Ensemble de Algoritmos – Random Forest

ROCKETSEAT

Plataforma completa de aprendizado contínuo em programação.

#BoostingPeople

rocketseat.com.br

Ensemble de Algoritmos

Random Forest

O objetivo deste módulo é apresentar o ensemble Random Forest que pertence a classe de ensembles Bagging e trabalharemos num projeto para uma empresa de tecnologia que deseja prever a probabilidade de churn de seus funcionários, além de entender quais os principais motivos que levam a este churn. Neste projeto faremos o processo completo desde o EDA até a visualização dos resultados, incluindo métricas de validação e importância das variáveis.

Agenda

- O que é Random Forest
- Etapas do Random Forest
- Desafios e Limitações do Random Forest
- Aplicações do Random Forest
- Principais Hiperparâmetros
- Como prevenir o Overfitting em Random Forest
- A métrica Log-Loss
- Projeto – Random Forest

O que é Random Forest

Random Forest é um algoritmo de aprendizado de máquina do tipo ensemble que combina várias árvores de decisão para melhorar a precisão e a robustez das previsões.

Foi desenvolvido por Leo Breiman (um estatístico americano) em 2001 e utiliza o método de bagging (bootstrap aggregating), no qual múltiplas amostras do conjunto de dados são criadas, e cada árvore é treinada com uma dessas amostras.

Etapas do Random Forest

1) Criação de amostras (bootstrap)

Em vez de usar todo o conjunto de dados de treino para construir cada árvore, uma amostra aleatória (com reposição) é selecionada para treinar cada árvore. Isso significa que algumas observações podem ser repetidas enquanto outras podem não ser selecionadas.

2) Construção de árvores de decisão:

Para cada nó de uma árvore, é selecionado um subconjunto aleatório das variáveis (ou features) em vez de todas as variáveis disponíveis. A árvore é então construída como uma árvore de decisão tradicional.

Etapas do Random Forest

3) Crescimento das árvores

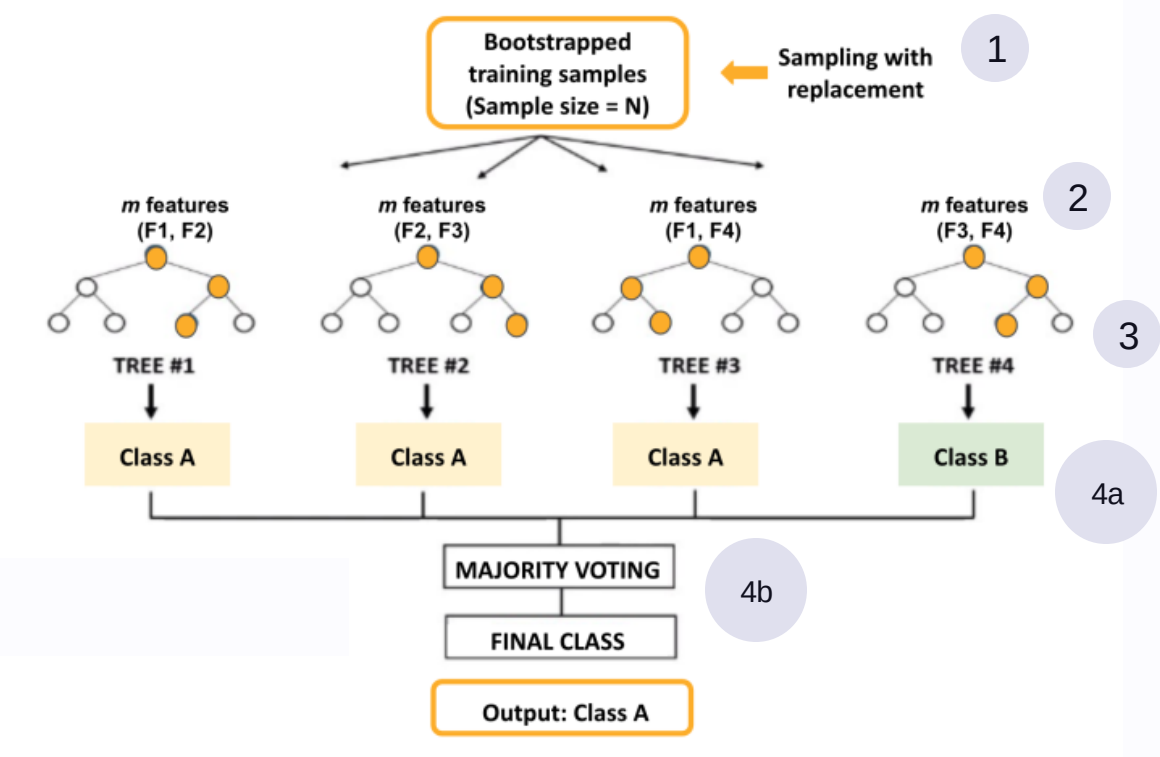
Cada árvore é treinada completamente, sem poda, até que todas as folhas sejam puras ou até um critério de parada ser atingido (como profundidade máxima, número mínimo de amostras por folha, etc.)

4) Previsão

Para problemas de classificação: cada árvore faz uma previsão e a classe final é decidida pela votação majoritária das árvores.

Para problemas de regressão: as previsões de todas as árvores são combinadas por média.

Etapas do Random Forest



Desafios e Limitações do Random Forest

Overfitting: Embora o Random Forest seja naturalmente resistente ao overfitting, pode ocorrer em dados muito complexos ou com poucas amostras.

Custo computacional: Treinar um grande número de árvores aumenta o tempo de processamento, especialmente com muitos hiperparâmetros para ajustar.

Interpretação: Os modelos de Random Forest são frequentemente criticados por sua falta de interpretabilidade comparados às árvores de decisão individuais.

Aplicações do Random Forest

Diagnóstico médico
Classificação de doenças ou condições com base em sintomas ou dados médicos (como ECG ou exames laboratoriais)

Análise de risco financeiro
Previsão de crédito ou detecção de fraudes bancárias.

Análise de churn
Identificação de clientes com maior probabilidade de deixar um serviço.

Modelos preditivos de dados tabulares
Problemas com dados estruturados, onde a relação entre variáveis e rótulos de classe é complexa.

Aplicações do Random Forest

Vantagens para estes cenários:

- Capaz de lidar bem com dados que possuem ruído e outliers.
- Eficiente mesmo com um grande número de variáveis explicativas.
- Requer menos pré-processamento de dados comparado a outros algoritmos.

Principais Hiperparâmetros

n_estimators

- Define o número de árvores no modelo. Um valor maior tende a melhorar o desempenho até certo ponto, mas aumenta o custo computacional.
- A principal função é de controlar o tamanho do ensemble
- Valor inicial sugerido: 100
- Justificativa: Um valor de 100 árvores costuma ser suficiente para a maioria dos problemas, garantindo uma boa performance sem custo computacional excessivo. Você pode aumentar esse valor se houver sinais de underfitting, mas valores muito altos podem aumentar o tempo de treinamento sem grandes ganhos.

Principais Hiperparâmetros

max_depth

- Limita a profundidade máxima de cada árvore. Árvores profundas podem modelar padrões complexos, mas correm o risco de overfitting.
- A principal função é de controlar o grau de complexidade das árvores.
- Valor inicial sugerido: None
- Justificativa: Deixar o parâmetro como None inicialmente ajuda a explorar a capacidade máxima das árvores. Caso o modelo demonstre overfitting, pode ser interessante limitar a profundidade (por exemplo, para 10 ou 20).

Principais Hiperparâmetros

min_samples_split

- O número mínimo de amostras necessárias para dividir um nó. Valores maiores resultam em árvores mais rasas.
- A principal função é de controlar o tamanho dos nós.
- Valor inicial sugerido: 2
- Justificativa: Este valor garante que as árvores possam explorar divisões em nós desde o início. Se houver sinais de overfitting, aumentar este valor (por exemplo, para 10 ou 20) pode ajudar a controlar a complexidade.

Principais Hiperparâmetros

min_samples_leaf

- O número mínimo de amostras necessárias para estar em uma folha. Ele afeta diretamente o tamanho das árvores.
- A principal função é de prevenir a criação de folhas muito pequenas.
- Valor inicial sugerido: 1 (Classificação) e 5 (Regressão)
- Justificativa: Um valor de 1 permite que as árvores cresçam ao máximo detalhamento, mas pode levar ao overfitting. Se o modelo parecer estar sobreajustado, aumentar para valores como 5 ou 10 pode ser útil.

Principais Hiperparâmetros

max_features

- O número máximo de variáveis consideradas para dividir um nó. Um valor menor resulta em árvores mais diversificadas e pode reduzir o overfitting.
- A principal função é de controlar a aleatoriedade na escolha das features.
- Valor inicial sugerido: sqrt (para problemas de classificação) ou auto (para problemas de regressão)
- Justificativa: A raiz quadrada do número total de características (sqrt) para classificação e todas as características (auto) para regressão são escolhas comuns e eficientes. Eles adicionam diversidade às árvores, ajudando a evitar overfitting.

Principais Hiperparâmetros

bootstrap

- Se verdadeiro, utiliza o método de amostragem com reposição. Caso contrário, usa toda a amostra de treino para cada árvore.
- A principal função é de controlar a forma de amostragem.
- Valor inicial sugerido: True
- Justificativa: Usar amostras bootstrap (True) é a configuração padrão do Random Forest, e normalmente ajuda a criar árvores menos correlacionadas entre si, o que melhora a generalização do modelo.

Principais Hiperparâmetros

oob_score

- Se deve usar amostras out-of-bag para estimar a precisão do modelo.
- Out-of-bag (OOB) é uma técnica de validação usada em modelos de ensemble, em que as amostras não selecionadas para treinar um estimador específico são usadas para avaliar o desempenho do modelo. Isso permite uma estimativa da performance sem a necessidade de validação cruzada explícita.
- A principal função é de fornecer uma estimativa imparcial do desempenho do modelo.

Principais Hiperparâmetros

oob_score

- Valor inicial sugerido: False
- Justificativa: Inicialmente, você pode não habilitar o oob_score, mas se quiser estimar a performance fora da amostra sem usar validação cruzada, ativar o True pode ser útil para modelos mais complexos

Como prevenir o Overfitting em Random Forest

Limitar a profundidade das árvores (max_depth): Árvores profundas tendem a superajustar aos dados de treino.

Aumentar o número mínimo de amostras para divisão (min_samples_split): Previne divisões muito específicas.

Definir um número mínimo de amostras por folha (min_samples_leaf): Ajuda a evitar divisões que criam folhas com poucos dados.

Como prevenir o Overfitting em Random Forest

Reduzir o número de características para cada divisão (max_features): Menos características aumentam a diversidade entre as árvores, reduzindo o overfitting.

Usar a validação out-of-bag (OOB): Avalia o desempenho em amostras que não foram usadas para treinar as árvores.

Cross-validation: Utilizar validação cruzada para garantir que o modelo generalize bem para novos dados.

A métrica Log-Loss

Log-Loss, também chamada de logarithmic loss ou binary cross-entropy, é uma métrica que mede a incerteza de um modelo de classificação ao prever uma classe.

Diferente da acurácia, que só avalia se uma previsão está correta ou não, o log-loss considera o quão confiante o modelo está nas suas previsões.

Essa métrica é particularmente útil quando o modelo está gerando probabilidades de cada classe como resultado.

A métrica Log-Loss

Em termos simples:

Baixo Log-Loss significa que o modelo está prevendo as probabilidades de maneira correta, com confiança alta nas classes certas.

Alto Log-Loss significa que o modelo está confuso, prevendo probabilidades distantes do valor real, mesmo que algumas vezes acerte a classe final.

A métrica Log-Loss

Vantagens do uso do Log-Loss no Random Forest:

Captura incerteza: Diferente da acurácia, que só olha se o modelo está certo ou errado, o log-loss também mede quão confiante o modelo está. Assim, ele penaliza previsões muito incertas ou muito confiantes que estão erradas.

Avaliação mais detalhada: Em situações de classes desbalanceadas, onde a acurácia pode ser enganosa, o log-loss dá uma visão mais precisa do desempenho real do modelo.

Melhor para otimização: Quando o objetivo é ajustar hiperparâmetros ou otimizar o modelo, usar log-loss é preferível, já que ele força o modelo a melhorar a qualidade das previsões probabilísticas.

A métrica Log-Loss

Exemplo:

Quando você usa o Random Forest para problemas de classificação, o modelo pode gerar probabilidades para cada classe. Por exemplo, para uma classificação binária, ele pode prever que a transação tem 80% de chance de ser fraudulenta e 20% de chance de não ser.

O log-loss entra em cena justamente para medir o quão “bem calibradas” estão essas probabilidades. Se o Random Forest estiver acertando muitas classes mas com baixa confiança, o log-loss será alto. Por outro lado, se o modelo estiver fazendo previsões corretas com probabilidades altas e errando com probabilidades baixas, o log-loss será baixo.

Projeto – Random Forest

Uma empresa de tecnologia tem buscado formas de reter seus funcionários, dado o aumento no turnover nos últimos anos, fruto de uma profusão de empresas da nova economia, que tem feito ofertas tentadores para roubar talentos, com desafios e remunerações mais atrativos.

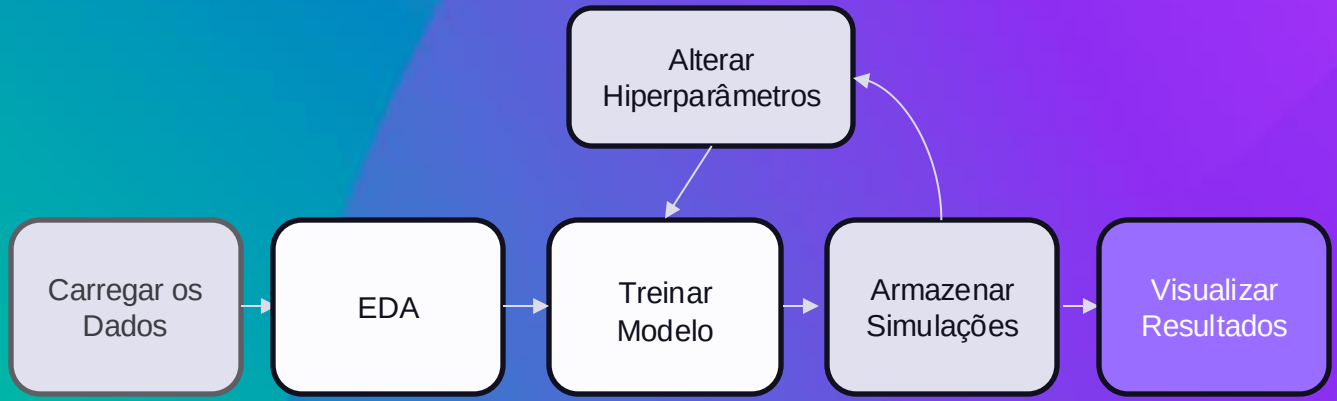
Mas a pergunta é: Será que somente um novo desafio ou um salário melhor é que estão motivando as pessoas a saírem desta empresa?

Com o objetivo de responder esta pergunta e propor mudanças drásticas na Diretoria de Pessoas, com o intuito de prevenir a evasão de talentos, com medidas personalizadas de retenção, o time de ciência de dados irá trabalhar num modelo capaz de prever a probabilidade de churn de um funcionário, mas também avaliar as principais motivações para este churn.

Pra isso, contará com um dataset histórico com dados de funcionários, contendo dados demográficos, dados sobre a relação do funcionário com a empresa e a variável target que indica se o funcionário saiu ou não da companhia.

E dada a complexidade do desafio e a quantidade de variáveis envolvida, iremos usar o ensemble Random Forest para fazer esta predição da variável target, da probabilidade do funcionário sair e as possíveis razões pra isso.

Estrutura do Projeto



Code Time . . .

rocketseat.com.br