**UNIVERSIDADE DE SÃO PAULO**

**FACULDADE DE FILOSOFIA, CIÊNCIAS E LETRAS DE RIBEIRÃO PRETO**

**DEPARTAMENTO DE COMPUTAÇÃO E MATEMÁTICA**

**CIÊNCIA DA COMPUTAÇÃO**

Larissa Magalhães Pereira - 13747904 - lamagalhaespereira@usp.br

Lucas Antonio P. dos Santos - 13861180 - lucasantonio.sanstos@usp.br

**REDES NEURAIS ARTIFICIAIS**

CLASSIFICAÇÃO DE MÚSICAS EM PLAYLISTS

RIBEIRÃO PRETO

2023

**1. INTRODUÇÃO**

Este trabalho possui o objetivo de, dada duas playlists públicas do aplicativo Spotify, criar uma rede neural do tipo MLP (Multilayer Perceptron) para classificar qualquer música dada na playlist mais adequada ao analisar os padrões das músicas das duas playlists e os atributos da música dada.

Foi utilizada a plataforma Google Colab para todo o código python3 desenvolvido ([trabalho RNA.ipynb](https://colab.research.google.com/drive/1d0ILALyZzolhIk0CZZuuNyIfcXMF1tnj?usp=sharing)). As bibliotecas utilizadas foram:

* Requests: Realizar requisições http
* Json, csv, pandas, numpy: Manipulação dos dados
* Sklearn: Desenvolvimento da rede neural e manipulação dos dados
* Matplotlib, seaborn: Imprimir gráficos

**2. METODOLOGIA**

Primeiramente, os dados são adquiridos da API web do aplicativo Spotify, conforme as suas diretrizes e termos de serviços. Com a biblioteca requests foi realizado requisições https com uma playlist ou track (música) para adquirir informações sobre em um arquivo json.

Utilizando a biblioteca csv, são criados dois arquivos em CSV - “playlist 1‘ e “playlist 2” - que armazenam os dados das músicas (tracks) de duas playlists escolhidas. Os dados das tracks armazenados são:

* id: identificação (string)
* name: nome (string)
* popularity: popularidade (numérico)
* release date: data de lançamento (string)
* duration: duração da música (numérico)
* explicit: se é ou não explícita (string)
* artists: lista dos artistas (lista de strings)
* genres: lista dos gêneros musicais (lista de strings)
* class: classe, no caso, playlist 1 ou playlist 2 (string)

Posteriormente, é feita a junção dos dois arquivos criados em um único arquivo CSV, nomeado de “database”.

**Código:** classe *‘SpotifyApi’;* métodos *‘writePlaylist’ e ‘writeDataBase’*

A próxima etapa é o tratamento dos dados do database para ser possível sua utilização pela Rede Neural. Inicialmente, os dados em formato de string são transformados em dados numéricos para ser possível sua normalização. Para isso são utilizados o *OrdinalEncoder* do sklearn.preprocessing e a biblioteca numpy. Depois, os dados numéricos, exceto a classe, são normalizados por meio do uso do *MinMaxScaler* da sklearn.preprocessing.

**Código:** método *‘dataProcessing’,* método *‘prepDataFrame’ e* método *‘normalizeDataFrame’*

Após a normalização, se inicia o treinamento e teste da MLP. Para isso foi criada a classe MLP, que contém os métodos classifier, confusionMatrix e getWrongResults e é inicializada com:

* self.mlp: objeto da classe MLPClassifier do scikit-learn é inicializado para representar o classificador MLP, com 1000 como número máximo de iterações (max\_iter), Adam como algoritmo de otimização (solver), 106 neurônios na camada oculta (hidden\_layer\_sizes) e RELU como função de ativação (activation)
* self.df: referência ao dataframe original
* self.dfNorm: normalização do dataframe
* self.entries: dados de entrada (features) são extraídos do dataframe normalizado excluindo a coluna 'class' id, name, popularity, release date, duration, explicit, artists e genres
* self.classes: dados de saída (classe) são extraídos da coluna 'class' do dataframe normalizado
* self.spotifyApi: objeto da classe SpotifyApi é criado para lidar com interações com a API do Spotify

O método classifier utiliza cross\_val\_predict do scikit-learn para realizar a validação cruzada - usando como modelo o self.mlp (MLPClassifier), como entradas (x) o self.entries e saídas (y - classes) o self.classes e o número de folds (cv\_folds=5) passado por parâmetro na função classifier - gerando previsões para cada parte do conjunto de dados. Depois, ele chama os métodos confusionMatrix e getWrongResults.

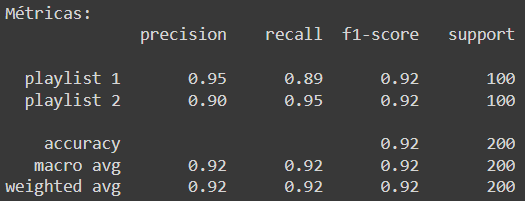
O método confusionMatrix imprime - usando as bibliotecas seaborn, matplotlib.pyplot e sklearn (*metrics*) - a matriz de confusão em um heatmap e exibe scores da validação cruzada e métricas do classificador, como precisão, recall e f1-score.

O método getWrongResults imprime informações sobre as músicas que o classificador errou durante o teste. Ele compara as previsões do modelo com as classes reais e imprime os detalhes das músicas para as quais houve discrepância.

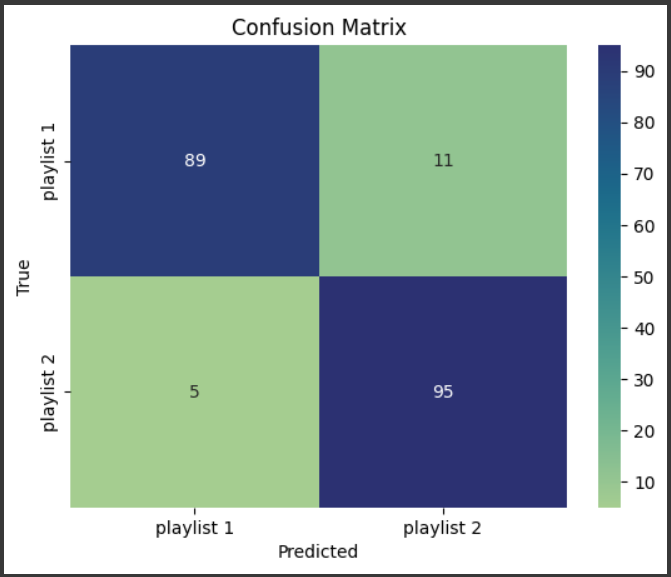
**Código:** classe *‘MLP’*, métodos *‘classifier’, ‘confusionMatrix’* e *‘getWrongResults’*

**3. RESULTADOS**

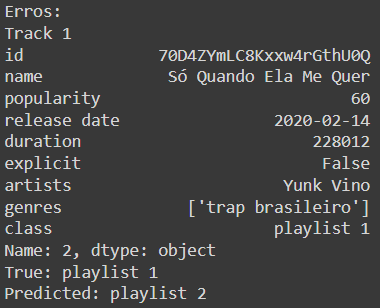
Métricas do classificador:



Matriz de Confusão:



Exemplo de erro do classificador (parte da saída do getWrongResults):



…

**4. CONCLUSÃO**

A implementação da Rede Neural MLP (Multilayer Perceptron) revelou-se eficaz para o problema em questão (classificação de músicas em playlists), apresentando uma ótima acurácia de 92%, evidenciando a capacidade do modelo em aprender padrões complexos em conjuntos de dados musicais.

No entanto, ao analisar os erros cometidos, percebe-se uma recorrência significativa no atributo explícito (explicit) das músicas. Observando a matriz de confusão e os resultados do método ‘getWrongResults’ com as músicas com classificação incorreta, identificamos que o erro de 11 músicas está possivelmente relacionado ao atributo explícito (explicit). Cinco músicas da playlist 2, com explicit = True, foram erroneamente previstas para a playlist 1, enquanto seis músicas da playlist 1, com explicit = False, foram previstas incorretamente para a playlist 2. Porém, as restantes cinco músicas foram músicas da playlist 1, com explicit = True, classificadas imprecisamente como pertencentes à playlist 2, sem ser possível identificar a origem específica do erro.

A complexidade intrínseca das playlists musicais, caracterizadas por uma variedade de músicas sem regras fixas e pela presença de faixas similares ou até mesmo idênticas em diferentes playlists, destaca a natureza desafiadora desse problema. Dado a variabilidade e subjetividade envolvidas nas preferências musicais, é difícil atingir uma acurácia de 100% na classificação.

Diante desse cenário, uma abordagem promissora para a melhoria do desempenho do modelo é a inclusão de mais dados. A expansão do conjunto de treinamento com uma variedade ainda maior de músicas em cada playlist oferece ao MLP a oportunidade de aprender nuances adicionais, ter uma visão mais abrangente e diversificada das características das playlists, e, assim, aprimorar sua capacidade de classificação, reduzindo a incidência de erros.

Em conclusão, o estudo destaca não apenas os sucessos, mas também os desafios intrínsecos à classificação de playlists musicais com MLP. A busca por uma acurácia perfeita pode ser ilusória dadas as peculiaridades desse problema, mas a contínua expansão e diversificação dos dados oferecem um caminho promissor para aprimorar a capacidade do modelo de compreender a complexidade e a subjetividade da experiência musical.