialmente considerando que dados padronizados e balanceados por SMOTE foram aplicados previamente.

A regressão logística busca encontrar uma fronteira de decisão linear entre as classes (fraude e não fraude), estimando a probabilidade de uma transação ser fraude com base em uma função sigmoide. Como estamos tratando um problema de classificação binária e com dados já tratados e escalados, essa configuração simples é suficiente para fornecer um bom ponto de comparação com os demais modelos.

Texto

O conteúdo gerado por IA pode estar incorreto. 

### 2.3.3 XGBoost

O modelo XGBoost (Extreme Gradient Boosting) é configurado para evitar alertas desnecessários com use\_label\_encoder=False e eval\_metric='logloss', apropriado para classificação binária. XGBoost é um dos algoritmos mais poderosos para dados estruturados e funciona treinando árvores de forma sequencial, onde cada nova árvore tenta corrigir os erros das anteriores.

Ele minimiza uma função de perda usando gradiente descendente, combinando o poder dos ensembles com controle refinado de regularização. Sua performance superior decorre da capacidade de ajustar-se rapidamente a padrões complexos, mesmo em dados desbalanceados (desde que tratados, como com SMOTE).

Texto

O conteúdo gerado por IA pode estar incorreto. Tela de celular com texto preto sobre fundo branco

O conteúdo gerado por IA pode estar incorreto.

### 2.3.4 Árvore de Decisão

O modelo DecisionTreeClassifier utilizado é configurado com uma semente aleatória (random\_state=42) para garantir reprodutibilidade dos resultados. Ele é treinado com os dados balanceados e padronizados (X\_train, y\_train) e, posteriormente, faz previsões sobre os dados de teste.

A predição de probabilidades é usada para calcular a AUC e a curva ROC, enquanto as predições diretas (classes) são usadas para as métricas como F1-Score e o relatório de classificação.

A árvore de decisão cria regras simples de "se-então" a partir dos dados, dividindo o espaço de decisão com base nos valores que melhor separam as classes. Sua vantagem está na interpretabilidade, já que os caminhos de decisão podem ser visualizados e explicados facilmente.

No entanto, por padrão, a árvore pode crescer demais e sofrer com overfitting, o que pode prejudicar a generalização se não for podada ou parametrizada corretamente.

Texto

O conteúdo gerado por IA pode estar incorreto. Tela de celular com texto preto sobre fundo branco

O conteúdo gerado por IA pode estar incorreto.

### 2.3.5 Rede Neural (MLP)

O código utilizado para o MLP cria uma rede neural com duas camadas ocultas: a primeira com 64 neurônios e a segunda com 32. A função de ativação usada em cada camada é a ReLU, que permite à rede aprender relações não lineares de forma eficiente, sendo especialmente útil em conjuntos de dados complexos.

O otimizador escolhido é o Adam, que combina as vantagens de AdaGrad e RMSProp, adaptando dinamicamente a taxa de aprendizado durante o treinamento. A taxa de aprendizado inicial (learning\_rate\_init=0.001) controla o tamanho dos passos de atualização dos pesos, sendo um valor padrão confiável para começar o treinamento estável.

O parâmetro alpha=0.0001 define a força da regularização L2, ajudando a evitar o overfitting ao penalizar pesos excessivamente altos. A rede é treinada por até 1000 épocas (max\_iter=1000), o que permite que ela converja mesmo com bases complexas, sem interromper precocemente o processo de aprendizado.

Por fim, o random\_state=42 garante reprodutibilidade dos resultados, mantendo os dados embaralhados da mesma forma em cada execução. Esses parâmetros, combinados com a padronização prévia dos dados e o balanceamento por SMOTE, criam uma configuração robusta e sensível para detecção de fraudes, ainda que sem validação cruzada ou tuning fino.

Interface gráfica do usuário, Texto, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

Texto

O conteúdo gerado por IA pode estar incorreto.

Tela de celular com texto preto sobre fundo branco

O conteúdo gerado por IA pode estar incorreto.

## 2.4 Comparação – Curva ROC

A curva ROC possibilita uma comparação entre cada modelo, através do cálculo da taxa de verdadeiros positivos e falsos positivos e plota suas curvas em um mesmo gráfico, juntamente com o valor da **AUC (Área sob a Curva)**, que indica a capacidade de separação entre fraudes e transações legítimas.

Taxa de **Falsos Positivos (False Positive Rate - FPR)**: Representa a proporção de casos negativos (transações legítimas) que foram classificados incorretamente como fraudes.

Uma imagem contendo Diagrama

O conteúdo gerado por IA pode estar incorreto.

**Taxa de Verdadeiros Positivos (True Positive Rate - TPR ou Recall)**: Indica a proporção de fraudes corretamente identificadas pelo modelo.

Uma imagem contendo Texto

O conteúdo gerado por IA pode estar incorreto.

**AUC (Área sob a Curva ROC)**: A métrica quantifica a capacidade do modelo em distinguir entre classes (fraudes e não fraudes).

* Um modelo perfeito tem **AUC = 1.0**.
* Um modelo aleatório tem **AUC ≈ 0.5**.
* Quanto mais próximo de 1, **melhor é o desempenho** do modelo em termos de separação entre as classes.

Texto

O conteúdo gerado por IA pode estar incorreto.Gráfico, Gráfico de dispersão

O conteúdo gerado por IA pode estar incorreto.

# Análise Final e Conclusão

## 3.1 Resultado dos modelos

O desenvolvimento de software exige uma abordagem cuidadosa e estruturada para garantir a criação de sistemas eficientes e de qualidade. A compreensão clara das necessidades dos usuários e a definição de requisitos adequados são essenciais para orientar o projeto e minimizar riscos. Além disso, práticas como o planejamento de protótipos e o versionamento de código asseguram um processo de desenvolvimento ágil e controlado.

A garantia da qualidade em cada fase do ciclo de vida do software, por meio de testes rigorosos, é fundamental para detectar e corrigir falhas, garantindo um produto robusto. Combinando essas estratégias, é possível criar soluções mais estáveis, confiáveis e alinhadas com as expectativas do cliente. Assim, a integração dessas práticas promove uma entrega contínua de valor e um aprimoramento constante. O resultado é um software de alta qualidade, adequado às necessidades do usuário e ao contexto em que será utilizado.

Importante frisar termos de destaque para as métricas:

* **F1-Score:** equilíbrio entre precisão e recall. Utilizada para medir o desempenho de modelos de classificação quando você tem dados desbalanceados porque leva em conta o tipo de erro — falso positivo e falso negativo — e não apenas o número de previsões incorretas, uma necessidade em áreas como prevenção de fraudes e outros casos de uso do setor.
* **Curva ROC:** gráfico que plota a taxa de verdadeiros positivos (TPR) contra a taxa de falsos positivos (FPR) em diferentes pontos de corte, permitindo visualizar a capacidade de um modelo de distinguir entre classes.
* **AUC-ROC:** métrica amplamente utilizada em aprendizado de máquina, especialmente para avaliar o desempenho de modelos de classificação. Ela representa a área sob a curva ROC e quantifica a capacidade de um modelo de discriminação entre classes positiva e negativa.
* **Recall:** essencial para reduzir falsos negativos (fraudes não detectadas). Mede a capacidade de um modelo de classificação em identificar corretamente todas as instâncias positivas reais em um conjunto de dados.

Com isso a inteligência artificial, aplicada nestes diferentes modelos, resultou nas seguintes taxas para cada um:

Tabela

O conteúdo gerado por IA pode estar incorreto.

De forma resumida:

* **XGBoost** se destaca pelo equilíbrio entre precisão, recall e AUC, sendo ideal para produção.
* **Random Forest** oferece robustez e interpretabilidade com ótimo desempenho.
* **Rede Neural (MLP)** apresenta performance competitiva, destacando-se em recall.
* **Árvore de Decisão** é uma opção interpretável e eficiente para uso prático.
* O uso de **SMOTE** no pré-processamento foi essencial para corrigir o desbalanceamento.
* O monitoramento do **Recall** é crítico em aplicações sensíveis como prevenção a fraudes.

## 3.2 Resultado dos modelos

PCA: (Análise de Componentes Principais) é uma técnica de redução de dimensionalidade que transforma um conjunto de variáveis originais em um novo conjunto menor de variáveis (chamadas componentes principais), preservando o máximo possível da variância dos dados.

Uma imagem contendo Texto

O conteúdo gerado por IA pode estar incorreto.

Texto, Carta

O conteúdo gerado por IA pode estar incorreto.

Texto

O conteúdo gerado por IA pode estar incorreto.

No contexto deste projeto de detecção de fraudes, o uso do PCA teve como objetivo reduzir a dimensionalidade dos dados, buscando melhorar a performance do modelo e diminuir possíveis redundâncias. Foram testadas duas abordagens com preservação de 95% e 99% da variância dos dados.

Apesar de seu potencial em problemas com muitas variáveis e alta colinearidade, o PCA não resultou em melhoria nas principais métricas de desempenho (F1-Score, AUC e Recall) em comparação ao modelo original. Isso indica que as variáveis descartadas no processo de redução carregavam informações relevantes para a predição de fraudes, e que o modelo já era capaz de lidar eficientemente com a dimensionalidade original.

Gráfico, Gráfico de linhas

O conteúdo gerado por IA pode estar incorreto.

Portanto, conclui-se que, neste caso específico, a aplicação do PCA não trouxe benefícios concretos ao modelo, reforçando a importância de validar empiricamente qualquer técnica de pré-processamento antes de adotá-la em produção.