1.

A nossa motivação para este trabalho foi o facto ambos gostarmos do tema, música está presente no nosso dia a dia, desperta em nós diferentes emoções, desde energia a calma, alegria a tristeza… todas as emoções que podemos experienciar. Achamos também interessante podermos fazer uma análise de dados de uma das maiores plataformas de streaming mundial.

Neste trabalho vamos fazer uma previsão das músicas mais e menos populares do dataset “spotify top hits…” usando diferentes modelos de machine learning, tornando o nosso problema num problema de classificação. Pretendemos obter a accuracy do modelos implementados e, posteriormente, fazer uma comparação entre eles.

2.

Aqui podemos ver as features do nosso dataset: artista, música, duração (em ms), se a música tem conteúdo explícito ou não, o ano do lançamento da música, popularidade, o quão suscetível uma música é para dançar (numa escala de 0-1), energia da música (escala de 0-1 também), key (que é a nota música em que música está. A escala usa uma notação chamada Pitch Class. Se não for detetada nota, então toma o valor -1), a loudness (medida em db), o modo (indica a modalidade da música: alto(1) ou baixo(0)), presença de palavras faladas na música, se a música é acústica ou não (classificado como 1 e 0), intrumentalness (prevê se uma música contem ou não vocals), liveness, valence (prevê se uma música é positiva/energética ou negativa/mais depressiva. É medida numa escala de 0 e 1), tempo (bpm) e, por fim, o género da música. São então 18 variáveis no total.

Quanto ao processamento dos nossos dados, não havia null values, por isso não foi necessário fazer nada quanto a isto. No entanto, havia duplicados e fizemos drop deles.

A partir do nosso objetivo, podemos assumir que certas colunas não causam grande ou praticamente nenhum impacto na nossa previsão, como é o caso de “song” que funciona só como um “id” e de “key”, que nos parece ser uma variável demasiado específica para a nossa previsão. Para além disso o key não nos faria sentido normalizar, visto que funciona também como um “id” para uma nota musical. No entanto a sua escala acaba por ainda ter grandes varações, por isso decidimos dar drop de forma a não influenciar os nossos modelos.

Estamos interessados em analisar os hits entre os anos 2000 e 2019, portanto todos as músicas relativas a anos fora deste intervalo foram eliminadas.

Convertemos a duração de ms para minutos.

Alguns dos géneros não estavam atribuídos, tinham apenas set(). Desta forma, procedemos a procurar os géneros da respetiva música e atribuir aos mesmos. Categorizamos a popularidade em baixa ou alta, baixa para popularidade menor que 65 e alta para maior que 65. Para popularidade alta atribuímos 1 e baixa 0. Removemos a col e 2una popularity.

3.

As colunas de string data “artist” e “genre” tiveram de ser codificadas para as podermos inserir nos nossos modelos. Assim fizemos uma coluna para cada género ou artista e, se música correspondesse a um género ou artista retornávamos 1 e o resto seria tudo 0.

A coluna explícit um booleano, desta forma convertemos também para binário inteiro.

Como o nosso dataset usa escalas diferentes de coluna para coluna é necessário normalizarmos os nossos dados. Como não faz sentido normalizarmos o ano, substituir por uma coluna com idade (2022-ano) e depois, sim, normalizamos a coluna. As colunas já classificadas como 0 e 1 não fazem sentido serem normalizadas. Portanto, as colunas normalizadas são: (ler do pp)

4.

Vamos mostrar algumas visualizações dos nossos dados (antes da classificação e normalização) de forma a compreendermos melhor os dados que estamos a estudar.

Começamos primeiro por mostrar a distribuição das nossas features. (descrever um pouco do que se passa)

5.

Aqui apresentamos a matriz de correlação das entre as nossas features. Onde as duas mais positivas são (pp) e as mais negativas (pp), (descrever relação)

6.

Por fim, visualizarmos a relação entre os géneros e a popularidade. Este tipo de visualização permite-nos tirar algumas informações estatísticas como mediana, quartis, e valores máximo e mínimo. Thus, we see that the **most popular** genre is **pop** and the **least popular** is the combination **folk acoustic, rock**.

7.

Os modelos que usamos na nossa análise foram Decision Tree Classifier, K-Nearest Neighbors e Logistic Regression.

De uma forma muito simples, decision tree classifier cria o modelo de classificação, construindo uma árvore de decisões. K-nearest neighbors usa a “semelhança” das features para prever os valores de qualquer dado novo. Logistic Regression estima a probabilidade de um evento acontecer, sendo dado um dataset de variáveis independentes. Uma vez que o outcome é uma probabilidade, a variável dependente varia entre 0 e 1.

Para cada modelo mostramos a sua accuracy (para o set de teste e treino) e fazemos também K-fold cross validation. Para além disso, apresentamos um classification report/performance metrics, a matriz de confusão e a ROC curve.

8. O nosso dataset foi dividido em 80% para treino e 20% para teste.

Dividimos isto em duas partes, primeiro aplicamos os modelos usando os parâmetros default e, posteriormente, escolhemos os melhores hyper-parametros para cada modelo.

9.

Apresentamos agora os resultados para o DT usando valores default.

Para o classification report usamos as performance metrics: Precision: tells us, out of all the positive predicted, what percentage is truly positive. Recall: tells us, out of the total positive, what percentage are predicted as positive. F1-score: that is the harmonic mean of precision and recall. It takes both false positive and false negatives into account. Macro average: is the usual average we’re used to seeing. Just add them all up and divide by how many there were; Weighted average: considers how many of each class there were in its calculation. E Support:tells how many of each class there were.

10.

Aqui apresentamos a cross validation.

Para melhorar a accuracy fizemos feature importance e corremos o modelo apenas com features cuja importância era maior do que 0.01. No entanto, a mudança da accuracy foi insignificante e deixamos o dataset como estava anteriormente.

11.

Pela accuracy vemos que há, claramente, um problema de overfitting. Há várias técnicas para tentar contornar o overfitting deste modelo nós vamos usar “pruning”. Neste caso “post-pruning” já que o “pre-prunning” é que vamos fazer usando o gridsearchCV. Ele permite o modelo crescer até à sua profundidade máxima, depois remove os ramos das árvores para evitar que o modelo sofra overfitting.

12. pp

13. Uma técnica para melhorar a precisão deste modelo é fazer scalling das nossas features, em vez de as normalizar. A normalização ajusta os valores dos seus dados numéricos a uma escala comum sem alterar a escala, enquanto que o scaling diminui ou estica os dados para se enquadrarem numa escala específica. Infelizmente, ao fazer isto, melhorámos, mas muito pouco. Assim, mantivemos a conjunto de dados anterior (com normalização em vez de escalonamento).

14. pp

15. pp

16.

Aqui fazemos uma comparação dos 3 modelos (descrever gráfico). Por fim, aplicamos um voting classifier. Um Voting Classifier é um modelo de machine learning que treina um conjunto de numerosos modelos e prevê um output baseado na sua maior probabilidade da classe escolhida como output. Ele simplesmente agrega os resultados de cada modelo passados no

Voting Classifier e prevê a classe de saída com base na mais alta maioria de votos. A ideia é, em vez de criar modelos separados dedicados e encontrar a precisão para cada um eles, criamos um modelo único que treina por estes modelos e prevê os resultados com base na sua maioria combinada de votos para cada classe do output.

17.

Agora passamos para o parameter turning usando o gridsearchCV. Para cada algoritmo utilizado, definimos um grupo de parâmetros e executar todos os algoritmos com esses parâmetros. Depois podemos ver que é a melhor combinação de parâmetros, escolhendo a que com a melhor accuracy.

Aqui representamos os gráficos da performance em função dos hyperparametros testados para este modelo. Foi a partir destes que escolhemos os parâmetros que aplicamos no grid-search. Para além da accuracy, tivemos também em conta o tempo de execução.

Hyperparameters:

criterion: function to measure the quality of a split. Supported criteria are “gini” for the Gini impurity and “log.loss” and “entropy” both for the Shannon information gain; • splitter: strategy used to choose the split at each node. Supported strategies are “best” to choose the best split and “random” to choose the best random split; • max.depth: maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min.samples.split samples; • max.features: number of features to consider when looking -for the best split.

18. pp

19

Hyperparameters:

• n.neighbors: number of neighbors to use by default for k-neighbors queries. • n.neighbors: weight function used in prediction. Possible values: ‘uniform’: uniform weights. All points in each neighborhood are weighted equally. ‘distance’: weight points by the inverse of their distance. in this case, closer neighbors of a query point will have a greater influence than neighbors which are further away. [callable]: a user-defined function that accepts an array of distances, and returns an array of the same shape containing the weights. • algorithm:algorithm used to compute the nearest neighbors: ’ball.tree’ will use BallTree; ’kd.tree’ will use KDTree; ‘brute’ will use a brute-force search.

20 pp

21

C:Inverse of regularization strength; must be a positive float. Like in support vector machines, smaller values specify stronger regularization. penalty: Specify the norm of the penalty: - ’none’: no penalty is added; - ’l2’: add a L2 penalty term and it is the default choice; - ’l1’: add a L1 penalty term; - ’elasticnet’: both L1 and L2 penalty terms are added. - solver: Algorithm to use in the optimization problem.

22 pp

23

Podemos concluir agora que obtivemos melhor accuracy quando usamos os parâmetros defaul, para todos os modelos.

24.

falar das conclusoes que estao no pp