ESPECIALIZAÇÃO EM CIÊNCIA DE DADOS – PUC-RIO
MACHINE LEARNING

AULA 4: ENSEMBLES E SELEÇÃO DE ATRIBUTOS

Tatiana Escovedo, PhD.
tatiana@inf.puc-rio.br

ENSEMBLES

ENSEMBLES

- Ajudam a melhorar os resultados, combinando vários modelos.
- Geralmente têm melhor desempenho preditivo em comparação com um único modelo, sendo os primeiros colocados em muitas competições de aprendizado de máquina de prestígio.
- **São meta-algoritmos que combinam várias técnicas de aprendizado de máquina em um modelo preditivo para diminuir a variância (bagging), o viés (boosting) ou melhorar as predições (stacking).**

Para saber mais: *https://blog.statsbot.co/ensemble-learning-d1dcd548e936*

PODEM SER DE 2 GRUPOS

- Métodos **sequenciais**, em que os modelos base são gerados sequencialmente.
  - Exploram a **dependência** entre os modelos de base. O desempenho geral pode ser aprimorado ponderando os exemplos previamente rotulados incorretamente com maior peso.
- Métodos **paralelos**, em que os modelos base são gerados em paralelo.
  - Exploram a **independência** entre os modelos base, pois o erro pode ser reduzido drasticamente se a média for utilizada.

ENSEMBLES HOMOGÊNEOS E HETEROGÊNEOS

- A maioria dos métodos ensemble usa um único modelo básico, produzindo **ensembles homogêneos**, mas também há métodos que usam modelos diferentes (**ensembles heterogêneos**).
- Para que os métodos de ensemble sejam melhores do que qualquer um de seus modelos individuais, os modelos base precisam boa precisão e serem **diversificados**.
- Os modelos precisam, idealmente, ter **alta variância** e **baixo viés (bias)**.

MÉTODOS ENSEMBLE MAIS POPULARES

- **Bagging:** constrói vários modelos (geralmente do mesmo tipo) a partir de diferentes sub-amostras do conjunto de dados de treinamento.
- **Boosting:** constrói vários modelos (geralmente do mesmo tipo) e cada um deles aprende a corrigir os erros de predição de um modelo anterior na sequência de modelos.
- **Votação (Stacking):** constrói vários modelos (geralmente de tipos diferentes) e calcula estatísticas simples (ex: média) para combinar as predições.

BOOSTRAP

- **Método estatístico poderoso para estimar uma quantidade (como média ou desvio padrão) a partir de uma amostra pequena de dados.**

- ***Exemplo:*** temos uma amostra de 100 valores e gostaríamos de obter uma estimativa da média da amostra. Podemos calcular a média diretamente da amostra

$$mean(x) = \frac{1}{100} \times \sum_{i=1}^{100} x_i$$

BOOSTRAP

- Como sabemos que a amostra é **pequena**, esta média provavelmente tem um **erro**.
- Podemos melhorar o estimativa de nossa média usando o **Bootstrap**:
  1. Crie muitas sub-amostras aleatórias (por exemplo, 1000) do conjunto de dados com substituição (o que significa que podemos selecionar o mesmo valor várias vezes).
  2. Calcule a média de cada sub-amostra.
  3. Calcule a média das médias coletadas e use-a como a média estimada para os dados.

BAGGING

- **Bagging:** Bootstrap Aggregation
- Uma forma de reduzir a **variância** de uma predição é calcular a **média** de várias predições.
- Usa Boostrap para coletar **várias amostras** do conjunto de treinamento (com reposição) e treina **um modelo para cada amostra**. A predição final de saída é a **classe mais votada** (classificação) ou a **média** (regressão) entre as predições de todos os submodelos.
- Pode ser usado para reduzir a **variância** dos algoritmos com alta variância, como as árvores de decisão, que são muito sensíveis aos dados específicos com os quais são treinados.

BAGGING

Funcionamento:

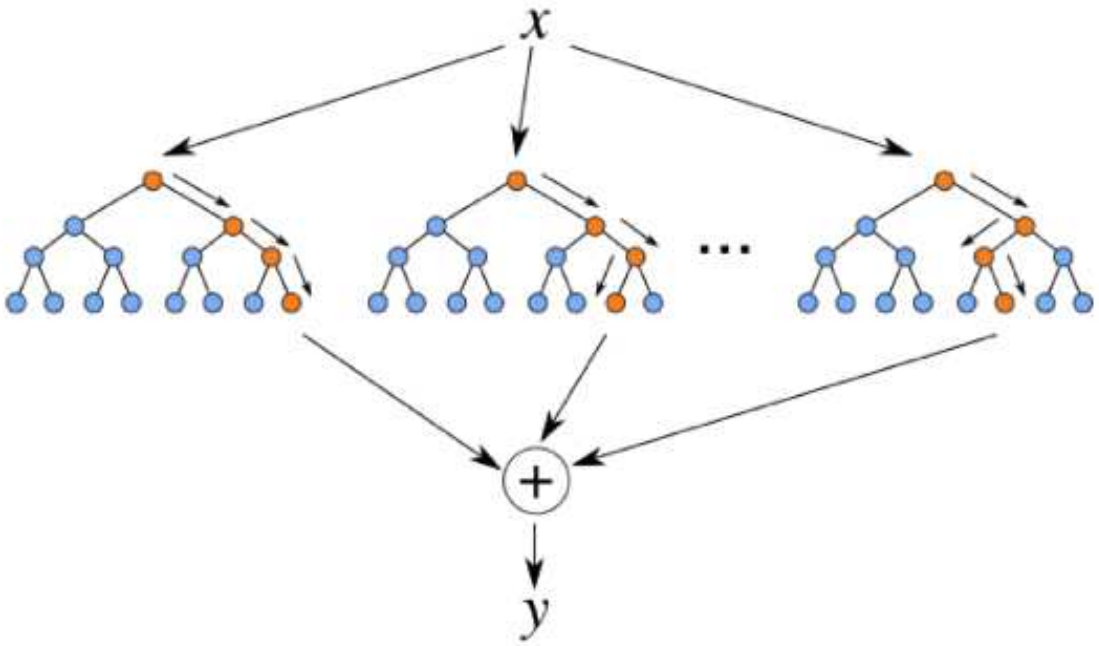
1. Crie várias sub-amostras (com aprox. 60% do dataset original) aleatórias do conjunto de dados de treino, com reposição.
2. Treine um modelo (por exemplo, CART) em cada amostra.
3. Dado um novo conjunto de dados, calcule a predição média do ensemble (classe mais frequente ou média dos valores de cada modelo).

- *Como estamos menos preocupados com as árvores individuais, elas crescem profundamente e não são podadas.*
- *O único parâmetro para o bagging de árvores de decisão é o número de árvores a serem criadas, que pode ser escolhido experimentalmente.*

RANDOM FORESTS (FLORESTAS ALEATÓRIAS)

- São uma melhoria em relação às árvores de decisão com Bagging.
- Um problema das árvores de decisão é que elas são **gulosas**: elas escolhem o atributo para o particionamento usando um algoritmo guloso que minimiza o erro.
  - Assim, as árvores de decisão geradas para o Bagging podem ter muitas semelhanças estruturais, resultando em **alta correlação** das predições.
- A combinação de predições de vários modelos em conjuntos funciona melhor se as predições dos submodelos não forem correlacionadas ou, na melhor das hipóteses, fracamente correlacionadas.

RANDOM FORESTS (FLORESTAS ALEATÓRIAS)



RANDOM FORESTS (FLORESTAS ALEATÓRIAS)

- No Random Forest, as sub-árvores são treinadas de forma que suas predições tenham menos correlação:
  - Em vez de utilizar todos os atributos para selecionar o melhor ponto de divisão, o algoritmo é limitado a uma amostra **aleatória** de atributos a pesquisar.
  - Para esta amostra de atributos, o algoritmo busca o ponto de corte ótimo para cada atributo.
  - O número de atributos que podem ser pesquisados em cada ponto de divisão deve ser especificado como um **parâmetro** para o algoritmo e pode ser escolhido experimentalmente, mas um bom valor padrão é:
    - **Para problemas de classificação:** $m = \sqrt{num\_atributos}$
    - **Para problemas de classificação:** $m = num\_atributos/3$

RANDOM FORESTS - DESEMPENHO ESTIMADO

- Para cada amostra de inicialização retirada dos dados de treinamento, haverá amostras que não foram incluídas, chamadas de *out-of-bag-samples* (OOB).
- O desempenho de cada modelo em suas amostras OOB pode fornecer uma precisão estimada do modelo (estimativa OOB).
- São uma estimativa confiável do erro de teste e se correlacionam bem com as estimativas de erro de validação cruzada.

RANDOM FORESTS - IMPORTÂNCIA DO ATRIBUTO

- À medida que as árvores de decisão bagging são construídas, podemos calcular quanto a função de erro cai para um atributo em cada ponto de divisão (soma quadrática dos erros ou índice Gini, por exemplo).
- As diminuições no erro podem ser calculadas como a média de todas as árvores de decisão e usados para fornecer uma estimativa da importância de cada atributo de entrada. **Quanto maior a queda do erro na escolha do atributo, maior a sua importância.**
- Isto pode ajudar a identificar subconjuntos de atributos de entrada que podem ser mais ou menos relevantes para o problema, sendo também uma **técnica de seleção de atributos**.

EXTRA TREES

- Também conhecido como Extremely Randomized Trees (Árvores Extremamente Aleatórias),
- As divisões são realizadas **aleatoriamente:**
  - A divisão a realizar é a melhor divisão usando pontos de corte **aleatórios** em um subconjunto de atributos selecionado **aleatoriamente** para a árvore corrente.
- Devido a aleatoriedade, é um modelo mais rápido computacionalmente do que as Random Forests, além de geralmente terem menor variância.

Para saber mais: *https://www.thekerneltrip.com/statistics/random-forest-vs-extra-tree/*
https://towardsdatascience.com/an-intuitive-explanation-of-random-forest-and-extra-trees-classifiers-8507ac21d54b

BOOSTING

- Cria uma sequência de modelos na qual um modelo tenta corrigir os erros dos modelos anteriores.
- Busca converter modelos **fracos** em modelos **fortes**.
- Mais peso é dado aos exemplos que foram classificados incorretamente em rodadas anteriores.
- As predições são combinadas por votação majoritária ponderada (classificação) ou média ponderada (regressão) para produzir a predição final.

ALGORITMOS DE BOOSTING

Adaboost

- Foi o primeiro algoritmo de boosting bem-sucedido.
- Pondera as instâncias no conjunto de dados de acordo com o quão fácil ou difícil elas são classificadas no modelo corrente, permitindo que o algoritmo preste menos ou mais atenção a elas na construção de modelos subseqüentes.

Gradient Boosting

- Variação do Adaboost.
- É uma das técnicas de ensemble mais sofisticadas e uma das melhores técnicas disponíveis para melhorar o desempenho via ensembles.

ADABOOST COM ÁRVORES DE DECISÃO

Funcionamento:

1. Treina uma primeira árvore de decisão na qual cada observação recebe um peso igual.
2. Avalia a primeira árvore, aumenta os pesos das observações difíceis de classificar e diminui os pesos para as fáceis de classificar.
3. Treina uma segunda árvore usando os dados anteriores ponderados, com o objetivo de melhorar as predições da árvore anterior.
4. Avalia a segunda árvore, ajusta os pesos das observações.
5. Repete os passos 3 e 4 para um número especificado de iterações.

GRADIENT BOOSTING

Envolve 3 elementos:

- **Uma função de perda a ser otimizada.**
  - Pode ser especificada pelo usuário. Depende do tipo de problema que está sendo resolvido e deve ser diferenciável.
- **Um modelo fraco para fazer predições.**
  - As árvores de decisão são geralmente usadas como modelo fraco, sendo comum restringi-las (número máximo de camadas, nós, divisões ou folha) para garantir que os modelos sejam fracos, mas que ainda possam ser construídos de maneira gulosa.
- **Um procedimento para adicionar novos modelos fracos para minimizar a função de perda.**
  - As árvores são adicionadas uma de cada vez, usando um procedimento de gradiente descendente para minimizar a perda ao adicionar uma nova árvore parametrizada de tal forma que reduza o erro (ou seja, na direção do gradiente).

GRADIENT BOOSTING

- Sua principal diferença em relação ao AdaBoost é a **forma de identificar as deficiências** dos modelos: o AdaBoost identifica as deficiências através do aumento de pesos altos, enquanto o Gradient Boosting aplica gradientes na função de perda, uma medida que indica quão bons são os coeficientes do modelo em ajustar os dados subjacentes.
- **Vantagem:** permite otimizar uma função de custo **especificada pelo usuário**, em vez de uma função de perda que geralmente oferece menos controle e não corresponde essencialmente a aplicativos do mundo real.
- **Desvantagem:** é um algoritmo guloso e pode gerar *overfitting* rapidamente um conjunto de dados de treinamento. Ele pode se beneficiar de técnicas de regularização que penalizam diversas partes do algoritmo, melhorando seu desempenho e reduzindo o *overfitting*. Estas técnicas estão fora do escopo deste curso.

Para saber mais: *https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/*
https://towardsdatascience.com/understanding-gradient-boosting-machines-9be756fe76ab

VOTAÇÃO (STACKING)

- É uma das técnicas mais simples de combinar predições de vários modelos de ML.
  1. Cria dois ou mais modelos independentes a partir do conjunto de dados de treinamento.
  2. Um classificador de votação agrupa os modelos e calcular a média das predições dos submodelos quando é solicitado a fazer predições para novos dados.
- As predições dos submodelos podem ser ponderadas, mas é difícil especificar os pesos dos classificadores manualmente ou mesmo heuristicamente.
  - Métodos mais avançados podem aprender a ponderar melhor as predições dos submodelos. Esta técnica é conhecida como *stacking (stacked aggregation)*, mas atualmente não é fornecida no Scikit-learn.

SELEÇÃO DE ATRIBUTOS

SELEÇÃO DE ATRIBUTOS (FEATURE SELECTION)

- Os atributos (X) dos dados usados para treinamento dos modelos de ML têm uma enorme influência no seu desempenho. atributos irrelevantes ou parcialmente relevantes podem afetar negativamente o desempenho do modelo, especialmente algoritmos lineares como regressão linear e logística.
- O processo de seleção de atributos realiza a seleção automática dos atributos que mais contribuem para a saída (Y) e tem como vantagens:
  - **Redução de *Overfitting*:** dados menos redundantes = menos risco de tomar decisões com base em ruído.
  - **Melhoria da precisão:** dados menos “enganosos” = melhor a precisão da modelagem.
  - **Redução do tempo de treinamento:** menos dados = treinamento mais rápido.

TÉCNICAS DE SELEÇÃO DE ATRIBUTOS

Seleção Univariada

- Testes estatísticos são usados para selecionar os atributos que têm relacionamento mais forte com a variável de saída.

Eliminação Recursiva de atributos (RFE)

- Remove recursivamente os atributos e vai construindo um modelo com os que permanecem.
- Usa a acurácia do modelo para identificar quais atributos (e a combinação de atributos) que mais contribuem para prever o atributo de destino.

TÉCNICAS DE SELEÇÃO DE ATRIBUTOS

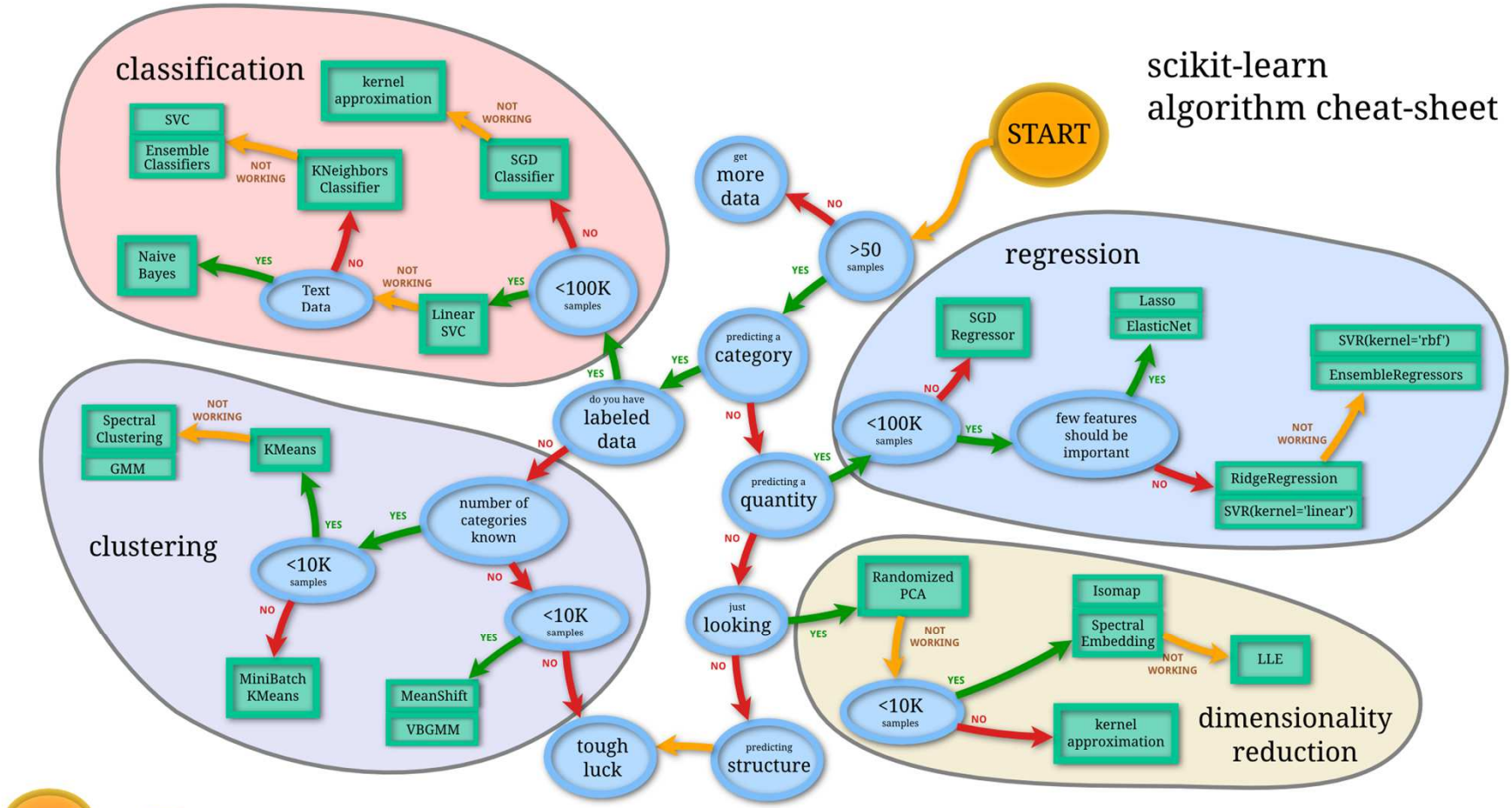
Análise de Componentes Principais (PCA)

- Usa álgebra linear (decomposição SVD) para transformar o conjunto de dados em um formato compactado (em um espaço dimensional inferior).
- É possível escolher o número de dimensões ou componentes principais no resultado transformado.

Importância de atributos

- Técnicas como Random Forest ou Extra Trees podem ser usadas para estimar a importância dos atributos.

Para saber mais: *https://towardsdatascience.com/feature-selection-for-machine-learning-1-2-1597d9ccb54a*


scikit-learn
algorithm cheat-sheet
START
classification
kernel approximation
NOT WORKING
SVC
Ensemble Classifiers
NOT WORKING
KNeighbors Classifier
SGD Classifier
NO
Naive Bayes
YES
NO
Text Data
NOT WORKING
YES
Linear SVC
YES
<100K samples
NO
get more data
NO
>50 samples
YES
regression
SGD Regressor
Lasso
ElasticNet
SVR(kernel='rbf')
EnsembleRegressors
NO
<100K samples
YES
few features should be important
YES
NOT WORKING
RidgeRegression
SVR(kernel='linear')
NO
predicting a category
YES
NO
do you have labeled data
YES
NO
predicting a quantity
YES
NO
clustering
Spectral Clustering
GMM
NOT WORKING
KMeans
YES
number of categories known
YES
NO
<10K samples
NO
MiniBatch KMeans
<10K samples
YES
NO
MeanShift
VBGMM
just looking
YES
NO
Randomized PCA
NOT WORKING
Isomap
Spectral Embedding
NOT WORKING
LLE
YES
<10K samples
NO
kernel approximation
dimensionality reduction
tough luck
predicting structure

Back