Questões_e_Relatório

February 4, 2022

1 1

O dilema descreve a dificuldade em se obter um modelo que tenha, simultâneamente, baixos erros de bias e de variância. Um modelo menos complexo possuirá menos erros de variância (os dados preditos por ele tendem a variar menos) mas terá maior erro de bias. Por serem menos complexos, esses modelos não conseguem capturar a complexidade dos problemas, e acabam sofrendo de *underfitting*.

Por outro lado, modelos mais complexos terão menor erro de bias (capturam melhor a complexidade do problema), porém maior erro de variância. À medida que a complexidade do modelo aumenta, ele tende a "decorar" os dados utilizados no treinamento (*overfitting*). Assim, quando lhe é apresentado um novo conjunto de dados, tal modelo apresenta uma alta variância.

2 2

2.1 a)

A afirmação é verdadeira caso os atributos sejam relevantes para o problema. Atributos não relevantes podem adicionar custo computacional desnecessário ao modelo fazendo com que ele tente aprender informações de atributos não irão adicionar nada de relevante ao modelo.

2.2 b)

Falso. Dados sem qualidade podem conter ruídos, valores incorretos e inconsistentes, prejudicando o desempenho do modelo, por exemplo, fazendo com que ele aprenda algo que não era o desejado. Sendo assim, a quantidade de dados "ruins" podem, sim, prejudicar o desempenho do modelo.

2.3 c)

Falso. A acurácia é uma métrica aceitável apenas em bases com uma distribuição balanceada das classes presentes nos dados. Em bases desbalanceadas, para a acurácia ser aceitável, ela precisa ser maior que acurácia obtida ao se atribuir todas as amostras à classe majoritária.

2.4 d)

Falso. O PCA é, na verdade, uma técnica utilizada na redução da dimensionalidade de dados com muitas dimensões.

3 Relatório

Todos os exercícios foram implementados na linguagem Python e executados no Google Colab.

Para a resolução da questão 5, utilizou-se a base de dados Wine. Primeiramente, dividiu-se a base de dados em 3 partes iguais de forma estratificada. A distribuição das classes das amostras impossibilitava que fosse obtida exatamente 3 bases iguais mantendo-se a proporção de amostras por classe. Assim, durante a amostragem estratificada, permitiu-se que algumas amostras aparecessem repetidas em cada uma das bases.

Para a amostragem estratificada, primeiramente verificou-se a proporção de amostras por classe na base de dados e separou-se as amostras de cada classe em 3 bases distintas. Com base na proporção identificada, criou-se três bases distintas a partir das amostras de cada uma das 3 bases separadas.

Na implementação do SFS, primeiro criou-se uma lista dos atributos da base de dados e identificou-se o melhor atributo dentre eles. A partir dos atributos restantes, foi-se adicionando os melhores atributos até que um total de 3 atributos fosse selecionado (**Questão 5a**) ou 8 atributos (**Questão 5b**). A implementação do SBE foi feita de maneira similar, com a diferença de que selecionava-se o atributo com pior acurácia.

Para a resolução da questão 6, foi utilizada a base de dados Electrical Grid Stability Simulated Data. Como solicitado, dividiu-se, aleatoriamente, a base em 8000 amostras de treino e 2000 de teste. Cada algoritmo dos itens a, b e c, foram executados 5 vezes, com a acurácia e tempo de execução de cada um sendo registrados.

Para garantir que os dados de treino e testes fossem diferentes em cada execução, implementou-se uma função que selecionava novas amostras de treino e de teste aleatórias.

O algoritmo Rocchio (**Questão 6a**) foi implementado em duas partes. A primeira identificava as classes presentes nos dados e calculava a média das amostras de cada classe. A segunda classificava uma base de teste com base nas informações calculadas pela primeira parte.

Para a (**Questão 6b**), implementou-se o algoritmo kNN também em duas partes. A primeira calcula a matriz de distância entre a base de treino e de teste e a segunda retorna os k vizinhos desejados, classificando a base de teste a partir da base de treino. A obtenção do melhor K foi feita como uma função para facilitar a geração de resultados e dos experimentos.

Por fim, na **Questão 6c** implementou-se o Edit kNN com inserção sequencial. Para facilitar o entendimento do algoritmo, utilizou-se uma matriz de distâncias auxiliar, em que as distâncias entre todas as amostras era igual a infinito. Assim, durante a verificação se uma amostra deveria ou não ser inserida, a matriz auxiliar era consultada e atualizada, caso uma amostra fosse inserida.