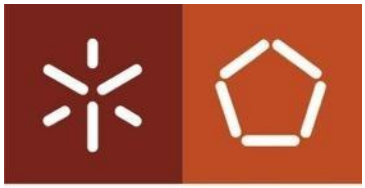


Universidade do Minho

Escola de Engenharia

Mestrado em Engenharia Informática
Dados e Aprendizagem Automática

Conceção e otimização de modelos de *Machine Learning*

Grupo 5

Pedro Veloso (pg47578)

Eduardo Coelho (pg47164)

Henrique Neto (pg47238)

Carlos Preto (pg47089)

Braga, Janeiro de 2022

Índice

Introdução

No âmbito da disciplina de Dados e Aprendizagem Automática, foi proposto o desenvolvimento e otimização de modelos de *Machine Learning* utilizando modelos de aprendizagem abordados ao longo do semestre.

Numa fase inicial foram fornecidos dois *datasets*, sendo estes um *dataset de aprendizagem (training_data.csv)* e um *dataset* de teste (*test_data.csv*). O *dataset* de treino contém informação referente ao tráfego de veículos na cidade do Porto, desde o dia 24 de julho de 2018 até ao dia 2 de outubro de 2019, e será utilizado para desenvolver e treinar o modelo de *Machine Learning.* Por sua vez, o *dataset* de teste será utilizado par validar a *accuracy* do modelo em dados ainda não analisados pelo mesmo. O objetivo passou por desenvolver modelos de *Machine Learning* capazes de prever o fluxo de tráfego rodoviário, numa determinada hora, na cidade do Porto.

Na fase seguinte, foi proposta a consulta, análise e seleção de um *dataset* de entre os que estão acessíveis a partir do *Kaggle*, sendo posteriormente necessário explorar e preparar o *dataset*, procurando extrair conhecimento relevante no contexto do problema em questão.

Neste relatório serão abordadas todas as etapas e decisões associadas ao desenvolvimento dos modelos de *Machine Learning* para cada uma das fases em questão.

Dataset Competição

A modelação do fluxo do tráfego rodoviário é um conhecido problema de características estocásticas, não-lineares. Desta forma, apareceram na literatura um conjunto de modelos que demonstram um potencial assinalável neste tipo de previsões. Assim, foi construído um *dataset* que contém dados referentes ao tráfego de veículos na cidade do Porto durante um determinado período de tempo (período superior a 1 ano). Este *dataset* cobre um período que vai desde o dia 24 de julho de 2018 até ao dia 02 de outubro de 2019.

Tendo por base o *dataset* mencionado, pretende-se prever o fluxo de tráfego rodoviário, para uma determinada hora, na cidade do Porto.

Data Understanding

A primeira etapa de desenvolvimento do modelo de *Machine Learning* consistiu em perceber os dados do problema. Após uma breve análise verificou-se que o *dataset* fornecido contempla 14 atributos:

- ***city_name***: nome da cidade em causa;
- ***record_date***: *timestamp* associado ao registo;
- ***average_speed_diff***: diferença entre a velocidade máxima que os carros podem atingir em cenários sem trânsito e a velocidade que realmente se verifica;
- ***average_free_flow_speed***: valor médio da velocidade máxima que os carros podem atingir em cenários sem trânsito;
- ***average_time_diff***: valor médio da diferença do tempo que se demora a percorrer um determinado conjunto de ruas;
- ***average_free_flow_time***: valor médio do tempo que se demora a percorrer um determinado conjunto de ruas quando não há trânsito;
- ***luminosity***: nível de luminosidade que se verifica na cidade do Porto;
- ***average_temperature***: valor médio da temperatura para o *record_date* na cidade do Porto;
- ***average_atmosp_pressure***: valor médio da pressão atmosférica para o *record_date;*
- ***average_humidity***: valor médio da humidade para o *record_date;*
- ***average_wind_speed***: valor médio da velocidade do vento para o *record_date;*
- ***average_cloudiness***: valor médio da percentagem de nuvens para o *record_date;*
- ***average_precipitation***: valor médio de precipitação para o *record_date;*
- ***average_rain***: avaliação qualitativa da precipitação para o *record_date;*

Destes atributos, aquele que se pretende prever corresponde ao atributo "*average_speed_diff",* que, tal como foi dito anteriormente, indica o nível de trânsito na cidade do Porto, numa determinada data.

Uma vez percebidos os dados do problema, avançou-se para a exploração e tratamento desses dados, sendo para tal necessário carregar para memória os *datasets* de treino e de teste fornecidos.

Através da análise do comportamento dos valores de cada atributo do *dataset,* é possível verificar que todos os valores correspondentes ao atributo "*average_precipitation"* estão com o valor zero. Desta forma, conclui-se que este atributo pode ser eliminado dos *datasets,* visto que não apresenta informação útil para a aprendizagem do modelo de *Machine Learning*.

```
traffic_train.describe()
```

	AVERAGE_FREE_FLOW_SPEED	AVERAGE_TIME_DIFF	AVERAGE_FREE_FLOW_TIME	AVERAGE_TEMPERATURE	AVERAGE_ATMOSP_PRESSURE	AVERAGE_HUMIDITY	AVERAGE_WIND_SPEED	AVERAGE_PRECIPITATION
count	6812.000000	6812.000000	6812.000000	6812.000000	6812.000000	6812.000000	6812.000000	6812.0
mean	40.661010	25.637111	81.143952	16.193482	1017.388139	80.084190	3.058573	0.0
std	4.119023	33.510507	8.294401	5.163492	5.751061	18.238863	2.138421	0.0
min	30.500000	0.000000	46.400000	0.000000	985.000000	14.000000	0.000000	0.0
25%	37.600000	2.275000	75.400000	13.000000	1015.000000	69.750000	1.000000	0.0
50%	40.700000	12.200000	82.400000	16.000000	1017.000000	83.000000	3.000000	0.0
75%	43.500000	36.200000	87.400000	19.000000	1021.000000	93.000000	4.000000	0.0
max	55.900000	296.500000	112.000000	35.000000	1033.000000	100.000000	14.000000	0.0

Figura 1: Análise atributos

De seguida, foi necessário verificar a existência de valores nulos em cada atributo. Após uma breve análise, verificou-se que os atributos "*average_cloudiness*" e "*average_rain*" apresentam 2682 e 6249 valores nulos, respetivamente, logo o grupo decidiu que estas deviam ser as primeiras *features* a serem tratadas.

Numa primeira fase, procedeu-se ao tratamento dos valores da "*average_cloudiness*". Este atributo pode apresentar 10 valores distintos, porém, algum dos valores representam a mesma situação, só que por palavras distintas. Assim, o grupo optou por reorganizar os dados, tal como se pode observar na Figura 3, obtendo no final apenas 3 valores distintos, sendo estes: céu claro, céu pouco nublado e céu nublado.

```
preencher = []
for ac3 in zip(traffic_train['AVERAGE_CLOUDINESS']):
    ac = ac3[0]
    if ac == 'céu claro' or ac == 'céu limpo':
        preencher.append('céu claro')
    elif ac=='nuvens quebrados' or ac=='nuvens quebradas' or ac == 'nuvens dispersas' or ac=='algumas nuvens' or ac=='céu pouco nublado':
        preencher.append('céu pouco nublado')
    elif ac=='tempo nublado' or ac=='nublado':
        preencher.append('céu nublado')
    else:#quando é nan
        preencher.append(ac)

traffic_train['AVERAGE_CLOUDINESS'] = preencher
```

Figura 2: Agrupar valores average_cloudiness

Após o agrupamento dos valores em "*average_cloudiness"*, avançou-se para o agrupamento dos valores do atributo "*average_rain*". Esta *feature* apresenta 14 valores distintos, porém, tal como se sucede no atributo "*average_cloudiness*", muitos desses valores correspondem a situações do quotidiano semelhantes. Após alguma reflexão, o grupo considerou que o fator trovoada não tem tanto impacto nas condições de trânsito, como, por exemplo, a chuva, logo não deverá haver uma distinção entre estes dois valores. Assim, o grupo optou por reorganizar os dados em 4 valores distintos, sendo estes: sem chuva, chuva fraca, chuva moderada e chuva forte.

```
preencher = []
for ar3 in zip(traffic_train['AVERAGE_RAIN']):
    ar = ar3[0]
    if ar == 'chuva fraca' or ar == 'chuva leve' or ar == 'chuvisco fraco' or ar == 'chuvisco e chuva fraca' or ar == 'trovoada com chuva leve':
        preencher.append('chuva fraca')
    elif ar == 'chuva moderada' or ar == 'chuva' or ar == 'aguaceiros fracos' or ar == 'trovoada com chuva':
        preencher.append('chuva moderada')
    elif ar == 'chuva de intensidade pesada' or ar == 'chuva de intensidade pesado' or ar == 'chuva forte' or ar == 'aguaceiros':
        preencher.append('chuva forte')
    else:#quando é nan
        preencher.append(ar)

traffic_train['AVERAGE_RAIN'] = preencher
```

Figura 3: Agrupar valores average_rain

Uma vez reestruturados os dados, foi necessário tomar uma decisão relativamente aos valores nulos em ambos os atributos, sendo que, ou se eliminava as linhas onde apareciam os valores a nulo, ou se tentava preencher esses mesmos valores. De maneira a não perder informação importante para a aprendizagem do modelo de *Machine Learning*, o grupo optou pelo preenchimento desses valores.

Após algumas pesquisas na *internet*, percebeu-se que, normalmente, a presença de nuvens no céu está bastante associada à pressão atmosférica, atributo ao qual temos acesso no *dataset*. Desta forma, utilizou-se a seguinte regra: quanto maior for o valor da pressão atmosférica, menor será a probabilidade de formação de nuvens no céu. Assim, para valores de "*average_atmosp_pressure*" superiores a 1025, considera-se que não há nuvens, para valores superiores a 1005 e inferiores a 1025 considera-se que o céu estará pouco nublado e para os restantes valores, considera-se que o céu estará nublado.

```
preencher = []
for ac, atm in zip(traffic_train['AVERAGE_CLOUDINESS'], traffic_train['AVERAGE_ATMOSP_PRESSURE']):
    if  str(ac) == 'nan':
        if atm > 1025:
            preencher.append('céu claro')
        elif atm > 1005:
            preencher.append('céu pouco nublado')
        else:
            preencher.append('céu nublado')
    else:
        preencher.append(ac)

traffic_train['AVERAGE_CLOUDINESS'] = preencher
```

Figura 4: Preencher average_cloudiness

Após preenchidos os valores na "*average_cloudiness*", procedeu-se ao preenchimento dos valores nulos na "*average_rain*". Analisado alguns artigos referentes aos fatores que podem afetar a precipitação numa dada zona, decidiu-se utilizar os atributos "*average_cloudiness*" e "*average_humidity*" para determinar os novos valores da "*average_rain*". Assim, quando o céu estiver nublado e os valores da "*average_humidity*" sejam superiores a 70, considera-se que há chuva forte, caso apenas sejam superiores a 50, então considera-se que ocorre uma chuva moderada. Já para valores na gama dos 30 considera-se que existe chuva fraca, não existindo a ocorrência de chuva nos restantes valores. Por sua vez, quando o céu estiver pouco nublado, definiu-se que não poderia haver chuva forte, por isso, caso os valores da "*average_humidity"* sejam superiores a 70, considera-se que há chuva moderada, caso apenas sejam superiores a 50, considera-se que há chuva fraca, e para os restantes valores considera-se que não há chuva.

```
preencher = []
for ac, ah, ar in zip(traffic_train['AVERAGE_CLOUDINESS'], traffic_train['AVERAGE_HUMIDITY'], traffic_train['AVERAGE_RAIN']):
    if  str(ar) == 'nan':
        if str(ac) == 'céu nublado':
            if ah > 70:
                preencher.append('chuva forte')
            elif ah > 50:
                preencher.append('chuva moderada')
            elif ah > 30:
                preencher.append('chuva fraca')
            else:
                preencher.append('sem chuva')
        elif str(ac) == 'céu pouco nublado':
            if ah > 70:
                preencher.append('chuva moderada')
            elif ah > 50:
                preencher.append('chuva fraca')
            else:
                preencher.append('sem chuva')
        else:
            preencher.append('sem chuva')
    else:
        preencher.append(ar)

traffic_train['AVERAGE_RAIN'] = preencher
```

Figura 5: Preencher valores average_rain

Além das restrições impostas, é necessário tratar todos os cenários impossíveis, que ocorrem quando a "*average_cloudiness"* indica que está céu limpo, e, no entanto, a "*average_rain"* indica que está a chover. Como se pode observar na Figura 6, sempre que estiver céu limpo, então é porque não está a chover.

```
preencher = []
for ac, ar in zip(traffic_train['AVERAGE_CLOUDINESS'], traffic_train['AVERAGE_RAIN']):
    if str(ac) == 'céu claro':
        preencher.append('sem chuva')
    else:
        preencher.append(ar)

traffic_train['AVERAGE_RAIN'] = preencher
```

Figura 6: Tratar cenários impossíveis

Feature Engineering

Uma vez analisados e tratados os valores nulos no *dataset*, de maneira a enriquecer o nível de informação deste, decidiu-se produzir novos atributos à custa de atributos já existentes. De todas as *features* presentes no *dataset*, considerou-se que a *feature "record_date"* apresenta bastante informação adicional que pode ser extraída e originar novos atributos.

Assim, à custa do "*record_date"*, foram adicionados três novos atributos ao *dataset*:

- **DiaSemana**: dia da semana associado a uma determinada data;
- **Dia**: dia do mês associado a uma determinada data;
- **Mês**: mês associado a uma determinada data;
- **Hora**: hora associada a uma determinada data;

```
def mes(data):
    from datetime import datetime, date
    data = datetime.strptime(data, '%Y-%m-%d %H:%M:%S').date()
    return data.month

traffic_train['Mes'] = traffic_train['record_date'].apply(mes)
traffic_test['Mes'] = traffic_test['record_date'].apply(mes)
```

Figura 7: Criação do atributo Mes

De seguida, considerou-se importante diferenciar os fins-de-semana dos restantes dias da semana, visto que os níveis de trânsito numa determinada hora também são afetados pelos dias de semana em questão. Assim, adicionou-se um novo atributo discreto denominado por "*FimDeSemana*", que toma o valor de 'Sim' quando o dia corresponde a um fim-de-semana, e toma o valor de 'Não' nos restantes casos.

Outro fator que pode afetar bastante as condições de trânsito numa determinada data é as situações de feriado. Numa cidade como o Porto, os feriados são bastante comemorados e atraem bastante gente, pelo que se espera que os níveis de trânsito nesses dias sejam relativamente altos. Como tal, através dos atributos *Dia* e *Mes*, é possível adicionar um novo atributo ao *dataset*, denominado por "*Feriado*", que indicará se uma determinada data coincide com um feriado na cidade do Porto. Como se pode observar

na Figura 8, tendo uma lista de feriados formada por pares, onde o primeiro elemento se refere ao dia e o segundo elemento se refere ao mês, é possível verificar se uma data coincide com alguns dos pares da lista, e caso coincida, então o atributo "*Feriado"* toma o valor 'Sim'.

```
feriados = [(1,1),(25,4),(1,5),(12,5),(13,5),(10,6),(24,6),(15,8),(5,10),(1,11),(1,12),(8,12),(25,12)]

#obter coluna com informação do Feriado
preencher = []
for d, m in zip(traffic_train['Dia'], traffic_train['Mes']):
    date = (d,m)
    if date in feriados:
        preencher.append('Sim')
    else:
        preencher.append('Não')

traffic_train['Feriado'] = preencher
```

Figura 8: Atributo Feriado

Posteriormente, identificou-se um fator bastante importante para identificação de situações de trânsito. Durante os dias da semana, os horários de maior trânsito decorrem quando as pessoas entram e saem dos seus trabalhos, por isso torna-se essencial distinguir as diferentes etapas do dia. Assim, à custa do atributo *Hora*, adicionou-se um novo atributo ao *dataset*, denominado por "*Parte_do_Dia*", que pode apresentar 3 valores distintos – Madrugada, Hora de Trabalho e Noite – tal como se pode observar na Figura 9.

```
#obter coluna com parte do dia
def daypart(hour):
    if hour > 0 and hour <= 7:
        return 'Madrugada'
    elif hour >= 8 and hour <= 19:
        return 'Hora de Trabalho'
    else:
        return 'Noite'

traffic_train['Parte_do_Dia'] = traffic_train['Hora'].apply(daypart)
traffic_test['Parte_do_Dia'] = traffic_test['Hora'].apply(daypart)
```

Figura 9: Atributo Parte_do_Dia

Por último, o grupo achou necessário distinguir entre as diferentes estações do ano, uma vez que algumas estações podem afetar negativamente a condição das estradas e por conseguinte originar transito mais lento, como por exemplo o Inverno. Por isso, adicionou-se um novo atributo denominado por "*Estacao"*, que irá indicar a estação do ano de uma determinada data.

Data Correlation

Depois de se ter adicionado novos conhecimentos ao *dataset*, procedeu-se à análise da matriz de correlação da base de conhecimento, de modo a verificar o grau de correlação entre pares de atributos.

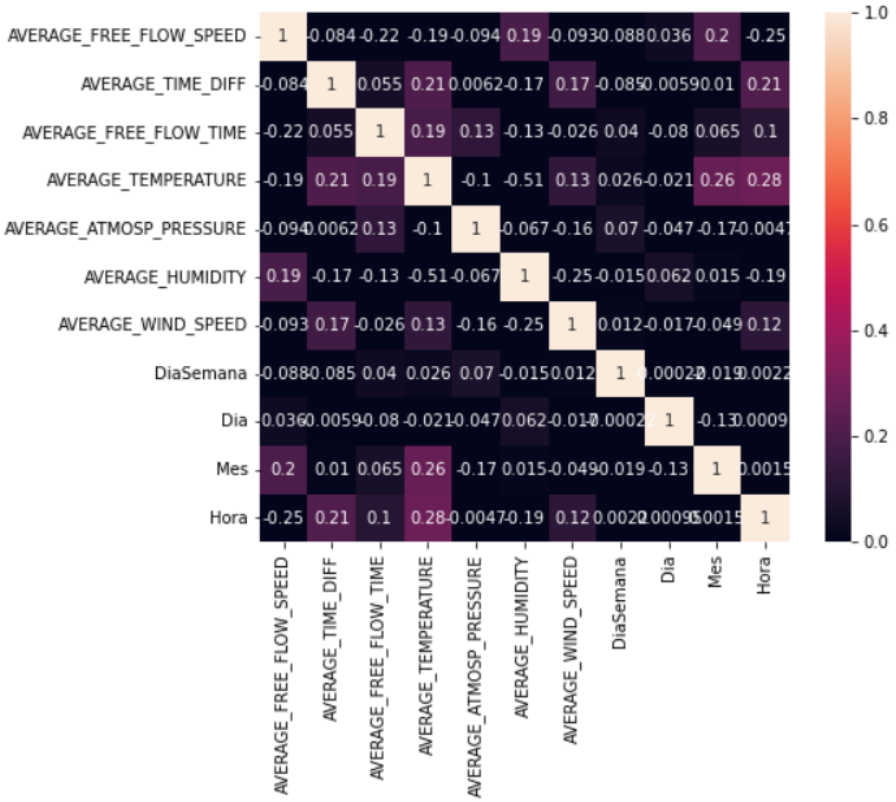


Figura 10: Matriz de Correlação

Analisando a Figura 10, percebe-se que não existe correlação forte entre nenhuns atributos, ou seja, todos os atributos do *dataset* contém informação importante e distinta de qualquer outra, pelo que não será necessário remover nenhuma informação redundante.

Outliers

A última etapa de análise de tratamento de dados incidiu sobre a correção dos dados que se diferenciam bastante dos restantes. Assim, para cada um dos atributos numéricos presentes no *dataset*, foi realizada uma intensa análise para perceber quais seriam os melhores valores a utilizar, de maneira a tratar esses mesmos dados.

Desta forma, os atributos selecionados para correção de *outliers* foram: *average_free_flow_speed*, *average_free_flow_time*, *average_time_diff*, *average_temperature* e *average_wind_speed***.**

Para cada um destes foram definidos valores máximos e mínimos, sendo que, sempre que um dado não pertença ao intervalo destes valores será reajustado para:

- **valor máximo**: se ultrapassar o valor máximo definido;
- **valor mínimo**: se for menor que o valor mínimo definido;

Os valores máximo e mínimo de cada atributo foram obtidos por observação do *boxplot* de cada uma das *features*, e por tentativa-erro, ajustou-se os valores de máximo e mínimo de modo a melhorar os resultados obtidos.

Como exemplo, considera-se o *boxplot* do atributo "*average_free_flow_time*", representado na Figura 11. Tal como se pode observar, existem alguns valores inferiores a 60 e outros superiores a 105 que são diferentes da generalidade dos valores do atributo.

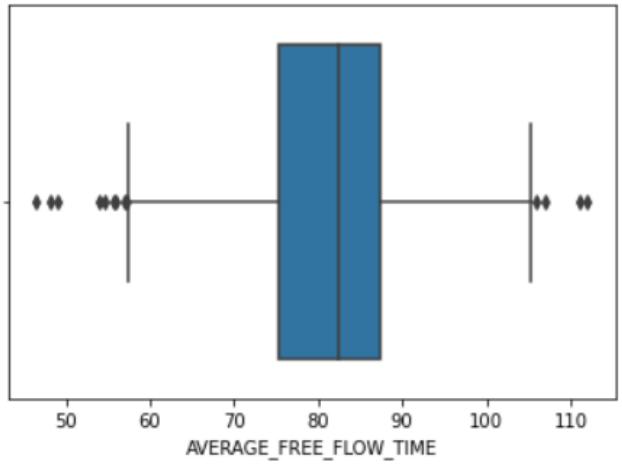


Figura 11: Outliers average_free_flow_time

Assim, definiu-se funções que, com base nos valores dos registos, verifica se estes são inferiores a 60 (no caso do atributo *average_free_flow_time*), sendo nesse caso necessário passá-los para esse valor, ou se os valores são superiores a 105, sendo também, nesse caso, atribui-lhes um novo valor. Com a implementação destas funções, pretende-se garantir uma melhor distribuição dos dados presentes no *dataset*.

```
qff1 = 60
qff2 = 105

preencher = []
for afft3 in zip(traffic_train['AVERAGE_FREE_FLOW_TIME']):
    afft = afft3[0]
    if afft > qff2:
        preencher.append(qff2)
    elif afft < qff1:
        preencher.append(qff1)
    else:
        preencher.append(afft)

traffic_train['AVERAGE_FREE_FLOW_TIME'] = preencher
```

Figura 12: Correção outliers average_free_flow_time

Modelos

Uma vez tratados todos os dados, foi possível começar a desenvolver o modelo que será utilizado para produzir os melhores resultados.

Analisando os resultados esperados, percebe-se que os dados são do tipo discreto, logo trata-se de um problema de classificação. Com base nesta informação, o grupo desenvolveu 3 modelos distintos, sendo eles um *Decision Tree Classifier*, um SVC e uma Rede Neuronal. Destes, aquele que produziu melhores resultados foi a Rede Neuronal, portanto apenas se refere as características associadas a esse modelo.

A primeira etapa a realizar foi converter todos os atributos nominais para atributos numéricos categóricos, para que seja possível normalizar todos os valores do *dataset*. Posteriormente definiu-se uma função responsável por produzir a Rede Neuronal.

```
def build_model(activation='relu', learning_rate=0.001):
    model=Sequential()
    model.add(Flatten(input_shape=(18,)))
    model.add(Dense(10, activation=activation))
    #model.add(Dense(16,activation=activation))
    model.add(Dense(5, activation='softmax'))

    model.compile(
        loss = 'sparse_categorical_crossentropy',
        optimizer = tf.optimizers.Adam(learning_rate),
        metrics = ['accuracy'])
    return model

model = build_model()
```

Figura 13: Rede Neuronal

Como se pode observar na Figura 13, a rede neuronal é constituída por 3 camadas distintas. A primeira camada, transforma as 18 *features* de input, através do método *Flatten.* A segunda camada é uma camada de ativação (*relu*) com 10 neurons. A última camada utiliza a função de ativação *softmax* e tem 5 neurons uma vez que se trata de um problema de classificação de múltiplas classes (neste caso são 5). A função de *loss* tem que ser a *sparse_categorical_crossentropy* pelo mesmo motivo. Estamos a aplicar o *optimizer Adam*, com um *learning rate* de 0.001. A métrica utilizada neste problema é a *accuracy* porque é este o fator de classificação da competição. Com a nossa rede neuronal definida, começámos a criação do modelo com 100 *epochs*, 32 de *batch_size* e *validation_split* de 0.1. Este valor foi extremamente importante para se ter a certeza de que não se está a dar *overfitting*. De seguida, utilizando a ferramenta *grid_search* consegui-se "afinar" melhor os hyper-parâmetros da rede neuronal, tais como a função de ativação e a *learning_rate.*

```
Best: 0.785285 using {'activation': 'relu', 'learning_rate': 0.001}
0.783333 (0.010708) with {'activation': 'relu', 'learning_rate': 0.01}
0.785285 (0.013371) with {'activation': 'relu', 'learning_rate': 0.001}
```

Figura 14:Melhores resultados para a rede neuronal (obtidos pelo grid_search)

Como podemos observar, os melhores valores para os hyper-parâmetros encontrados foram 0.001 para o *learning_rate* e a função *relu* para a função de ativação.

Avaliação dos resultados obtidos

Após obter a previsão, procedeu-se à sua avaliação. Para determinar se foi realizada uma boa extração de conhecimento dos dados do *dataset* foi necessário submeter os resultados obtidos no *Kaggle.* Como se pode observar na Figura 15, obteve-se uma *accuracy* final de 0.80190.

Figura 15: Avaliação Modelo

Dataset Secundário

Na segunda etapa do trabalho, optou-se por analisar um *dataset* relacionado com a indústria musical. Assim, pretende-se determinar o género musical, em que uma determinada música se encaixa, através de um conjunto de atributos extraídos da mesma.

Deste modo, aprofundou-se nas várias etapas propostas pela metodologia de extração de conhecimento SEMMA, para extrair conhecimento dos dados. Esta metodologia divide o processo de extração de conhecimentos em 5 passos: *Sampling* (extração dos dados), *Explore* (visualização dos dados obtidos), *Modify* (correção das imperfeiçoes dos dados), *Model* (aplicação de modelos de extração de conhecimento) e *Assess* (avaliação dos resultados obtidos nos modelos).

Sampling

Numa primeira fase, foi necessário construir um *dataset* que permitisse extrair o conhecimento pretendido (género musical). Assim, obteve-se um *dataset* com os seguintes atributos:

- ***Artist name***: nome do artista que produziu a música;
- ***Track name***: nome da música;
- ***Popularity***: valor que identifica a popularidade que a música obteve;
- ***Danceability***: métrica que indica o ritmo dançável de uma música;
- ***Energy***: métrica que avalia a energia transmitida por uma determinada música;
- ***Key***: valor estimado da *key* de uma música. Os valores mapeiam *pitches* usados na notação *standard*;
- ***Loudness***: métrica, em *decibéis*, que avalia o nível de ruído presente na música;
- ***Mode***: valor que indica a modalidade da música (*major* ou *minor*);
- ***Speechiness***: métrica que aponta para a quantidade de discurso presente numa música;
- ***Acousticness***: valor que representa a acústica de uma música;
- ***Instrumentalness***: métrica que classifica a presença de instrumentos musicais numa determinada música;
- ***Liveness***: métrica que representa a avaliação da presença de audiência numa música;
- ***Valence***: métrica que avalia o tom da música, indicando a felicidade que a música pode transmitir;
- ***Tempo***: valor que representa o ritmo de uma música;
- ***Duration***: duração total de uma música;
- ***Time_signature***: métrica que indica uma estimativa de batidas presentes numa música;
- ***Class***: Género musical de uma música (0- *Acoustic/Folk*, 1- *AltMusic*, 2- *Blues*, 3- *Bollywood*, 4- *Country*, 5- *HipHop*, 6- *Indie*, 7- *Instrumental*, 8- *Metal*, 9- *Pop*, 10- *Rock*).

Ao analisar os atributos, percebe-se a existência do atributo "*class*", que contém o resultado que se pretende prever. Assim, o paradigma de aprendizagem presente é o paradigma de aprendizagem supervisionada. Também é de notar a presença de um problema de classificação.

Explore

Nesta etapa, pretende-se aprofundar a análise dos dados obtidos anteriormente, de modo a perceber o seu comportamento. Através do seu comportamento deve-se determinar quais atributos devem permanecer ou sofrer alterações no *dataset*. Assim analisou-se o comportamento geral dos valores, a

existência de valores duplicados, a nulo ou com características distintas dos restantes valores (*outliers*), bem como a correlação entre atributos.

Análise dos valores

De modo, a perceber o estado geral dos valores de cada atributo, realizou-se a descrição dos mesmos, obtendo a seguinte tabela:

	Popularity	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo	duration_in min/ms	time_signature	Class
count	17568.000000	17996.000000	17996.000000	15982.000000	17996.000000	17996.000000	17996.000000	17996.000000	13619.000000	17996.000000	17996.000000	17996.000000	1.799600e+04	17996.000000	17996.000000
mean	44.512124	0.543433	0.662777	5.952447	-7.910660	0.636753	0.079707	0.247082	0.177562	0.196170	0.486208	122.623294	2.007445e+05	3.924039	6.695821
std	17.426928	0.166268	0.235373	3.196854	4.049151	0.480949	0.083576	0.310632	0.304048	0.159212	0.240195	29.571527	1.119891e+05	0.361618	3.206073
min	1.000000	0.059600	0.000020	1.000000	-39.952000	0.000000	0.022500	0.000000	0.000001	0.011900	0.018300	30.557000	5.016500e-01	1.000000	0.000000
25%	33.000000	0.432000	0.509000	3.000000	-9.538000	0.000000	0.034800	0.004300	0.000089	0.097500	0.297000	99.620750	1.663370e+05	4.000000	5.000000
50%	44.000000	0.545000	0.700000	6.000000	-7.016000	1.000000	0.047400	0.081400	0.003910	0.129000	0.481000	120.065500	2.091600e+05	4.000000	8.000000
75%	56.000000	0.659000	0.860000	9.000000	-5.189000	1.000000	0.083000	0.434000	0.200000	0.258000	0.672000	141.969250	2.524900e+05	4.000000	10.000000
max	100.000000	0.989000	1.000000	11.000000	1.355000	1.000000	0.955000	0.996000	0.996000	1.000000	0.986000	217.416000	1.477187e+06	5.000000	10.000000

Figura 16: Análise dos atributos

Ao analisar a tabela, percebe-se a existência de uma ampla gama de valores, como por exemplo valores na gama das dezenas ("*Popularity*"), enquanto outros encontram-se na casa das décimas ("*danceability*"). Por fim, também se percebe que o atributo relativo à duração da música pode se encontrar tanto em milissegundos, como em minutos.

Relativamente ao atributo "*Artist name*", notou-se que era possível uma música ser composta por um único artista, ou por uma parceria entre um conjunto de artistas.

Análise dos duplicados

Posteriormente, deve-se tentar perceber a existência de registos duplicados, tanto na sua totalidade, como também parcialmente. Desta forma percebe-se que o atributo "*Track name*" está altamente duplicado, sendo que existem vários artistas que compuseram músicas com o mesmo nome (figura 17).

```
['A musica "Dreams" tem os artistas: Sense Field'
 'A musica "Dreams" tem os artistas: Beck'
 'A musica "Dreams" tem os artistas: Langston Hughes'
 'A musica "Dreams" tem os artistas: The Game'
 'A musica "Dreams" tem os artistas: Fleetwood Mac'
 'A musica "Dreams" tem os artistas: Irish Women In Harmony'
 'A musica "Dreams" tem os artistas: chloe moriondo'
 'A musica "Dreams" tem os artistas: Van Halen'
 'A musica "Dreams" tem os artistas: D Smoke']

['A musica "Fire" tem os artistas: Kasabian'
 'A musica "Fire" tem os artistas: Diskopunk'
```

Figura 17: Exemplo de artistas que compuseram músicas com o mesmo nome

Análise de valores a nulo

De modo a perceber, a existência de valores que não foram preenchidos, realizou-se uma análise, de onde foi possível perceber a existência de três atributos com alguns valores a nulo: *popularity* (428 valores a nulos), *key* (2014 valores a nulo) e *instrumentalness* (4377 valores a nulo).

Artist Name	0
Popularity	428
danceability	0
energy	0
key	2014
loudness	0
mode	0
speechiness	0
acousticness	0
instrumentalness	4377
liveness	0
valence	0
tempo	0
duration_in min/ms	0
time_signature	0
Class	0

Figura 18: Quantidade de valores a nulo por atributo

Análise da correlação entre atributos

De modo a visualizar a correlação entre os vários atributos do *dataset*, recorreu-se à representação da matriz de correlação.

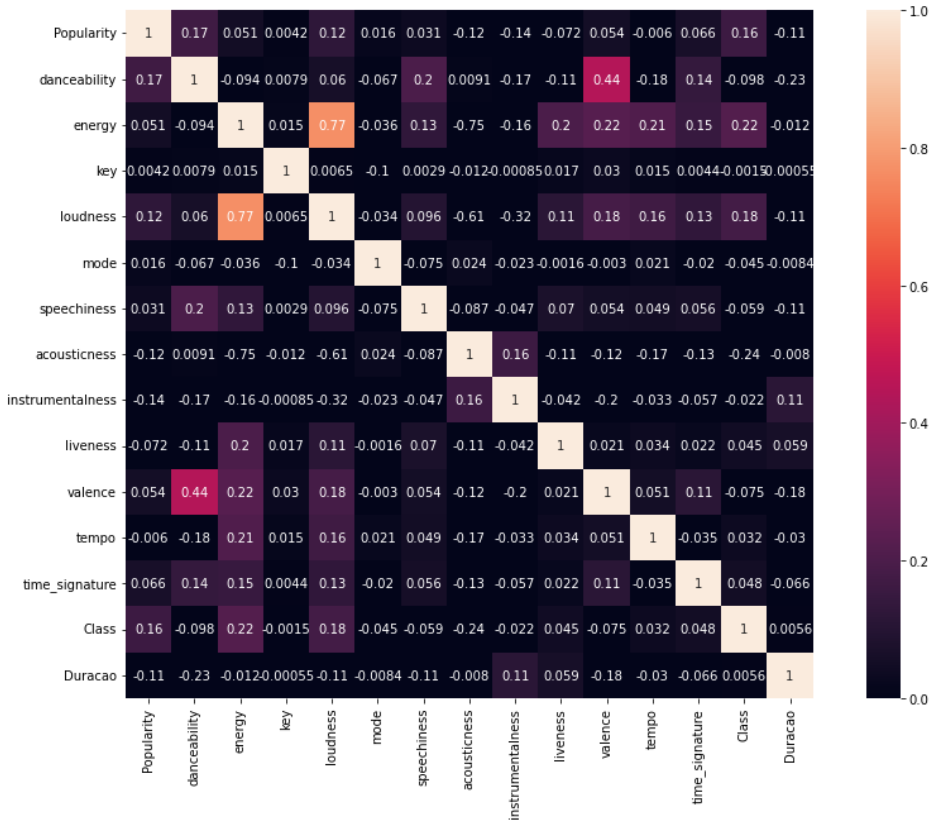


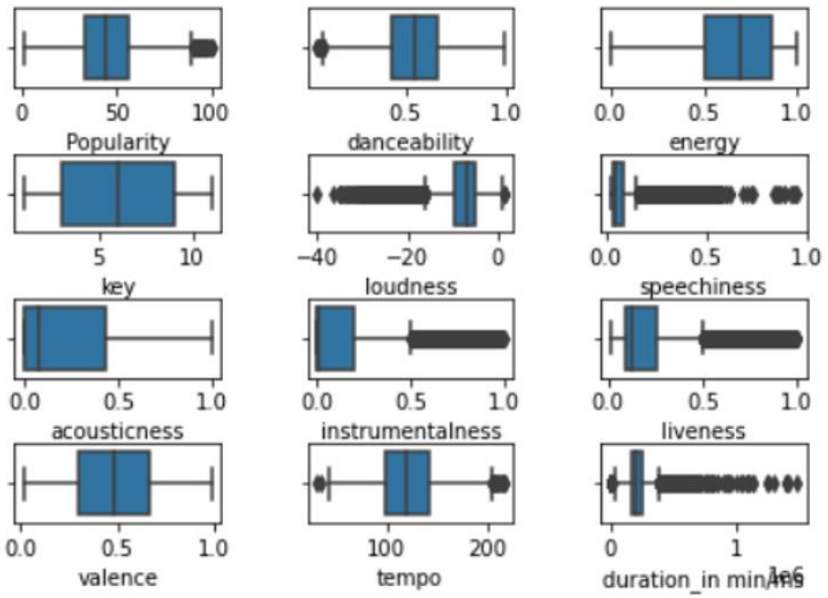
Figura 19: Matriz de correlação

Ao analisar a matriz de correlação (figura 19), percebeu-se que nenhum dos atributos evidenciam a tendência de alta correlação. Porém, é de notar uma correlação mais elevada (0.77) entre os atributos

"*loudness*" e "*energy*". Apesar, de não ser possível tirar conclusões de imediato, deve-se ponderar a necessidade da existência destes dois atributos, através da realização de testes.

Análise dos outliers

Por fim, analisou-se a existência de *outliers* (valores com características distintas dos restantes valores do mesmo atributo). Para realizar esta análise recorreu-se à representação de gráficos "*Box-Plot*", bem como a uma análise estatística que permitiu detetar a existência da quantidade exata de *outliers* para cada atributo, tal como se evidencia na figura seguinte.




```
# Outlieres:
Popularity : 41
danceability : 31
energy : 0
key : 0
loudness : 818
speechiness : 2034
acousticness : 0
instrumentalness : 2590
liveness : 956
valence : 0
tempo : 36
Duracao : 922
```

Figura 20: Análise dos outliers

Modify

Nesta fase, realizou-se a correção de problemas identificados na fase anterior. Assim tratou-se duplicados, valores a nulo, *outliers*, bem como os outros problemas já assinalados.

Tratar o atributo "Artist name"

De modo a garantir uma maior heterogeneidade neste atributo, decidiu-se dividir de uma forma mais clara e sucinta os diversos artistas que compuseram uma determinada música. Para tal, foram considerados 2 casos distintos:

- **Artistas separados por *feat*:** ocorre quando artistas colaboram na criação de uma determinada música;
- **Artistas separados por vírgula:** ocorre quando há mais que um artista a participar numa música.

De maneira a tratar estes casos, considerou-se que sempre que houvesse mais do que um artista associado a uma música, o registo original onde esses artistas aparecem, apenas conterá o primeiro artista referenciado, e posteriormente, acrescenta-se um novo registo para cada um dos restantes artistas, apenas alternando o nome do artista.

```
l = music_train['Artist Name']

novas_linhas = []
for index, row in music_train.iterrows():
    nome = row['Artist Name']
    if "feat." in nome:
        #obter todos os nomes
        temp = nome.split("(feat.")
        for n in temp:
            if ")" in n:
                temp2 = n.split(")")
                name = temp2[0].strip()
                #duplicar linha
                nova_linha = row
                nova_linha['Artist Name'] = name
                novas_linhas.append(nova_linha)
        #alterar linha antiga
        row['Artist Name'] = temp[0].strip()
    else:
        if "," in nome:
            #obter todos os nomes
            temp = nome.split(",")
            i = 0
            for n in temp:
                if i==0:
                    i+=1
                else:
                    name = n.strip()
                    #duplicar linha
                    nova_linha = row
                    nova_linha['Artist Name'] = name
                    novas_linhas.append(nova_linha)
                    i+=1
            #alterar original para o primeiro nome que aparece
            row['Artist Name'] = temp[0].strip()
```

Figura 21: Algoritmo para tratar os conjuntos de artistas

Tratar duplicados

De modo a garantir que não existia nenhum registo do *dataset* inteiramente duplicado, realizou-se a sua eliminação, recorrendo a uma função da biblioteca "*pandas*". Posteriormente, mediante a análise anterior decidiu-se eliminar também o atributo referente ao nome da música.

Tratar o atributo da duração da música

Uma vez que existia um atributo que continha a duração da música expressa tanto em milissegundos como em minutos, decidiu-se converter todos os valores para minutos, tal como se demonstra na figura seguinte:

```
def ms_to_min(duracao):
    if(duracao > 1000):
        return float(duracao / 60000.0)
    return duracao

music_train['Duracao'] = music_train['duration_in min/ms'].apply(ms_to_min)

music_train = music_train.drop(['duration_in min/ms'], axis=1)
```

Figura 22: Tratar o atributo da duração da música

Preencher os valores nulos

Como resultado da análise anterior, foi possível verificar a existência de alguns valores a nulo em certos atributos. Assim, realizou-se os seguintes testes de modo a decidir como proceder:

- Não realizar alterações;
- Prever os valores em falta – utilizar a técnica *bfill*, interpolar ou substituir por um valor (média);
- Apagar os registos;

Deste modo, obteve-se melhores resultados ao preencher os valores nulos com a média dos valores dessa *feature*.

Correção dos outliers

De modo a realizar o tratamento dos *outliers*, considerou-se apagar os registos ou simplesmente atribuir-lhes valores que se enquadrassem melhor na distribuição dos restantes elementos.

Ao realizar testes para cada uma das técnicas optou-se por aplicar a técnica de *cap*, ou seja, atribuir-lhes valores que se enquadrem melhor na distribuição dos restantes elementos.

Normalizar os dados

De modo a garantir que os atributos apresentem uma gama de valores semelhante, realizou-se a normalização dos dados. Assim, garantiu-se que todos os valores se encontravam entre 0 e 1.

```
# Scaler
from sklearn.preprocessing import MinMaxScaler

X = music_train.drop(['Class'],axis=1)
Y = music_train['Class'].to_frame()

scaler_x = MinMaxScaler(feature_range=(0,1)).fit(X)
x_scaled = pd.DataFrame(scaler_x.transform(X[X.columns]), columns=X.columns)
```

Figura 23:Escalar os dados para ficarem entre 0 e 1

Atributos correlacionados

Por fim, uma vez que os atributos "*loudness*" e "*energy*" tinham uma alta correlação testou-se os resultados obtidos pelo modelo, nos casos em que existiam os dois atributos, bem como nos casos em que apenas existia um deles.

Ao analisar os resultados percebeu-se que não houve benefício em apagar nenhum dos atributos, pelo que se optou por não remover conhecimento, deixando assim os 2 atributos para a criação do modelo final.

Model

Para obter o melhor modelo de extração de conhecimento, para este problema, testou-se a utilização de árvores de decisão, *suport vector machine*, bem como redes neuronais. Contudo, ao analisar os resultados obtidos por cada um dos algoritmos, optou-se por utilizar as redes neuronais.

Modelo selecionado (Redes neuronais)

```
def build_model(activation='relu', learning_rate=0.01):
    model=Sequential()
    model.add(Flatten(input_shape=(15,)))
    model.add(Dense(20, activation=activation))
    model.add(Dense(22,activation=activation))
    model.add(Dense(11, activation='softmax'))

    model.compile(
        loss = 'sparse_categorical_crossentropy',
        optimizer = tf.optimizers.Adam(learning_rate),
        metrics = ['accuracy'])
    return model
```

Figura 24: Construção da rede neuronal

Como se pode observar na figura 24, a rede neuronal é constituída por 4 camadas distintas. A primeira camada, transforma as 15 *features* de input, através do método *Flatten.* As camadas 2 e 3 são camadas de ativação (*relu*) com 20 e 22 neurónios, respetivamente. A última camada utiliza a função de ativação *softmax* e tem 11 neurónios uma vez que se trata de um problema de classificação de múltiplas classes (neste caso são 11). A função de *loss* tem que ser a *sparse_categorical_crossentropy* pelo mesmo motivo. Estamos a aplicar o *optimizer Adam*, com um *learning rate* de 0.01. A métrica utilizada neste problema é a *accuracy.* Com a nossa rede neuronal definida, começámos a criação do modelo com 50 *epochs*, 32 de *batch_size* e *validation_split* de 0.2. Este valor foi extremamente importante para termos a certeza de que não estávamos a dar *overfitting*. De seguida, utilizando a ferramenta *grid_search* conseguimos "afinar" melhor os *hyper*-parâmetros da reude neuronal tais como a função de ativação e a *learning_rate*.

```
Best: 0.467053 using {'activation': 'sigmoid', 'learning_rate': 0.01}
0.323330 (0.030619) with {'activation': 'relu', 'learning_rate': 0.1}
0.452924 (0.008744) with {'activation': 'relu', 'learning_rate': 0.01}
0.457307 (0.009975) with {'activation': 'relu', 'learning_rate': 0.001}
0.423153 (0.009160) with {'activation': 'sigmoid', 'learning_rate': 0.1}
0.467053 (0.013190) with {'activation': 'sigmoid', 'learning_rate': 0.01}
0.434033 (0.009725) with {'activation': 'sigmoid', 'learning_rate': 0.001}
```

Figura 25: Resultados da aplicação do GRID_SEARCH

Como podemos observar, os melhores valores para os *hyper*-parâmetros encontrados foram 0.01 para o *learning_rate* e a função *sigmoid* para a função de ativação.

Assess

Por fim, na última fase do método de extração de conhecimento adotado, realizou-se a análise dos resultados obtidos pelo modelo desenvolvido na etapa anterior. Assim, uma vez que o problema em questão é um problema de classificação, recorreu-se à matriz de confusão apresentada na figura seguinte.

```
[[ 61   0  33   5   0   2  18  10   0  30  76]
 [  6   0  14   2   0  20  34  14  23  44 263]
 [  8   0 127  28   0  10  19  12   5  48 134]
 [  2   0  18 114   1   1   5   5   0   3  44]
 [  9   0   7   1  10   1   1   0   0  13  83]
 [  1   0  13   3   0 293  11   2   1  93  35]
 [ 22   0  35  11   2  46 115  34  28  95 370]
 [  2   2   5   3   0   0   3 256   0   2   5]
 [  0   0   2   0   0   1   8   2 298   1 262]
 [ 24   0  32   6   2  60  25   4   3 382 227]
 [ 22   0  76  19  10  17  38   7 104 106 1083]]
```

Figura 26:Matriz de confusão

Análise dos valores de accuracy, precision, recall e *f1-score*

De modo a avaliar os resultados obtidos com o modelo gerado pela rede neuronal, e apresentados acima, calculou-se os valores de *accuracy, precision, recall e f1-score:*

- ***Accuracy***: aproximadamente 0.483;
- ***Precision***: aproximadamente 0.451;
- ***Recall***: aproximadamente 0.483;
- ***F1 score***: aproximadamente 0.443.

Ao analisar os resultados percebe-se que em aproximadamente 48% das vezes o modelo acerta na categoria da música. Contudo, entre todos os valores previstos para uma determinada classe, apenas 45% deles é que pertencem realmente a essa classe. Por outro lado, na totalidade de músicas pertencente a uma classe, cerca de 48% das previsões realizadas indicaram essa classe. Assim, é possível perceber que este modelo favorece a capacidade de prever positivos verdadeiros.

Por fim, refere-se que os valores, para a maior parte dos problemas, não são aceitáveis. No entanto, após alguma investigação, chega-se à conclusão que, para a complexidade envolvida na classificação do género de músicas, é extremamente difícil obter resultados melhores que estes, uma vez que, sendo a música uma forma expressiva de arte (de origem subjetiva e extremamente variada), os dados que existem podem gerar uma má interpretação e levar à contradição durante a criação dos modelos.

Conclusão

Neste projeto, pretendia-se a criação de modelos de *Machine Learning* para dois problemas distintos: um problema de previsão de trânsito (em formato de competição no *kaggle*) e um problema à nossa escolha.

Quanto ao primeiro problema, acreditámos ter feito um ótimo trabalho, uma vez que obtivemos uma boa classificação na tabela de classificações final.

Na verdade, este problema requereu imenso trabalho e estudo da nossa parte. Inicialmente, foi necessária a total compreensão e análise do contexto de negócio em que o problema se insere. Esta análise inclui a construção de relações entre os vários atributos da tabela e todas as variáveis envolvidas no trânsito automóvel no mundo real. Este estudo, juntamente com um consequente exaustivo tratamento dos dados permitiu-nos obter ótimos resultados com várias técnicas de criação de modelos de *Machine Learning*, destacando-se a técnica de redes neuronais.

Na segunda fase deste projeto, procurámos escolher um problema de *Machine Learning* que, não só nos permitisse colocar em prática técnicas que não conseguimos aplicar no primeiro problema, mas também queríamos um problema que nos proporcionasse um verdadeiro desafio, na medida em que seria necessária uma grande compreensão do seu contexto e, também, um ótimo tratamento de dados para obtermos resultados satisfatórios.

O problema da classificação de músicas, apesar de ser também um problema de classificação, proporcionou-nos a grande dificuldade de criar um modelo de *Machine Learning* para problemas baseados em áreas cuja subjetividade é um grande fator. Deste modo, fomos capazes de compreender que, para diferentes problemas, os valores de resultado aceitáveis estão em intervalos completamente diferentes.

Em suma, fomos capazes de cumprir, satisfatoriamente, todos os objetivos de ambos os problemas. Não só conseguimos um ótimo resultado no problema de previsão de trânsito como também fomos capazes de aplicar novas técnicas e interpretar problemas fora do comum, tal como o problema de classificação de músicas.