Classificação de gêneros musicais

Filipe Marques de Souza

Abstract

Dado o tema livre do trabalho, sendo necessário como base implementar um dos algorítimos visto em sala, para treinamento ou classificação de dados de um data set sendo assim o trabalho abaixo apresenta cinco métodos diferentes para classificar gêneros musicais, sendo eles regressão logica, Naive Bays, KNN-Classifier, Arvore de decisão, Floresta randômica. fazendo uso de um data set obtido na plataforma kaggle,*Data Set*. Dessa forma no decorrer desse artigo iremos explorar um poco mais sobre esse data set e o código para ver o funcionamento dele.

1 Introdução

Classificação de genero musical é um processo que demanda um que pode ser custoso no momento de criação e catalogação da musica, com isso o intuito desse projeto é justamente treinar modelos que sejam capazes de realizar esse processo afim de facilitar a catalogação desse processo

O trabalho aborda 5 sistemas de classificação bem conhecidos na literatura que podem ser treinados para uma melhor execução por base de determinados parâmetros que possibilitam a identificação de qual gênero a musica pertence, sendo assim o objetivo é comparar esses modelos entre si para ver qual pode ter a melhor capacidade e precisão para catalogar as musicas

2 Metodologia

A metodologia utilizada não foi algo complexo, com o decorrer desse capitulo demonstrarei parte do código usado para uma melhor abstração do conteúdo

A primeira parte do projeto era entender quais eram as métricas para o objetivo do trabalho partindo as características, para assim então partimos para o modelo usa, no caso desse projeto a comparação dos 5, ou seja, tive que escolher 5 métodos de classificação para poder observar um resultado melhor que possibilitasse uma melhor analise de como de qual o melhor algorítimo para esse data set

Com a dado o numero de possíveis classificações do projeto sendo dezesseis possibilidades a tendo ja como base a entrada com todos os dados pertinentes, podemos definir as métricas em:

- accuracy
- precision
- recall
- f1_score

```
df_metrics = pd.DataFrame(index=["name", "accuracy",
"precision", "recall", "f1_score"])
```

Com as métricas definidas para o que queríamos obter, parimos para o tratamento dos dados, dado que alguns dados são inicialmente não necessários para o processo de treinamento dos modelos, para que não aja nenhum problema quanto a integridade do resultado final, a melhor opção é eliminar esses itens removendo qualquer empecilho que venha a causar um problema no final do código, caso seja necessário

```
Y = df_train["Class"]
X = df_train.drop(["Class", "Artist_Name", "Track_Name"], axis=1)
cat_cols = X.select_dtypes("category").columns
num_cols = X.select_dtypes(include=np.number).columns
print(cat_cols)
print(num_cols)
x_train, x_test, y_train, y_test =
train_test_split(X, Y, test_size=0.2)
```

o passo seguinte foi verificar a exitância de elementos nulos ou vazios, dessa forma também vamos trata los para evitar possíveis problemas, nesse trabalho em especifico usamos moda e mediana para preencher os dados com os valores que mais se repetem e a media de todos como no código abaixo

```
    mode = df_train["Popularity"].mode()
    df_train["Popularity"] =
    df_train["Popularity"].fillna(mode[0])

    mode = df_train["key"].mode()
    df_train["key"] =
    df_train["key"].fillna(mode[0])

    median = df_train["instrumentalness"].median()
    df_train["instrumentalness"] =
    df_train["instrumentalness"].fillna(median)
```

Após todo esse processo partimos para o treinamento dos modelos, fazendo os ajustes necessários com a função scaler.fit_transform, para padronizar os objetos que vão ser trabalhados no processo de treinamento com isso podemos partir para o split de treinamento, indo em sequencia para o treinamento final baseado nos modelos

```
    models = [
    ("log_reg", LogisticRegression(multi_class="multinomial", max_ite
    ("naive_bay", GaussianNB()),
    ("kn_clf", KNeighborsClassifier()),
    ("dtree", DecisionTreeClassifier()),
    ("rforest", RandomForestClassifier()),
]
```

Após a definição dos modelos seguimos para o treinamento em questão usando funções de avaliação

```
df_metrics = pd.DataFrame(index=["name", "accuracy", "precision", "re
n = 0
for name, model in models:
    model.fit(df_train_prepro, y_train)
    y_pred = model.predict(df_test_prepro)
    metrics = [name]

    metrics.append(accuracy_score(y_test, y_pred))
```

```
metrics.append(precision_score(y_test, y_pred,
average="weighted"))

metrics.append(recall_score(y_test, y_pred,
average="weighted"))

metrics.append(f1_score(y_test, y_pred,
average="weighted"))
-reg
df_metrics[n] = metrics
n += 1
```

3 Resultados

Segundo os modelos de resultados temos os seguinte dados.

Name	$\text{Log}_{r}eg$	$\text{naive}_{b}ay$	$\text{kn}_{c}lf$	dtree	rforest
accuracy	0.497569	0.453674	0.417002	0.366857	0.510765
precision	0.464935	0.437683	0.418729	0.381028	0.498202
recall	0.497569	0.453674	0.417002	0.366857	0.510765
$\text{f1}_{s}core$	0.469316	0.43094	0.416749	0.373104	0.496696

Dado os resultados apresentados podemos perceber que para esse data set é a melhor opção de modelo para executar a clasificação