

FIAP

MBA

ADELAIDE ALVES DE OLIVEIRA

PROFESSORA



profadelaide.alves@fiap.com.br

Formação Acadêmica

- Bacharel em Estatística – UNICAMP
- Mestre em Ciências – FSP/USP

Atividades Profissionais

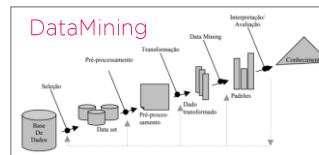
- Diretora Técnica Estatística da empresa **SD&W** - www.sdw.com.br
- Professora de Fundamentos Estatísticos, DataMining, Análise Preditiva e Machine Learning na FIAP dos cursos MBA: Big Data, Data Science, Business Intelligence & Analytics, Digital Data Marketing, IA & ML e Engenharia de Dados e nos Shift: People Analytics e Python Journey

MÉTODOS DE ENSEMBLE LEARNING

MÉTODOS DE ENSEMBLE LEARNING

INTRODUÇÃO

Para
resolver
um
problema



(*) No caso de problema supervisionado com variável target categórica.

Indicador de Performance
Acurácia(*), por exemplo:

Modelo 1 Árvore de Decisão 75%

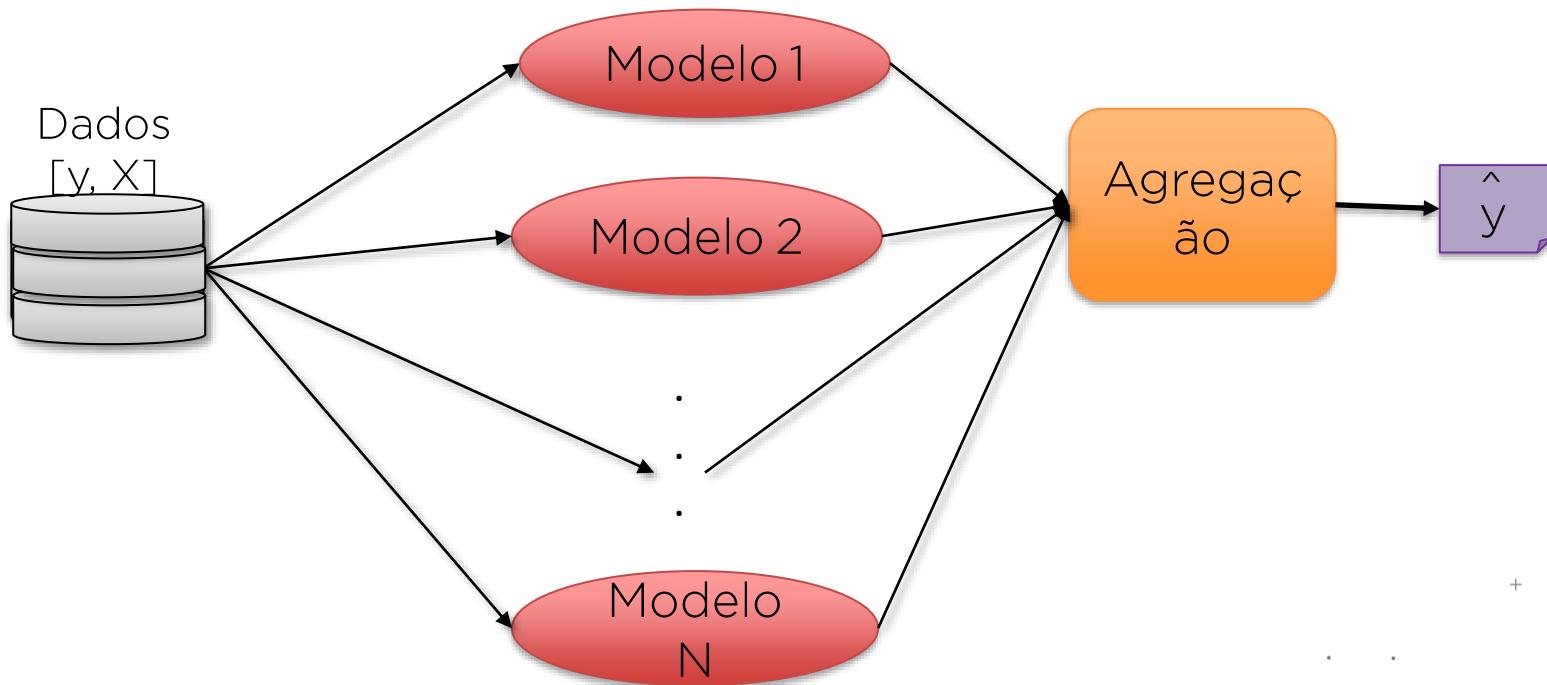
Modelo 2 KNN 73%

Modelo 3 Regressão Logística 72%

Ensemble Methods

Modelo combinado
Com acurácia de 78%, por exemplo

- MÉTODOS DE ENSEMBLE LEARNING
- INTRODUÇÃO



MÉTODOS DE ENSEMBLE LEARNING

INTRODUÇÃO

Junta múltiplos modelos mais “fracos”, com o objetivo de buscar a diminuir a suscetibilidade geral deles quanto o viés e a variância, tornando-os mais robustos.

Os métodos de Ensemble devem levar em conta a maneira com a qual eles agrupam os modelos, associando os algoritmos de forma a minimizar suas desvantagens individuais no modelo final.

Existem variados métodos de ensemble como: Bagging, Boosting e Stacking,

Stacking é de 1992, Bagging e Boosting são de 1996.

MÉTODOS DE ENSEMBLE LEARNING

INTRODUÇÃO: VISÃO GERAL

Características Principais:

- **Bagging** ➔ geralmente usa mesmo tipo de modelos individuais, cada um de forma independente em relação ao outro, de forma paralela. O algoritmo final é então feito a partir de algum tipo de resultado médio do que foi obtido a partir dos modelos bases.
- **Boosting** ➔ geralmente usa mesmo tipo de modelos individuais, que são aplicados de forma sequencial (o posterior depende do antecessor) e depois combinados no modelo final.
- **Stacking** ➔ geralmente usa tipos diferentes de modelos individuais, treinando-os em paralelo. É então aplicado um modelo no output dos weak learners (podendo incluir ou não as features utilizadas para treiná-los)

MÉTODOS DE ENSEMBLE LEARNING

INTRODUÇÃO

Uma técnica que precisamos entender antes:

Bootstrapping. Essa técnica consiste em subdividir nossa base treino, em amostras aleatória e **com reposição**.



População

Assumindo que
a amostra é
representativa
da população



Podemos “reconstruir” a
população criando várias
amostra.

Esperamos que essa
“reconstrução” se aproxime
da população inteira.

podemos criar novas
amostras e, dessa forma,
estudar as propriedades
das amostras em
relação à população.

MÉTODOS DE ENSEMBLE LEARNING

BAGGING

MÉTODOS DE ENSEMBLE LEARNING

BAGGING - Bootstrap Aggregating

- Como funciona:

- A diversidade é obtida com o uso de diferentes subconjuntos de dados aleatoriamente criados com reposição;
- Treina **modelos individuais** usando uma amostra aleatória para cada;
- Cada subconjunto é usado para treinar um classificador do mesmo tipo;
- Agrega os **modelos individuais depois de treinados** com suas respectivas amostras;
- No caso de problemas de regressão usa a média e no caso de classificação a moda;
- As saídas dos classificadores são combinadas por meio do voto majoritário com base em suas decisões.
- No caso de classificação, para uma dada instância, a classe que obtiver o maior número de votos será a resposta.

MÉTODOS DE ENSEMBLE LEARNING

BAGGING - Bootstrap Aggregating

- Vantagem:

- ajuda reduzir a variância (amostragem aleatória);
- pode reduzir o viés(pois estamos usando média e moda para combinar os modelos);
- fornece estabilidade e robustez (alto número de estimadores usados).

MÉTODOS DE ENSEMBLE LEARNING

BAGGING - Bootstrap Aggregating

- Desvantagem:

- tem um custo computacional alto (usa muito espaço e tempo - cada nova iteração é criada uma amostra diferente). A maioria dos sistemas combinados fazem uso de bootstrap ou de cross validation. E costumam envolver mais de uma fase (ou iterações);
- A técnica só funciona se o modelo base já tem uma boa performance. Usar o bagging em um modelo base ruim pode fazer com que o modelo final fique ainda pior. Como os modelos individuais usam o mesmo algoritmo, o bagging pode não reconhecer alguns padrões;
- Modelos combinados são mais difíceis de analisar.

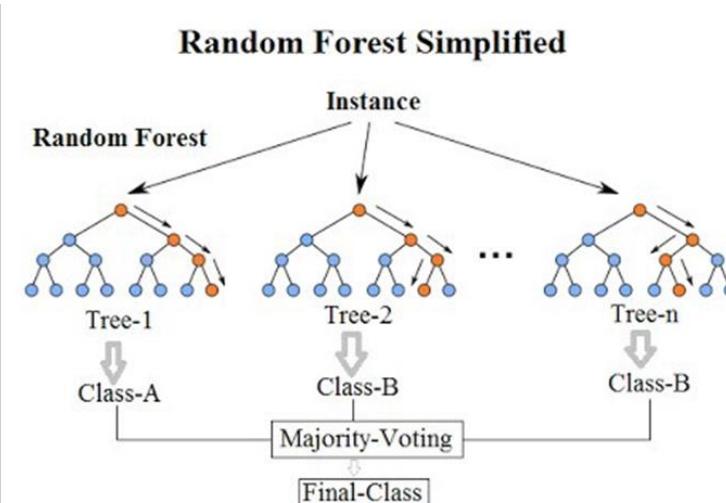
- MÉTODOS DE ENSEMBLE LEARNING
- **BAGGING – RANDOM FOREST**

Random Forest é uma técnica de bagging.

Usa **diversas árvores de decisão como modelos individuais**, além de fazer uma seleção aleatória de casos e de variáveis.

As árvores são extremamente interpretáveis, entretanto costumam ter um poder preditivo muito baixo quando comparados aos demais estimadores.

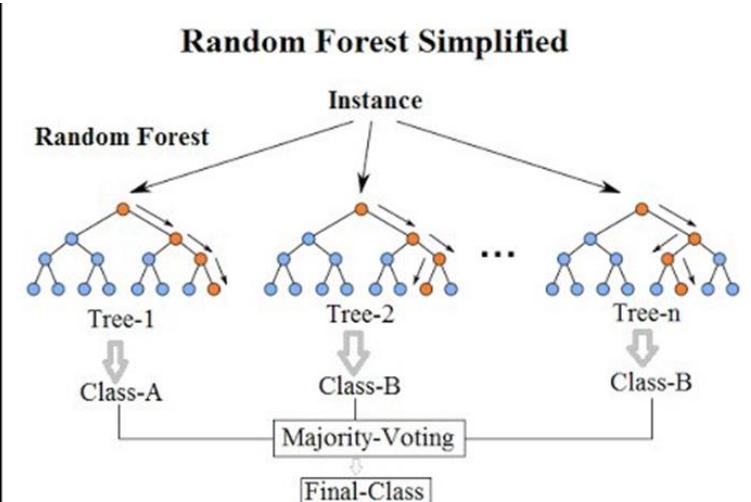
Uma forma de contornar isso é através da combinação da predição fornecida por diversas árvores para se fazer



<https://www.kdnuggets.com/wp-content/uploads/rand-forest-1.jpg>

- MÉTODOS DE ENSEMBLE LEARNING
- **BAGGING - RANDOM FOREST**

- Usado para a construção de Comitês com diferentes árvores de decisão. Variando a quantidade de dados e características. Usando árvores de decisão com diferentes inicializações. Cada árvore tenta estimar uma classificação e isso é chamado como “voto”. Idealmente, consideramos cada voto de cada árvore e escolhemos a classificação mais votada (estatística: Moda). No caso de problemas de regressão funciona similarmente, cada árvore tenta estimar a variável target e depois é considerada a média dos



<https://www.kdnuggets.com/wp-content/uploads/rand-forest-1.jpg>

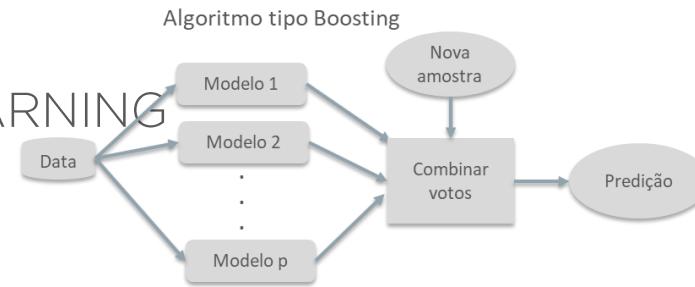
MÉTODOS DE ENSEMBLE LEARNING

BOOSTING

MÉTODOS DE ENSEMBLE LEARNING

BOOSTING

- Como funciona:
 - têm como base treinar **vários modelos mais simples** com a finalidade de produzir um modelo final mais robusto.
 - Os modelos não são mais treinados de forma independente entre si, mas de maneira sequencial, a partir de uma ajuste dos modelos treinados previamente.
 - Para maximizar o desempenho do preditor final, o Boosting treina iterativamente novos modelos com um enfoque nas observações que os modelos anteriores erraram mais, tornando a predição mais resistente ao viés.
 - Em seguida, atualiza-se o modelo para priorizar as previsões com maior erro nas observações da base de teste.
 - O modo como ocorre esse treinamento e essa atualização é onde diferem os diversos algoritmos de Boosting.
 - Como o principal foco dos métodos de Boosting é de reduzir o viés dos weak learners (modelos



MÉTODOS DE ENSEMBLE LEARNING

BOOSTING

- Como funciona:

- A partir de uma base de dados com N observações;
- Seleção de uma amostra aleatória com p observações;
- A partir das p observações sorteadas (mesma probabilidade) o primeiro modelo (M1) pode ser treinado.
- Com base nesse primeiro modelo (M1) treinado toda a base de dados é classificada. Algumas observações serão classificadas corretamente e outras não - pois se trata de um modelo fraco;
- As observações classificadas incorretas recebem um peso maior para a seleção de uma nova amostra (M2). Assim, as observações classificadas incorretamente terão uma probabilidade maior de seleção do que as demais. Com base nessa amostra o modelo é treinado. A base de dados pode ser classificada com esse modelo.
- Este processo é repetido para k modelos. No final, esses modelos são agregados e, um modelo final é construído - um modelo de melhor desempenho - uma vez que o erro de

MÉTODOS DE ENSEMBLE LEARNING

BAGGING X BOOSTING

- constrói os modelos separadamente,
- as saídas dos modelos são igualmente importantes.
- constrói modelos de forma iterativa: cada novo modelo é influenciado pela performance do anterior.
- as saídas dos modelos são ponderadas.
- Boosting funciona melhor do que bagging quando os algoritmos de aprendizagem são estáveis (como os modelos lineares).
- Boosting usualmente produz melhores resultados do que o bagging.

MÉTODOS DE ENSEMBLE LEARNING

BOOSTING

ADABOOST

MÉTODOS DE ENSEMBLE LEARNING

BOOSTING – ADABOOST – “Adaptive

Boosting”

E um algoritmo que consiste em combinar de forma sequencial vários modelos mais fracos (*weak learner*). Sendo assim, **cada modelo subsequente leva em consideração as previsões do anterior, para formar um preditor mais forte.**

- O diferencial desse algoritmo é que gera um conjunto de hipóteses que as combina por meio da votação ponderada. As previsões mais difíceis (aqueles em que o “modelo simples” da iteração atual não previu bem) recebem um peso maior no preditor seguinte, buscando assim uma maior otimização do algoritmo final.
- Foi projetado especificamente para problemas de classificação e pode ser aplicado combinado com qualquer algoritmo de aprendizagem de classificação.
- AdaBoost: Foi criado por Freund and Schapire em 1997.
- Foram criados os AdaBoost.M1 (manipulação de múltiplas classes) e AdaBoost.R

MÉTODOS DE ENSEMBLE LEARNING

BOOSTING – ADABOOST

- Começa por treinar árvores de decisão.
- Cada observação durante este procedimento tem um peso igual atribuído a ele.
- Depois de analisar a primeira árvore os pesos de cada observação que foi considerada errada na classificação são “aumentados”.
- Por outro lado, diminuem os pesos para aquelas em que a foi bem classificada.
- Assim a segunda árvore cresce sobre os dados ponderados, melhorando as previsões da primeira árvore.
- O novo modelo é a agregação da primeira e a segunda árvore. Em seguida, calcula-se os erros de classificação a partir do novo modelo “agregado” e uma terceira árvore é treinada para prever os resíduos alterados. É repetido este procedimento para uma determinada quantidade de iterações. As próximas árvores serão determinadas com base em observações onde as árvores anteriores falharam ou mostraram erros.

MÉTODOS DE ENSEMBLE LEARNING

BOOSTING – ADABOOST

- 1.^º peso para cada observação que representa o grau de foco que a iteração em questão deve dar para aquela observação. A cada novo modelo treinado, esses pesos das amostras são atualizados.

- O algoritmo inicia atribuindo pesos iguais a todas as instâncias nos dados de treinamento. Em seguida, executa o algoritmo de aprendizagem para gerar um classificador para estes dados e redistribui os pesos para cada instância de acordo com a saída deste classificador.
- Os pesos de instâncias corretamente classificadas são diminuídos e os pesos dos casos mal classificados são aumentados. Em todas as iterações subsequentes, um classificador é construído para os dados reponderados, concentrando-se, portanto, em classificar corretamente as instâncias erroneamente classificadas pelos classificadores anteriores.



- MÉTODOS DE ENSEMBLE LEARNING
- **BOOSTING – ADABOOST**

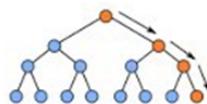
- 2.º peso é um peso para cada modelo que representa seu poder de decisão quando os modelos combinados. Eles são atribuídos apenas uma vez para cada estimador.



MÉTODOS DE ENSEMBLE LEARNING

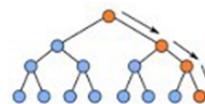
BOOSTING – ADABOOST

Amostras
Originais com
pesos iguais



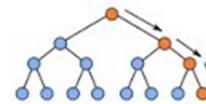
Peso 1

Amostras
com pesos
alterados



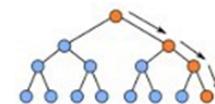
Peso 2

Amostras
com pesos
alterados



Peso 3

Amostras
com pesos
alterados



Peso n

Pesos Alterados
proporcionalmente ao
erro do modelo e
normalização dos pesos
de todas as instâncias

Em Boosting as
estimativas dos modelos
são ponderadas
proporcionalmente pelo
erro do modelo

Votação

Resposta

MÉTODOS DE ENSEMBLE LEARNING

BOOSTING – ADABOOST

- Ele pode ser usado em conjunto com outros tipos de algoritmos de aprendizagem para melhorar seu desempenho.
- AdaBoost é sensível a dados ruidosos e outliers.
- Em alguns problemas, pode ser menos suscetível ao problema de overfitting do que outros algoritmos de aprendizagem.

MÉTODOS DE ENSEMBLE LEARNING

BOOSTING – ADABOOST

- Diferença entre os métodos Adaboost e Bagging (incluindo Random Forests):

- é que, ao final do processo, quando todos os classificadores construídos durante as iterações forem solicitados a votar no alvo de uma nova observação, **haverá árvores com um voto mais pesado do que outros.** Essas são as árvores que tiveram o melhor desempenho durante todas as iterações (portanto, elas mostraram muito poucas classificações incorretas).

- ➔ O que acontece é que algumas árvores impactarão no resultado mais do que outras, isto foi mensurado/estimado ao longo do procedimento de iteração.

- outra característica importante do Adaboost é sua capacidade de

MÉTODOS DE ENSEMBLE LEARNING

BOOSTING

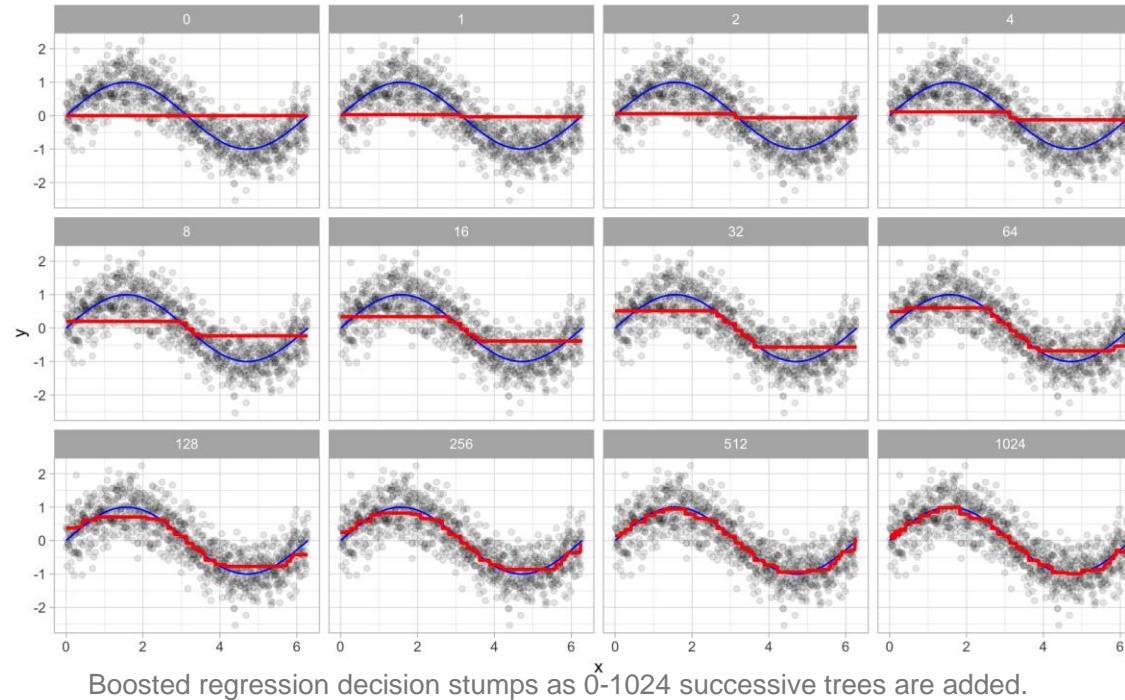
GRADIENT BOOSTING

- MÉTODOS DE ENSEMBLE LEARNING
- BOOSTING – GRADIENT BOOSTING

- O Gradient Boosting é um outro tipo de algoritmo de Boosting.
- Técnica de aprendizado de máquina para problemas de regressão e classificação, que produz um modelo de previsão na forma de um conjunto de modelos simples (baixa previsão), geralmente árvores de decisão.
- Ele constrói o modelo em etapas. O objetivo do algoritmo é criar uma corrente de modelos fracos, onde cada um tem como objetivo minimizar o erro do modelo anterior, por meio de uma função de perda.
- os erros são minimizado pelo algoritmo de gradiente descendente(*gradient descending*).
- Aos ajustes de cada modelo fraco é multiplicado um valor chamado de taxa de aprendizagem. Esse valor, tem como objetivo determinar o impacto de cada árvore no modelo final. Quanto menor o valor, menor a contribuição

MÉTODOS DE ENSEMBLE LEARNING

BOOSTING – GRADIENT BOOSTING



MÉTODOS DE ENSEMBLE LEARNING

BOOSTING – GRADIENT BOOSTING

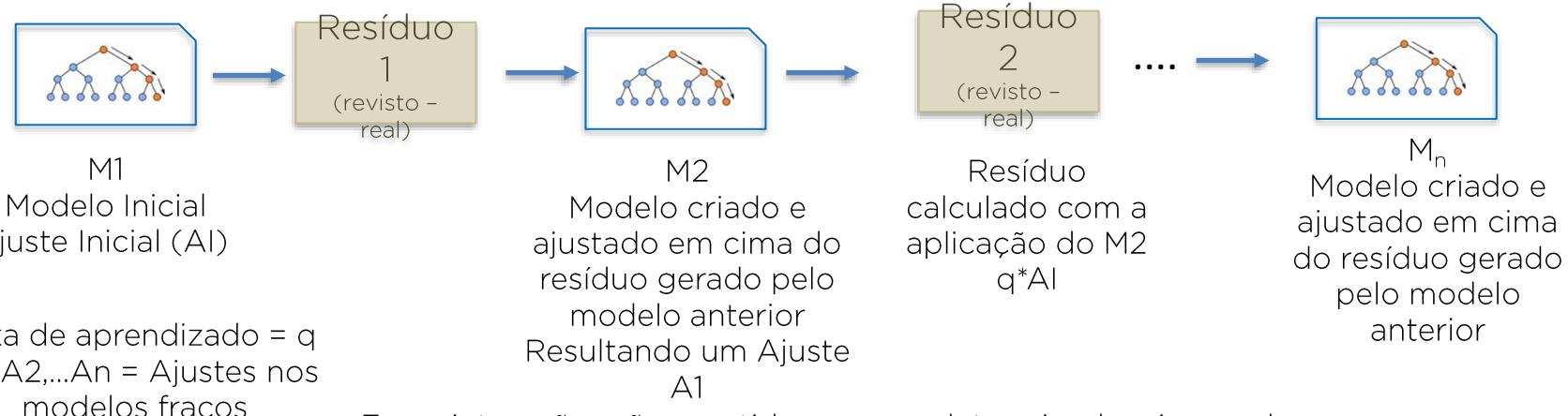
- O Gradient Boosting difere do Adaboost quanto à maneira com a qual os modelos são treinados com relação aos anteriores.
- Ao invés de estabelecer pesos para os “modelos simples” (weak learners), o Gradient Boosting treina novos modelos diretamente no erro dos modelos anteriores. Ou seja, os novos modelos tentam prever o erro dos modelos anteriores em vez de prever independentemente o target. Dessa forma, obtemos a predição final somando a predição de todos os “modelos fracos”.

- MÉTODOS DE ENSEMBLE LEARNING
- BOOSTING – GRADIENT BOOSTING

- O algoritmo do Gradient Boosting funciona assim:
 - o primeiro modelo faz uma aproximação bem simples da predição, e obtém-se os erros residuais (observado menos o previsto);
 - depois, é treinado mais modelos nesses erros residuais, para tentar predizer o erro do primeiro modelo;
 - A predição final é obtida com a soma das predições de cada modelo desenvolvido – como uma versão mais corrigida da primeira predição;
 - repetindo esse processo por várias iterações, obtém-se erros residuais cada vez menores.

MÉTODOS DE ENSEMBLE LEARNING

BOOSTING – GRADIENT BOOSTING



Essas interações são repetidas por um determinado número de vezes,
buscando minimizar o resíduo gerado pelos modelos fracos

O modelo final é a soma dos ajustes de todos os
A1 + q*M1 + q*A2 + ... + q*An

MÉTODOS DE ENSEMBLE LEARNING

BOOSTING – GRADIENT BOOSTING

Mais um pouco de teoria :

O Gradient Boosting Classifier depende de uma função de perda.

- Algoritmos de classificação frequentemente usam perda logarítmica.
- Algoritmos de regressão geralmente podem usar erros quadrados.

Os sistemas de aumento de gradiente têm duas partes necessárias: um modelo fraco e um componente aditivo:

- Os sistemas de aumento de gradiente usam árvores de decisão como “seus modelos fracos” (tanto para variável target classes ou para valores numéricos). Como as saídas são valores reais, à medida que novos modelos são adicionados ao modelo, a saída das árvores pode ser adicionada para corrigir erros nas previsões.
- O componente aditivo de um modelo de aumento de gradiente vem do fato de que as árvores são adicionadas ao modelo ao longo do tempo e, quando isso ocorre, as árvores existentes não são alteradas.



MÉTODOS DE ENSEMBLE LEARNING

BOOSTING – GRADIENT BOOSTING

Mais um pouco de teoria :

Um procedimento semelhante ao gradiente descendente é usado para minimizar o erro entre os parâmetros fornecidos. Isso é feito tomando a perda calculada e executando a descida do gradiente para reduzir essa perda. Em seguida, os parâmetros da árvore são modificados para reduzir a perda residual.

A saída da nova árvore é então anexada à saída das árvores anteriores usadas no modelo. Este processo é repetido até que um número de árvores previamente especificado seja alcançado, ou a perda seja reduzida abaixo de um certo limite.



BOOSTING

**XGBoost (Extreme Gradient
Boosting)**

MÉTODOS DE ENSEMBLE LEARNING

BOOSTING – XGBoost (Extreme Gradient

Boosting)

O XGBoost, baseado em algoritmos de árvores de decisão e que utiliza uma estrutura de Gradient boosting (aumento de Gradiente) ;

- XGBoost um Gradiente Boosting “tunado” - combina técnicas de otimização de software e hardware para produzir resultados superiores usando menos recursos de computação no menor período de tempo.
- O algoritmo XGBoost foi desenvolvido como um projeto de pesquisa na Universidade de Washington por Tianqi Chen e Carlos Guestrin.
- o mais vitorioso no Kaggle.
- tem sido utilizado em várias aplicações de ponta na indústria.
- há uma forte comunidade de cientistas de dados contribuindo para os projetos de código aberto XGBoost, com mais de 350 colaboradores e mais de 3.500 commits no GitHub;
- Em problemas de previsão envolvendo dados não estruturados, como imagens, textos e vídeos, as redes neurais artificiais tendem a superar todos os outros algoritmos ou frameworks. No entanto, quando se trata de dados estruturados/tabulares, algoritmos

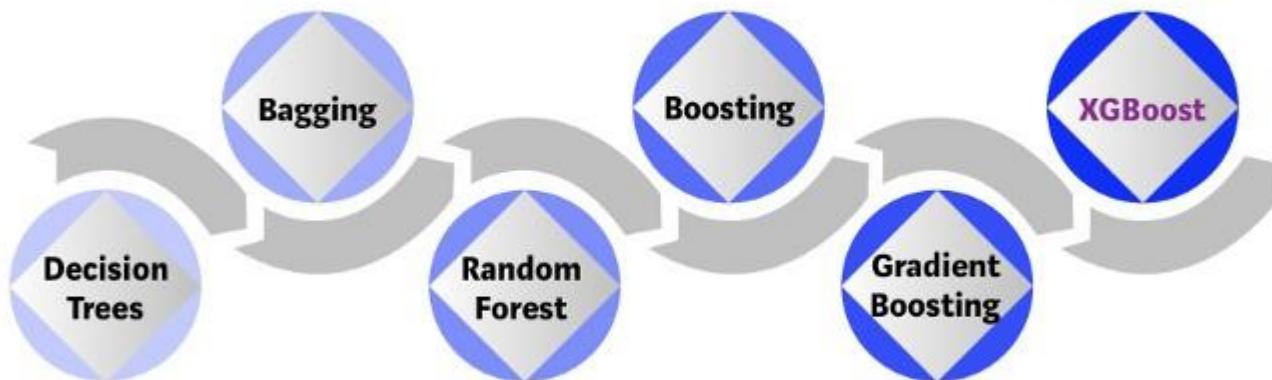
MÉTODOS DE ENSEMBLE LEARNING

Evolução dos algoritmos baseados em árvore ao longo dos anos.

Bootstrap aggregating or Bagging is a ensemble meta-algorithm combining predictions from multiple-decision trees through a majority voting mechanism

Models are built sequentially by minimizing the errors from previous models while increasing (or boosting) influence of high-performing models

Optimized Gradient Boosting algorithm through parallel processing, tree-pruning, handling missing values and regularization to avoid overfitting/bias



A graphical representation of possible solutions to a decision based on certain conditions

Bagging-based algorithm where only a subset of features are selected at random to build a forest or collection of decision trees

Gradient Boosting employs gradient descent algorithm to minimize errors in sequential models

Evolution of XGBoost Algorithm from Decision Trees

MÉTODOS DE ENSEMBLE LEARNING

BOOSTING – XGBoost (Extreme Gradient Boosting)

- O algoritmo se diferencia de outros das seguintes maneiras:
 - Aderente a uma ampla variedade de aplicações: Pode ser usado para resolver problemas de regressão, classificação, ranqueamento, entre muito outros.
 - Extremamente flexível – uma vez que possui um grande número de hiperparâmetros passíveis de aperfeiçoamento - você consegue ajustar adequadamente o XGBoost para o cenário do seu problema, seja ele qual for.
 - Portabilidade: Funciona em Windows, Linux e OS X.
 - Idiomas: Suporta todas as principais linguagens de programação, incluindo C++, Python, R, Java, Scala e Julia.
 - Integração em nuvem: oferece suporte a AWS, Azure, Clusters Yarn e funciona bem com Flink, Spark entre outros ecossistemas.

- MÉTODOS DE ENSEMBLE LEARNING
- **BOOSTING – GRADIENT BOOSTING - XGBoost**

- O primeiro passo no processo de treinamento do XGBoost é fazer uma predição inicial.
- Esta predição pode ser qualquer coisa, mas por padrão ela é 0.5, tanto para os problemas de regressão quanto para os problemas de classificação. Vale dizer aqui que o parâmetro `base_score` (considerada como probabilidade de partida) controla essa predição inicial, então editando esse valor, podemos mudar essa predição de 0.5.
- Para cada amostra, temos um resíduo, isto é, a diferença entre o valor observado e o predito, que nos mostra o quão boa a predição inicial é. Como o Gradient Boosting, o XGBoost treina uma árvore de decisão para os resíduos.
- Cada árvore começa com uma única folha, e todos os resíduos vão para essa folha.
- Assim como a primeira, as novas predições para as observações restantes têm resíduos menores do que antes, sugerindo que cada pequeno passo foi dado na direção certa.

- MÉTODOS DE ENSEMBLE LEARNING
- **BOOSTING – GRADIENT BOOSTING – XGBoost**

Uma nova árvore é construída baseada nos novos resíduos, e novas previsões são geradas com resíduos menores

E continua-se construindo árvores até que os resíduos sejam menores que um certo limite, ou até que tenha-se atingido o número máximo de árvores.

É calculado um score de qualidade, ou score de similaridade, para os resíduos. Este score de similaridade é:

$$\text{score de similaridade} = \frac{(\text{soma dos resíduos})^2}{\text{número de resíduos} + \lambda}$$

Onde, λ (lambda) é um parâmetro de regularização.

XGBoost e Gradient Boosting Machines (GBMs) são ambos métodos de árvore que aplicam o princípio de impulsionar weak learners usando a arquitetura de gradiente descendente. No entanto, o XGBoost aprimora a estrutura básica do GBM por meio da otimização de sistemas e aprimoramentos algorítmicos.

- MÉTODOS DE ENSEMBLE LEARNING
- **BOOSTING – GRADIENT BOOSTING - XGBoost**

Sobre as variáveis preditoras:

o XGBoost não pode lidar com recursos categóricos, ele só aceita valores numéricos (como o Random Forest). Portanto, é preciso executar várias codificações, como codificação de classes, codificação média ou codificação one-hot antes de fornecer dados categóricos ao XGBoost. (Ao contrário do CatBoost ou LGBM).

Tratamento de valores ausentes

No LightGBM e no XGBoost, os valores ausentes serão alocados para o lado que reduz a perda em cada divisão.

BOOSTING

CATBOOST(Categorial Boosting)

- MÉTODOS DE ENSEMBLE LEARNING
- **BOOSTING -**
- **CATBOOST(Categorial Boosting)**

- O CatBoost foi desenvolvido pela Yandex e supera muitos *boosting algorithms* como o XGBoost (que pode levar horas para ser treinado) e LightGBM em velocidade e também acurácia.
- O algoritmo é uma implementação de Gradient boosting que usa árvores de decisão binárias como preditores base. Durante o treinamento, ele cria um conjunto de árvores de decisão continuamente. Cada árvore seguinte é construída reduzindo o *loss** comparado com as anteriores
- O CatBoost executa o *gradient boosting* de uma forma diferente, utilizando ordered boosting. Se os dados não tiverem um "time", o CatBoost, cria aleatoriamente um tempo artificial para cada ponto de dados(*datapoints*)

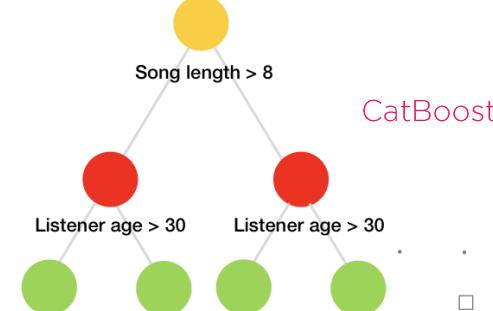
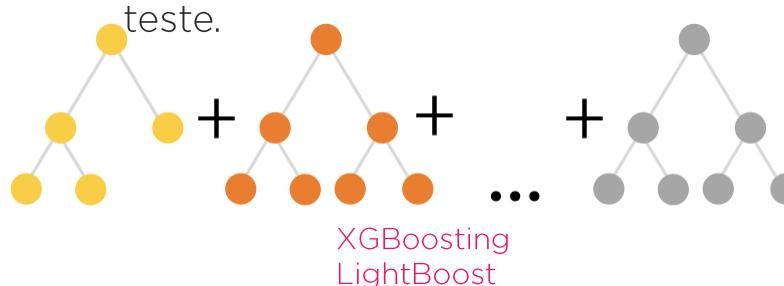
* real-estimado

Utiliza *Symmetric Trees* na geração de árvores

- MÉTODOS DE ENSEMBLE LEARNING
- **BOOSTING -**
- **CATBOOST(Categorial Boosting)**

Symmetric Trees

- Uma das principais diferenças entre o CatBoost e os demais algoritmos é sua implementação de symmetric trees (ou oblivious tree). O termo Oblivious significa que o mesmo critério de divisão é usado em todo o nível da árvore. Essas árvores são balanceadas, sendo menos propensas a overfitting e permitem acelerar significativamente a execução do modelo no momento do teste.



- MÉTODOS DE ENSEMBLE LEARNING
- **BOOSTING -**
- **CATBOOST(Categorial Boosting)**

- O CatBoost usa os mesmos recursos para dividir as instâncias de aprendizado nas partições esquerda e direita para cada nível da árvore. Neste caso, uma árvore de profundidade k tem exatamente 2^k folhas, e o índice de uma folha pode ser calculado com simples operações bit a bit.
- Assim, o esquema de aprendizado CatBoost é essencialmente profundo com alguma simplificação, obtido a partir do tipo de árvore de decisão.
- A escolha de árvores simétricas tem várias vantagens em relação às clássicas:
 - Esquema de montagem simples
 - Eficiente para implementar na CPU
 - Capacidade de fazer aplicadores de modelo muito rápido
 - Essa estrutura em árvore funciona como uma regularização, podendo trazer benefícios de qualidade para muitas tarefas

- MÉTODOS DE ENSEMBLE LEARNING
- **BOOSTING -**
- **CATBOOST(Categorial Boosting)**

Sobre as variáveis preditoras:

O CatBoost tem a flexibilidade de fornecer índices de colunas categóricas para que possa ser codificado como codificação one-hot usando one_hot_max_size (Usa codificação one-hot para todas as colunas com número de valores diferentes menor ou igual ao valor do parâmetro fornecido).

Se você não passar nada no argumento cat_features, o CatBoost tratará todas as colunas como variáveis numéricas.

Se uma coluna com valores de string não for fornecida em cat_features, CatBoost lançará um erro. Além disso, uma coluna que for do tipo int e seu conteúdo for

- MÉTODOS DE ENSEMBLE LEARNING
- **BOOSTING -**
- **CATBOOST(Categorial Boosting)**

Sobre as variáveis preditoras:

Tratamento de valores ausentes:

Catboost tem dois modos para processar valores ausentes, “Min” e “Max”.

- “Min”, os valores ausentes são processados como o valor mínimo para cada variável(eles recebem um valor menor que todos os valores existentes) ➔ garantido que uma divisão que separa os valores ausentes de todos os outros valores seja considerada ao selecionar as divisões.
- “Max” funciona exatamente da mesma forma que “Min”, apenas com valores máximos.

BOOSTING

LIGHTGBM

- MÉTODOS DE ENSEMBLE LEARNING
- **BOOSTING – LIGHTGBM (LIGHT GRADIENT BOOSTING MACHINE)**

LightGBM é uma estrutura de aumento de gradiente baseada em árvores de decisão para aumentar a eficiência do modelo e **reduzir o uso de memória**.

Ele foi projetado para ser distribuído e eficiente com as seguintes vantagens:

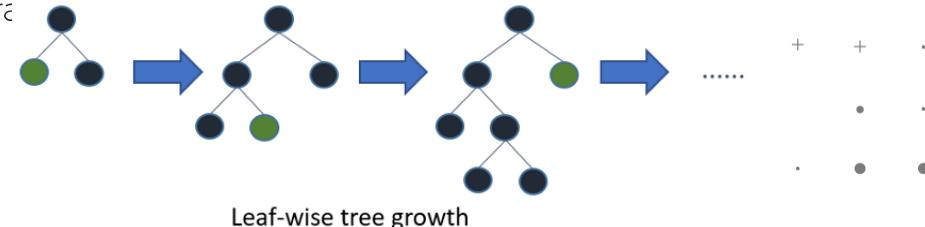
- Velocidade de treinamento mais rápida e maior eficiência.
- Menor uso de memória.
- Melhor precisão.
- Suporte de aprendizado paralelo, distribuído e GPU.
- Capaz de lidar com dados em grande escala.

- MÉTODOS DE ENSEMBLE LEARNING
- **BOOSTING – LIGHTGBM (LIGHT GRADIENT BOOSTING MACHINE)**

O LightGBM usa uma nova técnica de amostragem baseada em gradiente (GOSS) para filtrar as instâncias dos dados com maior valor de divisão, enquanto o XGBoost usa algoritmo pré-ordenado e baseado em histograma para calcular a melhor divisão.

GOSS (Gradient Based One Side Sampling) é um novo método de amostragem que tem como base os gradientes, isto é, instâncias com gradientes pequenos são bem treinadas (pequeno erro de treinamento) e aquelas com gradientes grandes são subtreinadas.

Folha com maior gradiente/erro é usada para

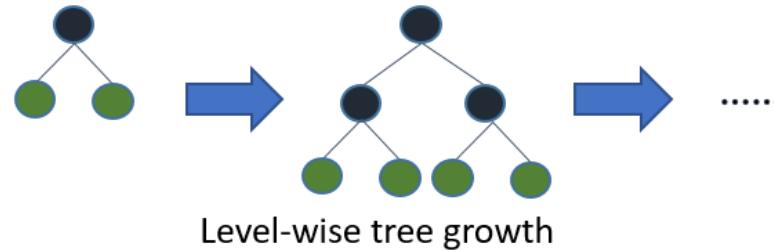


- MÉTODOS DE ENSEMBLE LEARNING
- **BOOSTING – LIGHTGBM (LIGHT GRADIENT BOOSTING MACHINE)**

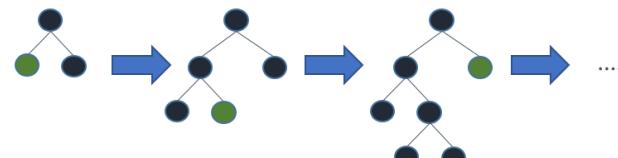
- Técnica de amostragem unilateral baseada em gradiente para LightGBM:
- diferentes instâncias de dados têm funções variadas no cálculo do ganho de informações. As instâncias com gradientes maiores (ou seja, instâncias mal treinadas) contribuirão mais para o ganho de informações. O GOSS mantém essas instâncias com grandes gradientes (por exemplo, maiores que um limite predefinido ou entre os percentis superiores) e apenas descarta aleatoriamente essas instâncias com pequenos gradientes para manter a precisão da estimativa de ganho de informação. Este tratamento pode levar a uma estimativa de ganho mais precisa do que a amostragem uniformemente aleatória, com a mesma taxa de amostragem alvo, especialmente quando o valor do ganho de informação tem uma grande faixa.

- MÉTODOS DE ENSEMBLE LEARNING
- **BOOSTING – LIGHTGBM (LIGHT GRADIENT BOOSTING MACHINE)**

- A maioria dos algoritmos de aprendizado de árvores de decisão crescem árvores por nível (profundidade):



LightGBM cresce árvores em folha (best-first). Ele escolherá a folha com perda máxima para crescer. Mantendo uma folha fixa, os algoritmos “leaf-wise” tendem a atingir uma perda menor do que os algoritmos “level-wise”.



- MÉTODOS DE ENSEMBLE LEARNING
- **BOOSTING – LIGHTGBM (LIGHT GRADIENT BOOSTING MACHINE)**

Vantagem:

- Menos uso de memória
- Redução no custo de comunicação para aprendizagem paralela
- Redução no custo para calcular o ganho para cada divisão na árvore de decisão.

Assim, como o LightGBM é treinado muito mais rápido, também pode levar ao caso de sobreajuste às vezes. Portanto, os parâmetros podem ser ajustados para obter um modelo ideal melhor.

- MÉTODOS DE ENSEMBLE LEARNING
- **BOOSTING – LIGHTGBM (LIGHT GRADIENT BOOSTING MACHINE)**

Sobre as variáveis preditoras:

Semelhante ao CatBoost, o LightGBM também pode lidar com recursos categóricos, inserindo os nomes das features. Ele não converte para codificação one-hot e é muito mais rápido do que codificação one-hot. LGBM usa um algoritmo especial para encontrar o valor de divisão de características categóricas.

Deve-se converter as variáveis categóricos para o tipo int antes de construir o conjunto de dados para LGBM. Ele não aceita valores de string, mesmo que seja apontado no parâmetro categorical_feature.

Tratamento de valores ausentes

No LightGBM e no XGBoost, os valores ausentes serão alocados para o lado que reduz a perda em cada divisão.

- MÉTODOS DE ENSEMBLE LEARNING
- **BOOSTING – LIGHTGBM (LIGHT GRADIENT BOOSTING MACHINE)**

Para obter o melhor ajuste, os seguintes parâmetros devem ser ajustados:

1. **num_leaves**: Uma vez que LightGBM cresce em folha, este valor deve ser menor que $2^{\text{(max_depth)}}$ para evitar um cenário de sobreajuste.
2. **min_data_in_leaf**: Para grandes conjuntos de dados, seu valor deve ser definido em centenas a milhares.
3. **max_depth**: um parâmetro-chave cujo valor deve ser definido de acordo para evitar sobreajuste.

Para obter uma melhor precisão, os seguintes parâmetros devem ser ajustados:

1. Mais dados de treinamento adicionados ao modelo podem aumentar a precisão.
2. **num_leaves**: aumentar seu valor aumentará a precisão conforme a divisão está ocorrendo em relação às folhas, mas também pode ocorrer overfitting.
3. **max_bin**: o valor alto terá um grande impacto na precisão, mas acabará resultando em sobreajuste.

COMO ESCOLHER OS MODELOS

COMO ESCOLHER OS MODELOS

Testar todos os algoritmos possíveis para os dados disponíveis.

Escolher o que melhor se aplica ao problema proposto.

Lembrar que há diversidade de problemas e diferentes conjunto de base de dados e diferentes features, isto é, base de dados diferentes podem mostrar algoritmos melhores, dependendo da situação.

Não há um modelo que é o melhor em todos os critérios!!!!

A escolha deve ser feita em função do objetivo, por exemplo se a velocidade de treinamento é nosso ponto problemático, usamos LightGBM.

COMO ESCOLHER OS MODELOS

Em tarefas em que precisamos de alta precisão, podemos treinar todos os algoritmos para encontrar aquele com o WAUC mais alto para esse conjunto de dados específico.

Para melhorar a velocidade, o treinamento em GPUs pode ser muito significativo.

Não é a escolha somente de um algoritmo mas é preciso escolher a configuração correta do algoritmo para um determinado conjunto de dados, ajustando os hiper-parâmetros.

Há um grande número de hiperparâmetros para serem ajustados, sendo alguns exclusivos para um algoritmo específico.

Ainda tem que se pensar em outras situações para a escolha do algoritmo,

BIBLIOGRAFIA

- KUHN, M. / JOHNSON K. *Applied Predictive Modeling*, 1st ed. 2013, Corr. 2nd printing 2018 Edition
- LESKOVEC, RAJAMARAM, ULLMAN. *Mining of Massive Datasets*, 2014. <http://mmds.org>.
- HAIR, J.F. / ANDERSON, R.E. / TATHAN, R.L. / BLACK, W.C. *Análise multivariada de dados*, 2009
- TORGÓ, L. *Data Mining with R: Learning with Case Studies*, 2.a ed. Chapman and Hall/CRC , 2007
- MINGOTI, S.A.; *Análise de dados através de métodos de estatística multivariada*, UFMG, 2005
- CARVALHO, L.A.V., *Datamining – A mineração de dados no marketing, medicina, economia, engenharia e administração*. Rio de Janeiro: Editora Ciência Moderna, 2005.
- BERRY,M.J.A., LINOFF,G. *Data Mining Techniques For Marketing, Sales and Customer Support*. 3a. ed. New York: John Wiley & Sons, Inc., 2011.
- DUNHAM, M.H. *Data Mining - Introductory and Advanced Topics*. Prentice Hall, 2002.
- DINIZ,C.A.R. , NETO F.L. *Data Mining: Uma Introdução*. São Paulo: XIV Simpósio Nacional de Probabilidade e Estatística. IME-USP, 2000.

OBRIGADA!



/AdelaideAlves



profadelaide.alves@fiap.com.br

FIAP

Copyright © 2022 | Professor (a) Adelaide Alves de Oliveira

Todos os direitos reservados. Reprodução ou divulgação total ou parcial deste documento, é expressamente proibido sem consentimento formal, por escrito, do professor/autor.

FIAP