Case Técnico - Credit Scoring

0. Objetivo e Escopo

O presente desafio tem como objetivo a construção de um modelo de *Credit Scoring*. A base disponibilizada contém 81 variáveis e 10.738 registros, sendo que o "id" é a chave única da tabela, "safra" é o mês ano de concessão do crédito, "y" é a variável target e as demais são variáveis preditoras mascaradas

O escopo do projeto abrange as etapas de exploração e tratamento dos dados, seleção de variáveis, construção e avaliação de modelos supervisionados, análise de estabilidade (tanto das variáveis quanto do modelo ao longo do tempo) e recomendação de ações com base nos resultados. Adicionalmente, são destacadas as métricas de desempenho e a análise de calibração do modelo, buscando assegurar aplicabilidade prática em cenários reais de tomada de decisão de crédito.

A base foi dividida respeitando a cronologia das concessões de crédito, utilizando-se as safras de outubro, novembro e dezembro como conjunto de validação Out-of-Time (OOT), a fim de simular o comportamento futuro do modelo e testar sua estabilidade temporal.

0.1 Estrutura do Projeto

O projeto foi organizado de forma modular para garantir escalabilidade, reprodutibilidade e facilidade de manutenção. Abaixo está a descrição dos principais diretórios e suas finalidades:

- `data/`
  Contém os conjuntos de dados utilizados no projeto, organizados por nível de processamento:
  - `raw/`: bases internas originais, sem qualquer tipo de tratamento.
  - `processed/`: bases de dados que já passaram por etapas de pré-processamento.
  - `external/`: fontes de dados externas que complementam a análise ou o modelo.

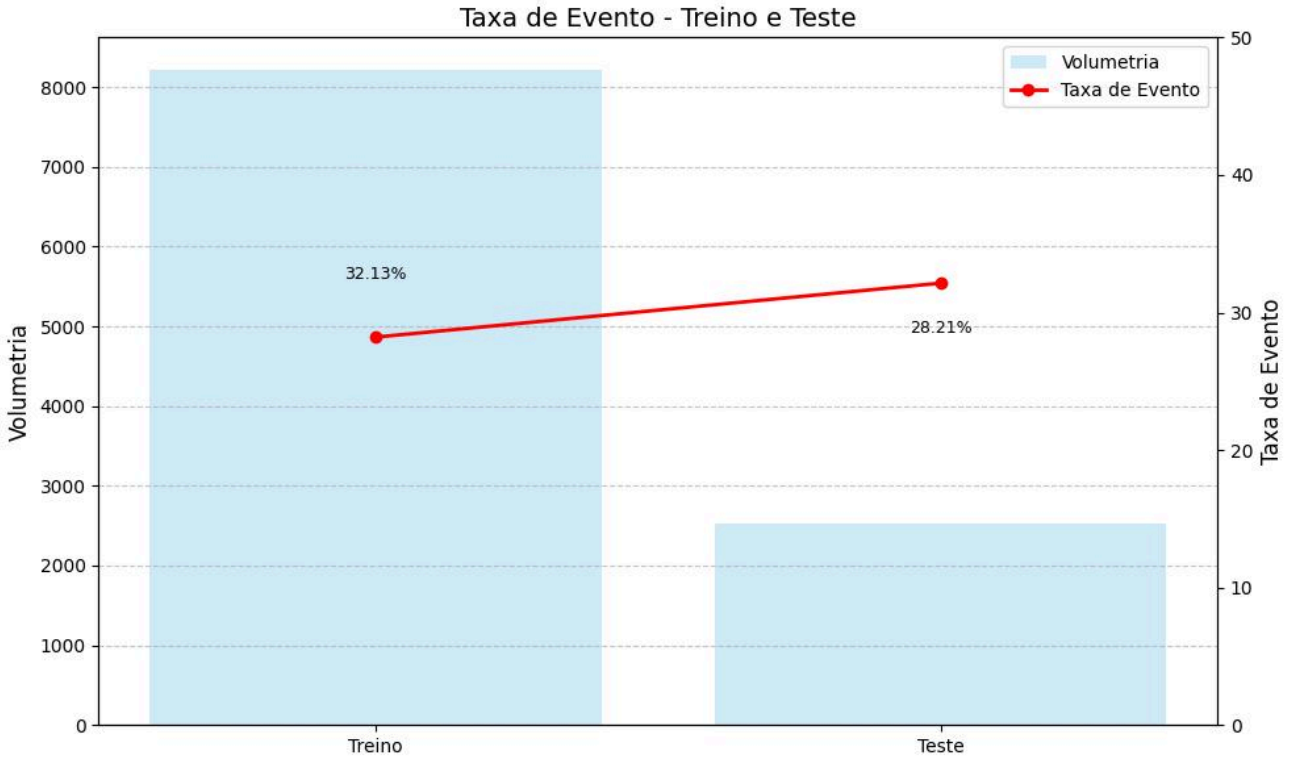
- `models/`
  Repositório para artefatos gerados durante o treinamento e avaliação dos modelos:
  - `artefacts/`: objetos utilizados no pipeline de processamento, como imputers, scalers e encoders.
  - `trained_model/`: modelo final treinado com os hiperparâmetros otimizados, pronto para uso em produção.
- `notebooks/`
  Contém os notebooks Jupyter utilizados nas diferentes etapas do projeto, incluindo exploração dos dados, pré-processamento, modelagem e análise de resultados.
- `reports/`
  Diretório destinado ao armazenamento de artefatos visuais e documentos gerados:
  - gráficos, logs de execução, relatórios analíticos e apresentações finais.
- `src/`
  Módulo com scripts e funções customizadas desenvolvidas ao longo do projeto, organizadas por finalidade (ex: pré-processamento, avaliação de modelos, utilitários).

---

1. Análise geral do público

A base de dados é composta por 10.738 registros e apresenta uma taxa média de evento de 29,13%. Não foram identificados valores nulos na variável target (`y`).

As safras analisadas apresentam uma volumetria relativamente estável ao longo do tempo, assim como a taxa de evento. Apesar de as duas primeiras safras apresentarem índices um pouco mais elevados, essa variação não é considerada significativa a ponto de comprometer a estabilidade do conjunto. Por isso, elas serão incluídas no processo de treinamento. As safras de outubro, novembro e dezembro, por sua vez, serão utilizadas como **Out-of-Time (OOT)** para validação do modelo, visando assegurar sua estabilidade e confiabilidade em cenários futuros.



Após a separação inicial entre as safras de treino e teste (Out-of-Time), a base de treino ficou composta por 8.211 registros, com uma taxa de evento de 28,21%, enquanto a base de teste contém 2.527 registros (23,50% do total), com taxa de evento de 32,13%. A amostra de teste apresenta volumetria suficiente para a validação do modelo, e a diferença entre as taxas de evento é considerada estável, permitindo prosseguir com o treinamento.

---

2. Tratamento das variáveis

2.1. Premissas Iniciais

Considerando que a modelagem é um processo iterativo, as premissas adotadas têm como objetivo simplificar o treinamento inicial e estabelecer uma base de parâmetros para futuras experimentações e iterações envolvendo o modelo, os dados e seu pré-processamento.

2.1.1. Exclusão de variáveis com alto percentual de nulos

As variáveis que apresentarem percentual de valores **nulos maior ou igual a 60,00%** serão **removidas** da base de treinamento, por serem consideradas pouco informativas e potencialmente prejudiciais à performance do modelo.

Quantidade de variáveis iniciais: 81
Quantidade de variáveis após filtragem dos nulos: 62

2.1.2. Identificando variáveis categóricas

Embora todas as colunas tenham sido inicialmente interpretadas como numéricas, aquelas com cardinalidade menor ou igual a 25 serão tratadas como variáveis categóricas. Todos os procedimentos subsequentes de pré-processamento e modelagem considerarão essa classificação.

Quantidade de variáveis numéricas: 44
Quantidade de variáveis categóricas: 16

2.1.3. Remoção de variáveis altamente correlacionadas - Variáveis numéricas

Com o objetivo de simplificar o modelo e garantir a qualidade dos dados utilizados no treinamento, foram removidas variáveis que apresentavam alta correlação (**igual ou superior a 90%**) com outras variáveis do conjunto. Como critério de escolha para remoção entre os pares correlacionados, foi excluída a variável com o maior percentual de valores nulos. Em casos de empate na quantidade de nulos, optou-se por manter a variável com maior cardinalidade, buscando preservar a variabilidade dos dados e, assim, permitir que o modelo capture padrões com maior precisão e alcance melhor performance.

Quantidade de variáveis numéricas: 39

2.1.4. Remoção de variáveis com baixa variância - Variáveis numéricas

Com o objetivo de identificar variáveis numéricas com baixa variância — que tendem a ter pouca contribuição para o poder preditivo do modelo e podem aumentar o risco de overfitting — foi realizada uma análise considerando um limiar mínimo de variância de 0,05. Apenas as variáveis que excederam esse valor foram mantidas na base de treino. No entanto, nenhuma variável foi removida nesse processo, indicando que todas apresentaram variabilidade suficiente para serem consideradas no modelo.

2.1.5. Tratamento dos nulos - Variáveis numéricas

Os valores nulos remanescentes foram substituídos pela média aritmética de ada variável. Essa abordagem visa preservar o volume total de dados e evitar a exclusão de informações relevantes para o treinamento do processo.

O arquivo para o tratamento dos dados de validação/produção estão salvos no diretório:

```
"/models/artefacts/num_imputer.pickle"
```

2.1.6. Normalização - Variáveis numéricas

Após a imputação dos valores ausentes, foi realizada a normalização das variáveis, uma vez que muitos dos algoritmos utilizados são sensíveis a diferenças de escala entre as variáveis, o que pode comprometer seu desempenho.

O arquivo para o tratamento dos dados de validação/produção estão salvos no diretório:

```
"/models/artefacts/num_scaler.pickle"
```

2.1.7. Remoção de colunas duplicadas - Variáveis categóricas

Durante a inspeção das colunas categóricas não foram identificadas quaisquer pares de colunas que sejam exatamente iguais.

2.1.8. Remoção de variáveis altamente correlacionadas - Variáveis categóricas

Embora não tenham sido identificadas colunas completamente idênticas, foram encontrados alguns pares com correlação superior a 90%, indicando um alto nível de redundância. Essa redundância pode aumentar o custo computacional e não contribui de forma significativa para a variabilidade do modelo. Por esse motivo, as colunas redundantes foram removidas do conjunto de dados.

Quantidade de Variáveis Categóricas: 11

2.1.9. Tratamento dos nulos - Variáveis categóricas

O tratamento de valores nulos em variáveis categóricas foi realizado com base no percentual de ausência de dados. Para variáveis com mais de 30% de valores nulos, foi utilizado um marcador genérico (`'9999'`) para sinalizar ausência de informação de forma explícita. Já para as variáveis com menor proporção de nulos (igual ou inferior a 30%), a imputação foi feita utilizando a moda de cada variável, preservando a consistência e a distribuição dos dados originais.

O arquivo para o tratamento dos dados de validação/produção estão salvos no diretório:

```
"/models/artefacts/cat_imputer.pickle"
```

2.1.10 Dummy encoding - Variáveis categóricas

Em seguida, foi aplicado o *dummy encoding* (One-Hot Encoding) às variáveis categóricas, permitindo que algoritmos que exigem entradas numéricas consigam processar essas informações de forma adequada.

O arquivo para o tratamento dos dados de validação/produção estão salvos no diretório:

```
"/models/artefacts/cat_one_hot.pickle"
```

2.1.11 Salvando colunas de interesse e criação da função de tratamento

As colunas de interesse selecionadas durante o processo de tratamento foram salvas em formato `pickle`, organizadas nas seguintes categorias:

- **Variáveis auxiliares**: colunas como `id` e `safra`
  - Caminho: `"/models/artefacts/aux_vars.pickle"`
- **Variável alvo (target)**: coluna que representa a variável resposta
  - Caminho: `"/models/artefacts/target_vars.pickle"`
- **Variáveis numéricas**: colunas com dados numéricos
  - Caminho: `"/models/artefacts/num_vars.pickle"`
- **Variáveis categóricas**: colunas com dados categóricos
  - Caminho: `"/models/artefacts/cat_vars.pickle"`

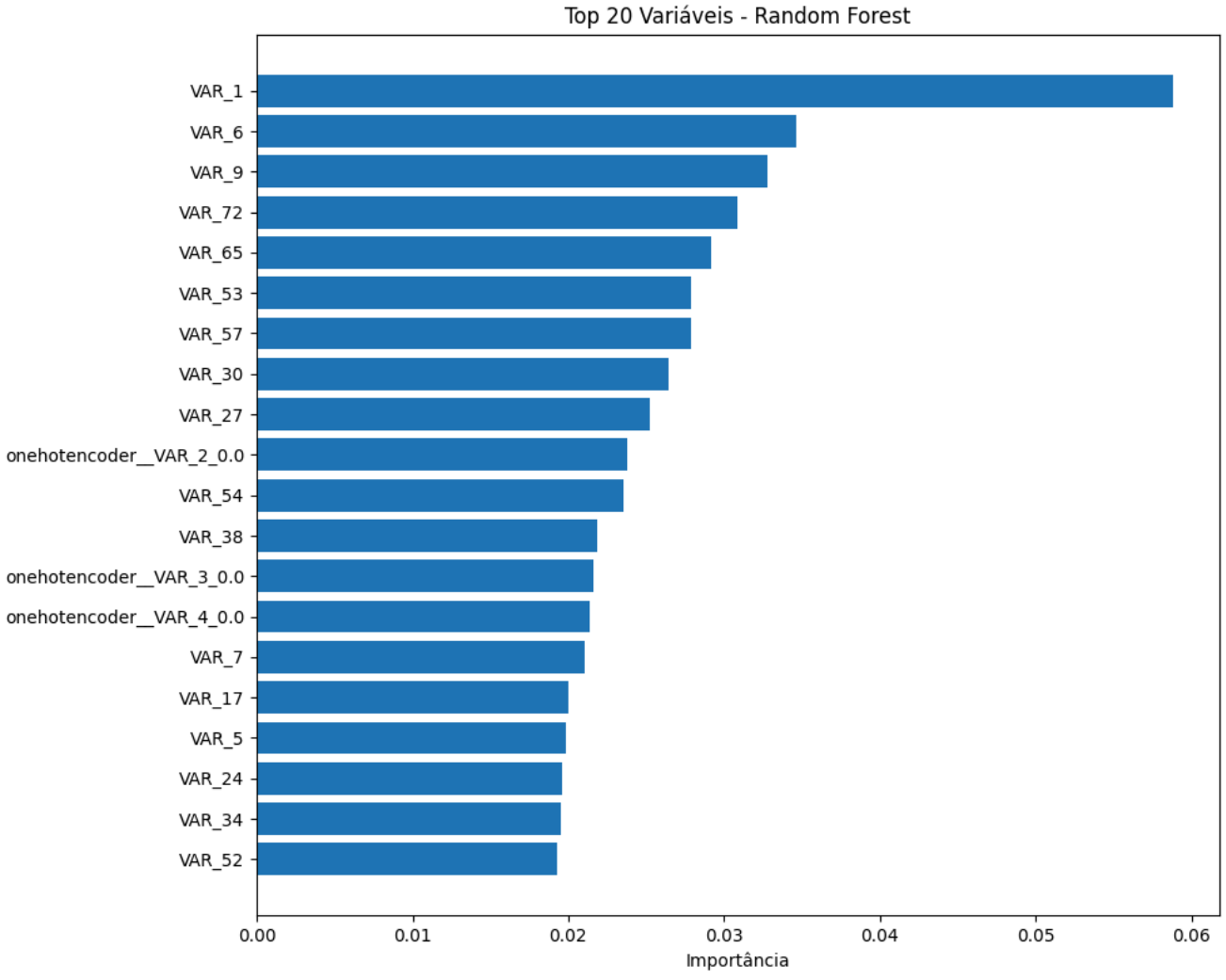
O pipeline completo de tratamento pode ser executado por meio da função `tratamento_completo`, localizada no arquivo `/src/features_processing.py`. Essa função é responsável por carregar todos os artefatos gerados e aplicar o tratamento às variáveis, retornando um DataFrame limpo e pronto para as etapas seguintes da modelagem.

---

3. Seleção de variáveis

Para selecionar as variáveis utilizamos o algoritmo **Random Forest**, que estima a importância de cada feature com base na redução média da impureza (Gini) durante o treinamento, disponível no atributo `feature_importances_`. Definimos um threshold mínimo de importância em **0,01 (1%)**, mantendo apenas as variáveis que excedem esse limite.

Esse procedimento reduz significativamente a dimensionalidade dos dados e minimiza o risco de *overfitting*, conservando apenas as variáveis que contribuem de forma relevante para a performance do modelo.



Ao final, foram selecionadas 42 variáveis para o treinamento do modelo de scoring. A lista dessas variáveis selecionadas foi salva em:

```
"/models/artefacts/selected_vars.pickle"
```

---

4. Treinamento do modelo

Nesta etapa utilizaremos um modelo de **Regressão Logística** como baseline, comparando-o com três algoritmos mais avançados: **XGBoost**, **LightGBM** e **Random Forest**. A seleção final levará em consideração o trade-off entre a alta interpretabilidade oferecida pela regressão logística, um modelo estatístico tradicional, e o melhor desempenho esperado dos modelos mais complexos, porém menos explicáveis em relação à influência individual das variáveis no resultado final.

Para o treinamento dos modelos foi utilizado cross-validation na base de treino e posteriormente um treinamento completo na base de treino e comparação das métricas com a base de testes, os resultados podem ser observados abaixo:

4.1 Resultado cross-validation - Treino

Modelo	KS Médio	KS Desvio	AUC Médio	AUC Desvio	Gini Médio	Gini Desvio
LightGBM	45.56	2.37	0.7991	0.0128	59.81	2.55
XGBoost	43.58	2.58	0.7838	0.0129	56.77	2.58
Logistic Regression	42.76	1.41	0.7822	0.0106	56.44	2.11
Random Forest	42.40	0.49	0.7796	0.0063	55.92	1.26

4.2 Resultado final - Teste

Modelo	KS	AUC	Gini
LightGBM	30.95	0.7024	40.48
Random Forest	29.08	0.6981	39.62

Modelo	KS	AUC	Gini
XGBoost	28.54	0.6913	38.26
Logistic Regression	28.71	0.6900	37.93

4.3 Modelo selecionado

Com base nos resultados obtidos, o modelo selecionado será o **LightGBM**. Apesar de possuir uma interpretabilidade inferior ao nosso modelo baseline (Regressão Logística), o LightGBM apresentou métricas substancialmente superiores em relação aos demais algoritmos avaliados, com um KS médio em cross-validation de **45,56**, comparado a **42,76** do baseline, e um KS em teste de **30,95**, superior aos **28,71** da regressão logística. Isso indica que o modelo escolhido possui maior equilíbrio entre o desempenho em treino e teste, além de melhor capacidade de generalização para dados não vistos.

5. Tunning de hiperparâmetros

Para otimizar os hiperparâmetros do modelo, foi utilizado a biblioteca **Optuna**, que tem integração de forma nativa com o **LightGBM**. Esse método realiza uma busca automatizada pelos melhores valores de parâmetros. Após o processo foi obtido uma leve melhora de desempenho tanto em treino quanto em teste.

Métrica	Treino	Teste
Ks	52,92	31,63
Auc	0,84	0,72
Gini	69,06	43,53

A lista com os melhores parâmetros foi armazenada em:

```
"/models/trained_model/final_params.pickle"
```

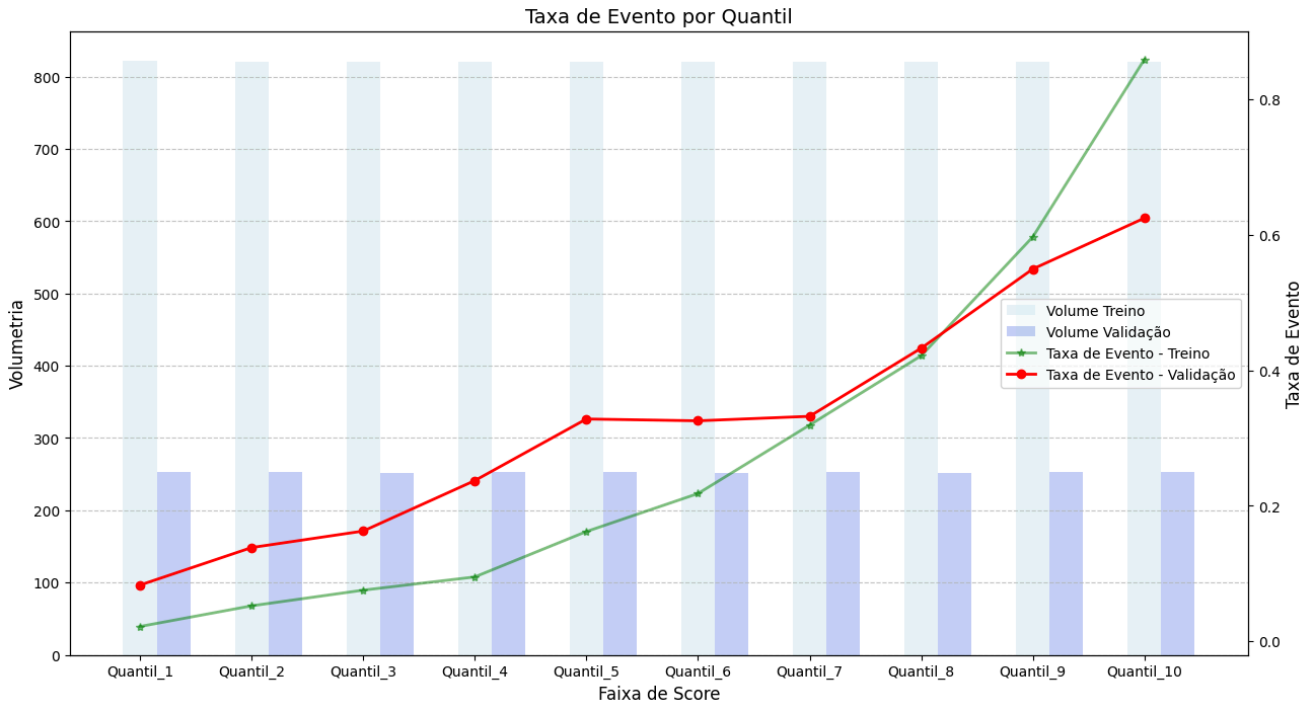
O modelo treinado final pode ser encontrado em:

```
"/models/trained_model/final_model.pickle"
```

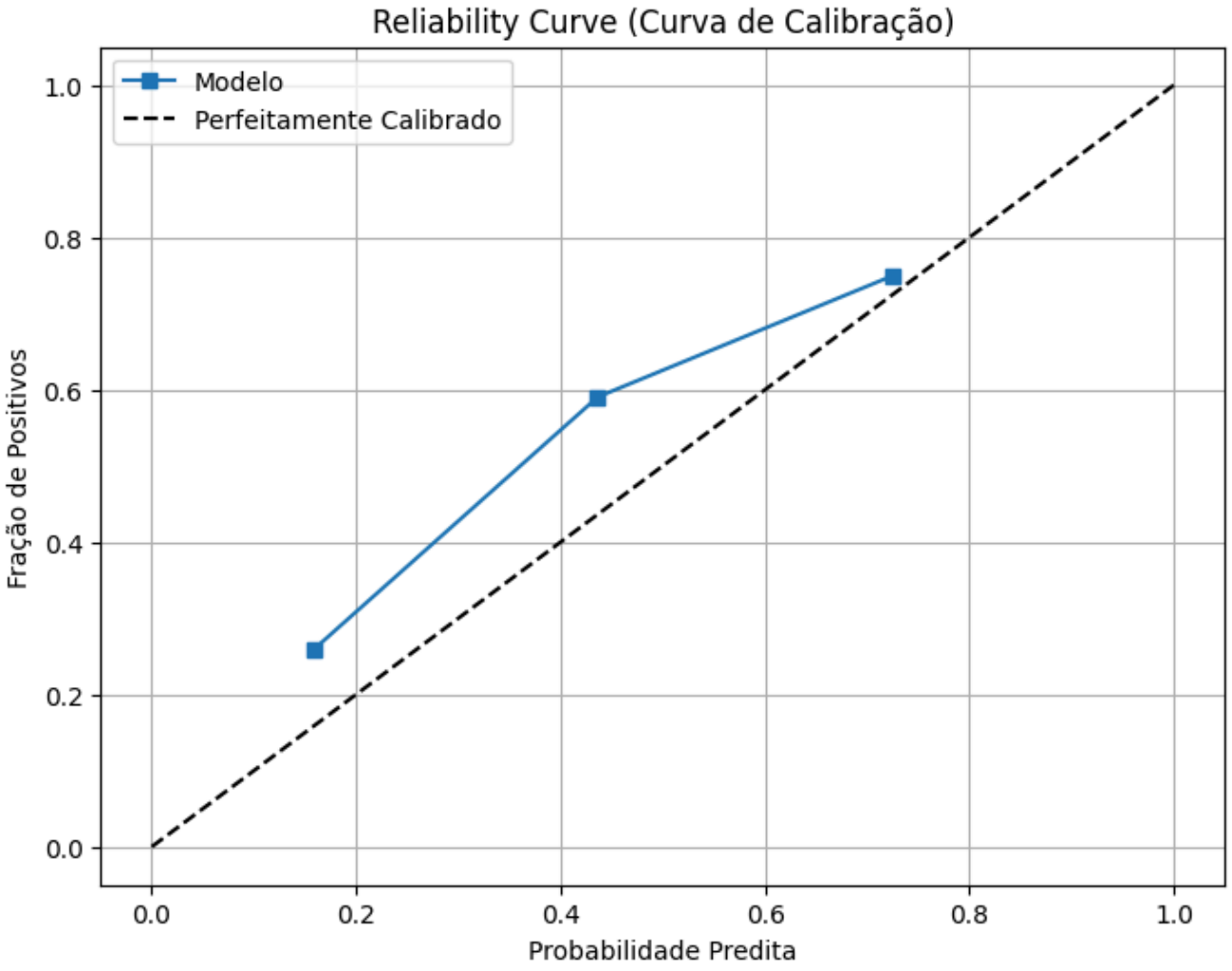
---

6. Avaliação Final

6.1. Ordenação dos scores



6.2 Calibração da curva de probabilidade

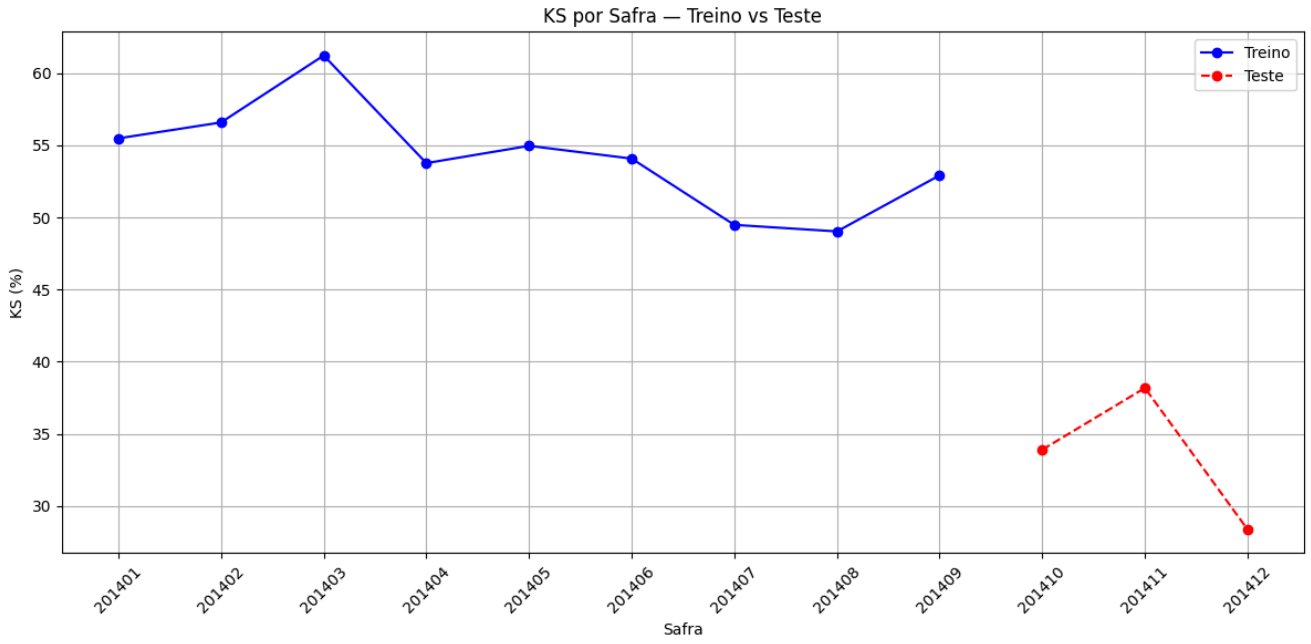


Brier Score: 0,2043
Expected Calibration Error (ECE): 0,1103

Apesar do modelo apresentar bom poder discriminativo — como visto pela taxa de eventos crescente nos quantis de score —, os indicadores de calibração (Brier Score = 0,2043 e ECE = 0,1103), junto à curva de confiabilidade, indicam que as probabilidades preditas estão sistematicamente subestimadas, especialmente nas faixas inferiores.

Diante disso, recomenda-se aplicar técnicas de calibração como **Platt Scaling** ou **Isotonic Regression** para alinhar melhor as previsões com a taxa real de eventos, ou ainda revisar a seleção de variáveis caso a calibração isolada não seja suficiente.

6.3. Estabilidade do KS ao longo das safras



A análise da curva de KS por safra evidencia uma **tendência de queda ao longo do tempo**. Esse comportamento sugere uma possível **instabilidade temporal do modelo**, que pode estar sofrendo com **overfitting** nos dados de treino ou refletindo uma **mudança significativa na distribuição da população** ao longo das safras.

Ao aplicar o modelo na base de teste, observamos um KS ainda **razoável (31,63%)**, o que indica alguma capacidade discriminativa. No entanto, a **forte queda em relação ao período de treino** — onde os valores oscilavam entre 49% e 61% — revela uma **perda considerável de performance e estabilidade**, reforçando a necessidade de investigar possíveis **drifts de dados**, revisar variáveis ou considerar técnicas de regularização e reamostragem para maior robustez temporal.

6.4. Estabilidade das variáveis

Ao comparar a estabilidade das variáveis entre o período de treino e o de teste, observou-se que algumas variáveis apresentaram **mudanças significativas na distribuição**. Especificamente, as variáveis `VAR_53` e `VAR_54` exibiram valores de PSI superiores a 0,6, o que indica uma **mudança severa** e potencialmente prejudicial para o desempenho do modelo — sendo recomendável **removê-las do treinamento**.

Além disso, as variáveis `VAR_30` e `VAR_1` apresentaram **mudanças moderadas** (PSI entre 0,1 e 0,25). Essas variáveis não precisam ser removidas de imediato, mas devem ser **monitoradas de perto**, pois podem indicar instabilidade futura.

As demais variáveis não apresentaram mudanças significativas em suas distribuições entre o período de treino e teste.

Variável	PSI	Mudança
VAR_53	0.983515	Mudança severa
VAR_54	0.623935	Mudança severa
VAR_30	0.209654	Mudança moderada
VAR_1	0.107737	Mudança moderada

7. Conclusão

O projeto alcançou seu objetivo principal ao construir um modelo preditivo eficaz para *Credit Scoring*, com destaque para o desempenho do algoritmo LightGBM, que obteve as melhores métricas entre os modelos testados. Apesar de uma leve queda de performance na base de teste (KS de 31,63), o modelo demonstrou bom poder discriminativo e consistência na ordenação dos scores.

Entretanto, a análise de calibração revelou que as probabilidades previstas estão subestimadas nas faixas inferiores, indicando a necessidade de aplicação de técnicas como Platt Scaling ou Isotonic Regression. Além disso, a avaliação da estabilidade das variáveis apontou mudanças severas em duas features (VAR*53 e VAR_54), sendo recomendada sua exclusão em versões futuras do modelo. A tendência de queda do KS ao longo das safras também sugere possível _data drift* ou sobreajuste, reforçando a importância de monitoramento contínuo e validações temporais regulares.

Como próximos passos, sugere-se:

- Aplicar técnicas de calibração probabilística;
- Reavaliar as variáveis instáveis e explorar novas combinações de features;
- Avaliar abordagens de reamostragem e regularização para ganho de robustez;
- Incluir validação contínua em ambiente de produção para detecção precoce de *concept drift*.

O pipeline completo foi implementado de forma modular e reprodutível, com artefatos salvos para reutilização futura. O código está estruturado para facilitar a

manutenção e novas iterações, cumprindo os critérios de clareza, documentação e boas práticas de programação.