Algoritmo Q-*Learning* na Aprendizagem por Reforço

Adaptado por: Dr. Arnaldo de Carvalho Junior – Março 2026

1. INTRODUÇÃO

Random Forest (RF), ou Floresta Aleatória, é um dos algoritmos de aprendizado de máquina (*machine learning* – ML) mais populares e versáteis. Ele é amplamente utilizado para tarefas de ***classificação*** e ***regressão***, devido à sua capacidade de lidar com grandes volumes de dados, evitar overfitting e fornecer resultados robustos. Este tutorial, apresenta um exemplo de RF do zero em Python, utilizando a biblioteca *scikit-learn*, compreendendo sua teoria e implementando suas principais funções passo a passo.

2. ALGORITMO DE RANDOM FOREST

O RF é um método de aprendizagem em conjunto (*ensemble learning*) que combina várias árvores de decisão para criar um modelo mais robusto e preciso. O princípio fundamental é treinar múltiplas árvores de decisão em subconjuntos aleatórios dos dados e, em seguida, combinar suas previsões para obter um resultado final. Esse processo ajuda a reduzir a variância e a melhorar a generalização do modelo. A Figura 1 apresenta este conceito. Os principais conceitos por trás do algoritmo incluem:

1. **Bagging (Bootstrap Aggregating):** Cada árvore é treinada com um subconjunto aleatório dos dados, obtido via amostragem com reposição.
2. **Seleção Aleatória de Features:** Cada árvore recebe apenas um subconjunto das features para tomar suas decisões, reduzindo correlações entre as árvores.
3. **Combinação de Previsões:** No caso de classificação, a previsão final é feita por maioria de votos entre as árvores; no caso de regressão, é usada a média das previsões.

Essa abordagem ajuda a reduzir o sobreajuste (*overfitting*) e melhora a acurácia do modelo.

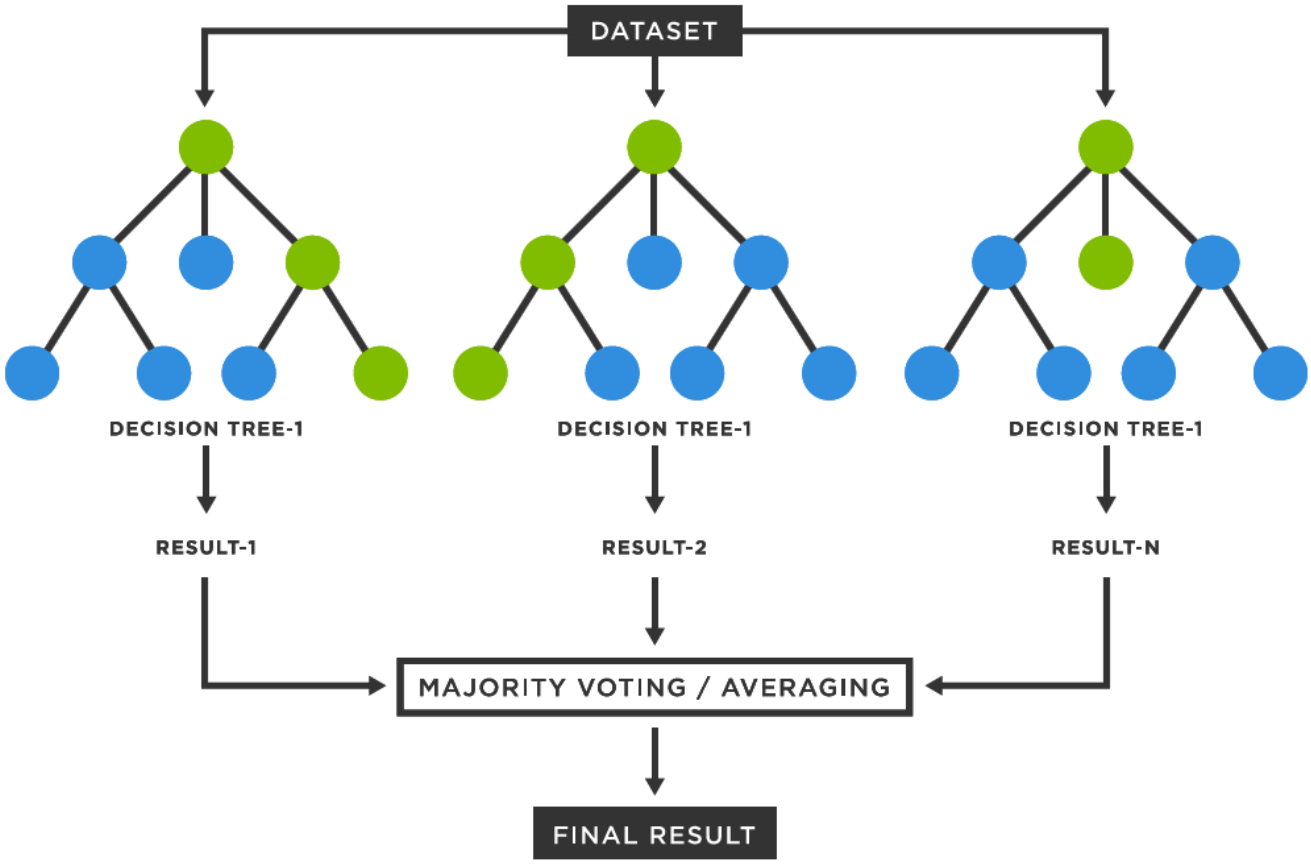

Figura 1 – Algoritmo de RF.

3. PASSOS PARA IMPLEMENTAR O RANDOM FOREST

1. **Criar uma Árvore de Decisão**: Começa por implementar uma árvore de decisão básica. Cada árvore será treinada em um subconjunto aleatório dos dados.
2. **Amostragem Aleatória (*Bootstrap*)**: Para cada árvore na floresta, seleciona-se um subconjunto aleatório dos dados de treinamento com reposição (*bootstrap*).
3. **Seleção Aleatória de Features**: Em cada divisão da árvore, seleciona-se um subconjunto aleatório de features para considerar a divisão. Isso ajuda a garantir que as árvores sejam diferentes umas das outras.

4. **Construir a Floresta**: Repete-se o processo de construção de árvores várias vezes para criar a “floresta”.
5. **Combinação de Previsões**: Para classificação, utiliza-se a moda das previsões das árvores. Para regressão, utiliza-se a média.

O Apêndice A apresenta um exemplo de RF para um banco de dados gerado aleatoriamente de altura, peso, idade e gênero, utilizando a biblioteca scikit-learn em Python. Este arquivo pode ser executado diretamente no Google Drive clicando aqui.

4. CONCLUSÃO

As principais vantagens do RF incluem a possibilidade de redução do *overfitting*, trabalhar com dados complexos, medição da importância das variáveis, e menor influência por valores discrepantes e dados contaminados por ruido. A desvantagem é a redução da interpretabilidade dos resultados, já que as previsões são baseadas na votação das árvores criadas. A biblioteca Scikit-Learn oferece implementações otimizadas, o que facilita a criação de algoritmos de RF.

REFERÊNCIAS

KUMAR, Rajender. Python Machine Learning: A Beginner's Guide to Scikit-Learn. Jamba Academy, 2023.

LUCAS, Hudson Barroso, Machine Learning from Scratch: Implementando Random Forest (Floresta Aleatória) em Python, IA com Café, 2025. Disponível em: https://iacomcafe.com.br/implementando-random-forest-em-python/. Acesso em: dez 12, 2025.

CARVALHO JUNIOR, A. COLETÂNEA DE DICAS, ORIENTAÇÕES E REFERÊNCIAS EM INTELIGÊNCIA ARTIFICIAL. EAILAB, IFSP, 7º ed, 233p, 2025. Disponível em: https://github.com/EAILAB-IFSP/PUBLICATIONS/blob/Publications/AI%20Cards%20Collection%20-%20EAILab%20v7%202025.pdf. Acessado em Set 19, 2025.

APÊNDICE A

```
# Exemplo de Algoritmo Random Forest em Python
# Usando biblioteca scikit-learn
# Referência - Book: KUMAR, Rajender. Python Machine Learning: A Beginner's
# Guide to Scikit-Learn. Jamba Academy, 2023.
# Adaptado por: Dr. Arnaldo de Carvalho Junior

# Importe as bibliotecas
import numpy as np
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt

# Definindo as sementes (seed) para reprodutibilidade
np.random.seed(42)

# Gerando um conjunto de dados aleatórios (random dataset)
# De altura, peso, idade e gênero
# distribuição normal, loc = media e scale = desvio padrão
height = np.random.normal(loc=170, scale=5, size=1000)
weight = np.random.normal(loc=70, scale=10, size=1000)
age = np.random.normal(loc=30, scale=10, size=1000)
gender = np.random.randint(low=0, high=2, size=1000)

# Criando um DataFrame para armazenar o conjunto de dados
# altura (height), peso (weight), idade (age), gênero (gender)
df = pd.DataFrame({
 'height': height,
 'weight': weight,
 'age': age,
 'gender': gender
})

# Definindo as variáveis dependentes e independentes
X = df[['height', 'weight', 'age']]
y = df['gender']

# Dividindo o conjunto de dados em treinamento (train) e teste (test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Instanciando um classificador Random Forest com 100 árvores (trees)
```

```
# *** Experimente outro número de estimadores ***
rfc = RandomForestClassifier(n_estimators=100)

# Ajustando o modelo aos dados de treinamento
rfc.fit(X_train, y_train)

# Usando o modelo para fazer previsões sobre os dados de teste
y_pred = rfc.predict(X_test)

# Avaliando a acurácia do modelo
# *** Experimente outros valores de seed, loc e scale e veja os resultados ***
accuracy = accuracy_score(y_test, y_pred)
print("Acurácia:", accuracy)

# Criando uma matriz de confusão para visualizar a performance do modelo
cm = confusion_matrix(y_test, y_pred)

# Imprimindo a Matriz de Confusão
print(cm)
plt.imshow(cm, cmap=plt.cm.Blues)
plt.colorbar()
plt.xticks([0, 1])
plt.yticks([0, 1])
plt.xlabel('Rótulo Previsto')
plt.ylabel('Rótulo Verdadeiro')
plt.title('Matriz de Confusão')
plt.show()
```