Universidade Federal do Rio de Janeiro

Relatório Trabalho 1:

Classificador para apoio à decisão de aprovação de crédito

Estudante: Felipe Vasconcellos Nunes Gurgel Farias (119154566)

Professor: Heraldo Almeida

EEL891 - Introdução ao Aprendizado de Máquina - 2025.2

Conteúdo

1 Introdução

A concessão de crédito é uma atividade central para instituições financeiras, estando diretamente associada à avaliação do risco de inadimplência dos solicitantes. Decisões imprecisas nesse processo podem resultar em perdas financeiras significativas, enquanto critérios excessivamente conservadores podem levar à rejeição de clientes com bom perfil de pagamento. Nesse contexto, técnicas de Aprendizado de Máquina são extremamente úteis para serem empregadas nesse meio como sistemas de apoio à decisão, auxiliando à análise de grandes volumes de dados e na identificação de padrões relevantes para a classificação de risco.

Este trabalho tem como objetivo desenvolver e avaliar modelos de classificação supervisionada capazes de prever a inadimplência de solicitantes de crédito a partir de um conjunto de dados tabular contendo informações demográficas, financeiras e cadastrais. A variável-alvo é binária, indicando se o solicitante se tornou inadimplente ou não, caracterizando um problema clássico de classificação.

Para a construção do sistema proposto, foram avaliados modelos representativos de diferentes abordagens de aprendizado, incluindo Regressão Logística, Random Forest e XGBoost. O desempenho dos modelos foi comparado por meio de um protocolo experimental padronizado, visando identificar a abordagem mais adequada para o problema em estudo.

2 Descrição dos Dados

Os dados utilizados neste trabalho foram fornecidos dentro do contexto da competição (no Kaggle) proposta pela disciplina e organizados em quatro arquivos principais, descritos a seguir:

- ***conjunto_de_treinamento.csv***: conjunto contendo 20.000 registros históricos de solicitações de crédito, com variáveis demográficas, financeiras e cadastrais dos solicitantes, além da variável-alvo *inadimplente*. Essa variável é binária e indica se o solicitante tornou-se inadimplente (1) ou quitou o contrato (0), caracterizando um problema de classificação supervisionada binária.
- ***conjunto_de_teste.csv***: conjunto composto por 5.000 registros com a mesma estrutura de variáveis explicativas do conjunto de treinamento, porém sem o desfecho do contrato. Esse conjunto é utilizado exclusivamente para a geração das previsões submetidas à plataforma Kaggle.
- ***exemplo_arquivo_respostas.csv***: arquivo ilustrativo que define o formato esperado para a submissão das previsões, contendo o identificador de cada solicitação e o respectivo valor predito para a variável-alvo.
- ***dicionario_de_dados.xlsx***: planilha contendo a descrição detalhada de todas as variáveis presentes nos arquivos CSV, utilizada para a identificação automática das variáveis numéricas e categóricas no processo de pré-processamento.

As variáveis disponíveis incluem atributos numéricos, como idade, renda e tempo de vínculo empregatício, bem como variáveis categóricas relacionadas a características pessoais e cadastrais dos solicitantes. A separação entre variáveis numéricas e categóricas foi realizada com base nas descrições fornecidas no dicionário de dados, garantindo maior reprodutibilidade e evitando classificações manuais.

Além disso, cabe destacar que a variável *id_solicitante* possui função identificadora e não representa uma característica do solicitante. Dessa forma, ela foi removida do conjunto de variáveis explicativas antes do treinamento dos modelos, a fim de evitar a indução de padrões artificiais e possíveis problemas de sobreajuste.

3 Metodologia

3.1 Carregamento e Organização dos Dados

As principais bibliotecas utilizadas nesta etapa do trabalho foram:

- `pandas`: utilizada para leitura, manipulação e organização dos dados tabulares provenientes dos arquivos CSV e Excel;
- `NumPy`: empregada como suporte para operações numéricas e manipulação eficiente de estruturas de dados;
- `scikit-learn`: utilizada nas etapas posteriores para pré-processamento, divisão dos dados, treinamento e avaliação dos modelos;
- `XGBoost`: biblioteca especializada em métodos de boosting, utilizada na construção do modelo final de classificação.

Os conjuntos de dados foram carregados em ambiente Python utilizando a biblioteca `pandas`. Os arquivos de treino e teste foram mantidos separados desde o início do processo, a fim de evitar vazamento de informação (*data leakage*).

O conjunto de treinamento foi utilizado tanto para a análise exploratória quanto para o ajuste dos modelos, enquanto o conjunto de teste permaneceu intocado até a etapa final de geração das previsões para submissão na plataforma Kaggle.

Adicionalmente, o dicionário de dados fornecido foi carregado e utilizado como base para a identificação automática dos tipos de variáveis, contribuindo para a reprodutibilidade e padronização do pipeline de pré-processamento.

Os dados foram carregados utilizando a biblioteca `pandas`, conforme ilustrado no trecho de código a seguir:

```
train_data = pd.read_csv("conjunto_de_treinamento.csv")
test_data  = pd.read_csv("conjunto_de_teste.csv")
dicionario = pd.read_excel("dicionario_de_dados.xlsx")
```

Listing 1: *Carregamento dos conjuntos de dados*

3.2 Seleção e Classificação das Variáveis

A identificação das variáveis numéricas e categóricas foi realizada de forma automatizada com base no dicionário de dados fornecido. Essa estratégia evita classificações manuais, reduz a possibilidade de erros humanos e aumenta a reprodutibilidade do experimento.

As variáveis foram classificadas a partir da descrição textual presente no dicionário, utilizando palavras-chave associadas a atributos categóricos (como códigos, categorias, sexo e estado civil) e numéricos (como valores monetários, idade e tempo em meses). Após essa etapa, foi realizada uma verificação para garantir que apenas variáveis efetivamente presentes no conjunto de dados fossem consideradas.

Adicionalmente, a variável `id_solicitante`, apesar de classificada como numérica no dicionário, foi removida do conjunto de variáveis explicativas. Essa variável possui caráter exclusivamente identificador e não representa uma característica intrínseca do solicitante, podendo induzir padrões artificiais e comprometer a capacidade de generalização dos modelos.

O processo de identificação e filtragem das variáveis é apresentado no trecho de código a seguir:

```
# Variaveis categoricas: detectadas pela descricao no dicionario
categorical_vars = dicionario[
    dicionario["Descricao"].str.contains(
        "codigo|categoria|forma|sexo|estado",
        case=False,
        na=False
    )
]["Nome da Variavel"].tolist()

# Variaveis numericas: detectadas pela descricao no dicionario
numerical_vars = dicionario[
    dicionario["Descricao"].str.contains(
        "valor|numero|idade|renda|meses",
        case=False,
        na=False
    )
]["Nome da Variavel"].tolist()

# Garantir que as variaveis existam no dataset
categorical_vars = [c for c in categorical_vars if c in
   train_data.columns]
```

```
numerical_vars = [c for c in numerical_vars if c in train_data.
    columns]

# Remocao explicita de identificadores
drop_ids = ["id_solicitante"]
categorical_vars = [c for c in categorical_vars if c not in
    drop_ids]
numerical_vars = [c for c in numerical_vars if c not in drop_ids]
```

Listing 2: *Seleção automática de variáveis numéricas e categóricas*

3.3 Análise Exploratória dos Dados (EDA)

A análise exploratória dos dados teve como objetivo compreender as características do conjunto de treinamento, avaliar a distribuição da variável-alvo e investigar possíveis relações entre as variáveis explicativas, fornecendo subsídios para as etapas subsequentes de modelagem. Cabe destacar que toda a análise foi realizada exclusivamente sobre o conjunto de treinamento, de modo a evitar vazamento de informação (*data leakage*).

Inicialmente, foi analisada a distribuição da variável-alvo *inadimplente*, conforme ilustrado na Figura 1. A partir deste gráfico foi observado que as classes estão balanceadas, com proporções semelhantes entre solicitantes inadimplentes e adimplentes. Esse comportamento indica que, a princípio, não há necessidade de aplicação de técnicas específicas de balanceamento de classes, permitindo a utilização direta de métricas tradicionais de avaliação, como acurácia, precisão, recall e F1-score.

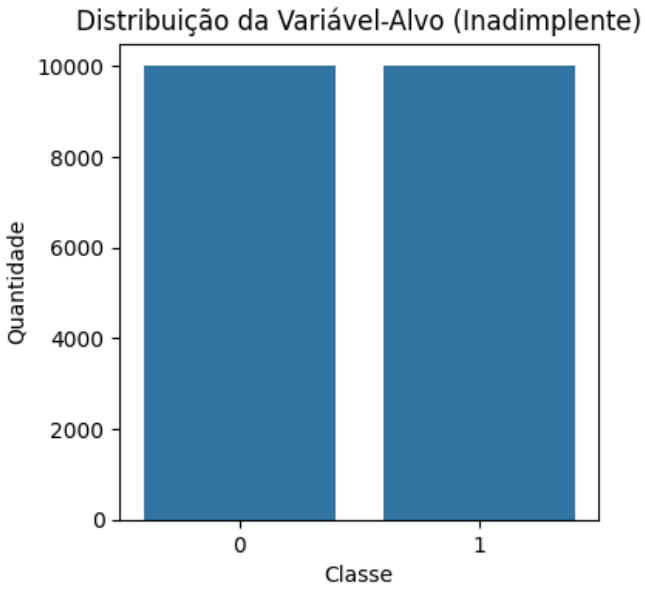

Figura 1: Distribuição da variável-alvo (inadimplente) no conjunto de treinamento.

Em seguida, foi realizada a análise de correlação entre as variáveis numéricas por meio do coeficiente de correlação de Pearson, cujo mapa de calor é apresentado na Figura 2.

Essa análise permite identificar relações lineares fortes entre variáveis, bem como possíveis indícios de multicolinearidade.

Os resultados indicam que, de modo geral, as variáveis numéricas apresentam correlações fracas entre si, com exceção de pares específicos, como a quantidade de contas bancárias e a quantidade de contas bancárias especiais, que exibem correlação mais elevada. Tal comportamento é esperado, dado o significado semântico dessas variáveis.

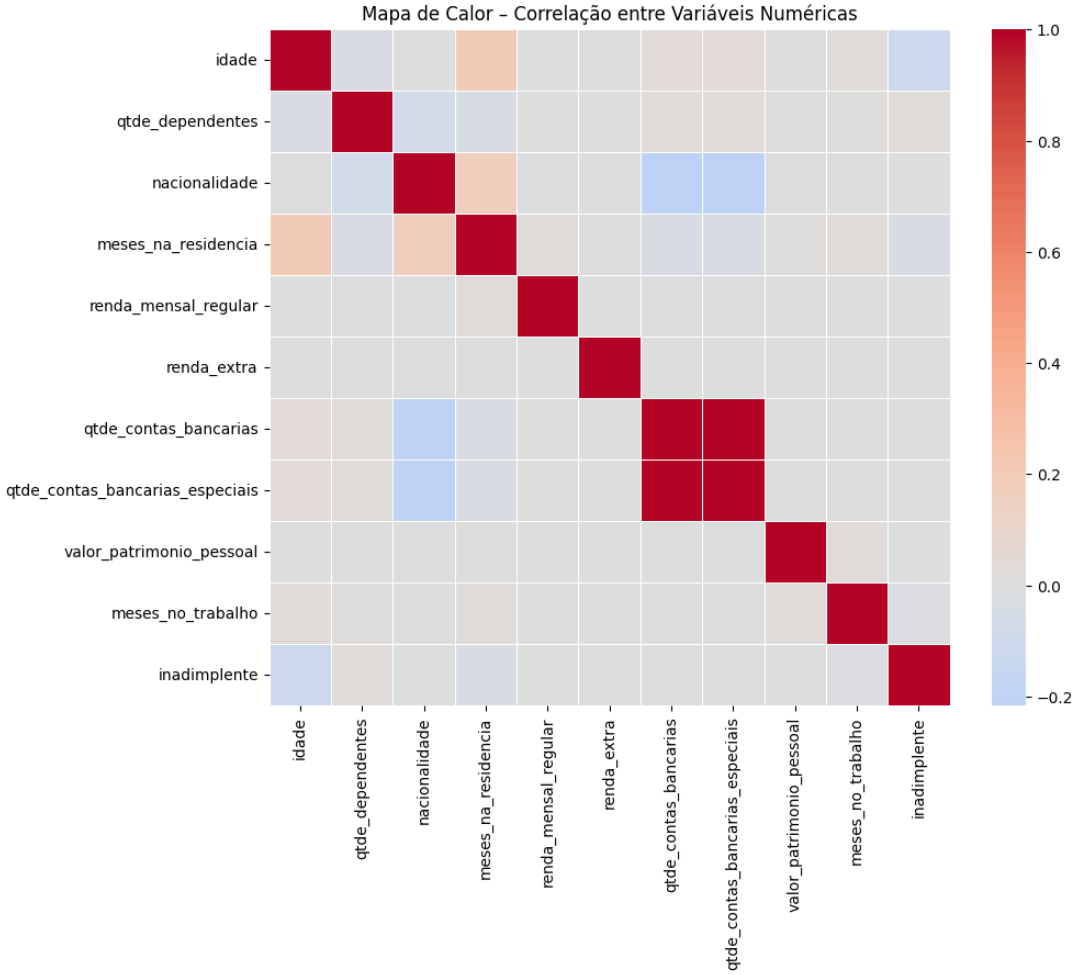

Figura 2: Mapa de calor da correlação entre variáveis numéricas do conjunto de treinamento.

Por fim, foi avaliada a correlação individual entre cada variável numérica e a variável-alvo, conforme apresentado na Figura 3, permitindo observar que os coeficientes de correlação são, em sua maioria, próximos de zero, indicando baixa relação linear direta entre as variáveis explicativas e a inadimplência.

Esse resultado sugere que modelos puramente lineares podem apresentar desempenho limitado, reforçando a motivação para a utilização de modelos capazes de capturar relações não lineares e interações complexas entre variáveis, como Random Forest e XGBoost, explorados nas etapas posteriores deste trabalho.

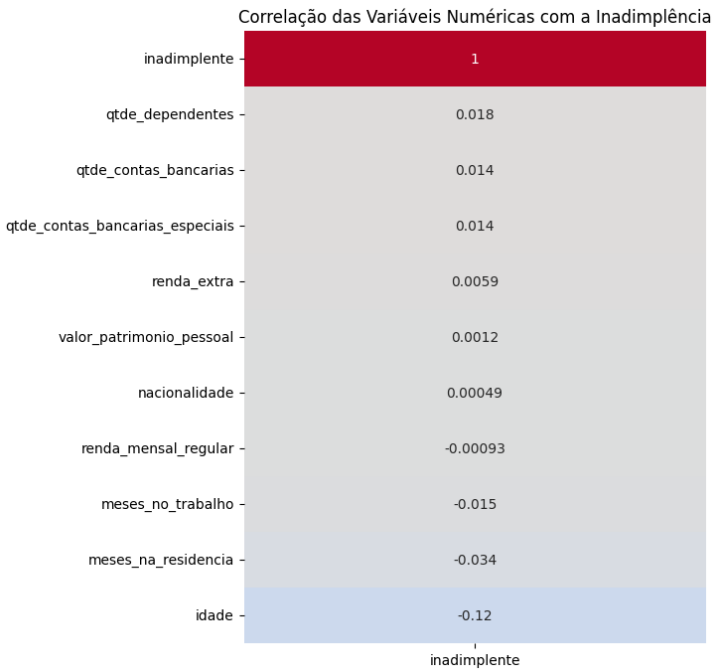

Figura 3: Correlação das variáveis numéricas com a variável-alvo (inadimplência).

3.4 Pré-processamento dos Dados

O pré-processamento dos dados foi realizado de forma sistemática por meio da utilização de *pipelines*, com o objetivo de garantir reprodutibilidade, organização do fluxo experimental e evitar vazamento de informação entre as etapas de treinamento e validação dos modelos.

As variáveis numéricas e categóricas foram tratadas de maneira distinta, respeitando suas características estatísticas e semânticas. Para as variáveis numéricas, adotou-se a imputação de valores ausentes pela mediana, por se tratar de uma medida robusta à presença de outliers. Em seguida, foi aplicada a padronização por meio do *StandardScaler*, assegurando que todas as variáveis numéricas apresentassem média zero e variância unitária, o que é especialmente relevante para modelos sensíveis à escala, como a Regressão Logística.

As variáveis categóricas foram submetidas à imputação pelo valor mais frequente, seguida da codificação *One-Hot Encoding*. Essa estratégia permite representar categorias de forma binária, sem impor qualquer ordem artificial entre seus valores. Além disso, foi utilizado o parâmetro `handle_unknown='ignore'`, garantindo robustez do modelo diante de categorias eventualmente não observadas no conjunto de treinamento.

O tratamento conjunto das variáveis foi realizado por meio do *ColumnTransformer*, que aplica transformações específicas a subconjuntos de atributos definidos previamente. Essa abordagem assegura que o mesmo pré-processamento seja aplicado de maneira consistente em todas as etapas do experimento, incluindo validação e predição final. O pipeline de pré-processamento adotado neste trabalho é apresentado no Listing 3:

```
numeric_transformer = Pipeline([
    ("imputer", SimpleImputer(strategy="median")),
    ("scaler", StandardScaler())
])

categorical_transformer = Pipeline([
    ("imputer", SimpleImputer(strategy="most_frequent")),
    ("ohe", OneHotEncoder(handle_unknown="ignore", sparse_output=
       False))
])

preprocessor = ColumnTransformer(
    [
        ("num", numeric_transformer, numerical_vars),
        ("cat", categorical_transformer, categorical_vars)
    ],
    remainder="drop"
)
```

Listing 3: *Pipeline de pré-processamento dos dados*

3.5 Divisão dos Dados (Treino / Validação)

Após as etapas de carregamento, organização e pré-processamento dos dados, o conjunto de treinamento foi dividido em subconjuntos de treino e validação com o objetivo de avaliar a capacidade de generalização dos modelos propostos.

Inicialmente, as variáveis explicativas (*features*) foram separadas da variável-alvo *inadimplente*. Em seguida, a variável *id_solicitante* foi removida do conjunto de variáveis explicativas, uma vez que possui caráter exclusivamente identificador e não representa uma característica intrínseca do solicitante. A inclusão desse tipo de atributo poderia induzir padrões artificiais e comprometer a capacidade de generalização dos modelos, caracterizando sobreajuste.

A divisão dos dados foi realizada utilizando o método *hold-out*, com 80% das amostras destinadas ao treinamento e 20% à validação. Para garantir a manutenção da proporção entre as classes da variável-alvo em ambos os subconjuntos, foi empregada a estratégia de amostragem estratificada, assegurando uma distribuição equilibrada das classes.

Essa abordagem permite uma avaliação consistente do desempenho dos modelos durante a fase de validação, ao mesmo tempo em que preserva um conjunto independente de dados para a etapa de predição final e submissão à plataforma Kaggle.

Essa abordagem permite uma avaliação consistente do desempenho dos modelos durante a fase de validação, ao mesmo tempo em que preserva um conjunto independente de dados para a etapa de predição final e submissão à plataforma Kaggle. O procedimento de separação e estratificação empregado está apresentado no *Listing* 4:

```python
X = train_data.drop(columns=["inadimplente"])
y = train_data["inadimplente"]

if "id_solicitante" in X.columns:
    X = X.drop(columns=["id_solicitante"])

X_train, X_val, y_train, y_val = train_test_split(
    X, y,
    test_size=0.20,
    random_state=42,
    stratify=y
)
```

Listing 4: *Divisão dos dados em treino e validação*

3.6 Modelagem

Nesta etapa, foram avaliados três modelos de classificação supervisionada pertencentes a diferentes famílias de aprendizado de máquina, selecionados de forma a permitir a comparação entre abordagens lineares e não lineares na tarefa de previsão de inadimplência.

Os modelos considerados neste trabalho são descritos a seguir:

- **Regressão Logística**: utilizada como modelo baseline, por sua simplicidade, eficiência computacional e elevada interpretabilidade, sendo amplamente empregada em problemas de classificação binária.
- **Random Forest**: modelo baseado em um conjunto de árvores de decisão treinadas de forma independente, capaz de capturar relações não lineares entre as variáveis e reduzir variância por meio do mecanismo de *bagging*.
- **XGBoost**: algoritmo da família de métodos de *boosting*, reconhecido por seu alto desempenho em dados tabulares, explorando gradativamente erros residuais e permitindo a modelagem de interações complexas entre atributos.

Todos os modelos foram implementados utilizando a mesma estrutura de pré-processamento por meio de *pipelines* do *scikit-learn*, assegurando uma comparação justa entre as abordagens e evitando vazamento de informações entre as etapas de transformação e treinamento.

A definição dos modelos e de seus principais hiperparâmetros é apresentada no *Listing* 5:

```python
# Regressao Logistica (baseline)
log_model = Pipeline([
    ("prep", preprocessor),
    ("clf", LogisticRegression(max_iter=900, C=0.7))
```

```
])

# Random Forest
rf_model = Pipeline([
    ("prep", preprocessor),
    ("clf", RandomForestClassifier(
        n_estimators=600,
        max_depth=20,
        min_samples_split=3,
        min_samples_leaf=2,
        random_state=42
    ))
])

# XGBoost
xgb_model = Pipeline([
    ("prep", preprocessor),
    ("clf", XGBClassifier(
        n_estimators=700,
        learning_rate=0.03,
        max_depth=6,
        subsample=0.85,
        colsample_bytree=0.85,
        min_child_weight=6,
        gamma=0.4,
        reg_alpha=0.8,
        reg_lambda=2.0,
        eval_metric="logloss",
        tree_method="hist",
        random_state=42,
        n_jobs=-1
    ))
])

models = {
    "Logistic Regression": log_model,
    "Random Forest": rf_model,
    "XGBoost": xgb_model
}
```

Listing 5: *Definição dos modelos de classificação*

3.7 Estratégias de Avaliação

```
for nome, modelo in modelos.items():
    print(f"\nTreinando o [Modelo {nome}]...")
    print("\nRelatorio:")

    modelo.fit(X_train, y_train)
    preds = modelo.predict(X_val)
```

```
7
8     print(classification_report(y_val, preds))
```

Listing 6: *Treinamento e validação dos modelos utilizando estratégia hold-out*

A avaliação do desempenho dos modelos foi realizada por meio da estratégia de validação do tipo *hold-out*. O conjunto de dados de treinamento foi dividido em duas partes: um subconjunto destinado ao ajuste dos modelos (80%) e um subconjunto reservado exclusivamente para validação (20%), garantindo a separação entre dados utilizados no treinamento e na avaliação do desempenho.

A divisão foi efetuada de forma estratificada, de modo a preservar a proporção das classes da variável-alvo em ambos os subconjuntos. Essa abordagem contribui para uma avaliação mais representativa do comportamento dos modelos em dados não vistos.

Durante a fase de validação, os modelos foram avaliados utilizando métricas clássicas de classificação binária, escolhidas de forma a capturar diferentes aspectos do desempenho preditivo. As métricas consideradas foram:

- **Accuracy**: proporção total de previsões corretas realizadas pelo modelo.
- **Precision**: razão entre previsões positivas corretas e o total de previsões positivas, indicando o grau de confiabilidade das predições de inadimplência.
- **Recall**: razão entre inadimplentes corretamente identificados e o total de inadimplentes reais, refletindo a capacidade do modelo em detectar clientes inadimplentes.
- **F1-score**: média harmônica entre precision e recall, utilizada como medida de equilíbrio entre ambas as métricas.

Essas métricas permitem uma análise abrangente do desempenho, considerando tanto a capacidade de identificar corretamente inadimplentes quanto a de evitar classificações incorretas de bons pagadores.

O processo de treinamento e avaliação dos modelos no conjunto de validação é ilustrado no *Listing* 6, no qual os modelos são treinados no conjunto de treino e avaliados no conjunto de validação por meio do relatório de classificação.

Cabe destacar que, embora métricas adicionais tenham sido consideradas para uma análise mais abrangente do comportamento dos modelos, a escolha do modelo final foi guiada prioritariamente pelo desempenho em termos de *accuracy*.

3.8 Predição Final e Submissão

Após a etapa de avaliação, o modelo com melhor desempenho em termos de *accuracy* no conjunto de validação foi selecionado para a geração das previsões finais. Esse modelo foi então treinado novamente utilizando todo o conjunto de dados de treinamento, com o objetivo de maximizar o uso das informações disponíveis.

Antes da etapa de predição, o conjunto de teste foi preparado de modo a garantir consistência com os dados utilizados no treinamento. Em particular, a variável *id_solicitante*, por possuir caráter exclusivamente identificador, foi removida, assegurando que apenas variáveis preditivas fossem utilizadas.

Em seguida, as previsões foram geradas e organizadas no formato exigido pela plataforma Kaggle, contendo o identificador de cada solicitação e o respectivo valor predito para a variável-alvo. O processo completo de treinamento final e geração do arquivo de submissão é apresentado no Listing 7:

```
best_model = xgb_model
best_model.fit(X, y)

X_test = test_data.copy()
X_test = X_test.drop(columns=["id_solicitante"], errors="ignore")
X_test = X_test[X.columns]

final_preds = best_model.predict(X_test)

submission = pd.DataFrame({
    "id_solicitante": test_data["id_solicitante"],
    "inadimplente": final_preds
})

submission.to_csv("submission.csv", index=False)
```

Listing 7: *Treinamento final do modelo selecionado e geração do arquivo de submissão*

4 Resultados

Durante a etapa de validação interna, os três modelos de classificação avaliados apresentaram desempenhos distintos. A Regressão Logística foi utilizada como modelo baseline, apresentando desempenho inferior em relação aos demais, especialmente na capacidade de capturar padrões mais complexos associados à inadimplência.

O modelo Random Forest obteve desempenho intermediário, demonstrando maior flexibilidade em relação à Regressão Logística por ser capaz de modelar relações não lineares entre as variáveis. Ainda assim, seu desempenho permaneceu inferior ao obtido pelo XGBoost, principalmente em termos de equilíbrio entre as métricas de avaliação.

O modelo XGBoost apresentou o melhor desempenho geral dentre os modelos avaliados. Sua estrutura baseada em boosting permitiu a construção sequencial de árvores, corrigindo erros dos modelos anteriores e proporcionando maior capacidade de generalização. Além disso, os hiperparâmetros adotados contribuíram para maior estabilidade e controle de sobreajuste.

Após a seleção do modelo com melhor desempenho na fase de validação, o XGBoost

foi treinado utilizando 100% dos dados rotulados disponíveis. Em seguida, o modelo foi empregado na geração do arquivo de submissão para a plataforma Kaggle.

A acurácia final obtida no leaderboard público da competição foi de **0,6076**, representando o melhor resultado alcançado ao longo do desenvolvimento do trabalho e confirmando a adequação do modelo selecionado para o problema proposto.

5 Discussão dos Resultados

Os resultados obtidos evidenciam diferenças claras entre as abordagens de modelagem avaliadas. A Regressão Logística, embora simples e interpretável, apresentou limitações por assumir relações lineares entre as variáveis explicativas e a variável-alvo, o que reduz sua capacidade de capturar padrões mais complexos presentes nos dados.

O modelo Random Forest apresentou melhorias em relação ao baseline, principalmente por explorar múltiplas árvores de decisão e reduzir a variância do modelo. Entretanto, sua abordagem baseada em bagging mostrou-se menos eficiente do que o boosting no contexto deste conjunto de dados.

O desempenho superior do XGBoost pode ser atribuído a diversos fatores, dentre eles:

- a construção sequencial de árvores, que permite corrigir erros cometidos em iterações anteriores;
- o uso de regularização explícita, reduzindo o risco de sobreajuste;
- a capacidade de lidar eficientemente com dados tabulares e variáveis categóricas codificadas via One-Hot Encoding;
- o ajuste criterioso de hiperparâmetros, contribuindo para maior estabilidade do modelo.

Apesar dos resultados satisfatórios, algumas limitações devem ser destacadas. A acurácia, métrica priorizada neste trabalho por ser a utilizada na competição, pode não ser a mais adequada em cenários com forte desbalanceamento de classes. Além disso, técnicas adicionais como validação cruzada, calibração de probabilidades ou ensembles mais sofisticados poderiam potencialmente melhorar o desempenho, embora estivessem fora do escopo proposto.

6 Conclusão

Este trabalho apresentou o desenvolvimento de um sistema completo de apoio à decisão para aprovação de crédito, utilizando técnicas de aprendizado de máquina supervisionado. O processo envolveu todas as etapas fundamentais de um fluxo de modelagem preditiva, incluindo:

- Carregamento e organização dos dados;
- Análise exploratória;
- Pré-processamento padronizado por meio de pipelines;
- Comparação entre diferentes algoritmos de classificação;
- Avaliação sistemática de desempenho;
- Geração de submissão para a plataforma Kaggle.

Dentre os modelos avaliados, o XGBoost apresentou o melhor desempenho, alcançando uma acurácia final de **0,6076**. Esse resultado evidencia a eficácia de métodos de boosting para problemas de classificação em dados tabulares, especialmente quando combinados com estratégias adequadas de pré-processamento.

É notório, portanto, que a abordagem adotada é apropriada para o problema de previsão de inadimplência, fornecendo uma base sólida para aplicações práticas e estudos futuros envolvendo modelos mais avançados ou métricas alternativas de avaliação.

Além disso, para aprimoramento de desempenho do algoritmo no futuro, será importante a investigação de estratégias adicionais, tais como a utilização de validação cruzada para seleção mais robusta de hiperparâmetros, a calibração das probabilidades preditas e a avaliação de métricas alternativas mais adequadas a cenários desbalanceados, como a área sob a curva ROC. Adicionalmente, a exploração de ensembles mais complexos e técnicas de seleção de atributos pode contribuir para ganhos adicionais de desempenho e interpretabilidade.

7 Referências

PEDREGOSA, F. et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011.

LESSMANN, S. et al. Benchmarking state-of-the-art classification algorithms for credit scoring. *European Journal of Operational Research*, v. 247, n. 1, p. 124–136, 2015.

CHEN, T.; GUESTRIN, C. XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

KE, G. et al. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. NBR 6023:2023 – Referências. Rio de Janeiro, 2023.