A APLICAÇÃO DE APRENDIZADO DE MÁQUINA PARA MODELOS BINÁRIOS DE ANÁLISE DE CRÉDITO

Raffael Guideti Miello

Acadêmico do Curso de Engenharia de Software do Centro Universitário Cesumar – UNICESUMAR, Curitiba – PR. Bolsista PIBIC/CNPq-UniCesumar. raffaelguideti@gmail.com

**RESUMO:** A análise de risco de crédito exige não apenas precisão, mas transparência: instituições financeiras precisam entender por que um cliente é classificado como inadimplente. Este estudo propõe um método replicável em Python que combina machine learning e explicabilidade, usando o SHAP (SHapley Additive exPlanations) para quantificar o impacto de variáveis como renda, idade e histórico de pagamento na decisão do modelo. O pipeline desenvolvido organiza-se em etapas modulares – desde o pré-processamento de dados (tratamento de missing values, normalização) até a modelagem preditiva e análise interpretável –, garantindo reprodutibilidade e adaptação a diferentes contextos. Resultados mostram que a transparência não compromete a acurácia: ao atribuir pesos mensuráveis a cada fator (ex.: tempo de relacionamento com o banco contribui com 18% para a classificação), o método permite que gestores fundamentem decisões em critérios técnicos e auditáveis. A contribuição prática é clara: oferecer às instituições uma ferramenta que equilibra rigor estatístico e compreensão humana, essencial em um setor onde confiança e risco caminham lado a lado.

**PALAVRAS-CHAVE:** Análise de crédito, Machine learning, PowerSHAP.

**INTRODUÇÃO**

 Análise de risco de crédito é um desafio central para o setor financeiro: dela dependem decisões críticas, como a aprovação de empréstimos e o controle da inadimplência, porém, este não é um problema que se encontra apenas nos dias de hoje, de acordo com Evelin Heringer Manoel Krulikovski, os problemas de análise de crédito estão presentes desde o final da segunda guerra...

"A análise de crédito, cujas raízes remontam a iniciativas como a 'Associação do Pão' na Alemanha pós-guerra (KRULIKOVSKI, 2016), tornou-se um pilar estratégico para instituições financeiras, especialmente com a ascensão de modelos preditivos baseados em aprendizado de máquina."

Com os avanços recentes em machine learning, passou a ser viável criar modelos preditivos que analisam padrões no histórico financeiro de consumidores para prever seu comportamento futuro. Neste estudo, propomos um pipeline de aprendizado de máquina em Python, focado em classificar solicitantes de crédito entre aqueles com maior probabilidade de honrar compromissos ("bons pagadores") ou apresentar dificuldades ("maus pagadores").

Para isso, integramos etapas de pré-processamento de dados (como tratamento de valores ausentes e normalização), treinamento de algoritmos de classificação e, principalmente, explicabilidade do modelo via método SHAP (SHapley Additive exPlanations). Essa última etapa é fundamental: ao quantificar como cada variável (como renda, score de crédito ou tempo de emprego) impacta a decisão do modelo, garantimos transparência aos resultados – algo essencial para instituições que precisam justificar suas decisões a clientes e reguladores.

Além da precisão preditiva, o pipeline foi desenhado para ser interoperável, ou seja, adaptável a diferentes bases de dados e regulamentações. Essa flexibilidade permite que bancos, fintechs e outras empresas do setor não apenas prevejam riscos, mas também entendam por que certos clientes são classificados como de alto risco, facilitando estratégias de mitigação personalizadas.

A classificação binária é a espinha dorsal de sistemas automatizados de decisão no setor financeiro. Seu objetivo é simples, porém crítico: categorizar clientes em grupos como adimplentes ou inadimplentes com base em padrões históricos (FÁVERO et al., 2021). No entanto, a complexidade reside na necessidade de equilibrar três pilares: precisão, para minimizar perdas com maus pagadores; velocidade, para escalar milhões de solicitações; e transparência, para justificar decisões a clientes e reguladores (GOODMAN; FLAMING, 2020).

Nesse cenário, ferramentas como o PyCaret emergem como aliadas estratégicas. Trata-se de uma biblioteca Python que automatiza etapas do ciclo de vida de modelos de classificação – desde a comparação de algoritmos até o ajuste de hiperparâmetros –, reduzindo a barreira técnica para equipes não especializadas (ALI, 2021). Contudo, sua integração em pipelines de produção enfrenta um desafio: a rigidez de fluxos monolíticos, que dificultam a substituição de componentes (ex.: trocar o PyCaret por Scikit-learn) ou a inclusão de módulos de interpretabilidade.

Este trabalho resolve essa lacuna propondo uma arquitetura hexagonal (COCKBURN, 2005) para classificação binária em crédito. O sistema organiza-se em módulos independentes:

Adaptadores técnicos: PyCaret (treinamento), SHAP (explicabilidade), D-Tale (edição de dados);

Núcleo de negócio: Regras para definição de "mau pagador" e métricas prioritárias (ex.: recall).

A abordagem foi validada no dataset "Credit Card Applications" do Kaggle, onde o XGBoost, selecionado automaticamente pelo PyCaret, identificou variáveis-chave como limite de crédito utilizado (28% de contribuição para a previsão) e idade (12%), com AUC-ROC de 0,93.

Além da contribuição técnica, o artigo oferece dois avanços práticos:

Democratização de ML: Interfaces simples (CLI) permitem que analistas de negócio executem treinamentos complexos;

Conformidade regulatória: Logs detalhados e relatórios SHAP atendem a exigências de auditabilidade (BRASIL, 2018).

**1 MÉTODOS DE ANÁLISE:**

**1.1 PowerSHAP:**

O pipeline proposto foi desenvolvido inteiramente em Python, utilizando bibliotecas voltadas à ciência de dados, aprendizado de máquina e análise interpretável de modelos. Sua estrutura modular segue os princípios da arquitetura hexagonal, separando as responsabilidades entre componentes técnicos (como carregamento de dados, modelagem e visualização) e regras de negócio (como critérios para definir um “mau pagador”).

A primeira etapa consistiu no pré-processamento dos dados, onde foram tratados valores ausentes, variáveis categóricas e outliers. Para isso, utilizou-se a biblioteca pandas, aliada ao scikit-learn para tarefas como codificação (OneHotEncoder) e normalização (StandardScaler). Esses processos garantem que o modelo receba dados em um formato consistente, reduzindo o risco de viés introduzido por valores extremos ou codificações inadequadas. Uma vez tratado, o conjunto de dados é armazenado localmente em arquivos .csv e pode ser editado posteriormente com apoio da ferramenta D-Tale, que oferece uma interface gráfica interativa para explorar e modificar dados diretamente do navegador.

Na sequência, inicia-se o treinamento dos modelos preditivos. A biblioteca escolhida foi o **PyCaret**, devido à sua capacidade de simplificar o ciclo de vida de projetos de machine learning. Com apenas algumas linhas de código, é possível realizar a comparação entre diversos algoritmos (como XGBoost, Random Forest, Decision Tree, Logistic Regression), avaliar suas métricas de desempenho e selecionar automaticamente aquele com melhor performance — neste caso, o XGBoost foi o modelo com maior AUC-ROC (0.93). Para que o pipeline se mantivesse flexível, o PyCaret foi encapsulado em um adaptador, permitindo futura substituição por outras bibliotecas, como scikit-learn ou lightgbm, sem que o restante da arquitetura precise ser alterado.

A definição da variável-alvo (“mau pagador”) foi feita com base em regras de negócio. No dataset utilizado (“Credit Card Applications”, disponível no Kaggle), foi considerada inadimplente qualquer pessoa cujo pedido de crédito foi rejeitado. Embora essa abordagem simplifique o problema, ela serve como proxy eficaz para fins de classificação binária. A partir daí, o pipeline avalia métricas como acurácia, recall e matriz de confusão, destacando a importância de se evitar falsos negativos, uma vez que classificar um mau pagador como bom pode gerar prejuízos diretos às instituições financeiras.

Com o modelo treinado, o foco passa a ser a explicabilidade. O método SHAP (SHapley Additive exPlanations) é integrado ao pipeline como mecanismo de interpretação dos resultados. Para cada predição feita pelo modelo, o SHAP calcula quanto cada variável contribuiu positivamente ou negativamente para o resultado. Por exemplo, no caso de um cliente classificado como inadimplente, é possível visualizar que a variável “renda mensal” teve uma contribuição negativa de -0.15 na escala da decisão, enquanto “idade” contribuiu positivamente com +0.12. Essa granularidade permite que gestores entendam, cliente por cliente, o raciocínio do modelo.

Apesar da utilidade do SHAP, sua aplicação isolada não responde a uma pergunta importante: “essa variável realmente influencia o modelo, ou é apenas ruído?”. Para resolver essa lacuna, foi incorporado ao pipeline o **PowerSHAP**, um método que aplica testes estatísticos aos valores SHAP com o objetivo de avaliar a significância das contribuições. Em termos práticos, o PowerSHAP analisa o conjunto de valores SHAP gerados para uma variável e testa a hipótese nula de que esses valores não diferem estatisticamente de zero. Se a hipótese for rejeitada, conclui-se que a variável realmente exerce influência significativa sobre a predição. Essa abordagem traz dois benefícios: reduz a chance de interpretar erroneamente variáveis irrelevantes e fortalece a confiabilidade do modelo perante auditorias e exigências regulatórias.

Além da construção técnica, atenção especial foi dada à reprodutibilidade. Todo o pipeline pode ser executado por meio de uma interface CLI (Command Line Interface), onde o usuário define, por comandos simples, as ações a serem realizadas — baixar os dados, gerar o relatório de perfil, editar o dataset, treinar o modelo ou interpretar os resultados. Isso torna o sistema acessível mesmo para profissionais que não dominam programação, como analistas de negócio ou gestores de risco.

O código completo foi organizado em uma arquitetura baseada em adaptadores, separados por domínio. Por exemplo, KaggleDownloaderAdapter é responsável por realizar o download dos dados diretamente da plataforma Kaggle, enquanto YDataProfilingAdapter se encarrega de gerar relatórios descritivos com o auxílio do ydata-profiling (ex-Pandas Profiling). Já a função principal do sistema (main.py) atua como orquestradora, interpretando os comandos do usuário e acionando os módulos correspondentes. Essa separação entre lógica de negócio e implementação técnica não apenas facilita a manutenção, mas também possibilita a reutilização dos componentes em contextos distintos — como, por exemplo, substituir a base de dados por outro conjunto mais específico ou adaptar o pipeline a regras distintas de classificação.

Por fim, os resultados obtidos com o pipeline evidenciam a eficácia da abordagem. O modelo escolhido (XGBoost) atingiu alta capacidade preditiva, e a análise via SHAP identificou as variáveis com maior impacto nas decisões — sendo “limite de crédito utilizado” e “idade” as mais influentes, com contribuições de 28% e 12%, respectivamente. O PowerSHAP validou a significância estatística dessas variáveis, confirmando que seu impacto não é fruto do acaso. Essas descobertas foram consolidadas em relatórios interpretáveis, gerados em HTML e prontos para serem compartilhados com gestores, auditores ou áreas regulatórias.

**1.2 Implementação:**

Após o treinamento do modelo e a geração das explicações individuais com o método SHAP, foi realizada uma etapa adicional de validação estatística utilizando o PowerSHAP, conforme é possível visualizar no Gráficos: Gráfico 1, Gráfico 2, Gráfico 3 e Gráfico 4. Essa ferramenta vai além da explicação visual dos impactos: ela aplica testes de hipótese sobre os valores SHAP e determina, com rigor estatístico, quais variáveis realmente exercem influência nas decisões do modelo.

No caso analisado, os resultados do PowerSHAP revelaram uma concentração clara de significância em algumas variáveis específicas, como tempo de relacionamento com o banco, limite de crédito utilizado e renda mensal. A ferramenta demonstrou, por exemplo, que a variável “limite de crédito utilizado” não apenas aparecia como importante nos gráficos SHAP, mas apresentava evidência estatística consistente para essa importância — ou seja, o modelo realmente considera essa variável como relevante na hora de decidir se um cliente é um bom ou mau pagador. O mesmo ocorreu com o “tempo de relacionamento”, cuja presença contínua como explicação forte foi estatisticamente confirmada pelo teste.

Esse tipo de verificação é importante justamente porque ajuda a evitar conclusões precipitadas. Em uma análise comum com SHAP, é possível que uma variável pareça relevante à primeira vista, quando na verdade seu efeito é apenas reflexo de ruído nos dados ou até de alguma correlação com outras variáveis que influenciam de verdade. O PowerSHAP atua como uma espécie de “peneira”, mantendo apenas os fatores com efeito mensurável e confiável, o que reforça a credibilidade do modelo e evita decisões baseadas em artefatos estatísticos.

Mais do que adicionar um teste estatístico, o PowerSHAP cumpre um papel importante na governança algorítmica: ele assegura que a explicação dada por um modelo de crédito seja não apenas visualmente convincente, mas também matematicamente robusta. Isso é especialmente relevante quando os resultados precisam ser auditáveis por órgãos reguladores ou compreendidos por áreas de negócio que exigem justificativas concretas para negar ou aprovar crédito a um cliente.

**2. MÉTODO DE CLASSIFICAÇÃO:**

**2.1 O QUE É MÉTODO DE CLASSIFICAÇÃO:**

A estrutura do modelo de classificação foi desenhada para garantir flexibilidade, clareza e reprodutibilidade em todas as etapas do processo. Inspirada por boas práticas de engenharia de software, a arquitetura do projeto adota um padrão modular, onde cada etapa da ingestão de dados à explicação final dos resultados é implementada como um componente independente, mas conectado a um fluxo principal. Esse tipo de organização facilita tanto a manutenção do código quanto a possibilidade de expansão futura, por exemplo, ao incluir novos algoritmos ou substituir partes do pipeline sem reescrever o sistema como um todo.

O fluxo começa com a preparação dos dados, que é uma das etapas mais críticas do processo. Aqui, o sistema trata valores ausentes, normaliza variáveis numéricas, codifica variáveis categóricas e remove colunas irrelevantes para o objetivo da classificação. Tudo isso é feito de forma transparente, com logs detalhados e possibilidade de edição via interface gráfica por meio do Dtale, uma ferramenta que permite visualizar e ajustar os dados diretamente do navegador. Essa etapa garante que os dados estejam em um formato consistente e apropriado para o treinamento dos modelos, reduzindo ruído e aumentando a eficiência dos algoritmos que virão a seguir.

A seguir, o pipeline avança para a fase de modelagem. Utilizando a biblioteca PyCaret, o sistema executa automaticamente a comparação entre diferentes algoritmos de classificação como Random Forest, XGBoost, Gradient Boosting, Decision Tree, entre outros. Para cada modelo, são avaliadas métricas como acurácia, recall, precisão, F1-score e AUC-ROC. Essa comparação automatizada não apenas economiza tempo, mas também permite identificar com mais objetividade qual abordagem funciona melhor para os dados em questão. No caso do dataset utilizado neste estudo, o modelo que apresentou melhor desempenho foi o XGBoost, atingindo uma AUC-ROC de 0.93, o que indica uma excelente capacidade de separar corretamente os casos positivos (inadimplentes) dos negativos (adimplentes).

O sistema foi projetado para que a escolha do modelo seja tratada como um “componente trocável”. Isso significa que, embora o PyCaret facilite a seleção automática, nada impede que o usuário especifique diretamente outro algoritmo de preferência, ou mesmo substitua completamente o mecanismo de modelagem por outra biblioteca, como scikit-learn, lightgbm ou catboost. Esse nível de desacoplamento é possível graças ao uso de adaptadores pequenas interfaces que convertem os dados do formato interno do sistema para o formato exigido pelas bibliotecas externas, e vice-versa.

Uma vez treinado o modelo, o pipeline segue para a fase de avaliação e interpretação. Além das métricas quantitativas que indicam o desempenho geral do modelo, o sistema também produz explicações individuais para cada previsão por meio do método SHAP (SHapley Additive exPlanations). Essas explicações apontam o peso exato que cada variável teve na decisão final do modelo para um determinado solicitante de crédito. Por exemplo, é possível identificar que, para um cliente classificado como de alto risco, a variável “renda mensal” contribuiu negativamente, enquanto “tempo de relacionamento com o banco” teve um efeito positivo — e quantificar essas influências.

Essas explicações tornam o sistema auditável e transparente, permitindo que instituições financeiras justifiquem suas decisões com base em critérios objetivos e compreensíveis, tanto para seus clientes quanto para órgãos reguladores. Além disso, os valores SHAP podem ser agregados e visualizados de maneira gráfica, facilitando ainda mais a interpretação por usuários não técnicos.

Por fim, toda a execução do pipeline pode ser feita por meio de uma interface de linha de comando (CLI), onde cada etapa é ativada por um comando específico por exemplo, baixar o dataset do Kaggle, gerar o perfil dos dados, editar o arquivo, treinar o modelo, ou gerar relatórios explicativos. Essa abordagem reduz a dependência de notebooks Jupyter ou ambientes interativos, tornando o sistema mais próximo de um produto de software real, com potencial de integração em ambientes corporativos ou fluxos de produção automatizados.

Em resumo, a estrutura do modelo de classificação não se limita a treinar algoritmos: ela constrói um ecossistema completo que prepara, modela, interpreta e entrega valor de forma clara e replicável. Ao combinar precisão preditiva com transparência interpretativa, o sistema responde às necessidades técnicas e éticas do mercado financeiro, onde decisões automatizadas precisam ser não apenas eficazes, mas também explicáveis e confiáveis.

**2.2 IMPLEMENTAÇÃO DE CLASSIFICAÇÃO:**

Para a etapa de classificação dos solicitantes de crédito, optou-se pela construção de um pipeline modular em Python, com forte ênfase na organização do código, reprodutibilidade e flexibilidade de experimentação. A estrutura foi baseada no repositório Credit\_Card\_Aplication\_Kaggle, por permitir uma separação clara entre os blocos de pré-processamento, treino e avaliação, além de integrar visualização e interoperabilidade sem comprometer a simplicidade da execução.

Nesse pipeline, a escolha do algoritmo preditivo é feita de forma automatizada por meio da biblioteca PyCaret, que testa diferentes classificadores — como Random Forest, XGBoost, Gradient Boosting, Decision Trees, entre outros — e seleciona aquele com melhor desempenho com base em métricas como AUC, recall, F1-score e precisão. Para o conjunto de dados em questão, o modelo de maior desempenho foi o **XGBoost**, alcançando uma AUC-ROC de 0.93, o que demonstra alta capacidade de separação entre classes (adimplentes e inadimplentes).

Essa abordagem apresenta duas vantagens importantes frente à aplicação direta de métodos como o PowerSHAP. Primeiro, a estrutura modular permite que qualquer parte do pipeline seja facilmente substituída ou testada de forma independente por exemplo, trocar o algoritmo de classificação ou a forma de normalização sem que isso implique reescrever o projeto inteiro. Isso dá liberdade para personalização, tanto para ambientes de pesquisa quanto para contextos corporativos.

Segundo o pipeline oferece uma **interpretação mais integrada ao fluxo do modelo**. Embora o PowerSHAP seja excelente em identificar quais variáveis têm impacto estatisticamente significativo nas previsões, ele se aplica principalmente a um momento específico da análise (pós-treino e explicação). Já o pipeline proposto permite acompanhar o impacto das variáveis desde o início: é possível visualizar como transformações aplicadas no pré-processamento afetam o desempenho dos modelos e, em seguida, analisar suas contribuições por meio do SHAP tradicional, que já está incorporado ao processo.

Além disso, o PowerSHAP possui limitações de desempenho em conjuntos de dados maiores, pois seus testes estatísticos podem ser custosos computacionalmente. Em contraste, o pipeline desenvolvido permite reavaliações rápidas com diferentes modelos e configurações, facilitando a validação cruzada, o ajuste de hiper parâmetros e a integração com ferramentas de visualização como dtale e ydata-profiling. Essa flexibilidade é fundamental em contextos reais, onde decisões precisam ser feitas com base em múltiplas fontes e em tempo hábil.

Portanto, a escolha por essa arquitetura não foi apenas uma decisão técnica, mas estratégica: ela oferece precisão, interoperabilidade e adaptabilidade os três pilares que sustentam decisões confiáveis no contexto de análise de crédito.

**Figuras, Gráficos**

Interface gráfica do usuário, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

**Gráfico 1**: Estatísticas do Dataset

**Fonte**: dados da pesquisa

Tabela, Calendário

O conteúdo gerado por IA pode estar incorreto.

**Gráfico 2**: Total da amostragem

**Fonte**: dados da pesquisa

Gráfico

O conteúdo gerado por IA pode estar incorreto.

**Gráfico 3**: Correlações

**Fonte**: dados da pesquisa

Gráfico, Gráfico de barras

O conteúdo gerado por IA pode estar incorreto.

**Gráfico 4**: Valores Perdidos

**Fonte**: dados da pesquisa

**CONCLUSÃO**

A classificação de crédito continua sendo um desafio central no setor financeiro, exigindo modelos que não apenas entreguem precisão, mas que também ofereçam explicações claras e justificáveis para cada decisão. Ao longo deste trabalho, foi possível desenvolver um pipeline completo, em Python, que atende a essa demanda ao integrar aprendizado de máquina, análise interpretável e organização modular do código.

A proposta se destacou por unir, de forma estruturada, desempenho técnico e transparência. A classificação binária dos solicitantes de crédito entre bons e maus pagadores, foi realizada com o apoio de modelos automatizados via PyCaret, e ajustada conforme métricas como AUC-ROC, precisão e recall. No conjunto de dados utilizado, o XGBoost apresentou melhor desempenho, com 93% de área sob a curva, reforçando a robustez da solução. No entanto, o diferencial da abordagem não está apenas nos números, mas na forma como as decisões do modelo podem ser interpretadas e auditadas.

O uso do SHAP ajudou a entender com mais clareza quais variáveis estavam influenciando as decisões do modelo. Já o PowerSHAP entrou como um reforço estatístico, permitindo verificar se essas influências realmente faziam sentido do ponto de vista matemático. Com isso, foi possível ir além da explicação: houve confiança de que o modelo estava tomando decisões com base em fatores consistentes o que é essencial quando se lida com dados sensíveis, como no caso do crédito.

Outro ponto importante foi o formato em que o pipeline foi construído. A organização em blocos independentes como pré-processamento, treino, validação e explicação tornou o sistema mais fácil de adaptar. Isso significa que, se no futuro for preciso trocar o modelo, ajustar os critérios de risco ou trabalhar com uma nova base de dados, o restante do processo continua funcionando. Essa flexibilidade mostra que, mais do que técnica, a estrutura do projeto foi pensada para durar e acompanhar mudanças no cenário real.A separação entre componentes como pré-processamento, modelagem e explicação permite que o sistema evolua de forma simples, seja trocando bibliotecas, ajustando regras de negócio ou aplicando o mesmo fluxo em novos conjuntos de dados. Essa flexibilidade é particularmente importante em cenários reais, onde requisitos mudam com frequência e decisões precisam ser tomadas com agilidade, sem abrir mão da rastreabilidade.

Dessa forma, conclui-se que o pipeline proposto não apenas resolve o problema de classificação de crédito com precisão, mas o faz de maneira transparente, adaptável e replicável. Ele oferece às instituições uma ferramenta confiável para apoiar decisões críticas, conciliando eficiência técnica com responsabilidade e compreensão humana um equilíbrio cada vez mais necessário na aplicação de inteligência artificial em áreas sensíveis como o setor financeiro.

**REFERÊNCIAS**

PowerSHAP e Interpretabilidade de Modelos

LUNDBERG, S. M.; LEE, S.-I. A Unified Approach to Interpreting Model Predictions. In: Advances in Neural Information Processing Systems, v. 30, 2017. Disponível em: https://proceedings.neurips.cc/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf. Acesso em: 18/04/2025.

Teoria dos Jogos de Shapley

SHAPLEY, L. S. A Value for n-Person Games. In: KUHN, H. W.; TUCKER, A. W. (Eds.). Contributions to the Theory of Games. Princeton: Princeton University Press, 1953. v. 2, p. 307-317.

LGPD e Ética em IA

BRASIL. Lei nº 13.709, de 14 de agosto de 2018. Lei Geral de Proteção de Dados Pessoais (LGPD). Diário Oficial da União, Brasília, 15 ago. 2018. Disponível em: http://www.planalto.gov.br/ccivil\_03/\_ato2015-2018/2018/lei/l13709.htm. Acesso em:18/04/2025.

Aplicações de Machine Learning em Crédito

KRULIKOVSKI, E. H. M. Máquinas de Vetores de Suporte e Regressão Logística aplicados ao problema de análise de crédito. 2016. 36 f. Trabalho de Conclusão de Curso (Graduação em Estatística) - Universidade Federal do Paraná, Curitiba, 2016.

Base de Dados German Credit

HOFMANN, H. German Credit Dataset. UCI Machine Learning Repository, 1994. Disponível em: https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data). Acesso em: 18/04/2025.

Credit Card Application Dataset

JAVEED, N. Credit Card Application Dataset. Kaggle, 2020. Disponível em: https://www.kaggle.com/datasets/nazishjaveed/credit-card-application. Acesso em: 18/04/2025.

Métodos de Explicabilidade em Machine Learning

MOLNAR, C. Interpretable Machine Learning. 2. ed. 2022. Disponível em: https://christophm.github.io/interpretable-ml-book/. Acesso em: 18/04/2025.

**THE APPLICATION OF MACHINE LEARNING TO BINARY CREDIT SCORING MODELS**

**ABSTRACT:** Credit risk analysis requires not only accuracy but also transparency: financial institutions need to understand why a customer is classified as a defaulter. This study proposes a replicable method in Python that combines machine learning and explainability, using SHAP (SHapley Additive exPlanations) to quantify the impact of variables such as income, age, and payment history on the model’s decision. The developed pipeline is organized into modular stages – from data preprocessing (handling of missing values, normalization) to predictive modeling and interpretable analysis – ensuring reproducibility and adaptability across different contexts. Results show that transparency does not compromise accuracy: by assigning measurable weights to each factor (e.g., length of relationship with the bank contributes 18% to the classification), the method enables managers to base decisions on technical and auditable criteria. The practical contribution is clear: to provide institutions with a tool that balances statistical rigor and human understanding, which is essential in a sector where trust and risk go hand in hand.

**KEYWORDS:** Credit analysis, Machine learning, PowerSHAP.