

# football - ML

## Sistema que prevê resultados do Brasileirão 2023

**"Como um computador pode prever quem vai ganhar um jogo de futebol?"**

### O QUE O SISTEMA FAZ?

Imagine que você quer saber quem vai ganhar entre Flamengo x São Paulo. O que você faria?

1. Veria como os times estão jogando
2. Quantos gols estão fazendo
3. Se estão ganhando ou perdendo
4. Quem está jogando em casa

Nosso sistema faz exatamente isso, mas usando matemática e computação!

### COMO ELE FAZ ISSO?






#### 1. COLETA DE DADOS

É como se o sistema fosse um super jornalista que:

- Sabe todos os resultados dos jogos
- Acompanha todos os times
- Atualiza os dados a cada jogo
- Guarda tudo organizado

#### 2. ANÁLISE DOS TIMES

Para cada time, o sistema estuda:

Time: São Paulo  
- Últimos 5 jogos:     

- Gols marcados: 🏆🏆🏆 (média de 3 por jogo)
- Gols sofridos: 🏆 (média de 1 por jogo)
- Posição na tabela: 4º lugar

# O Cérebro do Sistema: Entendendo o Random Forest

## Por que Random Forest?

Imagine uma reunião de 200 especialistas em futebol, onde:

- Cada um tem sua própria experiência
- Cada um olha o jogo de um jeito diferente
- A decisão final é tomada pela maioria

Isso é o Random Forest! Uma "floresta" de 200 árvores de decisão.

## Como uma Árvore de Decisão Funciona?

É como um fluxograma de perguntas:

```
Time fez mais de 2 gols nos últimos jogos?  
├─ SIM: Time está invicto em casa?  
│   └─ SIM: Provável VITÓRIA (80% de chance)  
│   └─ NÃO: Verificar forma recente  
└─ NÃO: Time está em má fase?  
    └─ SIM: Provável DERROTA (70% de chance)  
    └─ NÃO: Possível EMPATE (60% de chance)
```

## Os Hiperparâmetros (Configurações do Modelo)

### 1. `n_estimators = 500`

O que é? Número de árvores de decisão

- Por que 500?
  - Mais árvores = mais "opiniões"

- Aumenta a precisão do modelo
- Melhor que 200, pois temos dados suficientes
- **Impacto:** Decisões mais robustas e confiáveis

## 2. **max\_depth = 10**

O que é? Quantidade máxima de "perguntas" em sequência

- **Por que 10?**
  - Permite análises mais profundas que 8
  - Captura padrões mais complexos
  - Ainda evita overfitting (decorar demais)
- **Impacto:** Análises mais detalhadas de cada situação

## 3. **min\_samples\_split = 4**

O que é? Número mínimo de amostras para dividir um nó

- **Por que 4?**
  - Garante decisões baseadas em dados suficientes
  - Evita divisões com poucos exemplos
  - Balanceia generalização e precisão
- **Impacto:** Decisões mais estáveis e confiáveis

## 4. **min\_samples\_leaf = 2**

O que é? Número mínimo de amostras em cada nó folha

- **Por que 2?**
  - Cada previsão baseada em pelo menos 2 casos
  - Evita conclusões de casos isolados
  - Mantém o modelo mais robusto
- **Impacto:** Previsões mais consistentes

## 5. **max\_features = 'sqrt'**

**O que é?** Número de features consideradas em cada divisão

- **Por que 'sqrt'?**
  - Usa raiz quadrada do total de features
  - Reduz correlação entre árvores
  - Aumenta a diversidade de análises
- **Impacto:** Melhor generalização do modelo

## 6. class\_weight Personalizado

```
class_weight = {  
    0: 1.0, # Vitória fora  
    1: 1.5, # Empate  
    2: 1.0 # Vitória casa  
}
```

**O que é?** Peso diferente para cada resultado

- **Por que estes valores?**
  - Empates são mais raros
  - Peso 1.5 para empates compensa isso
  - Vitórias casa/fora peso normal (1.0)
- **Impacto:** Modelo mais atento a possibilidade de empates

## 7. n\_jobs = -1

**O que é?** Uso de processamento paralelo

- **Por que -1?**
  - Usa todos os núcleos do processador
  - Acelera o treinamento
  - Melhor aproveitamento do hardware
- **Impacto:** Treinamento mais rápido

## 8. random\_state = 42

**O que é?** Semente para reproducibilidade

- **Por que 42?**
  - Garante mesmos resultados sempre
  - Valor padrão na comunidade
  - Permite comparações justas
- **Impacto:** Resultados consistentes e reproduzíveis

## Como o Modelo Aprende?

### 1. Fase de Treino:

```
# Para cada jogo no histórico:
jogo = {
    'gols_recentes_casa': 2.5, # Média de gols do mandante
    'aproveitamento_casa': 70%, # % de pontos em casa
    'gols_recentes_fora': 1.2, # Média de gols do visitante
    'aproveitamento_fora': 40%, # % de pontos fora
    'resultado_real': 'Vitória Casa'
}

# O modelo aprende padrões:
"Times que fazem +2 gols/jogo e têm +60% em casa costumam v
encer"
```

### 1. Fase de Previsão:

```
# Novo jogo:
Flamengo x São Paulo = {
    'gols_recentes_casa': 2.8,
    'aproveitamento_casa': 75%,
    'gols_recentes_fora': 1.5,
    'aproveitamento_fora': 45%
}

# Cada árvore dá seu "voto":
Árvore 1: Vitória Casa (65% confiante)
Árvore 2: Vitória Casa (70% confiante)
```

Árvore 3: Empate (55% confiante)

...até Árvore 200

# Resultado Final = Média ponderada dos votos

## O QUE O SISTEMA MOSTRA?

### 1. Classificação

- Tabela do campeonato
- Pontos de cada time
- Gráficos fáceis de entender

### 2. Previsões

Você escolhe dois times e o sistema mostra:

Flamengo x São Paulo

Chances de:

- Flamengo ganhar: 45%
- Empate: 30%
- São Paulo ganhar: 25%

Por quê?

- Flamengo: Vem de 3 vitórias
- São Paulo: Não ganha fora há 2 jogos
- Histórico recente: Flamengo ganhou último jogo

### 3. Análise de Times

Mostra tudo sobre um time:

- Como está jogando
- Quantos gols faz
- Se está melhorando ou piorando

## RESULTADOS

### Quanto o sistema acerta?

- Chute aleatório: 33% (como jogar um dado de 3 lados)
- Nosso sistema: 55% (melhor que o chute!)

### Por que não acerta mais?

- Futebol é imprevisível
- Time favorito nem sempre ganha
- Muitas coisas podem acontecer em um jogo

## Por que Este Modelo é Bom?

### 1. Vantagens:

- Considera vários aspectos do jogo
- Não se baseia em um único fator
- Aprende padrões complexos
- Fácil de entender as decisões

### 1. Limitações:

- Precisa de bastante dados
- Pode ser lento com muitas árvores
- Ocupa mais memória que modelos simples

## CONCLUSÃO

É uma ferramenta que:

- Ajuda a entender melhor o futebol
- Usa matemática de forma prática
- Mostra informações interessantes
- Mas não substitui a emoção do jogo!