# Ensemble de Algoritmos – LightGBM

ROCKETSEAT

Plataforma completa de aprendizado contínuo em programação.

#BoostingPeople

rocketseat.com.br

# Ensemble de Algoritmos

## LightGBM

O objetivo deste módulo é apresentar o ensemble LightGBM que pertence a classe de ensembles Boosting e trabalharemos num projeto para uma empresa do segmento de importação que deseja fazer boas escolhas de compras, prevendo a qualidade de vinhos que deseja importar. Neste projeto faremos o processo completo desde o EDA até a visualização dos resultados, incluindo métricas de validação e importância das variáveis.

# Agenda

- O que é LightGBM
- Etapas do LightGBM
- Desafios e Limitações do LightGBM
- Principais Hiperparâmetros
- Diferenças entre Catboost e LightGBM
- Otimização Bayesiana
- Projeto – LightGBM

# O que é LightGBM

LightGBM é um algoritmo de aprendizado de máquina baseado em árvores de decisão que também utiliza a técnica de boosting por gradiente, assim como Catboost ou XGBoost, mas que é especializado em grandes volumes de dados e cenários de alta dimensionalidade, otimizado para velocidade e eficiência.

O LightGBM foi desenvolvido pela Microsoft, mais especificamente pela equipe da Microsoft Research, mantido como um projeto open-source, disponível no GitHub desde 2016.

# O que é LightGBM

## Principais características:

Crescimento de Árvore baseado em Folhas (Leaf-wise): Ao invés do método tradicional baseado em níveis (level-wise), essa abordagem expande a folha que maximiza o ganho de informação, resultando em um treinamento mais eficiente e em modelos capazes de capturar padrões complexos nos dados.

Suporte a Histogramas: O LightGBM utiliza uma técnica de discretização baseada em histogramas, onde os valores contínuos das features são agrupados em bins (faixas). Isto reduz significativamente o custo computacional durante o treinamento, o que acelera a performance, especialmente em datasets grandes e com alta dimensionalidade.

# O que é LightGBM

## Principais características:

Paralelismo e Suporte a GPU): O LightGBM suporta paralelismo integrado, permitindo o uso eficiente de múltiplos núcleos de CPU.  Além disso, ele oferece treinamento acelerado por GPU, o que é particularmente útil para datasets grandes e de alta complexidade

Regularização e Configuração Avançada de Hiperparâmetros: O LightGBM possui uma robusta configuração de regularização e ajuste de hiperparâmetros, como num_leaves, min_data_in_leaf, feature_fraction e bagging_fraction. Essas opções ajudam a evitar overfitting e permitem ajustes precisos, garantindo o equilíbrio entre performance e generalização

# Etapas do LightGBM

## Visão Geral

- Inicializa com uma previsão base
- Calcula os resíduos (erros)
- Cria uma árvore de decisão para ajustar os resíduos
- Atualiza as previsões com base na nova árvore
- Repete o processo até atingir o critério de parada
- Aplica regularização e validação para garantir generalização
- Validação e ajuste
- Modelo final

# Etapas do LightGBM

## Inicializa com uma previsão base

O algoritmo começa com uma previsão inicial simples, geralmente a média dos valores do alvo (para regressão) ou a probabilidade inicial (para classificação binária).

Essa previsão é usada como base para calcular os resíduos iniciais, que representam o erro entre as previsões e os valores reais.

# Etapas do LightGBM

## Calcula os resíduos (erros)

Os resíduos, ou erros, são calculados como a diferença entre os valores reais e as previsões atuais.

Eles se tornam o novo alvo para as próximas árvores, pois o objetivo do algoritmo é ajustar essas diferenças.

# Etapas do LightGBM

## Cria uma árvore de decisão para ajustar os resíduos

### Método Leaf-wise

Nesta etapa, uma nova árvore de decisão é criada para ajustar os resíduos. O LightGBM utiliza o método de crescimento de árvore leaf-wise, que prioriza a expansão das folhas que oferecem o maior ganho de informação (redução no erro).

No método leaf-wise, a árvore cresce desbalanceada, expandindo seletivamente as folhas mais importantes.

# Etapas do LightGBM

## Cria uma árvore de decisão para ajustar os resíduos

### Divisão por Histogramas

Para aumentar a eficiência, o LightGBM converte os valores contínuos das variáveis em bins (grupos ou faixas), construindo um histograma de frequência.

Em vez de avaliar todas as possíveis divisões, o algoritmo analisa as divisões com base nesses bins, reduzindo significativamente o custo computacional.

# Etapas do LightGBM

## Cria uma árvore de decisão para ajustar os resíduos

### Cálculo dos Pesos (Gradientes)

Após construir a árvore, o algoritmo ajusta os pesos das folhas para minimizar os resíduos. Esses ajustes são feitos com base no gradiente da função de perda, que mede o impacto dos pesos no erro.

O gradiente é uma indicação da direção e magnitude da mudança necessária para melhorar a previsão.

# Etapas do LightGBM

## Atualiza as previsões com base na nova árvore

Com a nova árvore construída, as previsões do modelo são atualizadas somando-se as contribuições da nova árvore às previsões anteriores.

A atualização é controlada pelo hiperparâmetro *learning_rate*, que determina o impacto da nova árvore no modelo global.

# Etapas do LightGBM

## Repete o processo até atingir o critério de parada

O processo de calcular resíduos, construir árvores e atualizar previsões é repetido várias vezes. O número de iterações é controlado por um critério de parada, que pode ser:

- O número máximo de árvores (n_estimators).
- Um erro mínimo atingido.
- O uso de early stopping, que interrompe o treinamento se o desempenho no conjunto de validação não melhorar após um número pré-definido de iterações.

# Etapas do LightGBM

## Regularização

Durante o treinamento, o LightGBM aplica técnicas de regularização para controlar a complexidade do modelo e evitar overfitting. Algumas delas incluem:

- Limitação do número de folhas (num_leaves): Restringe a complexidade das árvores.
- Mínimo de amostras por folha (min_data_in_leaf): Evita folhas pequenas, que podem gerar overfitting.
- Regularização L1 e L2 (lambda_l1 e lambda_l2): Penaliza pesos excessivamente grandes ou complexos.

# Etapas do LightGBM

## Validação e Ajuste

O modelo é avaliado periodicamente em um conjunto de validação para monitorar seu desempenho e evitar overfitting.

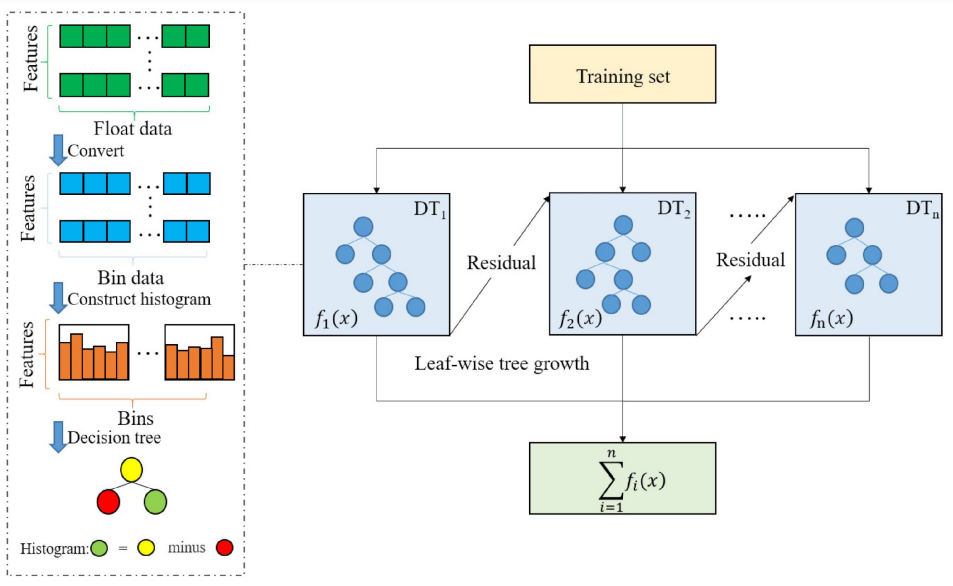
Se o desempenho no conjunto de validação não melhorar após um número determinado de iterações, o treinamento pode ser interrompido antecipadamente (early stopping).

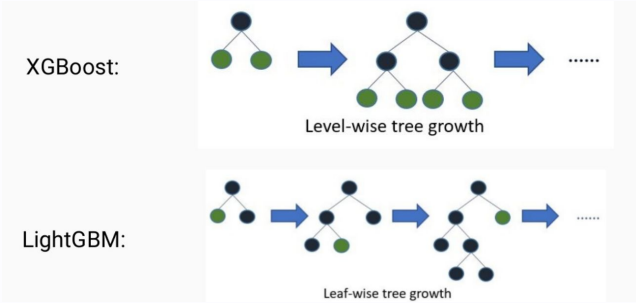
# Etapas do LightGBM

## Modelo Final

Ao final do processo, o modelo combina todas as árvores criadas em um ensemble. Ele utiliza essas árvores para prever novos dados, aplicando a soma ponderada das contribuições de cada uma.

# Etapas do LightGBM



# Desafios e Limitações do LightGBM

O LightGBM, apesar de eficiente e poderoso, apresenta alguns desafios que devem ser considerados. Ele é altamente sensível à configuração dos hiperparâmetros, o que significa que ajustes inadequados podem comprometer significativamente o desempenho do modelo. Além disso, o método de crescimento leaf-wise, embora eficiente, aumenta o risco de overfitting, especialmente em datasets pequenos ou com poucos exemplos por classe. O algoritmo também não lida nativamente bem com variáveis categóricas, exigindo pré-processamento ou transformação dos dados, o que pode adicionar complexidade ao fluxo de trabalho. Outra limitação é o consumo de memória, que pode ser elevado ao lidar com grandes volumes de dados, especialmente durante o uso de histogramas para discretização.

# Desafios e Limitações do LightGBM

Por fim, o LightGBM pode exigir ajustes adicionais para lidar com classes desbalanceadas, como a introdução de pesos ou a aplicação de técnicas de amostragem, tornando sua configuração mais desafiadora em cenários com desbalanceamento severo.

# Principais Hiperparâmetros

## n_estimators

- Define o número total de árvores no modelo.
- A principal função é controlar o tamanho do ensemble, impactando diretamente no desempenho do modelo e no tempo de treinamento.
- Valor inicial sugerido: 100
- Justificativa: Para a maioria dos problemas, 100 árvores são suficientes para alcançar uma boa performance inicial. Esse valor pode ser ajustado dependendo da taxa de aprendizado e do tamanho do dataset.

# Principais Hiperparâmetros

## learning_rate

- Controla o impacto de cada árvore no modelo final. Valores menores reduzem o ajuste excessivo, mas exigem mais iterações.
- A principal função é determinar a taxa com que o modelo converge para uma solução ideal.
- Valor inicial sugerido: 0.1
- Justificativa: Um valor de 0.1 equilibra a velocidade de aprendizado e a estabilidade do modelo. Para maior precisão, reduza para 0.01 ou 0.001, mas aumente o número de iterações.

# Principais Hiperparâmetros

## max_depth

- Limita a profundidade máxima das árvores, restringindo a complexidade.
- A principal função é controlar o overfitting em problemas de baixa dimensionalidade ou com poucos dados.
- Valor inicial sugerido: -1 (sem limite)
- Justificativa: O valor padrão permite que o modelo cresça livremente, mas em dados menores, limitar a profundidade pode ser útil para evitar overfitting.

# Principais Hiperparâmetros

## num_leaves

- Define o número máximo de folhas em cada árvore. Esse parâmetro controla a complexidade do modelo.
- A principal função é determinar a capacidade de aprendizado da árvore, influenciando diretamente o overfitting ou underfitting.
- Valor inicial sugerido: 31
- Justificativa: É um bom ponto de partida para problemas gerais. Aumentar esse valor permite que o modelo capture mais padrões complexos, mas valores muito altos podem levar ao overfitting.

# Principais Hiperparâmetros

## min_data_in_leaf

- Define o número mínimo de amostras necessárias em cada folha da árvore.
- A principal função é evitar a criação de folhas pequenas que podem causar overfitting.
- Valor inicial sugerido: 20
- Justificativa: Este valor reduz a divisão excessiva em folhas pequenas, melhorando a generalização, especialmente em datasets menores.

# Principais Hiperparâmetros

## feature_fraction

- Especifica a proporção de features usadas para treinar cada árvore.
- A principal função é introduzir variabilidade no treinamento, reduzindo o risco de overfitting.
- Valor inicial sugerido: 0.8
- Justificativa: Utilizar 80% das features em cada iteração é eficiente para a maioria dos problemas. Reduzir esse valor pode ajudar em datasets muito grandes.

# Principais Hiperparâmetros

## bagging_fraction

- Especifica a proporção de amostras usadas em cada iteração do treinamento.
- A principal função é introduzir diversidade nas árvores, reduzindo o risco de overfitting.
- Valor inicial sugerido: 0.8
- Justificativa: Amostrar 80% dos dados geralmente equilibra diversidade e desempenho. Pode ser ajustado para datasets muito grandes.

# Principais Hiperparâmetros

## lambda_l1

- Coeficiente de regularização L1 que adiciona penalização ao modelo para reduzir complexidade.
- A principal função é simplificar o modelo, eliminando pesos irrelevantes.
- Valor inicial sugerido: 0.0
- Justificativa: A regularização L1 pode ser ativada para lidar com problemas de overfitting, especialmente em datasets com muitas features.

# Principais Hiperparâmetros

## lambda_l2

- Coeficiente de regularização L2 que adiciona penalização aos pesos para reduzir sua magnitude.
- A principal função é estabilizar o modelo e prevenir overfitting.
- Valor inicial sugerido: 0.0
- Justificativa: Assim como a L1, a regularização L2 é útil em modelos complexos. Ativar esse parâmetro é uma boa prática ao lidar com overfitting.

# Principais Hiperparâmetros

## max_bin

- Define o número máximo de bins usados para discretizar as variáveis contínuas.
- A principal função é controlar a granularidade da representação dos dados, impactando o custo computacional.
- Valor inicial sugerido: 255
- Justificativa: O valor padrão é eficiente para a maioria dos problemas. Aumentar o número de bins pode melhorar a precisão, mas aumenta o uso de memória e tempo de treinamento.

# Diferenças entre Catboost e LightGBM

## Manipulação de Dados Categóricos

LightGBM:

- Requer pré-processamento manual para variáveis categóricas.
- Transforma variáveis categóricas em números inteiros ou utiliza técnicas como one-hot encoding.
- Embora tenha suporte limitado a variáveis categóricas, o desempenho não é otimizado diretamente para esses tipos de dados.

CatBoost:

- Oferece suporte nativo para dados categóricos.
- Usa técnicas avançadas como Encoding Baseado em Target (Target-Based Encoding) com regularização para lidar com viés e overfitting.
- É ideal para datasets com muitas variáveis categóricas.

# Diferenças entre Catboost e LightGBM

## Eficiência no Treinamento

### LightGBM:

- Muito rápido para datasets grandes e de alta dimensionalidade, graças ao uso de histogramas e crescimento leaf-wise.
- Geralmente mais rápido que o CatBoost, mas depende de mais pré-processamento dos dados.

### CatBoost:

- Mais lento que o LightGBM devido ao cálculo iterativo para evitar overfitting.
- Utiliza a estratégia de Ordenação por Conjuntos Aleatórios (Ordered Boosting), que divide os dados em diferentes subconjuntos para calcular os gradientes sem vazamento de informações, aumentando a robustez.

# Diferenças entre Catboost e LightGBM

## Prevenção de Overfitting

### LightGBM:

- Utiliza regularizações como L1, L2, min_data_in_leaf, e early stopping para evitar overfitting.
- O crescimento leaf-wise é eficiente, mas pode levar a overfitting em datasets pequenos.

### CatBoost:

- Desenvolvido com foco em overfitting minimizado. Sua abordagem de boosting ordenado é mais robusta e reduz o vazamento de informações nos gradientes.
- Mais confiável para dados com alto risco de overfitting, especialmente em datasets menores.

# Diferenças entre Catboost e LightGBM

## Sensibilidade a Classes Desbalanceadas

### LightGBM:

- Pode lidar com classes desbalanceadas ajustando os pesos das classes (class_weight) ou parâmetros como scale_pos_weight.
- O desempenho depende de ajustes manuais, o que pode demandar mais experimentação.

### CatBoost:

- Inclui suporte integrado para classes desbalanceadas sem a necessidade de muitos ajustes manuais.
- Lida melhor automaticamente com problemas de desbalanceamento por causa de sua otimização nativa.

# Diferenças entre Catboost e LightGBM

## Interpretação e Facilidade de Uso

### LightGBM:

- Oferece boa flexibilidade, mas exige maior esforço do usuário para pré-processamento, ajuste de hiperparâmetros e controle de overfitting.

### CatBoost:

- É mais user-friendly, com um pré-processamento mínimo.
- Configurações padrão são bem ajustadas, tornando-o uma escolha preferida para iniciantes ou quem deseja resultados rápidos sem muita experimentação.

# Diferenças entre Catboost e LightGBM

## Treinamento em GPU

### LightGBM:

- Suporta treinamento em GPU, mas apenas para algumas funções de perda, como classificação binária.
- A aceleração em GPU é limitada a configurações específicas e pode exigir otimizações adicionais.

### CatBoost:

- Tem suporte completo e otimizado para GPU, sendo mais eficiente que o LightGBM em problemas complexos ou em datasets grandes ao usar hardware acelerado.

# Diferenças entre Catboost e LightGBM

## Suporte a Dados Ordenados

LightGBM:

- Não tem estratégias específicas para dados ordenados; trata todas as amostras igualmente.

CatBoost:

- É particularmente eficiente com dados sequenciais ou temporais, graças ao seu boosting ordenado.

# Diferenças entre Catboost e LightGBM

## Desempenho Geral

### LightGBM:

- Geralmente se destaca em problemas de alta dimensionalidade e datasets muito grandes devido à sua velocidade e eficiência.

### CatBoost:

- É mais indicado para datasets menores, dados categóricos complexos e problemas onde o risco de overfitting é elevado.

# Diferenças entre Catboost e LightGBM

**Resumo: Qual escolher?**

**Use LightGBM se:**

- O dataset for grande e contínuo.
- Você tiver tempo para ajustar hiperparâmetros e pré-processar os dados.
- Desempenho e velocidade forem prioridades absolutas.

**Use CatBoost se:**

- O dataset contiver muitas variáveis categóricas.
- O dataset for pequeno ou moderado.
- Você desejar um algoritmo robusto com menos ajustes manuais.

# Otimização Bayesiana

A Otimização Bayesiana é uma técnica para encontrar o conjunto ideal de hiperparâmetros de um modelo com base na maximização ou minimização de uma função objetivo. Ela é especialmente útil quando:

- A função objetivo é cara de avaliar (como treinar um modelo complexo).
- O espaço de busca dos hiperparâmetros é contínuo e/ou discreto.
- Não há uma expressão explícita para derivar a melhor solução.

A ideia central é usar um modelo probabilístico (geralmente um Processo Gaussiano) para estimar a relação entre os hiperparâmetros e a métrica de avaliação, iterando para melhorar essa estimativa com base nos resultados obtidos.

# Otimização Bayesiana

## Etapas

- Definir a Função Objetivo
- Modelar a Função com um Processo Gaussiano
- Selecionar a Próxima Combinação de Hiperparâmetros
- Avaliar os Hiperparâmetros
- Atualizar o Modelo Probabilístico
- Repetir até Convergir

# Otimização Bayesiana

- Definir a Função Objetivo

A métrica que você quer otimizar (exemplo: AUC, Log Loss, Accuracy).
Os hiperparâmetros do modelo são os inputs dessa função.

- Modelar a Função com um Processo Gaussiano

O modelo cria uma distribuição de probabilidade sobre a função objetivo (representando a incerteza sobre o espaço de busca).

# Otimização Bayesiana

- Selecionar a Próxima Combinação de Hiperparâmetros

Um Critério de Aquisição é usado para determinar onde buscar a próxima avaliação.Critérios comuns:

- Expected Improvement (EI): Seleciona o ponto que maximiza a melhoria esperada.
- Probability of Improvement (PI): Maximiza a probabilidade de obter uma métrica melhor.
- Upper Confidence Bound (UCB): Faz um trade-off entre exploitation e exploration.

- Avaliar os Hiperparâmetros

O modelo é treinado com a nova combinação de hiperparâmetros, e o resultado da métrica é registrado

# Otimização Bayesiana

- Atualizar o Modelo Probabilístico

A função objetivo é ajustada com base nos novos resultados.

- Repetir até Convergir

O processo continua até atingir o número máximo de iterações ou o critério de parada.

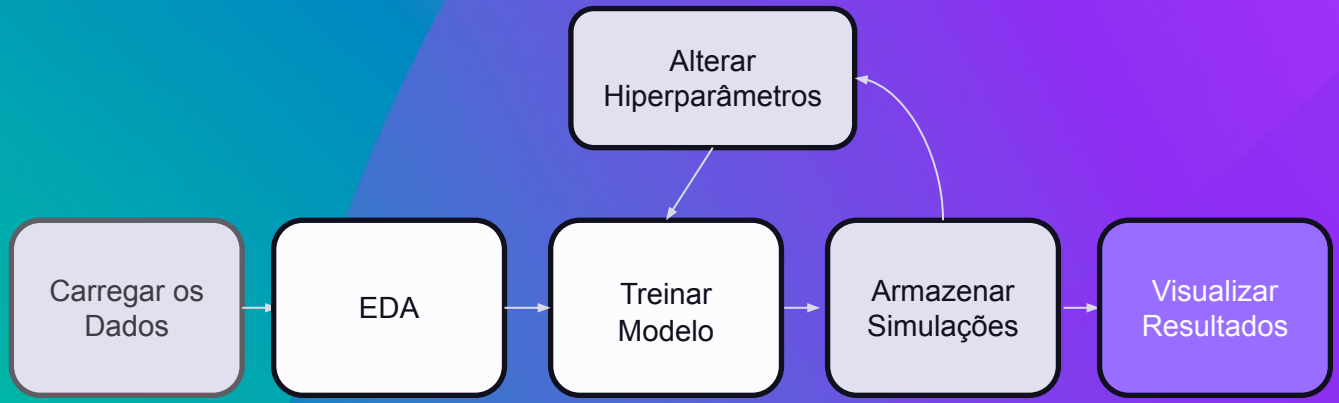
# Projeto – LightGBM

Uma empresa do segmento de importação de bebidas precisa planejar as próximas aquisições de vinhos vindos do exterior e, para isso, deseja estimar a qualidade do vinho com base nas características de cada produto.

Pra isso, o time de ciência de dados irá construir um modelo que contará com um dataset com informações dos vinhos, tais como total de ácidos cítricos, densidade, pH, dentre outras características técnicas.

E dada a complexidade do desafio e o tamanho do dataset (mais de 20.000 linhas), iremos usar o ensemble LightGBM para fazer a predição da qualidade, bem como entender quais variáveis mais contribuem para esta predição.

# Estrutura do Projeto



# Code Time ...



rocketseat.com.br