

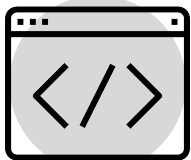
Profissão: Cientista de Dados



BOAS PRÁTICAS



Combinação de modelos II



- **Conheça o Boosting**
- **Conheça o AdaBoost**
- **Aplique Gradient Boosting Machine - GBM**
- **Conheça Stochastic Gradient Boosting Machine**
- **Utilize eXtreme Gradient Boosting - XGBoost**
- **Realize Boosting no Python**



Conheça o Boosting

- Embora o Boosting tenha semelhanças com o Bagging, eles são fundamentalmente diferentes. Compreender essas diferenças pode ajudar a escolher a técnica mais adequada para cada problema.
- O Boosting utiliza vários "weak learners" para criar um modelo forte. Entender esse conceito pode ajudar a entender como o Boosting funciona e como ele pode ser aplicado.



Conheça o AdaBoost

- Ao usar o algoritmo AdaBoost, é importante lembrar que ele constrói 'tocos' de árvores em vez de árvores completas. Isso significa que cada modelo é simples e se concentra em um aspecto específico dos dados.
- O AdaBoost é um algoritmo sequencial, o que significa que cada modelo é influenciado pelo anterior. Portanto, a ordem em que os modelos são treinados é importante.
- Ao usar o AdaBoost, é importante lembrar que os modelos têm pesos diferentes. Os modelos que têm um bom desempenho têm um peso maior, enquanto os modelos que não têm um bom desempenho têm um peso menor.





Conheça o AdaBoost

- Ao usar o AdaBoost, é importante entender como ele faz previsões. O AdaBoost faz previsões através da soma das performances ponderadas de cada modelo para cada classe possível, selecionando a classe com a maior soma.
- O AdaBoost raramente sofre de overfitting, independentemente do número de iterações realizadas. Isso torna o AdaBoost uma boa escolha para conjuntos de dados grandes e complexos.



Aplique Gradient Boosting Machine – GBM

- 
 Saiba como diferenciar e escolher entre algoritmos semelhantes: O professor discutiu a diferença entre o GBM e o AdaBoost, dois algoritmos que usam uma abordagem de combinação de modelos. Saber como esses algoritmos diferem pode ajudá-lo a escolher o mais adequado para o seu problema.
- 
 Busque sempre minimizar os resíduos: No caso do GBM, o objetivo é fazer com que os resíduos se aproximem de zero, indicando que o modelo está se ajustando bem aos dados. Isso é um lembrete importante de que, em aprendizado de máquina, nosso objetivo é sempre minimizar o erro.



Conheça Stochastic Gradient Boosting Machine

- Utilize técnicas como o Gradient Boosting Machine (GBM) para lidar com problemas de classificação e regressão. Essa técnica é eficaz e menos onerosa, pois seleciona uma subamostra do conjunto de dados de treinamento de forma aleatória e sem reposição.
- Explore técnicas híbridas que combinam diferentes abordagens. O GBM, por exemplo, é um híbrido entre as técnicas de bagging e boosting.
- Mantenha-se atualizado sobre as pesquisas e desenvolvimentos na área de aprendizado de máquina.



Utilize eXtreme Gradient Boosting – XGBoost

- Considere a escalabilidade: O XGBoost é capaz de ser executado mais de dez vezes mais rápido do que soluções populares existentes em uma única máquina e pode ser dimensionado para bilhões de exemplos em configurações distribuídas ou limitações de memória. Portanto, ao escolher um algoritmo, considere sua capacidade de escalar.
- Otimize os hiperparâmetros: O XGBoost tem muitos hiperparâmetros que podem ser otimizados para melhorar ainda mais seu desempenho. Isso inclui parâmetros relacionados à velocidade de execução, ao uso da máquina e à seleção de colunas. Portanto, não se esqueça de otimizar os hiperparâmetros do seu algoritmo para obter o melhor desempenho possível.



Utilize eXtreme Gradient Boosting – XGBoost

- Utilize algoritmos que lidam bem com dados esparsos: O XGBoost inclui um novo algoritmo de aprendizado de árvore para lidar com dados esparsos. Portanto, se você estiver trabalhando com dados esparsos, considere usar algoritmos que foram projetados para lidar com esse tipo de dados.



Realize Boosting no Python

- Ao usar o algoritmo de Boosting, comece com parâmetros padrão e calcule as métricas de desempenho. Isso fornecerá uma linha de base para comparação à medida que você otimiza o modelo.
- Se o desempenho do modelo no conjunto de treinamento for significativamente melhor do que no conjunto de teste e validação, isso pode indicar overfitting. Nesse caso, você deve ajustar os parâmetros do modelo para melhorar sua capacidade de generalização.



Realize Boosting no Python

- Use a otimização de hiperparâmetros para melhorar o desempenho do seu modelo. Experimente diferentes valores para cada hiperparâmetro e use uma métrica de desempenho, como ROC AUC, para determinar os melhores valores.
- Após otimizar os hiperparâmetros, treine o modelo novamente e calcule as métricas de desempenho. Se o desempenho melhorar em todos os conjuntos de dados (treino, teste e validação), isso indica que a otimização foi bem-sucedida.



Bons estudos!

