UNIVERSIDADE DE ÉVORA
CURSO DE ENGENHARIA INFORMÁTICA
2023/2024

Aprendizagem Automática

→ Relatório ←

KNN e Naïve Bayes

António Carvalho - 51483
Tiago Morgado - 51717
Mariana Cavaco - 51820

1. Objetivo

Este trabalho tem como objetivo implementar em Python os algoritmos KNN e Naïve de Bayes para problemas de classificação que permitam a integração com o ambiente scikit-learn.

2. Implementação

- **KNeighborsClassUE**

A classe KNeighborsClassUE é um modelo de k-vizinhos mais próximos (KNN), utilizado para tarefas de classificação. No método de inicialização, definimos parâmetros como o número de vizinhos (k) e o parâmetro da distância de Minkowski (p). Durante o treinamento, o método fit recebe conjuntos de treinamento (X e y) e os converte em arrays numpy, atribuindo esses conjuntos a X_train e y_train. O modelo utiliza a distância de Minkowski para encontrar os kvizinhos mais próximos durante a previsão, realizada pelo método predict. A precisão do modelo é avaliada pelo método score, que compara as previsões com os rótulos verdadeiros.

- **NBayesClassUE**

A classe NBayesClassUE implementa um modelo de Naive Bayes, adequado para tarefas de classificação probabilística. No método de inicialização, definimos o parâmetro alpha para suavização de Laplace e inicializamos o dicionário probs para armazenar probabilidades. Durante o treinamento, o método fit calcula contagens de instâncias para cada classe, contagens condicionais para cada valor de característica dada uma classe, e as probabilidades associadas. As previsões são realizadas pelo método predict, que calcula a pontuação posterior de cada classe para uma instância e retorna a classe com a pontuação mais alta. A avaliação da precisão do modelo é feita pelo método score, que compara as previsões com os rótulos verdadeiros, fornecendo a taxa de acerto do modelo Naive Bayes.

3. Análise Crítica do Desempenho

K: 1 |P: 1 | KNN_UE: 0.9473684210526315
K: 3 |P: 1 | KNN_UE: 0.9473684210526315
K: 4 |P: 1 | KNN_UE: 0.9210526315789473
K: 9 |P: 1 | KNN_UE: 0.9736842105263158
K: 1 |P: 2 | KNN_UE: 0.9473684210526315
K: 3 |P: 2 | KNN_UE: 0.9473684210526315
K: 4 |P: 2 | KNN_UE: 0.9473684210526315
K: 9 |P: 2 | KNN_UE: 0.9736842105263158

Para K=9 com p=1 e p=2 a precisão é a mais elevada, a precisão é de 97.37%.

Para K=4 com p=1 é onde se obtém o valor mais baixo, a precisão é de 92.11% .

Para os restantes valores o modelo tem um desempenho consistente, onde a precisão é de 94.74%.

Com isto, podemos concluir que o melhor desempenho é obtido quando o K=9.

alpha: 0 | NBUE: 0.7857142857142857
alpha: 1 | NBUE: 0.7857142857142857
alpha: 3 | NBUE: 0.7857142857142857
alpha: 5 | NBUE: 0.7857142857142857

Os resultados acima são os obtidos para o classificador NBAYES da variável alpha para o ficheiro ‘bc-nominal.cvs’. Como podemos perceber o alpha não está a ter impacto no desempenho do classificador. podemos afirmar que a exatidão do modelo não é afetada pela variação do alpha. Provavelmente o classificador não está a conseguir avaliar corretamente o modelo.

4. Conclusão

Em resumo, ambos os modelos têm vantagens e desvantagens, e a escolha entre eles depende das características do conjunto de dados e dos requisitos específicos do problema. A otimização de parâmetros e a consideração das suposições subjacentes são aspectos críticos ao usar esses modelos. Concluímos que com este trabalho tivemos uma introdução ao desenvolvimento de classificadores de modelos de inteligência artificial em python com a biblioteca sklearn mesmo que não tenham ficado bem implementados no final.