UNIVERSIDADE SÃO JUDAS TADEU

DOMINIC NASCIMENTO LAUBHOUET - RA: 822.141.082
EDUARDO VIEIRA DE JESUS - RA: 823.123.930
JULIA MARIA SILVA SIRINO - RA: 822.138.749
MARCOS VINÍCIUS PEREIRA PINTO - RA: 823.122.151
NICOLE LUANA DA SILVA - RA: 822.167.650

ENTREGA 06 – DOCUMENTAÇÃO FINAL DO PROJETO - MODELO DE MACHINE LEARNING

Inteligência Artificial

ENTREGA PARTE DO TRABALHO ACADÊMICO A3 REFERENTE À UC DE INTELIGÊNCIA ARTIFICIAL

SÃO PAULO
2023

DOMINIC NASCIMENTO LAUBHOUET - RA: 822.141.082
EDUARDO VIEIRA DE JESUS - RA: 823.123.930
JULIA MARIA SILVA SIRINO - RA: 822.138.749
MARCOS VINÍCIUS PEREIRA PINTO - RA: 823.122.151
NICOLE LUANA DA SILVA - RA: 822.167.650

ENTREGA 06 – DOCUMENTAÇÃO FINAL DO PROJETO - MODELO DE MACHINE LEARNING

Inteligência Artificial

Entrega 06 (seis - final) do componente de trabalho acadêmico referente à atividade avaliativa A3 apresentado à UC de Inteligência Artificial como parte dos requisitos necessários para pontuação no cálculo final da nota da disciplina

Professor(a): Evandro Catelani Ferraz e Renato Alexandre de Medeiros
Disciplina: Inteligência Artificial
Turma: CCP1AM-BUC1-3198699

São Paulo
2023

RESUMO

A força motriz das Inteligências Artificiais são os dados, que representam um papel fundamental no seu uso, funcionamento e treinamento.

Em vista disso, é inerente a conclusão que a qualidade do modelo de Inteligência Artificial está diretamente proporcional à qualidade dos dados usados durante seu desenvolvimento. Dados mais precisos, relevantes, consistentes e completos tornam modelos de IA mais eficientes e exatos em uma ampla gama de cenários.

Logo, escolher um bom *database* para ser usado durante o desenvolvimento do software de Inteligência Artificial é uma etapa essencial para garantir sua qualidade, usabilidade e capacidade de ser atualizado.

Portanto, nós, como grupo, escolhemos uma base de dados que não só esteja de acordo com os objetivos que buscamos com o modelo de IA, mas que também cumpram requisitos de excelência para que o software desenvolvido atinja os objetivos que estabelecemos para nosso projeto.

ABSTRACT

The driving force of Artificial Intelligence is data, which represents a fundamental role in its use, operation and training.

Given this, it is inherent to conclude that the quality of the Artificial Intelligence model is directly proportional to the quality of the data used during its development. More accurate, relevant, consistent, and complete data makes AI models more efficient and accurate across a wider range of scenarios.

Therefore, choosing a good database to be used during the development of Artificial Intelligence software is an essential step to guarantee its quality, usability and ability to be updated.

Hence, we, as a group, chose a database that not only complies with the objectives we seek with the AI model, but also meets excellence requirements so that the software developed achieves the objectives we set for our project.

SUMÁRIO

1. Introdução.....4
2. Documentação de Tabelas e Colunas.....5
3. Documentação da Variável Target.....9
4. Documentação das Transformações.....10
5. 1° Modelo - Árvore de Decisão.....12
5. 2° Modelo - Rede Neural MLP.....14
6. 3° Modelo - Naive Bayes.....17
7. Conclusão.....18
8. Fontes Bibliográficas.....19

INTRODUÇÃO

É sabido que a Inteligência Artificial, em especial o campo de estudo do Aprendizado de Máquina (*Machine Learning* - ML) são extremamente dependentes de dados e de sua qualidade. Diante disso, entender a organização e o que cada dado representa dentro de um *database* é essencial para julgar não só sua integridade, mas o quão útil ele é para ser usado no treinamento e desenvolvimento de modelos de IA e ML.

As Tabelas são um dos meios mais comuns de se organizar dados e são empregadas em inúmeros implementações de ML e IA e, para verificar sua qualidade, é necessário entender que tipo de dado cada coluna e o que eles representam.

Tendo em vista isso será efetuada nas seções subsequentes uma análise profunda de todas as tabelas e suas respectivas colunas assim como da variável *target* se será prevista com base nas interações entre os outros dados.

DOCUMENTAÇÃO DE TABELAS E COLUNAS

A *database* selecionada é composta de 3 tabelas diferentes, todas no formato de Valores Separados por Vírgulas (.csv), que garante simplicidade e legibilidade aos dados, sendo elas; *train.csv*; *test.csv* e *submission.csv*.

A tabela *train.csv* é a principal tabela do database, contendo todos os dados necessários para treinar o modelo de Machine Learning, que inclui 17.996 linhas distribuídos em 17 colunas, sendo elas:

- **Artist Name** *(Nome do Artista)* – Coluna que armazena dados em formato *string* a qual representa o(s) nome(s) do(s) artista(s) que participaram da faixa musical;
- **Track Name** *(Nome da Faixa)* – Coluna que armazena dados em formato *string* e representa o nome da faixa musical cuja informações estão reunidas na linha correspondente;
- **Popularity** *(Popularidade)* – Coluna cujos dados são armazenados em formato *int* e transmitem a ideia do quão aceita uma música é recebida pelo público, onde maiores valores representam maior popularidade;
- ***Danceability*** *(Capacidade de Dança)* – Coluna que armazena dados em formato *float* que com base em características como ritmo, andamento, estabilidade e batida da música mede o quão adequada ela é para ser dançada;
- **Energy** *(Energy)* – Coluna que retém dados no formato *float* e refere-se a sua intensidade e entusiasmo, onde valores numéricos mais altos são traduzidos como músicas mais empolgantes, rápidas e barulhentas, enquanto valores mais baixos representam músicas mais calmas, lentas e suaves;
- **Loudness** *(Volume)* – Coluna que armazena dados em formato *float* e que indica a intensidade do som, se aquela música é mais alta ou baixa;
- **Mode** *(Modo)* –Coluna que armazena dados em formato int e indica a modalidade da música (Maior ou Menor), ou seja, a escala usada na melodia da música.
    - Por exemplo, uma música C maior tem a escala musical mais ‘feliz’ (C, D, E, F, G, A, B – Intervalos que seguem uma sequência específica de

tons e semitons, com uma progressão mais aberta e arejada); enquanto uma música C menor tem a escala musical mais 'melancólica' (C, D, E♭, F, G, A♭, B – Notas 'achatadas', com alterações nos intervalos de tons e semitons).

- **Speechiness** *(Intensidade da Fala)* – Coluna que armazena dados em formato *float* e detecta a presença de palavras em uma faixa, indicando se a música é mais falada/cantada – se a música tiver esse valor numérico alto, ela terá letras e vocais mais sobressaídos;
- **Acousticness** *(Intensidade Acústica)* – Coluna que armazena dados em formato *float* representando se a música tem mais elementos acústicos do que elementos eletrônicos;
- **Instrumentalness** *(Intensidade Instrumental)* – Coluna que armazena dados em formato *float* e representa se a música é mais instrumental, onde valores maiores significando instrumentos serem mais ressaltados;
- **Key** *(Tom)* – Coluna que armazena dados em formato *float* que representa a "nota musical" em que a faixa é performada com base na notação padrão de tom musical, ou seja, a tonalidade em que uma peça musical está escrita, sendo sua nota fundamental e a escala utilizada na composição.
    - Por exemplo, se uma música tem a tonalidade de C# (C sustenido) maior, isso quer dizer isso que a nota fundamental da música é o C sustenido maior, e a escala utilizada será a escala maior de C sustenido (C#, D#, E#, F#, G#, A#, B#).

Classificação de Tom Musical

Classificação Numérica de Tonalidade	Contrapartes tonais	Solfejo
0	C (também B ♯, D 𝄫)	do
1	C♯, D ♭ (também B 𝄪)	
2	D (também C 𝄪, E 𝄫)	ré
3	D♯, E ♭ (também F 𝄫)	
4	E (também D 𝄪, F ♭)	mi
5	F (também E ♯, G 𝄫)	fa
6	F♯, G ♭ (também E 𝄪)	
7	G (também F 𝄪, A 𝄫)	sol
8	G♯, A ♭ _	
9	A (também G 𝄪, B 𝄫)	la
10, t ou A	A♯, B ♭ (também C 𝄫)	
11, e ou B	B (também A 𝄪, C ♭)	si

Figura 1.1 - Notação Padrão de Tom Musical

- **Liveness** *(Vivacidade)* – Coluna que armazena dados em formato *float* se refere à detecção da presença de público na gravação, onde valores mais altos indicam que há maior probabilidade que a faixa foi gravada ao vivo e valores mais baixos indicam menor probabilidade;
- **Valence** *(Valência)* – Coluna que armazena dados em formato *float* representando se a música é mais positiva ou negativa. Altas valências indicam maior alegria, enquanto uma música com baixa valência será mais triste;
- **Tempo** *(BPM)* – Coluna que armazena dados em formato *float* que representa o BPM (Batidas por Minuto) de uma música;
- **Duration_in min/ms** – Coluna cujos dados armazenados em *float* representam a duração de uma música em milissegundos;
- **Time_signature** – Coluna que armazena dados em formato *int* que descreve a estrutura rítmica da música, especificando quantas batidas há em cada compasso da música. Por exemplo, uma faixa com assinatura de tempo 5 significa uma estrutura rítmica de 5/4;
- **Class** – Coluna que armazena dados em formato *int* os quais representam o gênero musical que determinada música pertence.

Já a tabela *test.csv* é uma tabela com valores não rotulados, contendo as mesmas colunas, com os mesmos nomes e mesmos tipos de dados da tabela *train.csv*; com exceção da coluna *class* (que foi removida da tabela), totalizando 7.713 linhas em 16 colunas. O objetivo principal contendo é testar o modelo de machine learning desenvolvido em dados “não vistos” por ele antes.

Por fim, a tabela *submission.csv* também possui 7.713 linhas, porém, em 11 colunas. Por se tratar de um database criado para uma competição Hackathon, a tabela possui algumas informações sobre como enviar os resultados obtidos com a predição da tabela test.csv; onde as primeiras sete linhas armazenam uma matriz identidade e as restantes armazenam apenas valores *int* iguais a 0.

O cabeçalho de coluna dessa tabela possui um índice, que relaciona o valor *int* da coluna *class* com um gênero musical que o database permite o treinamento de um modelo de IA:

- **Acoustic/Folk** – 0;
- **Alt Music** – 1;
- **Blues** – 2;

- **Bollywood** – 3;
- **Country** – 4;
- **HipHop** – 5;
- **Indie/Alt** – 6;
- **Instrumental** – 7;
- **Metal** – 8;
- **Pop** – 9;
- **Rock** – 10;

DOCUMENTAÇÃO DA VARIÁVEL TARGET

A variável *target* (ou variável dependente) pode ser descrita como a variável que deseja ser prevista ou estimada a partir de um modelo de Machine Learning, chamada de *target* à medida que é o alvo e interesse dentro de um conjunto de dados.

A partir de algoritmos de Machine Learning, é possível realizar manipulações matemáticas sob os dados para encontrar padrões e/ou fórmulas matemáticas que sejam capazes de descrever as relações entre os dados e possam ser usadas para estimar valores desconhecidos, gerando então um modelo de IA baseado em Machine Learning.

A variável target do database escolhido para treinar os modelos de machine learning será a variável ***class***, armazenada na tabela *train.csv* que é responsável por rotular o gênero musical que uma faixa se encaixa.

Ela possui 11 possíveis resultados que são influenciados pelas outras variáveis (independentes).

Como cada gênero musical possui sua própria assinatura - características, que, independentemente da faixa, são semelhantes por todo o estilo (como por exemplo, músicas pop possuem normalmente um BPM entre 110 a 150). Isso torna a escolha de ***class*** como variável target extremamente apropriada, tendo em vista que a database possui um amplo conjunto de faixas dos mais diversos tipos, rotuladas e com as métricas que permitem encontrar padrões de cada categoria.

DOCUMENTAÇÃO DAS TRANSFORMAÇÕES

Para garantir que a *database* esteja nos padrões de qualidade desejados, é necessário analisar as suas tabelas e verificar se é preciso transformar os dados que armazenam com o objetivo de adequá-la para o uso no treinamento, em um processo de **Limpeza de Dados**, à medida que dados brutos podem conter erros que afetam a precisão do modelo de Machine Learning, levando a previsões incorretas.

É necessário localizar e corrigir dados duplicados; dados irrelevantes; exceções; dados ausentes e erros estruturais.

Na *database* escolhida há 3 tabelas, das quais será usada somente 1 - *trains.csv* – que será renomeada para “*musicas_categorizadas.csv*” (a tabela *submission.csv* não será usada diretamente, apenas o seu cabeçalho, que relaciona os valores da variável *class* com gêneros musicais e a tabela *test.csv* será descartada à medida que serão usados dados de músicas obtidas pela API do Spotify mediante entrada de usuário).

Das colunas da tabela *train.csv*, serão excluídas as que armazenam o nome do artista (*Artist Name*) e o nome da faixa (*Track Name*), à medida que não contêm valores numéricos e não apresentam repetição frequente o suficiente para serem consideradas significativas. Além disso, músicas com nomes idênticos podem pertencer a gêneros diferentes, como "*Karma*" de Taylor Swift e "*Karma*" de Alicia Keys e um único artista pode criar músicas de gêneros diversos, como as faixas "*Would You - Acoustic*" e "*Sometimes it Rains in LA*" de The Vamps.

A coluna *Popularity* também será desconsiderada, a medida que, mesmo representando um valor numérico, é extremamente volátil, sujeita a mudanças significativas com o passar do tempo (o que pode torná-la imprecisa se comparada com os valores atuais, já que o *database* foi publica em 2021). Além disso, músicas dentro do mesmo gênero podem apresentar amplas variações em termos de popularidade (como por exemplo “*Follow Through*” de David Kennedy com 64% de popularidade e “*Full Throttle*” de Ben Schuller com 33%), tornando-a uma métrica inadequada para o desenvolvimento e treinamento de um modelo de Machine Learning.

A coluna *duration_in min/ms* também será desprezada, à medida que a database já conta com uma vasta gama de características das faixas, os gêneros

musicais que serão previstos não possuem grandes diferenças na duração média de suas músicas e músicas específicas como *All To Well (10 Minutes Version)* da Taylor Swift podem não ser classificadas de maneira correta pelo algoritmo treinado com os valores de duração.

Logo, após a retirada das 4 colunas (*Artist Name, Track Name, Popularity* e *duration_in min/ms*), restam 13 colunas (*danceability, energy, key, loudness, mode, speechiness, acousticness, instrumentalness, liveness, valence, tempo, time_signature* e *Class*) na tabela "*musicas_categorizadas.csv*".

No entanto, ainda será necessário imputar dados, à medida que existem valores faltantes nas colunas *key* e *instrumentalness* em ambas as tabelas. Em *train.csv*, estão ausentes 2014 valores na coluna *key* e 4377 na coluna *instrumentalness*, enquanto na tabela *test.csv*, há 808 valores faltantes em *key* e 1909 em *instrumentalness*.

Para corrigir esse problema, dados faltantes na coluna *key* serão imputados usando a abordagem empregada pelo Spotify em sua Web API, onde valores nulos de key serão transformados em -1.

Já na coluna *instrumentalness,* será adotado um procedimento semelhante, usando a média dos valores de *instrumentalness* dos gêneros musicais para realizar a imputação dos dados faltantes.

1° MODELO - ÁRVORE DE DECISÃO

Para classificar os gêneros musicais com base em seus dados numéricos que representam seus aspectos essenciais das músicas, usar **Árvores de Decisão** é uma decisão pertinente. Essas características são importantes, mas sua relação não linear e influência variável na classificação demandam uma abordagem flexível.

As árvores de decisão, ao considerarem esses atributos de forma hierárquica, podem mapear os padrões existentes entre elas de maneira eficiente.

O algoritmo de árvore de decisão tem estrutura visual que recorda a de um fluxograma, o que permite fácil entendimento e visualização.

Seu funcionamento é análogo à uma estrutura de tomada de decisões, tornando-a acessível, conseguindo lidar tanto com problemas de regressão como de classificação.

Ela se estabelece em **nós** (*decision nodes*) que se relacionam entre si por uma hierarquia – há dois tipos de nós: o **nó-raiz** (*root node*) que é o mais importante e os **nós-folha** (leaf nodes). No contexto do Machine Learning, a raiz é um dos atributos da base de dados e as folhas são a classe ou valor que busca como resultado.

Esses nós são ligados por caminhos, chamados de ramos, escolhidos a partir das condições e conferências realizadas pelos nós.

Figura 2.1 - Exemplo simples de árvore de decisão com 2 nós e 4 ramos.

Nos ramos, encontram-se regras semelhantes às estruturas de controle "*if-else*" (por exemplo, se a **X > 15**, o caminho será para o **nó 1**; caso contrário, o caminho será rumo o **nó 2**).

Uma árvore de decisão é um algoritmo chamado de recursivo (conceito que define funções que chamam a si mesmo), ou seja, repete o mesmo padrão sempre na medida em que entram novos níveis de profundidade – O algoritmo se invoca em cada nó, tentando encontrar a melhor divisão de dados.

Essa melhor divisão de dados é a tarefa que a árvore de decisão precisa cumprir com objetivo de encontrar que nós serão encaixados em cada posição, qual será o nó raiz, etc.

Para realizar esses cálculos e obter resultados satisfatórios, uma abordagem comum é usar a **"impureza" dos dados** – quanto mais alta a impureza, menor a uniformidade dos dados – junto ao **ganho de informações**, que mede a mudança na entropia quando usa-se um atributo para dividir os dados – quanto maior o ganho da informação, melhor o atributo usado e maior a redução da incerteza sobre a classe que se quer prever.

Primeiro, se calcula a impureza da classe saída (seja por meio da entropia ou do índice de Gini), calcula-se o ganho de informações para cada atributo analisado e escolhe-se o atributo cuja métrica é maior; repetindo o processo, criando novos nós e ramos.

No modelo desenvolvido para o database escolhido leva como critério de impureza o índice de Gini; a divisão de cada nó é baseada na melhor divisão possível com base na impureza; a profundidade máxima da árvore é 10 (ou seja, o comprimento mais longo de uma raiz até uma folha é 10 – ou seja, o número máximo de decisões consecutivas de decisões é 10); o número mínimo de amostras necessárias para formar uma folha é 1 enquanto o número mínimo de amostras para se dividir um nó interno é de 10. Por fim, o estado aleatório é de 42, que define a aleatoriedade do estimador.

```
clf = DecisionTreeClassifier(criterion = 'gini', splitter = 'best', max_depth = 10, min_samples_leaf = 1, min_samples_split = 10, random_state = 42)
```

2.2 - Código que define o modelo de árvore de decisão

O modelo de IA possui precisão de 52.28%

2° MODELO - REDES NEURAIS MLP

A utilização de redes neurais de *perceptron* multicamada (MLP) também é uma abordagem eficiente para a classificação de gêneros musicais com base em seus metadados numéricos, a medida que são capazes de abstrair padrões complexos e relações não lineares nos dados, características cruciais para lidar com características musicais, as quais que sofrem influências sutis e possuem interações multifacetadas.

As MLPs são formadas de duas partes principais: Os **neurônios** e as **conexões** entre eles.

Um neurônio armazena um número entre 0 e 1, que representa sua “ativação”, obtida a partir de uma função específica.

Ao agrupar neurônios e conectá-los, camadas são criadas. Uma rede neural possui diversas camada, inclusive camada entre o *input* e *output* dos modelos (camadas ocultas).

Definir o número de camadas ocultas bem como a quantidade de neurônios presentes em cada uma é uma decisão arbitrária, tomada durante o desenvolvimento com o objetivo de aumentar a eficiência do modelo.

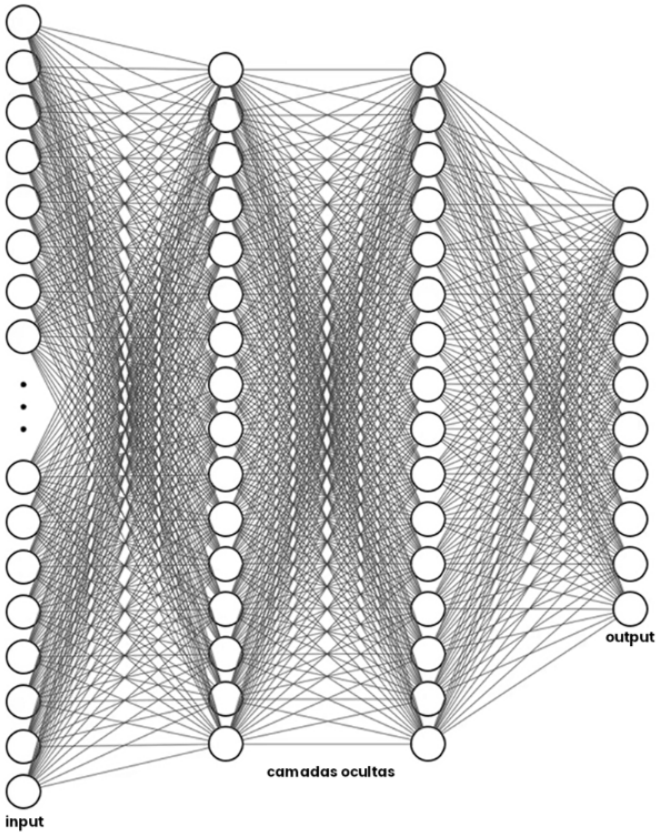

Figura 3.1 - Estrutura de uma Rede Neural de 4 camadas (1 input, 1 output e 2 ocultas)

Para que essa rede funcione, as ativações de uma camada determinam as ativações na próxima junto aos pesos de cada uma das conexões (um parâmetro numérico que determinam a importância relativa das entradas para a ativação de um neurônio seguinte) por meio do cálculo da soma ponderada desses valores:

$$w1a1 + w2a2 + w3a3 + w4a4 + ... + wNaN$$

Onde ***w*** é o valor do peso da conexão e ***a*** é o valor da ativação do neurônio.

No entanto, a soma ponderada pode resultar em qualquer número real, mas as ativações devem estar entre 0 e 1, logo é comum usar funções capazes de reduzir o valor da soma para um intervalo entre 0 e 1; como a função Sigmóide; a ReLU (*Rectified Linear Unit*); a Tangente (tanh) ou a Softmax.

Outro fator que pode influenciar o valor de ativação de um neurônio é a sua tendência (ou *bias*), que pode ser traduzida como uma tendência do neurônio ser ou não ser ativado, sendo muitas vezes adicionado à soma ponderada antes de transformá-la com as funções de redução.

O seu treinamento envolve algoritmos com o objetivo de otimizar os valores dos pesos e tendências das conexões para adequá-los da melhor maneira possível ao conjunto de dados de treino, buscando minimizar a diferença entre as previsões geradas e os rótulos reais associados às amostras de treino, encontrando uma configuração de pesos que seja uma boa generalização e eficiente para realizar previsões precisas e úteis para novos exemplos além do conjunto de treinamento.

Há diversos algoritmos que permitem realizar essa tarefa, como SGD (Descida de Gradiente Estocástica); LBFGS (algoritmo BFGS – Broyden-Fletcher-Goldfarb-Shanno – com memória limitada) e adam (Estimação Adaptativa de Momento).

Todos esses algoritmos buscam ajustar iterativamente os parâmetros do modelo para minimizar a função de custo (função que quantifica o quão distantes as previsões do modelo estão dos valores reais nos dados de treinamento), implementam esse conceito de maneiras diferentes.

No modelo desenvolvido para o problema de classificação, são especificadas 3 camadas ocultas com tamanhos de 50, 100 e 50 neurônios, respectivamente. A função de ativação 'tanh' é empregada, enquanto o otimizador 'adam' é escolhido para guiar o processo de aprendizado. Já o parâmetro de regularização alpha que controla a complexidade do modelo é configurado como 0.05. A taxa de

aprendizado é constante, e o número máximo de iterações é definido como 200. O estado aleatório é fixado em 42, contribuindo para a reprodutibilidade do modelo e sua aleatoriedade controlada durante o treinamento.

```
clf = MLPClassifier(hidden_layer_sizes=(50, 100, 50), activation='tanh', solver='adam', alpha=0.05, learning_rate='constant', max_iter=200, random_state=42)
```

Figura 3.2 - Código da Rede Neural MLP

3° MODELO - NAIVE BAYES

O algoritmo Naive Bayes é um método de classificação que também pode ser usado para classificar o gênero musical a partir das características de uma faixa.

Baseado no teorema de Bayes, o é chamado de “*naive*” (ingênuo) à medida que assume no que todas as características (*features*) em uma base de dados são independentes entre si.

Ele cria uma tabela de probabilidades a partir de uma técnica de classificação de dados, assumindo que a presença de uma característica específica não está relacionada a qualquer outra característica presente na base de dados.

Ele modela a distribuição de probabilidade de cada classe, calculando a probabilidade de um ponto de dados pertencer a uma determinada classe, dada a distribuição.

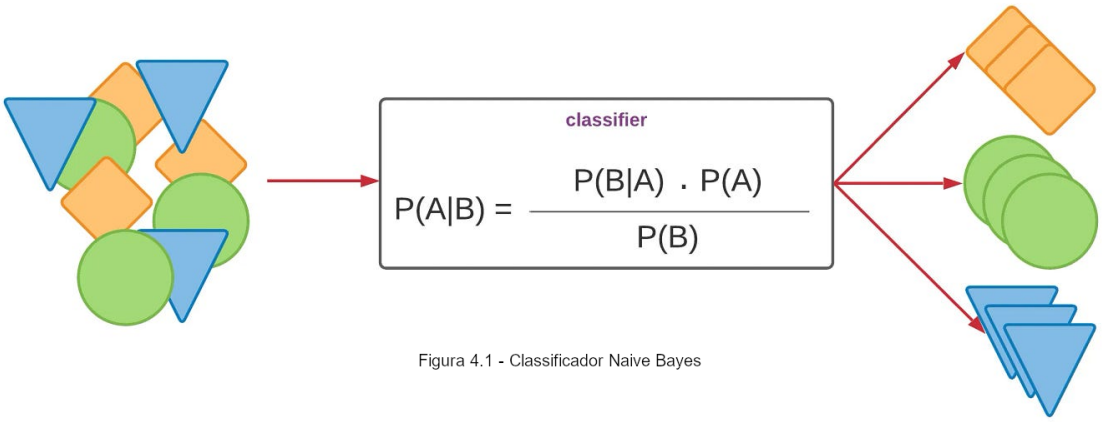

Figura 4.1 - Classificador Naive Bayes

CONCLUSÃO

Por mais úteis e inovadoras que sejam, criar soluções de Inteligência Artificial e Machine Learning é algo extremamente complicado. No decorrer deste projeto, usamos três modelos de ML para classificar músicas em onze gêneros distintos e o mais preciso deles (Árvore de Decisão) teve acuracidade de somente 52%.

A primeira observação que emerge da análise da precisão alcançada é a possível influência da base de dados utilizada – por não estar normalizada, com quantidades proporcionais de amostras para cada gênero, isso gera vieses significativos, impactando diretamente os resultados.

Ademais, classificar músicas em gêneros musicais é algo de extrema complexidade intrínseca do problema em questão - sua natureza subjetiva e dinâmica, torna as fronteiras entre os gêneros difusas, o que afeta o desempenho dos modelos.

Outro fator é a quantidade de resultados possíveis, o que pode ter dificultado a tarefa de generalização dos dados, diminuindo a acuracidade dos modelos.

Logo, é intrínseco concluir que, para desenvolver um bom modelo de IA que seja preciso e tenha uso amplo no mundo real, é necessário não só considerar cuidadosamente a qualidade e a representatividade da base de dados, mas também pré-processá-la da melhor forma possível. Além disso, reconhecer a complexidade do problema que deseja ser resolvido e suas limitações pode tornar o desenvolvimento menos frustrante, além de permitir que novas abordagens sejam adotadas com para tornar os resultados ainda mais satisfatórios.

FONTES BIBLIOGRÁFICAS

Figura 1.1 – Pitch class. Disponível em: <https://en.wikipedia.org/wiki/Pitch_class> - Acesso em 25 out. 2023.

Figura 2.1 – Como funciona o algoritmo de Árvore de Decisão (Decision Tree). Disponível em: <https://didatica.tech/como-funciona-o-algoritmo-arvore-de-decisao/>. Acesso em 15 nov. 2023.

Figura 2.2 - VINÍCIUS, Marcos “DeVinc1”. PROJETO A3 - Inteligência Artificial e Machine Learning Disponível em: <https://github.com/DeVinc1/PROJETO-A3-Inteligencia-Artificial-e-Machine-Learning/blob/main/ENTREGA%2004/Modelo%20de%20Predi%C3%A7%C3%A3o%20de%20G%C3%AAnero%20Musical%20-%20%C3%81rvore%20de%20Decis%C3%A3o.ipynb>. Acesso em: 15 nov. 2023.

Figura 3.1 - But what is a Neural Network? | Deep learning, chapter 1. Disponível em: <https://www.youtube.com/watch?v=aircAruvnKk&list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi>. Acesso em: 22 nov. 2023.

Figura 3.2 - VINÍCIUS, Marcos “DeVinc1”. PROJETO A3 - Inteligência Artificial e Machine Learning Disponível em: <https://github.com/DeVinc1/PROJETO-A3-Inteligencia-Artificial-e-Machine-Learning/blob/main/ENTREGA%2005/Modelo%20de%20Predi%C3%A7%C3%A3o%20de%20G%C3%AAnero%20Musical%20-%20Rede%20Neural%20MLP.ipynb> Acesso em: 22 nov. 2023.

Figura 4.1 - AJAYMEHTA. “Demystifying Naïve Bayes: Simple yet Powerful for Text Classification”. Disponível em: <https://medium.com/@dancerworld60/demystifying-na%C3%AFve-bayes-simple-yet-powerful-for-text-classification-ad92b14a5c7>.

Music Genre Classification. Disponível em: <https://www.kaggle.com/datasets/purumalgi/music-genre-classification> Acesso em: 25 out. 2023.

SPOTIFY. Web API Reference | Spotify for Developers. Disponível em: <https://developer.spotify.com/documentation/web-api/reference/get-audio-features>. Acesso em: 25 out. 2023.

PARSONS, C. O que é um Modelo de Machine Learning? | Blog da NVIDIA. Disponível em: <https://blog.nvidia.com.br/2021/09/28/o-que-e-um-modelo-de-machine-learning/>. Acesso em: 31 out. 2023.

Entenda: Aprendizado Supervisionado ou Não Supervisionado. Disponível em: <https://didatica.tech/aprendizado-supervisionado-ou-nao-supervisionado/>. Acesso em: 31 out. 2023.

KIM, M. The secret math behind feel-good music. Disponível em: <https://www.washingtonpost.com/news/to-your-health/wp/2015/10/30/the-mathematical-formula-behind-feel-good-songs/>. Acesso em: 31 out. 2023.

O que é Limpeza de dados? — Limpeza de dados explicada — AWS. Disponível em: <https://aws.amazon.com/pt/what-is/data-cleansing/>. Acesso em: 1 nov. 2023.

SACRAMENTO, G. Árvore de decisão: entenda esse algoritmo de Machine Learning. Disponível em: <https://blog.somostera.com/data-science/arvores-de-decisao>. Acesso em: 15 nov. 2023.

But what is a Neural Network? | Deep learning, chapter 1. Disponível em: <https://www.youtube.com/watch?v=aircAruvnKk&list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi>.

O que é Naive Bayes e como funciona esse algoritmo de classificação. Disponível em: <https://rockcontent.com/br/blog/naive-bayes/>.