Estudo Dirigido 3

📌 Lucas de Araújo - 18.2.4049

Questão 1)

a) Overfitting: O overfitting é um conceito de aprendizado de máquina que ocorre quando um modelo é treinado de maneira tão específica nos dados de treinamento que ele tem dificuldade em generalizar para novos dados. Em outras palavras, o modelo "aprende demais" dos dados de treinamento, incluindo ruídos e outliers, o que diminui sua eficácia ao lidar com novos dados.

Exemplo: modelo de aprendizado de máquina para identificar e-mails de spam. Se este modelo for overfitting, pode funcionar perfeitamente nos dados de treinamento, mas pode ter um desempenho ruim ao encontrar novos e-mails que não estão no conjunto de treinamento.

b) Underfitting: O underfitting ocorre quando um modelo de aprendizado de máquina é muito simples para capturar a complexidade subjacente dos dados. Ele não aprende o suficiente a partir dos dados de treinamento e, portanto, tem um desempenho ruim tanto nos dados de treinamento quanto nos dados de teste.

Exemplo: sistema que tente prever o preço de casas com base em vários recursos, mas seu modelo só considera um único recurso (como a área total da casa), provavelmente acontecerá um underfitting pois o modelo é muito simplista para capturar todas as nuances dos dados.

c) Viés de busca: O viés de busca refere-se à tendência de um algoritmo de aprendizado de máquina em preferir determinados tipos de soluções ou modelos em detrimento de outros, com base no design do algoritmo.

Exemplo: Um algoritmo de árvore de decisão tem um viés de busca natural para modelos que usam uma hierarquia de perguntas de decisão "se-então". Essa estrutura pode levar a um desempenho subótimo quando se lida com problemas onde essa hierarquia não é a melhor abordagem.

d) Viés de representação: O viés de representação é uma tendência nos dados de treinamento que faz com que o modelo aprenda e reproduza esse viés, o que pode levar a resultados imprecisos ou injustos.

Exemplo: treinamento de um algoritmo de aprendizado de máquina para prever a admissão à universidade e seu conjunto de dados de treinamento inclui principalmente candidatos de um determinado grupo étnico, o modelo pode aprender a favorecer esse grupo étnico e ter um desempenho subótimo (ou até injusto) ao avaliar candidatos de outros grupos étnicos.

Questão 2)

Razões para utilizar modelos de aprendizado com composição:

1. **Melhorar a precisão**: Diferentes modelos podem aprender e se concentrar em diferentes características dos dados. Ao combiná-los, você pode aproveitar o melhor de cada um, geralmente resultando em uma precisão superior à de qualquer modelo individual.
2. **Aumentar a robustez**: Se um único modelo fizer uma previsão errada, é possível que outros modelos no conjunto possam corrigir esse erro. Assim, os modelos de ensemble são geralmente mais robustos a erros individuais.
3. **Reduzir overfitting/underfitting**: Ensemble learning pode ajudar a equilibrar o viés/variação, pois diferentes modelos podem ter diferentes níveis de complexidade. Isso pode ajudar a evitar o overfitting (quando o modelo é muito complexo e se ajusta demais aos dados de treinamento) e o underfitting (quando o modelo é muito simples e não se ajusta suficientemente aos dados de treinamento).

Exemplo: Imagine que estamos tentando decidir se devemos investir em uma determinada ação na bolsa. Podemos pedir conselhos a diferentes pessoas: um corretor de ações, um economista, um amigo que investe há muitos anos e um algoritmo de previsão de ações. Cada um desses "modelos" pode ter uma perspectiva diferente e oferecer um conselho diferente com base em diferentes aspectos da situação. Ao considerar todos esses conselhos juntos (ou seja, usar um "modelo de ensemble"), fará com que tomemos uma decisão melhor do que se confiassemos apenas em um único modelo. Cada modelo individual pode ter suas falhas ou viéses, mas ao combiná-los, você pode compensar essas deficiências e chegar a uma decisão mais precisa e robusta.

Questão 3)

A) Entropia e erro de classificação:

$$Entropy(S) = 0$$
$$Error(S) = 0$$

B) Entropia e erro de classificação:

$$Entropy(S) = \frac{5 \log \frac{6}{5}}{6 \log 2} + \frac{\log 6}{6 \log 2}$$
$$Error(S) = \frac{1}{6}$$

C) Entropia e erro de classificação:

$$Entropy(S) = \frac{2 \log \frac{3}{2}}{3 \log 2} + \frac{\log 3}{3 \log 2}$$
$$Error(S) = \frac{1}{3}$$

D) Entropia e erro de classificação:

$$Entropy(S) = 1$$
$$Error(S) = \frac{1}{2}$$

Questão 4)

- Tipos de Carro:

$$Gini(A) = 1 - (p_1^2 + p_2^2) = 1 - ((1/5)^2 + (4/5)^2) = 8/25$$

$$Gini(B) = 1 - (p_1^2 + p_2^2) = 1 - ((2/3)^2 + (1/3)^2) = 4/9$$

$$Gini(C) = 1 - (p_1^2 + p_2^2) = 1 - ((1/2)^2 + (1/2)^2) = 1/2$$

- Tipos de Renda:

$$Gini(A) = 1 - (p_1^2 + p_2^2) = 1 - ((3/5)^2 + (2/5)^2) = 12/25$$

$$Gini(B) = 1 - (p_1^2 + p_2^2) = 1 - ((2/6)^2 + (4/6)^2) = 4/9$$

Questão 5)

Cálculo do Gini inicial:

P(doente) = 3/6 = 0.5
P(saudável) = 3/6 = 0.5

Gini Total = 1 - [P(doente)^2 + P(saudável)^2] = 1 - [0.25 + 0.25] = 0.5

Cálculo do Gini para Febre:

- **Febre = sim:**
  P(doente) = 2/4 = 0.5
  P(saudável) = 2/4 = 0.5

  Gini(Febre=sim) = 1 - [0.5^2 + 0.5^2] = 0.5

- **Febre = não:**
  P(doente) = 1/2 = 0.5
  P(saudável) = 1/2 = 0.5

  Gini(Febre=não) = 1 - [0.5^2 + 0.5^2] = 0.5

Gini Ponderado para Febre = (4/6)*0.5 + (2/6)*0.5 = 0.5

Cálculo do Gini para Enjôo:

- **Enjôo = sim:**
  P(doente) = 1/3 = 0.333
  P(saudável) = 2/3 = 0.667

  Gini(Enjôo=sim) = 1 - [0.333^2 + 0.667^2] = 0.444

- **Enjôo = não:**
  P(doente) = 2/3 = 0.667
  P(saudável) = 1/3 = 0.333

  Gini(Enjôo=não) = 1 - [0.667^2 + 0.333^2] = 0.444

Gini Ponderado para Enjôo = (3/6)*0.444 + (3/6)*0.444 = 0.444

Cálculo do Gini para Mancha:

- **Mancha = pequenas:**
  P(doente) = 1/3 = 0.333

P(saudável) = 2/3 = 0.667

Gini(Mancha=pequenas) = 1 - [0.333^2 + 0.667^2] = 0.444

- **Mancha = grandes:**
  P(doente) = 2/3 = 0.667
  P(saudável) = 1/3 = 0.333

  Gini(Mancha=grandes) = 1 - [0.667^2 + 0.333^2] = 0.444

Gini Ponderado para Mancha = (3/6)*0.444 + (3/6)*0.444 = 0.444

Cálculo do Gini para Dores:

- **Dores = sim:**
  P(doente) = 3/4 = 0.75
  P(saudável) = 1/4 = 0.25

  Gini(Dores=sim) = 1 - [0.75^2 + 0.25^2] = 0.375

- **Dores = não:**
  P(doente) = 0/2 = 0
  P(saudável) = 2/2 = 1

  Gini(Dores=não) = 1 - [0^2 + 1^2] = 0

Gini Ponderado para Dores = (4/6)*0.375 + (2/6)*0 = 0.25

Com base no Gini ponderado, o atributo "Dores" possui o menor Gini e, portanto, será o primeiro nó da nossa árvore de decisão.

Árvore de Decisão:

1. **Dores = sim:**
   - Maioria dos exemplos são "doente", então classificamos como "doente"
2. **Dores = não:**
   - Todos os exemplos são "saudável", então classificamos como "saudável".

Teste para os dados:

2.1) (Luis, não, não, pequenas, sim)

- Dores = sim → Doente

2.2) (Laís, sim, sim, grandes, sim)

- Dores = sim → Doente

Ambos os dados testados são classificados como "doente" pela árvore de decisão induzida.

Questão 6)

Vou começar por calcular os pesos (w) após dois ciclos do algoritmo Perceptron.

Primeiro, vamos definir nossos valores iniciais com base na informação dada:

η = 0,05
Número de matrícula: 18.2.4049 → m = 49
w = [0,1; 0,49; 0,1]

O algoritmo Perceptron ajusta os pesos w de acordo com a seguinte regra:
w = w + η * (d - y) * x
Onde:

- d é o valor desejado (valor verdadeiro)
- y é a saída do neurônio (0 ou 1)
- x é o vetor de entrada
- η é a taxa de aprendizado

Vamos calcular os pesos após 2 ciclos:

Primeiro Ciclo:

1. Para a Entrada (1):
   x = [1, 2, 3]
   d = 1
   y = sgn(w.x)
   y = sgn(0,1*1 + 0,49*2 + 0,1*3)
   y = sgn(1,08) → Como é positivo, y=1.
   O valor calculado coincide com o valor desejado, portanto, os pesos não são atualizados.
2. Para a Entrada (2):
   x = [1, 4, 1]
   d = 1
   y = sgn(0,1*1 + 0,49*4 + 0,1*1)
   y = sgn(2,06) → y=1.
   Os pesos não são atualizados.
3. Para a Entrada (3):
   x = [1, 1, 1]
   d = 0
   y = sgn(0,1*1 + 0,49*1 + 0,1*1)
   y = sgn(0,69) → y=1.

Aqui, temos um erro, pois a saída é diferente do valor desejado. Atualizamos os pesos:
Δw = η * (d - y) * x
Δw = 0,05 * (-1) * [1, 1, 1]
Δw = [-0,05; -0,05; -0,05]
w (novo) = [0,05; 0,44; 0,05]

4. Para a Entrada (4):
   x = [1, 1, 0]
   d = 0
   y = sgn(0,05*1 + 0,44*1 + 0,05*0)
   y = sgn(0,49) → y=1.
   Também temos um erro:
   Δw = 0,05 * (-1) * [1, 1, 0]
   Δw = [-0,05; -0,05; 0]
   w (novo) = [0; 0,39; 0,05]

Segundo Ciclo:
Repetimos as entradas com os novos pesos.

1. Para a Entrada (1):
   x = [1, 2, 3]
   d = 1
   y = sgn(0*1 + 0,39*2 + 0,05*3)
   y = sgn(0,87) → y=1.
   Os pesos não são atualizados.

2. Para a Entrada (2):
   x = [1, 4, 1]
   d = 1
   y = sgn(0*1 + 0,39*4 + 0,05*1)
   y = sgn(1,61) → y=1.
   Os pesos não são atualizados.

3. Para a Entrada (3):
   x = [1, 1, 1]
   d = 0
   y = sgn(0*1 + 0,39*1 + 0,05*1)
   y = sgn(0,44) → y=1.
   Δw = 0,05 * (-1) * [1, 1, 1]
   Δw = [-0,05; -0,05; -0,05]
   w (novo) = [-0,05; 0,34; 0]

4. Para a Entrada (4):
   x = [1, 1, 0]
   d = 0
   y = sgn(-0,05*1 + 0,34*1 + 0*0)
   y = sgn(0,29) → y=1.
   Δw = 0,05 * (-1) * [1, 1, 0]
   Δw = [-0,05; -0,05; 0]
   w (novo) = [-0,1; 0,29; 0]

Então, após 2 ciclos de execução do algoritmo Perceptron, os valores de peso (w) do neurônio são:
w = [-0,1; 0,29; 0].

Questão 7)

- Alternativa correta: B

Questão 8)

- Alternativa correta: C

Questão 9)

- Alternativa correta: A