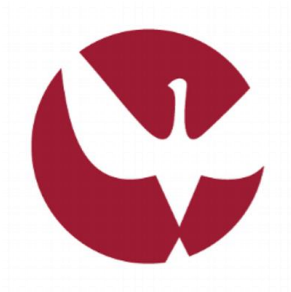

Aprendizagem Automática

Relatório
Trabalho Prático 1

2023/2024

Trabalho realizado por:
Alexandre Costa nº 48039
Manuel Pereira nº 45855

Docente:
Professora Teresa Gonçalves

1. K-N-Neighbors (k-n-vizinhos/knn)

1.1 Algoritmo

O algoritmo tem como parâmetros o nº de vizinhos, o peso destes (sendo uniforme ou inversamente proporcional à distância), a função distância (_min_kowski, p):

- Euclidiana (p>=2);
- Manhattan (p=1);

$$D(X,Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{\frac{1}{p}}$$

O cálculo dos vizinhos mais próximos pode ser realizado por "Força Bruta" (calculando a distância para todos os pontos) ou por "Cálculo Otimizado" (BallTree, KDTree). Este algoritmo é fácil de perceber e, normalmente, não precisa de muitos ajustes. Porém, quando confrontado com conjuntos de treino grandes (nº de exemplos ou atributos) a previsão é lenta, o que também se verifica quando os conjuntos são esparsos (i.e., a maioria dos atributos tem o valor 0).

1.2 Código

Existe um construtor que aceita dois parâmetros, 'k' e 'p' (sendo estes o nº de vizinhos e o parâmetro de distância respetivamente), com valores padrões definidos. O método "fit" recebe os conjuntos 'X' e 'y' e converte os dados para arrays. A distância euclidiana é calculada pela função "_min_kowski". A função "predict" aplica o modelo e devolve as etiquetas previstas para o conjunto fornecido à função, onde depois a função "score" vai averiguar a exatidão do modelo.

1.3 Observações

Conjunto de dados	Parâmetro K	Parâmetro P	Conjunto treino (Exatidão)	Conjunto teste (Exatidão)
Íris	1	1	1.00000000	0.94736842
	1	2	1.00000000	0.94736842
	3	1	0.96428571	0.94736842
	3	2	0.96428571	0.94736842
	5	1	0.97321429	0.94736842
	5	2	0.97321429	0.94736842
	9	1	0.97321429	0.97368421
	9	2	0.97321429	0.97368421
Rice	1	1	1.00000000	0.87932844
	1	2	1.00000000	0.87093389
	3	1	0.93489674	0.89821616
	3	2	0.93069653	0.88772298
	5	1	0.91774589	0.90556139
	5	2	0.91004550	0.90031480
	9	1	0.90619531	0.90451207
	9	2	0.89324466	0.89192025
WDBC	1	1	1.00000000	0.83916084
	1	2	1.00000000	0.80419580
	3	1	0.90140845	0.80419580
	3	2	0.88028169	0.80419580
	5	1	0.84741784	0.76923077
	5	2	0.81220657	0.73426573
	9	1	0.76760563	0.71328671
	9	2	0.75117371	0.67832168

Os conjuntos de treino e teste foram avaliados em termos de exatidão. Relativamente ao conjunto de dados Iris, o modelo parece ter um desempenho bastante bom nos conjuntos de teste, com exatidão variando entre 94,74% e 97,37%. Para K=1, o modelo atinge uma exatidão de 100% no conjunto de treino, mas há uma pequena queda para o conjunto de teste. Isso pode indicar um pouco de sobre-ajustamento. Ao aumentar o K para 3 e 5 resulta numa exatidão estável no conjunto de teste, sugerindo uma boa generalização. Quando o K=9, a exatidão atinge 97,37%, indicando que considerar um número maior de vizinhos pode ser benéfico.

Quanto à análise feita no conjunto de dados Rice, o desempenho é geralmente bom. Aumentar o K geralmente leva a uma queda na exatidão do conjunto de treino, mas melhora a generalização para o conjunto de teste. Com o valor K=1 apresenta uma exatidão perfeita no conjunto de treino, indicando possivelmente sobre-ajuste. Valores maiores de K parecem mais equilibrados.

Por fim, quanto ao conjunto de dados WDBC, o desempenho é mais variado neste conjunto, com exatidão variando entre 67,83% e 83,94% no conjunto de teste. Para K=1 o modelo atinge 100% no conjunto de treino, mas a exatidão do conjunto de teste continua baixa, indicando sobre-ajustamento. Com o K a 5 e 9 conseguimos observar um pouco de sub-ajustamento, dando a entender que o modelo se encontra demasiado simples para conseguir capturar todos os aspetos e variabilidade dos dados.

1.4 Estrutura de Dados

Foram usados arrays para guardar “X” e “y” convertendo-os em arrays Numpy para treino, sendo feito na função “fit”. Na função “predict” , as previsões e labels foram guardadas em um array Numpy, onde estão, para cada índex, guardadas as labels mais comuns para a distância k. Para a função “fit” foram usados arrays Numpy para guardar o número de ocorrências de cada classe assim como para as contagens de ocorrência de cada combinação de atributo e classe (respetivamente, “feature_probs” e “feature_counts”).

2. Naive-Bayes

2.1 Algoritmo

O algoritmo tem como objetivo, dados os valores dos atributos de um conjunto de dados, atribuir a classe c mais provável a um exemplo.
Usando o teorema de Bayes: $P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)}$

Tem-se:

$$\arg\max_{c_j \in C} \frac{P(a_1, a_2, \ldots, a_n|c_j)P(c_j)}{P(a_1, a_2, \ldots, a_n)} \qquad \arg\max_{c_j \in C} P(c_j|a_1, a_2, \ldots, a_n)$$

- Assume a independência condicional dos valores dos atributos dada a classe:

$$P(a_1, a_2, \ldots, a_n|c_j) = P(a_1|c_j) * P(a_2|c_j) * \ldots * P(a_n|c_j)$$

- Construção do modelo: - estimar as probabilidades:
  - de cada classe: P(c_j)
- do valor de cada atributo dada a classe: P(a_i | c_j)
- Previsão de um exemplo:
  - Escolher a classe que maximiza a expressão

$$P(c_j) \prod P(a_i|c_j)$$

- Estimador de Probabilidades:

- P(x) = n_x / total
    - n_x : nº de vezes que x ocorre
    - total : nº máximo possível
- Características: Se o valor estimado for 0, o termo domina o classificador:
    - Sempre que um valor de atributo não aparece no conjunto de treino

- Estimador Suavizado/ Lidstone:

$$P(x) = \frac{n_x + \alpha}{total + \alpha * nvals}$$

2.2 Código

A classe NBayesClasseUe possui um construtor que permite ajustar o parâmetro de suavização denominado "alpha". O método "fit" é utilizado para treinar o modelo, onde são contabilizadas as frequências de ocorrência de cada valor de atributo em cada classe. O método "predict" realiza previsões para novas instâncias, utilizando a fórmula do Teorema de Bayes. O grupo decidiu que ao aparecer um novo valor de atributo nunca visto durante o treino do modelo seria atribuído zero ao número de vezes que X ocorre. Por fim, no método "score" é calculada a exatidão do modelo em relação ao conjunto testado.

2.3 Observações

Conjunto de dados	Parâmetro Alpha	Conjunto treino (Exatidão)	Conjunto teste (Exatidão)
Bc-Nominal	0	0.75362	0.82857
	1	0.75845	0.81429
	3	0.73430	0.78571
	5	0.75362	0.78571
Contact-Lenses	0	0.88889	0.83333
	1	0.88889	1.00000
	3	0.88889	1.00000
	5	0.88889	1.00000
Weather-Nominal	0	0.90000	0.75000
	1	0.90000	0.75000
	3	0.90000	0.75000
	5	0.70000	0.75000

Os conjuntos de treino e teste foram avaliados em termos de exatidão. Relativamente ao conjunto de dados Bc-Nominal, o desempenho no conjunto de teste não apresenta melhoria significativa com diferentes valores de alpha. A exatidão é relativamente baixa, indicando que o modelo pode não estar bem generalizado para novos dados. Podemos concluir que o algoritmo Naive Bayes pode não ser a melhor escolha para este conjunto de dados.

Quanto ao conjunto de dados Contact-Lenses, os resultados são perfeitos no conjunto de teste. Isso levanta a questão de, se o modelo está a avaliar corretamente o conjunto de teste ou se o conjunto de teste é bastante simples e previsível, o que permite ao modelo atingir altos níveis de exatidão.

Por fim, no conjunto de dados Weather-Nominal apesar da variação do alpha a exatidão do modelo relativamente ao conjunto de teste permanece sempre baixa. Claramente estamos perante um problema de sobre-ajustamento muito provavelmente devido ao conjunto de dados ser pequeno e/ou o próprio conjunto de treino não ser variado o que faz com que faça um bom ajustamento aos dados de treino, mas não está a generalizar efetivamente para novos dados.

2.4 Estrutura de Dados

É usado o módulo 'numpy' para criar arrays a partir dos dados fornecidos pelo arquivo CSV. O módulo 'pandas' é utilizado para ler o conjunto de dados de um ficheiro CSV e transformá-lo em um DataFrame. A instância 'classes' é dicionário que mapeia rótulos de classe para a contagem de instâncias de cada classe. A instância 'atributo' é um dicionário aninhado que mapeia rótulos de classe para atributos e suas contagens.
O "valor_atributo" faz de dicionário que mapeia índices de atributos para os valores únicos que esses atributos podem ter.