



Projeto 3

Grupo 3

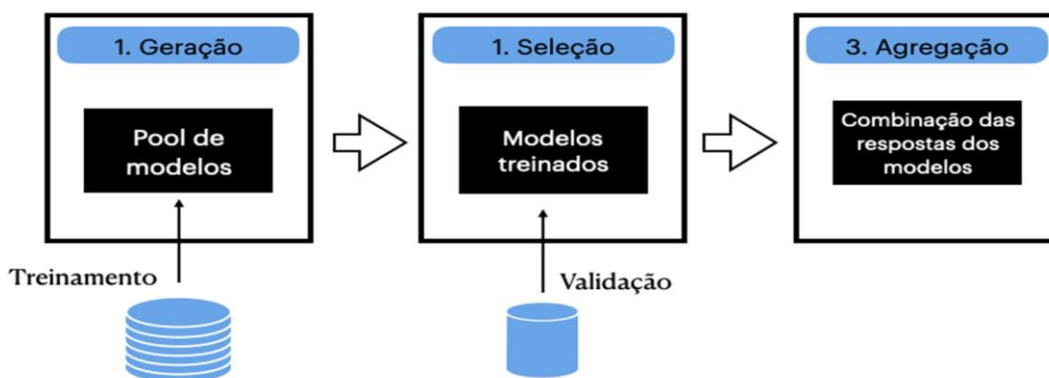
Ana Claudia Quintino Oliveira
André Souza
Bruno de Sousa Donato
Lucas dos Santos Garcia
Nicolas Spogis
Roger Trezza
Thiago Martins
Thomaz Barros Pires
Vinicius Vizenzo

1. Ensemble Methods

Ensemble Methods podem ser traduzidos livremente como “Métodos de Conjunto”. Esta tradução faz sentido, pois as técnicas de ensemble consistem justamente em combinar múltiplos modelos individuais para tentar melhorar a performance preditiva sobre um determinado problema. Podemos fazer uma analogia desta técnica a um princípio conhecido como “A Sabedoria das Multidões”, que estabelece que estimativas e respostas mais precisas podem ser obtidas combinando os julgamentos de diferentes avaliadores.

Em um ensemble, você combina as previsões de vários modelos para produzir uma previsão final mais precisa.

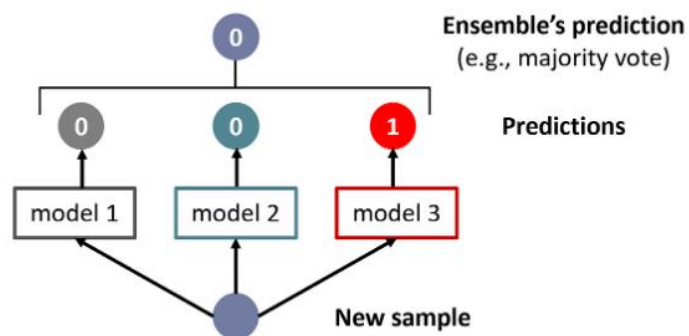
Podemos fazer uma analogia com decisões democráticas onde quem vence é a maioria. Com isto, é possível alcançar maiores níveis de acurácia e precisão na resposta final e diminuir os vieses que podem levar a conclusões erradas.



Funcionamento: Agregamento de várias árvores de decisão para se obter uma resposta final com base em todas as considerações finais individuais

Benefícios Principais: Respostas mais assertivas (Precisão), maior desempenho (Performance) e menores vieses.

Principais Métodos: Bagging, Boosting, AdaBoost, Extra Trees, Random Forest, LightGBM, CatBoost e XG Boost.



Fonte Imagem: Towards AI

Aqui está um exemplo simples usando uma média simples das previsões de dois modelos:

```
from sklearn.ensemble import VotingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Gerar dados de exemplo
X, y = make_classification(n_samples=1000, n_features=20, random_state=42)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Inicializar modelos
model1 = LogisticRegression(random_state=42)
model2 = DecisionTreeClassifier(random_state=42)

# Criar ensemble
ensemble_model = VotingClassifier(estimators=[('lr', model1), ('dt', model2)],
voting='soft')

# Treinar ensemble
ensemble_model.fit(X_train, y_train)

# Fazer previsões
predictions = ensemble_model.predict(X_test)

# Avaliar precisão
accuracy = accuracy_score(y_test, predictions)
print("Accuracy:", accuracy)
```

1.1. Desvantagens dos Modelos de Ensemble:

Complexidade computacional: por conta do algoritmo e fluxograma mais complexos, os modelos ensemble são mais complexos computacionalmente visto que são um agregado de modelos de decisão (strong learner) e não apenas um (weak learner).

Dificuldade de interpretação em comparação com modelos simples: é um modelo mais robusto tecnicamente, logo a sua interpretação é mais difícil.

Aplicações dos Modelos de Ensemble: Classificação, Regressão e Detecção de anomalias.

Dicas para Utilização Eficiente de Modelos de Ensemble: Escolha do modelo certo para o problema, ajuste dos hiperparâmetros e validação cruzada.

1.2. Bagging:

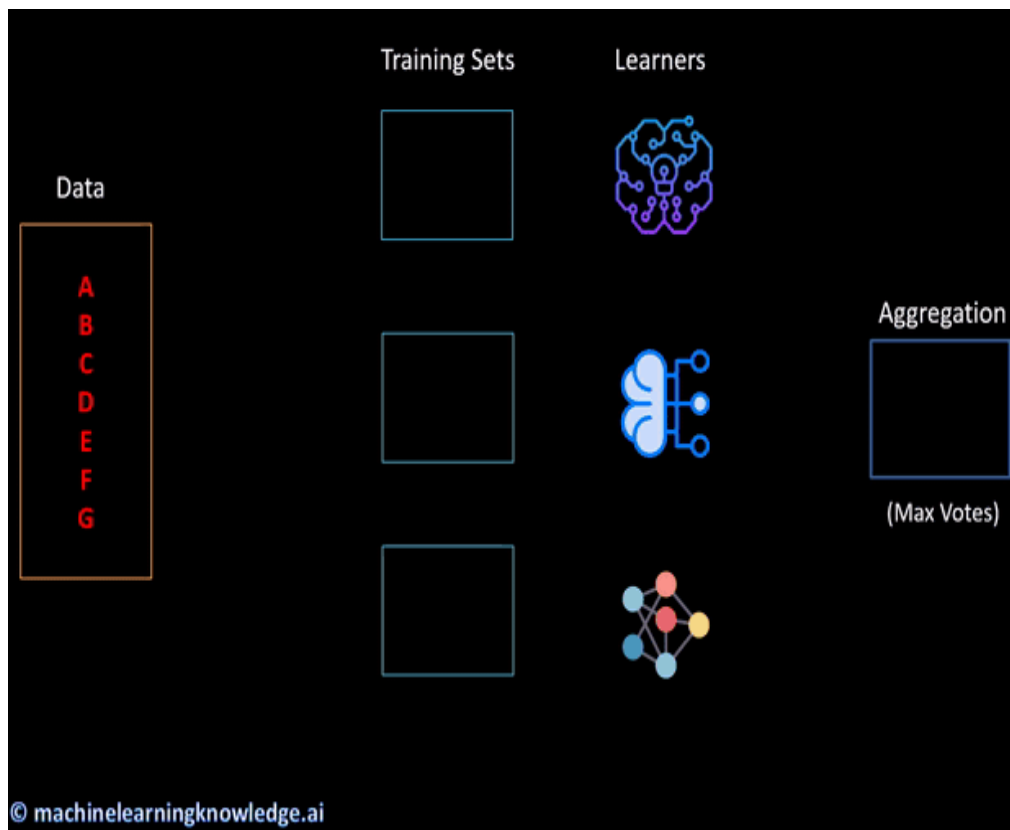
Bagging, ou Bootstrap Aggregating, é uma técnica de ensemble onde várias instâncias do mesmo algoritmo de aprendizado são treinadas em subconjuntos aleatórios do conjunto de dados original. Depois, as previsões desses modelos são agregadas, geralmente por média ou votação, para produzir uma previsão final.

Principais Características:

Amostragem por Bootstrap: O Bagging utiliza amostragem por bootstrap para criar várias subamostras do conjunto de dados original, permitindo que cada modelo seja treinado em uma amostra diferente.

Modelos Independentes: Cada modelo no *ensemble* é treinado de forma independente nos conjuntos de dados bootstrap, o que ajuda a reduzir a correlação entre os modelos e a evitar o overfitting.

Combinação por Votação ou Média: Para fazer previsões, o Bagging combina as previsões de cada modelo por meio de votação (classificação) ou média (regressão).



1.2.1. Vantagens do Bagging:

- Reduz a variância do modelo, tornando-o mais estável e menos sensível a pequenas variações nos dados de treinamento.
- Melhora a precisão do modelo, especialmente em modelos propensos ao overfitting.

Aplicações Comuns:

- Classificação e regressão em uma variedade de conjuntos de dados, especialmente úteis em conjuntos de dados grandes e complexos.

Dica de Uso: Experimente diferentes modelos de base (por exemplo, árvores de decisão, SVMs) e ajuste os hiperparâmetros para obter o melhor desempenho do Bagging em seu problema específico

Algoritmo 1: Bagging**Entrada:** Base de treinamento D **Saída** : Classificador *ensemble*

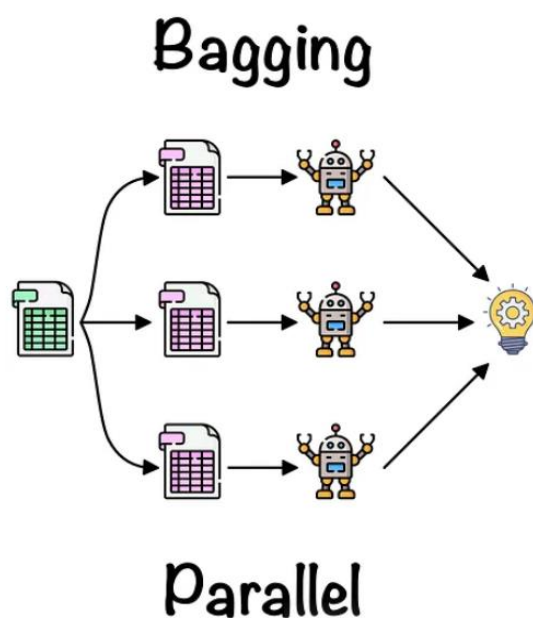
```

1 para  $i \leftarrow 1$  até  $n$  faça
2   | Construa uma amostra  $D_i$  da base  $D$ 
3   | Aprenda um modelo de classificação  $H_i$  com base em  $D_i$ 
4 fim
5 retorna Classificador ensemble  $H$ 

```

Um classificador Bagging é um comitê que treina (fit) os classificadores base em subconjuntos aleatórios do conjunto de dados original e, em seguida, agrega suas previsões individuais por votação ou por média para formar uma previsão final.

O Bagging pode normalmente ser usado como uma forma de reduzir a variância de um estimador (por exemplo, uma árvore de decisão), introduzindo aleatoriedade por meio do procedimento na construção dos datasets.



1.2.2. Existem 4 tipos de Bagging:

- Pasting: quando os subconjuntos são subconjuntos aleatórios do conjunto de dados (não há reposição de amostras)
- Bagging: quando as amostras são retiradas com reposição.
- Random Subspaces: quando os subconjuntos de dados são gerados a partir de subconjunto de atributos.
- Random Patches : quando estimadores de base são construídos em subconjuntos de amostras e atributos.

1.2.3. O que é Bootstrapping

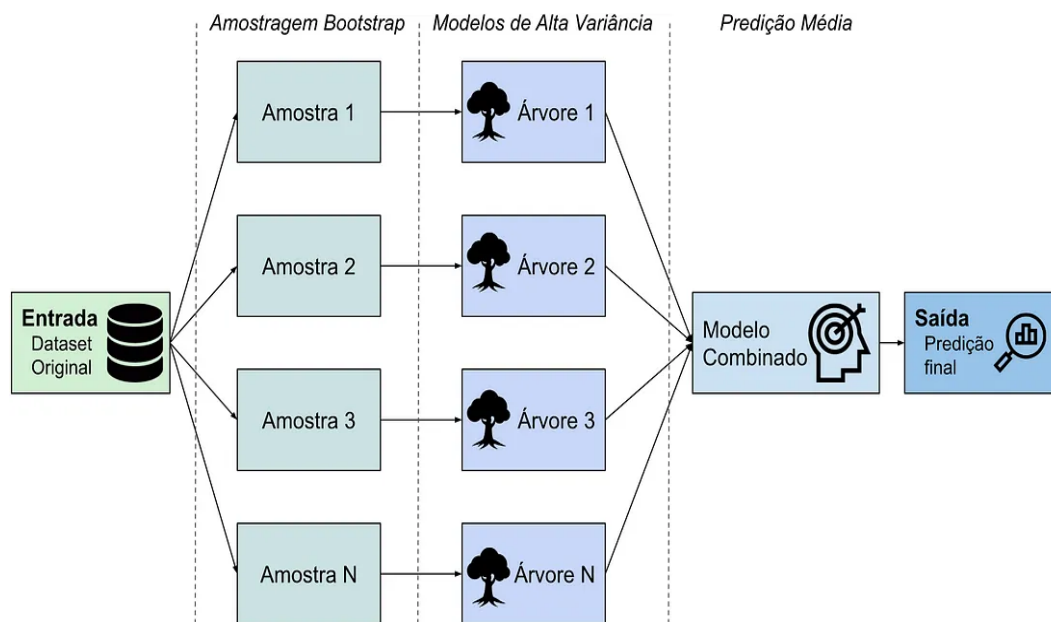
Esta técnica estatística consiste em gerar amostras de um tamanho B (chamadas de amostras bootstrap) a partir de um conjunto de dados inicial de tamanho N , selecionando aleatoriamente com reposição B observações.



Sob algumas suposições, essas amostras têm boas propriedades estatísticas: em aproximação, elas podem ser vistas como sendo retiradas diretamente da verdadeira distribuição de dados adjacente (e muitas vezes desconhecida) e independente umas das outras. Assim, elas podem ser consideradas como amostras representativas e independentes da verdadeira distribuição de dados.

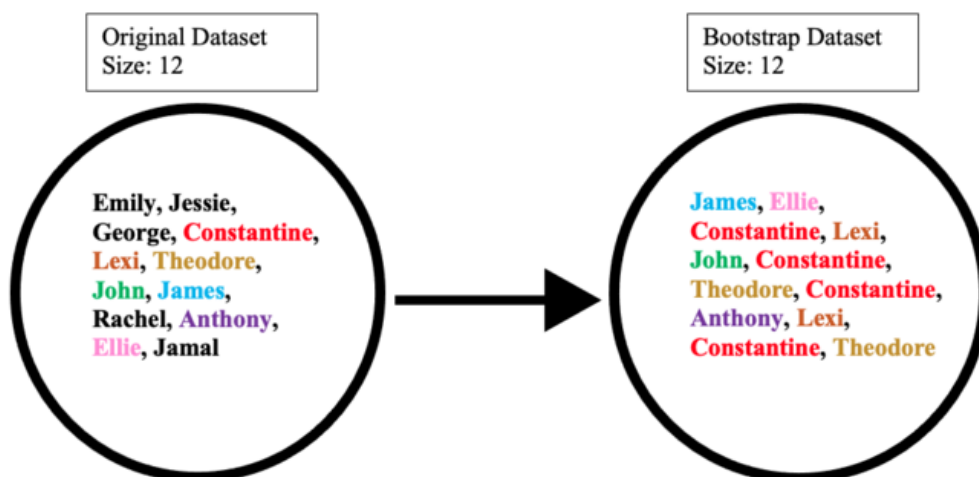
As hipóteses que devem ser verificadas para que essa aproximação seja válida são duas. Primeiro, o tamanho N do conjunto de dados inicial deve ser grande o suficiente para capturar a maior parte da complexidade da distribuição subjacente, de modo que a amostragem do conjunto de dados seja uma boa aproximação da amostragem da distribuição real. Em segundo lugar, o tamanho N do conjunto de dados deve ser grande o suficiente em comparação com o tamanho B das amostras de bootstrap para que as amostras não estejam muito correlacionadas (independentes).

Amostras bootstrap são frequentemente usadas, por exemplo, para avaliar a variância ou intervalos de confiança de estimadores estatísticos. Por definição, um estimador estatístico é uma função de algumas observações e, portanto, uma variável aleatória com variância proveniente dessas observações. Para estimar a variância de tal estimador, precisamos avaliá-lo em várias amostras independentes retiradas da distribuição de interesse. Na maioria dos casos, considerar amostras verdadeiramente independentes exigiria muitos dados em comparação com a quantidade realmente disponível. Podemos então usar o método de bootstrap para gerar várias amostras de bootstrap que podem ser consideradas como sendo “quase-representativas” e “quase-independentes”. Essas amostras de bootstrap nos permitirão aproximar a variância do estimador, avaliando seu valor para cada uma delas.



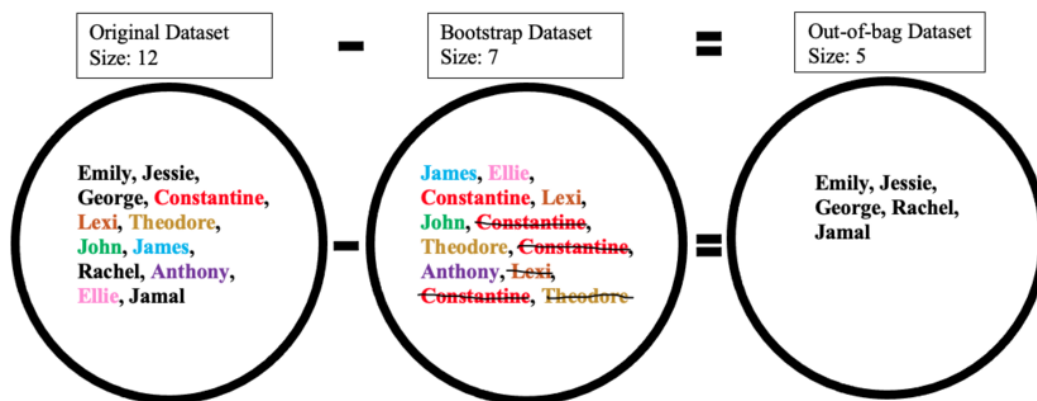
1.2.4. Bootstrap dataset

O conjunto de dados de bootstrap é feito escolhendo aleatoriamente objetos do conjunto de dados original. Além disso, deve ter o mesmo tamanho do conjunto de dados original. No entanto, a diferença é que o conjunto de dados de bootstrap pode ter objetos duplicados. Aqui está um exemplo simples para demonstrar como funciona junto com a ilustração abaixo:



1.2.5. Out-of-bag dataset

O conjunto “Out-of-bag” representa as pessoas restantes que não estavam no conjunto de dados de inicialização. Pode ser calculado calculando a diferença entre os conjuntos de dados originais e de bootstrap. Nesse caso, as demais amostras que não foram selecionadas são Emily, Jessie, George, Rachel e Jamal. Tenha em mente que, como ambos os conjuntos de dados são conjuntos, ao calcular a diferença, os nomes duplicados são ignorados no conjunto de dados de inicialização. A ilustração abaixo mostra como a matemática é feita:



Aqui está um exemplo de como usar o algoritmo RandomForestClassifier, que utiliza a técnica de bagging para treinar várias árvores de decisão em subconjuntos aleatórios do conjunto de dados:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Gerar dados de exemplo
X, y = make_classification(n_samples=1000, n_features=20, random_state=42)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

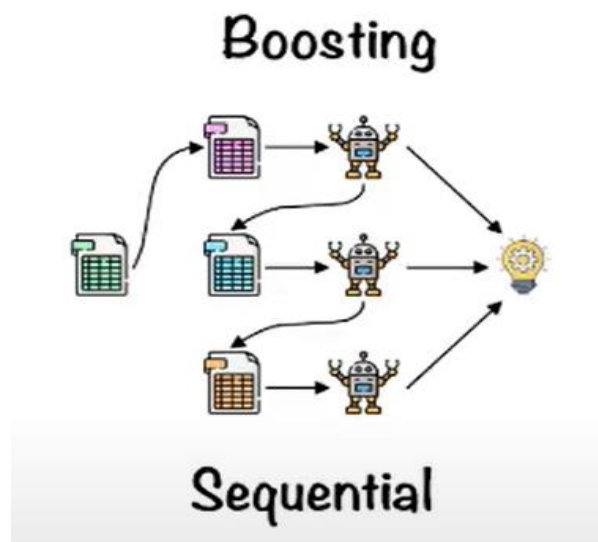
# Inicializar e treinar o modelo Random Forest
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Fazer previsões
predictions = rf_model.predict(X_test)

# Avaliar precisão
accuracy = accuracy_score(y_test, predictions)
print("Accuracy:", accuracy)
```

1.3. Boosting

Boosting é uma técnica de ensemble que reúne diversos modelos denominados “weak learners” para criar um modelo forte. Concentra-se em reduzir o viés, ajustando iterativamente os modelos fracos para dar peso aos exemplos classificados incorretamente pelo modelo anterior.



Como funciona:

Treinamento Progressivo:

Um modelo inicial é treinado em todo o conjunto de dados.

Erros do modelo são identificados e ponderados.

Novos modelos são treinados com ênfase nos erros anteriores.

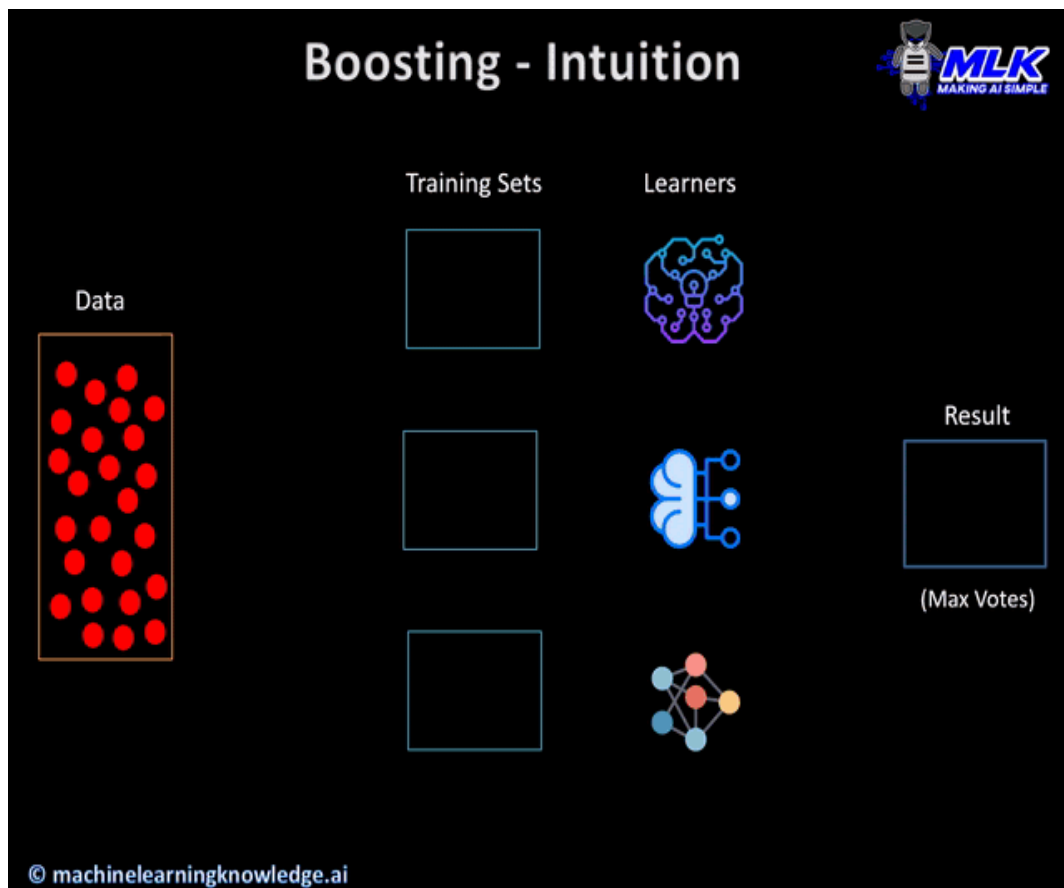
O processo se repete até um número definido de modelos.

Combinação de Previsões:

As previsões de todos os modelos individuais são combinadas.

O objetivo é alcançar maior precisão geral.

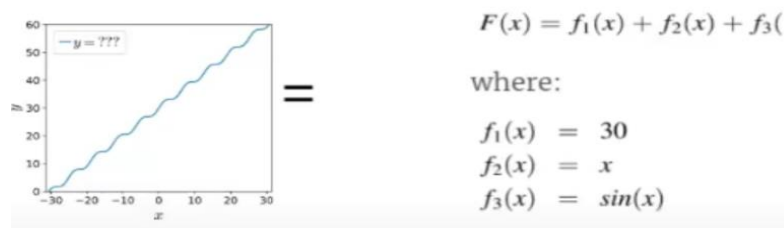
Diferentes métodos de combinação podem ser utilizados.



Resultado: Combina as previsões de todos os modelos para fazer uma previsão final.

Matematicamente falando:

Um modelo em machine learning é, essencialmente, uma tentativa de descobrir a função por trás dos dados. No método boosting, ele combina diferentes funções para encontrar a "função forte".



Como ele atua na árvore de decisão:

Boosting cria um modelo de conjunto combinando várias árvores de decisão fracas sequencialmente. Ele atribui pesos à saída de árvores individuais. Em seguida, dá um peso maior às classificações incorretas da primeira árvore de decisão e uma entrada para a próxima árvore. Após vários ciclos, o método de boosting combina essas regras fracas em uma única regra de previsão poderosa.

Como é feito o treinamento em boosting?Etapa 1

O algoritmo de boosting atribui peso igual a cada amostra de dados. Ele alimenta os dados para o primeiro modelo de máquina, chamado de algoritmo base. O algoritmo base faz previsões para cada amostra de dados.

Etapa 2

O algoritmo de boosting avalia as previsões do modelo e aumenta o peso das amostras com um erro mais significativo. Também atribui um peso com base na performance do modelo. Um modelo que produz previsões excelentes terá uma grande influência sobre a decisão final.

Etapa 3

O algoritmo passa os dados ponderados para a próxima árvore de decisão.

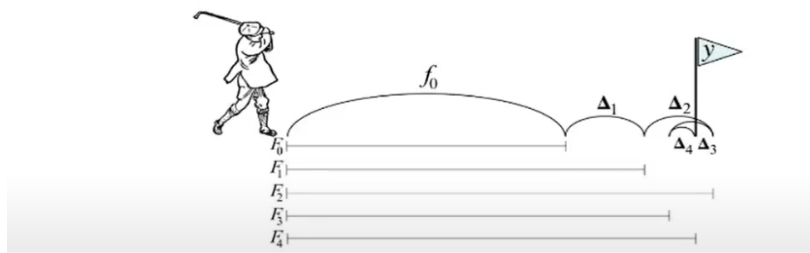
Etapa 4

O algoritmo repete as etapas 2 e 3 até que as instâncias de erros de treinamento estejam abaixo de um determinado limite.

Caso de uso

Se um modelo de identificação de gato foi treinado apenas com imagens de gatos brancos, ele pode ocasionalmente identificar erroneamente um gato preto. O boosting tenta superar esse problema treinando vários modelos sequencialmente para melhorar a precisão geral do sistema.

A cada passo nosso modelo aprender a ir um pouco melhor



Principais Características:

- **Treinamento Iterativo:** O Boosting treina uma sequência de modelos fracos, onde cada novo modelo foca nos erros dos modelos anteriores.
- **Ponderação dos Exemplos:** Exemplos mal classificados recebem maior peso durante o treinamento de cada novo modelo, o que ajuda a corrigir os erros cometidos pelos modelos anteriores.
- **Combinação Ponderada:** As previsões finais são feitas por meio de uma combinação ponderada das previsões de todos os modelos fracos.

Vantagens do Boosting:

- Pode alcançar um desempenho preditivo superior a outros métodos de ensemble e modelos individuais.
- Reduz a variância e o viés do modelo, resultando em previsões mais robustas.

Desvantagem do Boosting:

- **Overfitting:** Combinação de diversos modelos fracos pode levar a um ajuste excessivo aos dados de treinamento.
- **Interpretabilidade Limitada:** Dificuldade em entender as decisões finais do algoritmo.

Algoritmos de Boosting Populares:

- AdaBoost (Adaptive Boosting).
- Gradient Boosting Machines (GBM).
- XGBoost (Extreme Gradient Boosting)

Aqui está um exemplo de como usar o algoritmo GradientBoostingClassifier, que é uma implementação do algoritmo de boosting Gradient Boosting:

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Gerar dados de exemplo
X, y = make_classification(n_samples=1000, n_features=20, random_state=42)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Inicializar e treinar o modelo Gradient Boosting
gb_model = GradientBoostingClassifier(n_estimators=100, random_state=42)
gb_model.fit(X_train, y_train)

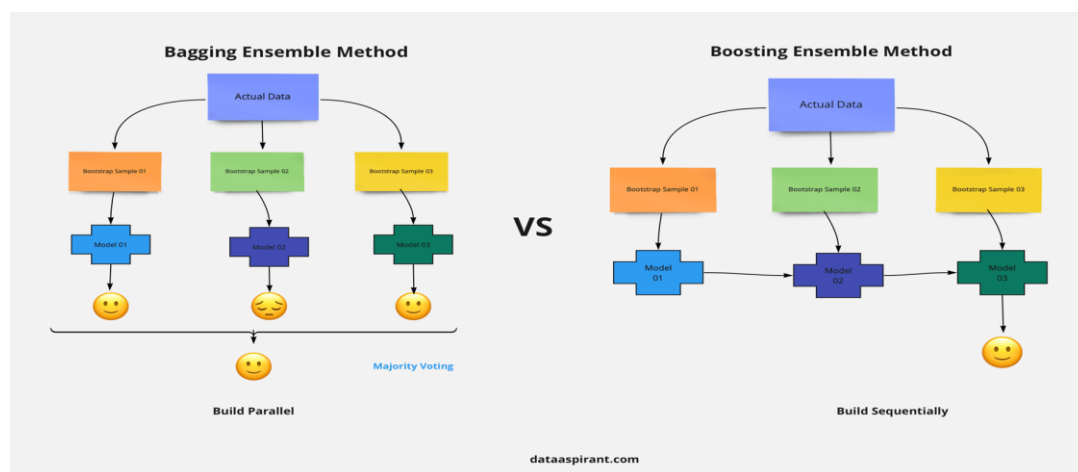
# Fazer previsões
predictions = gb_model.predict(X_test)

# Avaliar precisão
accuracy = accuracy_score(y_test, predictions)
print("Accuracy:", accuracy)
```

1.4. Bagging vs. Boosting

A principal diferença entre bagging e boosting reside na maneira como os modelos são combinados. Enquanto o bagging combina modelos independentes por meio de média ou votação, o boosting constrói uma sequência de modelos onde cada um tenta corrigir os erros dos modelos anteriores.

A diferença entre bagging e boosting é principalmente na maneira como os modelos são treinados e combinados. No exemplo acima, RandomForestClassifier usa bagging enquanto GradientBoostingClassifier usa boosting.



2. Random Forest

Random Forest é um algoritmo de aprendizado de máquina baseado em ensemble que utiliza bagging como técnica de combinação. Ele consiste em uma coleção de árvores de decisão treinadas em subconjuntos aleatórios do conjunto de dados original. As previsões de cada árvore são então agregadas para produzir uma previsão final.

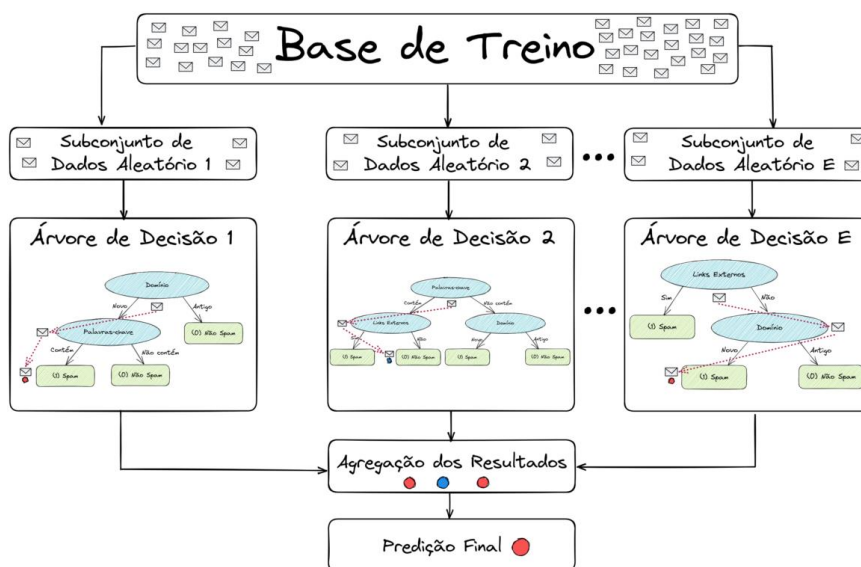
Machine Learning, 24, 123–140 (1996)
© 1996 Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

Bagging Predictors

LEO BREIMAN
Statistics Department, University of California, Berkeley, CA 94720

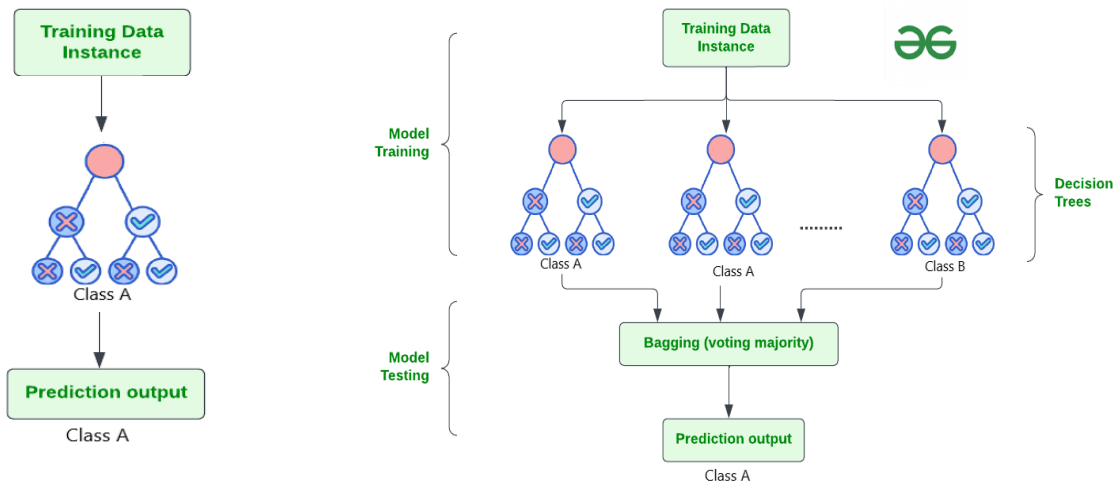
leo@stat.berkeley.edu

Paper proposto por Leo Breiman e Adele Cutler: extensão da árvore de decisão



Melhor ponto de corte segundo entropia ou Gini

- Exército/Conjunto de árvores de decisão (floresta): analogia com a opinião de vários especialistas para chegar a uma conclusão mais acertada por votação (tomada de decisão colaborativa)



Construída a partir de várias árvores de decisão ($n_estimators$), cada uma treinada com uma amostra aleatória do conjunto de dados (bootstrapping). Durante a construção de cada árvore, em cada nó, é considerado apenas um subconjunto aleatório de features (aumento da diversidade/variabilidade)

- **Pontos positivos:**

- Reduz overfitting
- Performa bem com dados de alta dimensionalidade
- Paralelização
- Regressão ou classificação
- Lida bem com dados desbalanceados
- Checagem de Feature importance
- Lida com valores faltantes

- **Pontos negativos:**

- Menor Interpretabilidade
- Pode ser computacionalmente caro



Clique na Figura para uma Aula do Prof. Nicolas Spogis sobre Random Forest!

I2A2 Public Pin Unwatch

master 1 Branch 0 Tags Add file Code

Spogis Add Challenge 3 037e8cc · last week 15 Commits

Challenge 1	Add Challenge 2	2 months ago
Challenge 2	Add Challenge 3	last week
Challenge 3	Add Challenge 3	last week
.gitignore	Add Challenge 2	2 months ago
LICENSE	Create LICENSE	3 months ago
README.md	First Commit	3 months ago
requirements.txt	Add Challenge 2	2 months ago

Acesse no GitHub um Código Exemplo (Jupyter Notebook) usando o Random Forest!

<https://github.com/Spogis/I2A2/blob/master/Challenge%203/00%20-%20Random%20Forest%20Regression.ipynb>

3. Extra trees (Extreme randomized trees)

Extra Trees, ou Extremely Randomized Trees, é uma variação do algoritmo Random Forest. Assim como o Random Forest, ele treina várias árvores de decisão em subconjuntos aleatórios do conjunto de dados. No entanto, ao contrário do Random Forest, o Extra Trees toma decisões completamente aleatórias nos pontos de divisão durante a construção das árvores, o que pode levar a uma maior diversidade entre as árvores e, em alguns casos, a um melhor desempenho preditivo.

Mach Learn () :
DOI 10.1007/s10994-006-6226-1

Extremely randomized trees

Pierre Geurts · Damien Ernst · Louis Wehenkel

Received: 14 June 2005 / Revised: 29 October 2005 / Accepted: 15 November 2005 /
Published online: 2 March 2006
Springer Science + Business Media, Inc. 2006

Variação de random forest proposta por Pierre Geurts, Damien Ernst e Louis Wehenkel

- Camada adicional de aleatoriedade por isso o nome “extreme”: amostra aleatória (algumas fontes citam que o conjunto de dados completo é utilizado para construção das árvores), features aleatórias e divisão aleatória (random forest = best_split, extra trees = random_split)
- Mais rápida na construção comparada a random forest por não buscar melhor ponto de divisão - pode resultar em maior variância sem afetar significativamente a precisão
- Combinação de classificadores fracos (baixa acurácia) podem resultar em boa acurácia/performance


- **Pontos positivos:**


- Menor tempo de treinamento (comparada a random forest, pelo split aleatório)
- Modelo menos enviesado
- Reduz overfitting

- **Pontos negativos:**


- Menor interpretabilidade
- Pode ter performance inferior a random forest
- Variáveis não relevantes podem influenciar na má performance do modelo (pela escolha aleatória).
- Necessário atenção ao pré-processamento para selecionar variáveis mais relevantes

	Decision Tree	Random Forest	Extra Trees
Number of trees	1	Many	Many
No of features considered for split at each decision node	All Features	Random subset of Features	Random subset of Features
Boostrapping(Drawing Sampling without replacement)	Not applied	Yes	No
How split is made	Best Split	Best Split	Random Split






 **MÓDULO 3**
MODELAGEM




Usando o Scikit-Learn
Extremely Randomized Trees Regression














Clique na Figura para uma Aula do Prof. Nicolas Spogis sobre Extra Trees!


I2A2
Public

 Pin
  Unwatch 1

 master
  1 Branch
  0 Tags

 Add file
  Code

 Spogis Add Challenge 3	037e8cc · last week	 15 Commits
 Challenge 1	Add Challenge 2	2 months ago
 Challenge 2	Add Challenge 3	last week
 Challenge 3	Add Challenge 3	last week
 .gitignore	Add Challenge 2	2 months ago
 LICENSE	Create LICENSE	3 months ago
 README.md	First Commit	3 months ago
 requirements.txt	Add Challenge 2	2 months ago

Acesse no GitHub um Código Exemplo (Jupyter Notebook) usando o Extra Trees!

<https://github.com/Spogis/I2A2/blob/master/Challenge%203/01%20-%20Extremely%20Randomized%20Trees%20Regression.ipynb>

Aqui está um exemplo de como usar o algoritmo `ExtraTreesClassifier`, que é uma implementação do algoritmo Extra Trees:

```
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Gerar dados de exemplo
X, y = make_classification(n_samples=1000, n_features=20, random_state=42)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Inicializar e treinar o modelo Extra Trees
et_model = ExtraTreesClassifier(n_estimators=100, random_state=42)
et_model.fit(X_train, y_train)

# Fazer previsões
predictions = et_model.predict(X_test)

# Avaliar precisão
accuracy = accuracy_score(y_test, predictions)
print("Accuracy:", accuracy)
```

Vamos criar um projeto teórico de classificação de dados usando Python e a biblioteca Scikit-learn. Neste projeto, usaremos o conjunto de dados "Give Me Some Credit", que contém informações sobre clientes de um banco e se eles pagaram ou não seus empréstimos.

#Importando as bibliotecas necessárias:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier,
ExtraTreesClassifier
from sklearn.metrics import accuracy_score, classification_report
```

#Carregando e explorando os dados:

```
# Carregar os dados
data = pd.read_csv('GiveMeSomeCredit.csv')

# Visualizar as primeiras linhas dos dados
print(data.head())

# Verificar informações sobre os dados
print(data.info())

# Verificar estatísticas descritivas
print(data.describe())
```

#Preparando os dados:

Remover valores ausentes

```
data.dropna(inplace=True)
```

Dividir os dados em features e target

```
X = data.drop('SeriousDlqin2yrs', axis=1)
```

```
y = data['SeriousDlqin2yrs']
```

Dividir os dados em conjuntos de treinamento e teste

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,  
random_state=42)
```

#Construindo e treinando os modelos:

Inicializar os modelos

```
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
```

```
gb_model = GradientBoostingClassifier(n_estimators=100, random_state=42)
```

```
et_model = ExtraTreesClassifier(n_estimators=100, random_state=42)
```

Treinar os modelos

```
rf_model.fit(X_train, y_train)
```

```
gb_model.fit(X_train, y_train)
```

```
et_model.fit(X_train, y_train)
```

#Fazendo previsões e avaliando os modelos:

Fazer previsões

```
rf_predictions = rf_model.predict(X_test)
```

```
gb_predictions = gb_model.predict(X_test)
```

```
et_predictions = et_model.predict(X_test)
```

Avaliar precisão dos modelos

```
rf_accuracy = accuracy_score(y_test, rf_predictions)
```

```
gb_accuracy = accuracy_score(y_test, gb_predictions)
et_accuracy = accuracy_score(y_test, et_predictions)
```

```
print("Random Forest Accuracy:", rf_accuracy)
print("Gradient Boosting Accuracy:", gb_accuracy)
print("Extra Trees Accuracy:", et_accuracy)
```

```
# Relatório de classificação
print("Random Forest Classification Report:")
print(classification_report(y_test, rf_predictions))
```

```
print("Gradient Boosting Classification Report:")
print(classification_report(y_test, gb_predictions))
```

```
print("Extra Trees Classification Report:")
print(classification_report(y_test, et_predictions))
```

4. Referências

- <https://towardsdatascience.com/what-when-how-extratrees-classifier-c939f905851c>
- https://stats.stackexchange.com/questions/175523/difference-between-random-forest-and-extremely-randomized-trees?source=post_page-----c939f905851c-----
- <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>
- <https://didatica.tech/o-que-e-e-como-funciona-o-algoritmo-randomforest/>
- <https://www.datacamp.com/tutorial/random-forests-classifier-python>
- <https://www.geeksforgeeks.org/random-forest-algorithm-in-machine-learning/>
- <https://didatica.tech/como-funciona-o-algoritmo-extratrees/>
- <https://www.geeksforgeeks.org/random-forest-algorithm-in-machine-learning/>
- <https://www.ibm.com/br-pt/topics/random-forest>
- <https://aws.amazon.com/pt/what-is/boosting/>
- <https://www.youtube.com/watch?v=dwxvPurdgpk>