Métodos Probabilísticos para Engenharia Informática-42432

Projeto MPEI
(Gestão de lesões de atletas)

Tomás Hilário - 119896

Diogo Duarte - 120482

deti universidade de aveiro
departamento de eletrónica,
telecomunicações e informática

Introdução

Neste trabalho foi-nos proposto a criação de um conjunto de funções/módulos em *MatLab* que aplicassem os conceitos lecionados na disciplina de MPEI, entre eles, o classificador de *Naïve Bayes,* filtro de *bloom* e *MinHash,* de modo a produzir um resultado de valor.

O tema por nós escolhido foi a gestão de lesões em atletas. Estes atletas serão definidos por um conjunto de características, sendo elas, género (*gender),* idade *(age),* altura *(height),* peso *(weight),* desporto praticado *(sport),* horas de treino semanais *(weekly training hours),* intensidade de treino *(training intensity),* condição física *(physical conditioning),* índice de nutrição *(nutrition score),* número de lesões contraídas no passado *(previous injuries)* e a classe do atleta no que toca à facilidade de contração de lesões *(injury risk)*.

Aplicação conjunta

A aplicação conjunta dos nossos módulos funcionará do seguinte modo: o programa começa, preferencialmente, com a inicialização do filtro de bloom da variável "BF.mat" que vai guardando e atualizando os atletas que possuem alto risco de contração de lesão, porém, também é possível a reinicialização do filtro "descomentando" o código em causa. Posto isto, é pedido ao utilizador que insira um valor para cada característica que representa o atleta e estes valores são verificados e guardados.

Em seguida, é verificado se o atleta pertence ou não ao filtro de *bloom*. No caso de pertença podemos concluir, embora que com erro, que o atleta inserido pelo utilizador é propenso à contração de lesões, enquanto que no caso em que a pertença não foi verificada, teremos de utilizar o módulo de *Naïve Bayes* para prever a classe do atleta (baixo ou alto risco de lesão) e, nos casos em que o atleta é classificado como de alto risco é adicionado ao *bloom filter* e a variável "BF.mat" é atualizada. Após a classificação do atleta, é perguntado ao utilizador se pretende receber um conjunto de atletas com um perfil semelhante a esse atleta, bem como a dimensão desse conjunto e é utilizado o módulo de *MinHash* para o cálculo dessas similaridades, sendo depois impresso no ecrã os resultados.

***Naïve Bayes*:**

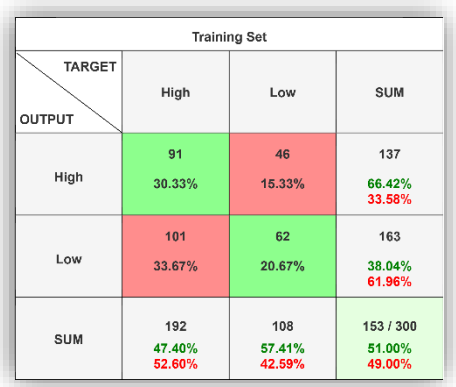
Para o teste deste módulo foi realizado a divisão aleatória dos dados dos atletas em que 70% são utilizados para treino e os restantes para teste. Depois disso, é utilizado o módulo para prever a classe dos atletas de teste e é comparado o valor previsto com o valor real e calculado o número de verdadeiros e falsos positivos e negativos. Dado o nosso objetivo foi considerado como positiva a previsão correspondente a um atleta pertencente à classe de risco. Após isso, foram calculados os valores de *precision*, *recall* e F1-score. Este processo foi realizado "n" vezes para que fossem obtidos valores mais precisos para estas variáveis.

***Output*:**

>> Precision: 66.25%

>> Recall: 47.12%

>> F1-score: 54.85%



Training Set

TARGET / OUTPUT	High	Low	SUM
High	91 30.33%	46 15.33%	137 66.42% 33.58%
Low	101 33.67%	62 20.67%	163 38.04% 61.96%
SUM	192 47.40% 52.60%	108 57.41% 42.59%	153 / 300 51.00% 49.00%

Dados estes valores, podemos concluir que cerca de 66% das previsões positivas feitas pelo classificador estão corretas, porém apenas cerca de 47% dos casos positivos são realmente identificados e, dado o nosso objetivo de acertar grande parte dos casos positivos, estes valores são apenas aceitáveis.

Pensamos que estes valores baixos se devam, maioritariamente, à natureza dos dados utilizados, embora tenhamos tentado classificar atletas baseados num outro conjunto de características, tendo obtido resultados semelhantes. Características como a altura e o peso de um atleta podem não ter tanta influência para a contração de lesões visto serem subjetivas ao desporto praticado pelo atleta. É previsível que um atleta que pratique *basketball* seja mais alto e mais pesado do que um atleta que pratique natação, o que não necessariamente implica que o risco de lesão do atleta de basketball seja mais alto. A forma como foram gerados os dados, nomeadamente o facto da divisão dos atletas não ser exatamente 50/50 pode também ter influência no resultado destas variáveis. Concluindo, o classificador será apenas razoável.

***Bloom Filter*:**

Foi inicializado o *bloom filter* com n bits e, recorrendo às funções deste módulo, foram colocados no filtro os atletas cuja classe era *high* (alto risco de lesão). Após isso, iteramos pelos atletas da classe *low* (baixo risco de lesão) e verificamos a pertença dos mesmos ao filtro. Uma vez que estes atletas não foram inseridos no filtro inicialmente, não devem ser considerados pelo filtro como pertencentes ao mesmo. Os casos em que isto acontece são considerados como falsos positivos. Embora, numa implementação correta do filtro de bloom fosse conhecido o número de falsos negativos (0), optamos por realizar o teste da mesma maneira. Foi também calculada a percentagem de erro teórica e comparado esse valor ao valor por nós obtido.

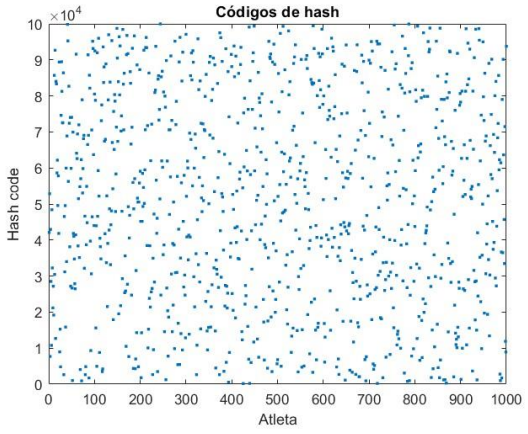
***Output*:**

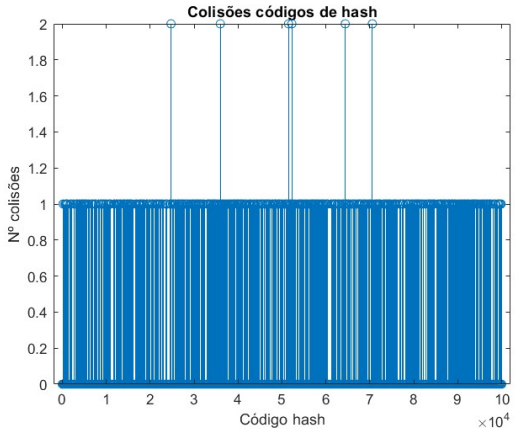
>> Falsos positivos: 3 (em 369)

>> Percentagem de falsos positivos: 0.81%

>> Falsos negativos: 0 (em 631)

>> Percentagem de falsos negativos: 0.00%





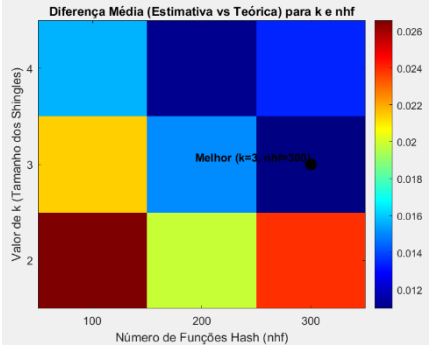
Os valores obtidos confirmam o bom funcionamento do nosso filtro de bloom, nomeadamente o baixo valor de falsos positivos, visto que, como já foi dito, o número de falsos negativos teria de ser sempre 0. Recorrendo à fórmula que calcula a probabilidade de erro, obtemos 0.025% de erro e, embora este valor seja ligeiramente inferior ao valor por nós obtido, esta diferença não será significativa e pode ser

explicada pela estrutura da nossa função de hash, que pela análise dos gráficos podemos concluir funcionar de maneira correta. Podemos então concluir que o nosso filtro de bloom cumpre os seus objetivos, apresentando uma baixa taxa de falsos positivos e garantindo a ausência de falsos negativos.

***MinHash*:**

Para o teste do nosso módulo *MinHash*, foram verificados maioritariamente dois aspetos: para atletas com características iguais, a distância de *Jaccard* terá de ser igual a zero e o valor da diferença entre as distâncias de Jaccard obtidas pelo módulo e os valores dessas distâncias na teoria.

Além disso concluímos também o comprimento ideal para os *shingles* seria de k = 3 pois cada entrada teria por volta de 40 a 50 caracteres de texto, por isso seria apenas necessário criar *shingles* deste tamanho. Também concluímos que 300 funções de *hash* seria um número suficiente para obter resultados muito satisfatórios, como se pode concluir na figura abaixo. Valores superiores de ambas as variáveis não iam resultar numa melhoria significativa, aumentando apenas o tempo de execução e o espaço ocupado em memória.



***Input*:**

'Men 24 184 82.4 Football 13.2 High 3.4 2.1 3'

***Output*:**

>> Os 5 atletas mais similares são:

>> Men 24 184 62.9 Football 15 High 2.2 4.6 3 high com distância 0.5433 (Diferença Teórica: 0.0356)

>> Men 36 194 77.4 Football 18.3 High 3.9 2.3 4 high com distância 0.6500 (Diferença Teórica: 0.0057)

>> Men 18 191 67.8 Football 19.1 High 3.1 3.8 5 high com distância 0.6500 (Diferença Teórica: 0.0110)

>> Men 35 187 78.6 Football 19.6 High 2.4 2.3 1 high com distância 0.6533 (Diferença Teórica: 0.0188)

>> Men 21 177 72 Football 16.8 High 3.2 3 3 high com distância 0.6533 (Diferença Teórica: 0.0077)

Modo de funcionamento

Os testes aplicados a cada módulo podem ser consultados na pasta "testes" contida no ficheiro zip enviado. Existem 3 ficheiros de teste, um para cada módulo e basta correr cada um dos scripts para obter o resultado dos testes, que já foram abordados acima no relatório. Para correr a demonstração conjunta dos módulos basta correr o ficheiro "main.m" e seguir os *prompts* que são dados. Espera-se que este programa realize o que foi definido acima, relativamente à utilização de cada módulo. Existe ainda uma pasta denominada "Data" que possui os ficheiros CSV com os dados utilizados e ainda alguns scripts para a geração de dados. Muitos destes scripts foram escritos com auxílio de IA em python e adaptados face às nossas necessidades.

Conclusões

O sistema consegue lidar com dados reais e contínuos, como altura, peso, idade e outras características físicas, integrando essas variáveis no processamento e análise. No entanto, a previsão da probabilidade de lesões dos atletas apresenta limitações devido à natureza genérica dos dados utilizados. Esses dados não possuem detalhes suficientes para capturar nuances específicas, como histórico médico detalhado, tipo de treino, genética ou fatores externos que influenciam o risco de lesão.

Essa situação reflete um cenário real em que prever lesões com alta precisão é extremamente desafiador. Lesões são eventos multifatoriais, influenciados por condições complexas e variáveis não mensuráveis, como esforço acumulado, biomecânica e contexto desportivo. Mesmo com ferramentas avançadas, seria difícil alcançar previsões altamente corretas sem informações especializadas e granularidade nos dados.