Classificador LGBM Aplicado na Recomendação de Músicas para o Conjunto de dados Deezer

Caique Henrique Santos Matos[1]

[1]Departamento de Ciências Exatas – Universidade Federal de Alfenas
Rua Gabriel Monteiro da Silva, 714 – Alfenas – MG – Brasil – CEP: 37130-000

caiquehsm@gmail.com

Abstract. *In 2017 there was a competition on the Kaggle platform in which a set of data from the company Deezer was made available. This data set contained information about users of the music application who used a music recommendation tool. During the competition the objective was to classify which user had or did not accept this recommendation. Therefore, this work proposes to demonstrate the construction of a Microsoft LightGBM® algorithm applied to this task. The classifier was able to achieve ratings of approximately 0.7 on the AUC rating metric.*

Resumo. *Em 2017 houve uma competição na plataforma Kaggle na qual um conjunto de dados da empresa Deezer foi disponibilizada. Neste conjunto de dados continham informações sobre usuários do aplicativo de música que utilizaram uma ferramenta de recomendação musical. Durante a competição o objetivo era classificar qual usuário havia ou não aceitado esta recomendação. Diante disso, este trabalho propõe demonstrar construção de um algoritmo Microsoft LightGBM® aplicado a esta tarefa. O classificador conseguiu atingir avaliações de aproximadamente 0,7 na métrica de avaliação AUC.*

1. Introdução

Atualmente, o aprendizado de máquina está presente em diversas tarefas do dia a dia, mesmo nas mais simples, como fazer compras ou visitar o banco. É impossível nos livrarmos de qualquer recomendação, classificação ou previsão no nosso dia-a-dia com tanta tecnologia. É por isso que algoritmos de aprendizado de máquina se tornaram tão populares. Desta maneira, trabalhar em problemas de recomendações para usuários pode ser muito importante.

O algoritmo mencionado no título deste trabalho é baseado em aumento do vetor gradiente e foi implementado pela Microsoft em abril de 2017 com o objetivo de otimizar a performance das estruturas de árvore de decisão utilizadas até o o momento [Abou Omar 2018].

Um grande número de algoritmos com objetivos similares foram desenvolvidos e melhorados em um passado recente. Dessa forma, recomendar um produto ou prever a escassez em uma loja passou a ser meta de muitos programadores.

Neste trabalho iremos desenvolver um classificador utilizando o framework Microsoft LightGBM® com objetivo de predizer a aceitação de um usuário em relação à uma ferramenta de recomendação de faixas de um aplicativo de música. Esta ferramenta se chama Flow e faz parte das funções do aplicativo de músicas francês Deezer.

A principal tarefa de um sistema de recomendação de música é propor músicas interessantes, consistindo em uma mistura de artistas conhecidos e desconhecidos, bem como as faixas disponíveis, de acordo com um perfil de usuário, colocando-as em uma lista de artistas ou criar uma sequência ordenada de músicas (uma lista de reprodução personalizada) [Celma 2010].

Portanto, dividido por seções, este relatório pretende demonstrar a criação e os resultados obtidos de um modelo classificador capaz de avaliar a assertividade de recomendação da ferramenta Flow utilizando dados de treinamento providos pela empresa Deezer.

2. Fundamentação Teórica

2.1. Recomendação de Música

Segundo a revista online businessofapps.com, sobrevivendo à transição para a tecnologia móvel, uma das maiores empresas do mercado de streaming de música abriu o capital em abril de 2018, com um valor de mercado de 26,5 bilhões de dólares após o primeiro dia de negociação. Ainda com ótimos números, outro concorrente bateu a marca de 7 milhões de usuários pagantes durante o ano de 2019.

Estes fatos são evidências da importância do mercado musical para os consumidores finais, por isso utilizar bons sistemas de recomendação de músicas pode fazer grande diferença.

Os ouvintes estão cada vez mais encontrando músicas de seu interesse na web, em vez de através de outros canais de distribuição. Isso representa uma grande oportunidade para novos artistas apresentarem sua música a grandes públicos, uma vez que a internet tem barreiras de entrada relativamente baixas em relação a gravadores e rádios[Logan 2004].

No entanto, as técnicas estabelecidas para a criação dos sistemas de recomendação podem influenciar muito neste processo, pois podem levar muito em consideração opiniões ou listas de reprodução geradas pelo público, ou metadados gerados por especialistas. Em contra partida, também possível utilizar apenas os dados do áudio para a recomendação.

No cenário da ferramenta Flow, base de dados deste trabalho, utilizaremos informações que vão além do áudio. Dados como o gênero do álbum, a plataforma de utilização do aplicativo, o álbum da música entre outros estarão presentes em nossa base de dados.

2.2. Conceitos Computacionais

2.2.1. Árvores de Decisão

Voltando a abordagem técnica, na base do algoritmo escolhido para este trabalho estão presentes como estruturas de dados principais as árvores de decisão.

A estrutura de dados em árvore tem sido uma das estruturas de dados mais populares estudadas por cientistas da computação universitários no passado recente.

Uma estrutura em árvore pode modelar soluções para pesquisar, organizar, armazenar e outras funções adicionais, que ajudam em problemas do mundo real. Dessa forma, pode ser útil em estruturas de aprendizado de máquina, com certeza.

Este modelo de dados consiste em organizar e dividir os dados em partições denominadas nós a partir do primeiro nó e conectá-los por meio de arestas. No entanto, essa estrutura simples deu origem ao modelo de árvore de decisão.

A árvore de decisão é uma ferramenta de apoio à tomada de decisão. Podemos querer usá-lo para organizar nossa sequência e a consequência de cada decisão. Também podemos calcular os resultados dos eventos casuais, custos de recursos e utilidade [Freitas Junior 2019].

2.2.2. Microsoft LightGBM®

LGBM, uma abreviatura de Light Gradient Boosting Machine, é uma estrutura de aumento de gradiente distribuída gratuita e de código aberto para aprendizado de máquina desenvolvida pela Microsoft. É baseado em algoritmos de árvore de decisão e pode ser aplicado para regressão, classificação e outras tarefas de aprendizado de máquina.

Usando esta ferramenta gratuita da Microsoft, podemos modelar um classificador para o problema de recomendação de músicas do Deezer.

Além disso, é significativo notar que as melhorias do LGBM e o foco no desempenho ajudam muito na experimentação. O objetivo era tornar o aumento de gradiente nas árvores de decisão mais rápido, modificando a forma de dividir e criar mais folhas nas árvores [Abou Omar 2018].

2.3. Tecnologias Utilizadas

As tecnologias utilizadas foram: a linguagem de integração Python 3, alguns pacotes suportados por ela e dois tipos de arquivos de leitura e escrita melhor descritos na subseção 2.4. A utilização do Python foi fundamental para integração de bibliotecas que facilitaram todo o desenvolvimento.

Os pacotes e bibliotecas utilizadas foram Pandas, Numpy, Matplotlib, Seaborn, Scikit-learn e o LightGBM, utilizando as classes DataFrame e Series e, por último, o tipo de arquivo utilizado foi CSV.

O pandas oferece leitura e escrita de arquivos do tipo CSV transformando-os em objetos DataFrame. Esses objetos são similares a tabelas de banco de dados e possuem funções de busca e manipulação de dados que, em conjunto com o Numpy, são rápidas. Além disso, oferece a criação de imagens e gráficos das tabelas através do pacote Matplotlib [Freitas Junior 2019].

O pacote Pandas Profiling também foi utilizado para análise exploratória de dados.Por final, para modelagem do classificador, foi utilizado o pacote LightGBM possui o algoritmo LGB. Este algoritmo além das funções de aprendizado para classificação contém outras métricas e objetivos que não foram utilizados neste trabalho.

2.4. Conjunto de Dados Deezer

Deezer é um aplicativo de música online francês gratuito que oferece a seus usuários mais de 43 milhões de faixas em mais de 180 países. Em 2017 em parceria com o Data Science Game, grupo também francês dedicado a organização de competições na área de ciência de dados para estudantes, o Deezer disponibilizou uma base de dados para uma competição entre estudantes por todo o mundo por meio da plataforma Kaggle.

Esta base esteve disponível para download entre os anos de 2017 e 2020, ainda pelo link oficial da competição[1].

Neste este desafio online, a empresa deu foco a sua própria ferramenta de recomendação. O conceito do Flow é simples: ele usa filtragem colaborativa para fornecer a um usuário a música que ele deseja ouvir na hora que ele deseja. E caso ele não queira ouvir algumas faixas específicas ele pula músicas pressionando o botão 'Próxima música'. Nesse contexto, acertar na recomendação da primeira música é muito importante.

2.4.1. Objetivo

O objetivo deste desafio foi prever se os usuários do conjunto de dados de teste ouviram a primeira faixa que Flow os propôs ou não. O Deezer o considera que uma faixa é "ouvida"se o usuário a ouviu por mais de 30 segundos (is_listened = 1). Se o usuário pressionar o botão 'Próxima música' antes de 30 segundos, a faixa não será considerada como ouvida (is_listened = 0).

2.4.2. Métrica de Avaliação

A métrica de avaliação AUC, abreviação de Área sob a curva, foi escolhida para este trabalho. É uma métrica de avaliação criada para simplificar a curva ROC. A curva ROC representa dois valores significativos para um modelo de aprendizado de máquina:

- Taxa de Verdadeiro Positivo;
- Taxa de falso positivo.

Taxa de Positivo Verdadeiro (TPR) é definido da seguinte forma:

$$TPR = \frac{TP}{TP+FN}$$

No nosso caso, o TPR ocorre quando o modelo classifica corretamente uma recomendação que o usuário aceitou.

A taxa de falso positivo (FPR) é definida da seguinte forma:

$$FPR = \frac{FP}{FP+TN}$$

[1] https://www.kaggle.com/c/dsg17-online-phase

2.4.3. Conjunto de Dados de Teste

O conjunto de dados de teste consiste em uma lista das primeiras faixas recomendadas no Flow para vários usuários. Cada linha representa um usuário.

2.4.4. Conjunto de Dados de Treinamento

O conjunto de dados do treinamento foi gerado usando o histórico desses usuários do Deezer por um mês. Cada linha representa uma música ouvida. A lista de usuários distintos no conjunto de dados do trem corresponde exatamente ao do conjunto de dados de teste.

Nesta base de dados encontraremos campos contendo informações sobre os usuários que utilizaram a ferramenta de recomendação do Deezer: Flow.

Desta forma, teremos informações sobre o usuário e seu histórico de consumo de faixas.

3. Desenvolvimento

3.1. Análise de Dados

A análise exploratória de dados e a estatística estão presentes na humanidade desde a antiguidade e servem como base inicial para análises de fenômenos em dados [MEDRI 2011].

Desta maneira, nosso primeiro passo foi desenvolver esta análise em busca de identificar e entender as informações contidas dentro da base de dados Deezer em relação a coluna (is_listened) que será alvo do nosso classificador para avaliar a possibilidade da modelagem.

3.1.1. Lista de Colunas

O conjunto de dados contém os seguintes campos de dados:

- Media_id: Identificador da música ouvida pelo usuário;
- Album_id - Identificador do álbum da música;
- Media_duration - Duração da música;
- User_gênero - gênero do usuário;
- User_id - Identificador anônimo do usuário;
- Context_tipo - Tipo de conteúdo onde a música foi ouvida: lista de reprodução, álbum...
- Release_date - Data de lançamento da música com o formato AAAAMMDD
- Ts_listen - Carimbo de data / hora da escuta no horário UNIX
- Platform_nome - Tipo de sistema operacional do usuário;
- Platform_family - Tipo de dispositivo;
- User_age - Idade do usuário;
- Listen_type - Se as músicas foram ouvidas em um fluxo ou não;
- Artist_id - Identificador do artista da música;
- Genre_id - Identificador do gênero da música;
- Is_listened - 1 se a faixa foi escutada, 0 caso contrário - um id anônimo único para um determinado cliente.

3.1.2. Análise de Informações

Observando o significado das colunas presentes no conjunto de dados, podemos perceber uma alta quantidade de variáveis categóricas. Estas variáveis são aquelas em que a variável assume "valores" em categorias, classes ou rótulos. São, portanto, por natureza, dados não numéricos que denotam características individuais das unidades sob análise [MEDRI 2011]. Porém, em nossa base podemos notar que estes dados foram mapeados para valores numéricos para rotular suas características.

Devido as informações nestas variáveis podemos conjecturar que elas sejam de grande relevância para um usuária durante a utilização do Flow. Por isso, foram realizadas análises em colunas que poderiam ser de alta relevância para o modelo LGBM.

A primeira coluna a ser estudada foi a de gênero que assim como as demais variáveis categóricas foi explorada a partir da contagem da quantidade de valores, em função de notar os mais comuns.

Tabela 1. Contagem da Quantidade de Valores na Coluna de Gênero

Valor	Quantidade	Frequência(%)
0	3635386	48.5%
7	922421	12.3%
10	286455	3.8%

Como pudemos perceber o gênero 0 aparece em quase metade da base de dados seguido do gênero 7 e 10. Isto nos guia a explorar as quantidade de músicas e usuários distintos em nossa base para verificarmos a variedade que temos neste conjunto.

Tabela 2. Contagem da Quantidade de Valores Distintos

Coluna	Quantidade
Genre_id	2.917
Media_id	450.827
Album_id	150.593
User_id	19.426
Artist_id	66.764

Observando as tabelas podemos perceber que o conjunto de dados do Deezer fornece variedade de informações relevantes para uma possível modelagem de um classificador. Para a obtenção destas informações foi utilizado o pacote pandas profiling como observado nas linhas 51 e 71 das listas de código em anexo.

3.2. Modelagem

A primeira abordagem utilizada neste trabalho foi a de um modelo classificador LGBM sem nenhuma técnica de validação implementada.

3.2.1. Bibliotecas

Apresentada por completo na seção 6, a primeira modelagem deste relatório traz em suas primeiras linhas a importação das suas bibliotecas citadas na subseção 2.3 sobre tecnologias.

3.2.2. Parâmetros Globais

Logo após, são definidos os parâmetros globais que têm como proposito guardar os resultados das avaliações do modelo, guardar o valor da taxa de aprendizado e a quantidade de rounds de treinamento respectivamente.

3.2.3. Modelo e Função de Treinamento

Por final, envoltos por um laço de repetição, estão presentes a definição do classificador e de sua respectiva função de treinamento. Nota-se nesta parte a definição dos parâmetros de treinamento.

```
    #parameters for LightGBM
    def train_lgb(dtrain, val_sets, n_round):
        params = {
            'boosting_type': 'gbdt',
            'objective': 'binary',
            'metric': 'auc',
            'learning_rate': learn_start,
            'lambda_l1': 0.1,
            'random_state': 123,
            'verbosity': 1}

        model = lgb.train(params,
                          dtrain,
                          num_boost_round = n_round,
                          valid_sets = val_sets,
                          verbose_eval=100,
                          evals_result=evals_result,
                         early_stopping_rounds = 300)
        return model
```

Listing 1. Parâmetros de Aprendizado Utilizados para Treinamento

3.3. Validação de Aprendizado

Após a implementação da primeira abordagem, uma validação de aprendizado também foi desenvolvida em busca de validar as pontuações obtidas em treinamento em relação as pontuações obtidas no conjunto de dados de teste.

3.3.1. Técnica Train/Test split

A técnica de validação escolhida para este trabalho foi a validação chamada de divisão treino/teste devido a temporalidade do conjunto de dados. Consiste em separar seu banco de dados em duas partes e cada iteração usa cada parte especificamente para sua função. Isso significa que, a cada iteração, o modelo é treinado com o banco de dados de treinamento e testado em um banco de dados de teste criado artificialmente a partir do banco de dados original.

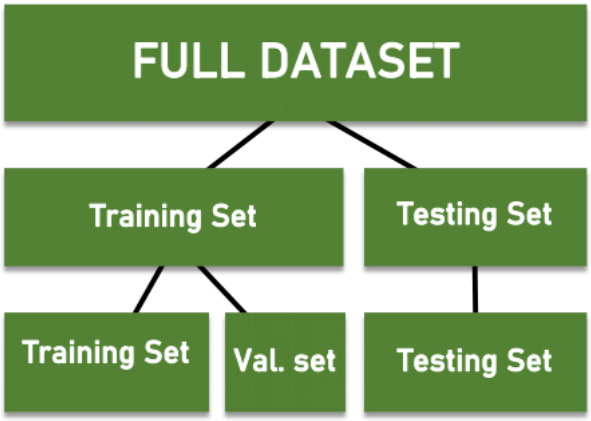


Figura 1. Ilustração da Técnica Train/Test split
Fonte do próprio autor

O modelo com base com a técnica de validação implementada é demonstrado na lista 3 de códigos em anexos na seção 6. É importante a enfase para as linhas 40 e 46 onde separamos os conjuntos e garantimos nenhum vazamento de dados para as predições.

4. Resultados

Após as implementações dos modelos com objetivo de classificarmos a coluna Is_listened que nos diz se um usuário aceitou ou não a recomendação da ferramenta Flow, partimos para as execuções dos algoritmos e seus resultados.

Utilizando o código da lista 2 dos anexos foram gerados 14 modelos, de modo com que os parâmetros variem dentro do laço de repetição e os parâmetros globais são colocados manualmente, os seguintes resultados foram encontrados:

Tabela 3. Modelos com Taxa de Aprendizado fixa em 1.2 com variação em Rounds de Treinamento

Modelo	Taxa de Aprendizado	Rounds	Treino AUC	Teste AUC	Delta AUC
1	1.2	500	0.724402	**0.60172**	0.122682
4	1.2	1000	0.5988	**0.57336**	0.02544
7	1.2	5000	0.5887	**0.53336**	0.02534
10	1.2	10000	0.541038	**0.51198**	0.029058

Observando a tabela 3 podemos perceber que a melhor nota desta tabela está no Modelo 1. Porém, com o aumento de rounds de treinamento houve um decréscimo da nota tanto em treinamento quanto em teste.

Tabela 4. Modelos com Taxa de Aprendizado fixa em 0.24 com variação em Rounds de Treinamento

Modelo	Taxa de Aprendizado	Rounds	Treino AUC	Teste AUC	Delta AUC
2	0.24	500	0.757835	**0.62609**	0.131745
5	0.24	1000	0.781775	**0.64293**	0.138845
8	0.24	5000	0.84115	**0.68726**	0.15389
11	0.24	10000	0.869386	**0.70129**	0.168096
13	0.24	20000	0.900622	**0.70071**	0.199912

Entretanto, a tabela 4 demonstrou um cenário completamente diferente. É notável o aumento das pontuações de treinamento e teste durante o aumento do tempo de treinamento, isto só não ocorre na última linha, no Modelo 13.

Tabela 5. Modelos com Taxa de Aprendizado fixa em 0.048 com variação em Rounds de Treinamento

Modelo	Taxa de Aprendizado	Rounds	Treino AUC	Teste AUC	Delta AUC
3	0.048	500	0.713389	**0.60260**	0.110789
6	0.048	1000	0.73081	**0.61176**	0.11905
9	0.048	5000	0.781672	**0.64542**	0.136252
12	0.048	10000	0.808194	**0.66345**	0.144744
14	0.048	20000	0.834298	**0.68726**	0.147038

Por fim, na tabela 5 foi encontrado um cenário muito parecido com o da tabela 4, sendo que não foi notado nenhuma linha decréscimo nas avaliações. Todos os gráficos de treinamento destes modelos e todos os outros modelos estão presentes na seção de anexos.

Ao fim destas execuções, mais quatro foram feitas. Duas afim de executar validações e outras respectivas para gerarem suas saídas.

Os resultados encontrados com a execução do algoritmo com as técnicas de validação foram:

```
Early stopping, best iteration is:
[621]   training's auc: 0.737891        valid_1's auc: 0.630863
Plot metrics during training with Learning rate: 1.2
```

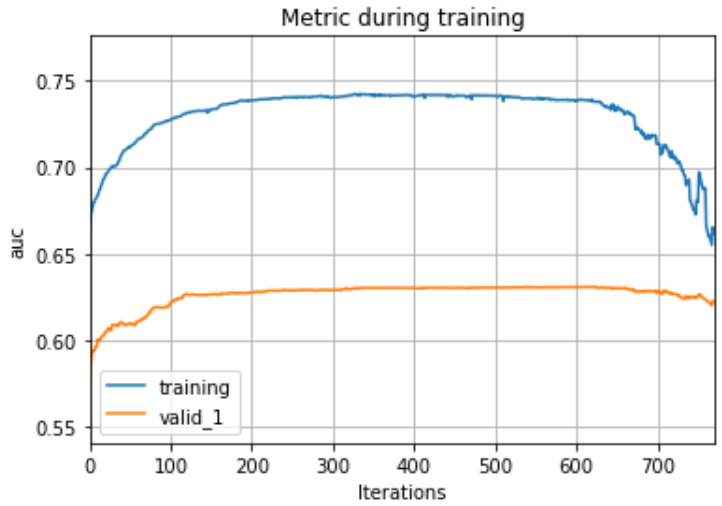


Figura 2. Gráfico do Treinamento e Validação do Modelo com Taxa de Aprendizado de 1.2

Fonte do próprio autor

Tabela 6. Resultado do Modelo de Validação com Modelo Base com Taxa de Aprendizado de 1.2

Modelo	Taxa de Aprendizado	Rounds	Treino AUC	Teste AUC	Delta AUC
1	1.2	500	0.724402	**0.60172**	0.122682
4	1.2	1000	0.5988	**0.57336**	0.02544
7	1.2	5000	0.5887	**0.53336**	0.02534
10	1.2	10000	0.541038	**0.51198**	0.029058
Valid 1.2	1.2	621	**Valid(0.630863)**	**0.61196**	0.018903

A tabela 6 tem em sua última linha o resultado da execução de um modelo implementado com a técnica de validação Train/Test split. Note que a nota de validação com a métrica de avaliação AUC consegue diminuir o valor da coluna delta, além de aumentar a nota do teste.

```
Early stopping, best iteration is:
[4980]  training's auc: 0.843005        valid_1's auc: 0.709713
Plot metrics during training with Learning rate: 0.24
```

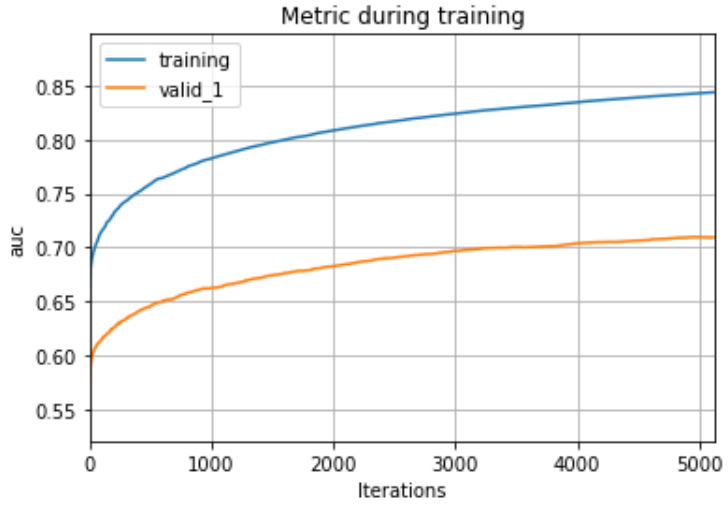


Figura 3. Gráfico do Treinamento e Validação do Modelo com Taxa de Aprendizado de 0.24

Fonte do próprio autor

Tabela 7. Resultado do Modelo de Validação com Modelo Base com Taxa de Aprendizado de 1.2

Modelo	Taxa de Aprendizado	Rounds	Treino AUC	Teste AUC	Delta AUC
2	0.24	500	0.757835	**0.62609**	0.131745
5	0.24	1000	0.781775	**0.64293**	0.138845
8	0.24	5000	0.84115	**0.68726**	0.15389
11	0.24	10000	0.869386	**0.70129**	0.168096
13	0.24	20000	0.900622	**0.70071**	0.199912
Valid 0.24	0.24	4980	**Valid(0.709713)**	**0.69856**	0.011153

Já a tabela 7, tem em sua última linha o resultado da execução de um modelo implementado com a técnica de validação Train/Test split com a taxa de aprendizado com valor 0.24. Note que neste caso a nota de validação com a métrica de avaliação AUC consegue diminuir o valor da coluna delta, e fica com uma avaliação no conjunto de teste muito próxima ao modelo 11 com apenas aproximadamente metade dos rounds de treinamento.

5. Considerações

Pode-se então concluir que foi possível a construção do classificador para as recomendações e objetivo para o conjunto de dados do Deezer. Observando os resultados também conseguimos notar que utilizando a validação conseguimos ter uma ideia melhor de qual a avaliação do modelo no conjunto de teste durante o treinamento, uma vez que as notas de validação são sempre mais próximas as notas de teste do que as de treinamento.

Portanto, utilizar a validação de aprendizado pode deixar este modelo de classificação mais robusto e em alguns casos aumentar até sua avaliação, como observado na tabela 6.

6. Códigos e Anexos

```
import numpy as np
import pandas as pd
import lightgbm as lgb
import matplotlib.pyplot as plt

# True for validation or False to generate predictions
is_valid = False
evals_result= {}
learn_start=1.2
times=500
for learn in range(3):
    #parameters for LightGBM
    def train_lgb(dtrain, val_sets, n_round):
        params = {
            'boosting_type': 'gbdt',
            'objective': 'binary',
            'metric': 'auc',
            'learning_rate': learn_start,
            'lambda_l1': 0.1,
            'random_state': 123,
            'verbosity': 1}

        model = lgb.train(params,
                          dtrain,
                          num_boost_round = n_round,
                          valid_sets = val_sets,
                          verbose_eval=100,
                          evals_result=evals_result,
                         early_stopping_rounds = 300)
        return model

    train = pd.read_csv(path+'input/train.csv')
    test = pd.read_csv(path+'input/test.csv')
    # to record eval results for plotting

    y_train = train['is_listened']
    del train['is_listened']

    features = train.columns
    d_train = lgb.Dataset(train, y_train)

    model = train_lgb(d_train, val_sets=[d_train], n_round=times)
    preds = model.predict(test[features])

    sub = pd.DataFrame({'sample_id': test['sample_id'], 'is_listened':
   preds})
    sub.to_csv(str(learn_start)+'-'+str(times)+'sub.csv', index=False)
    learn_start=learn_start/5
    print('Plot metrics during training with Learning rate:',
   learn_start)
    ax = lgb.plot_metric(evals_result, metric='auc')
    plt.show()
train.profile_report()
```

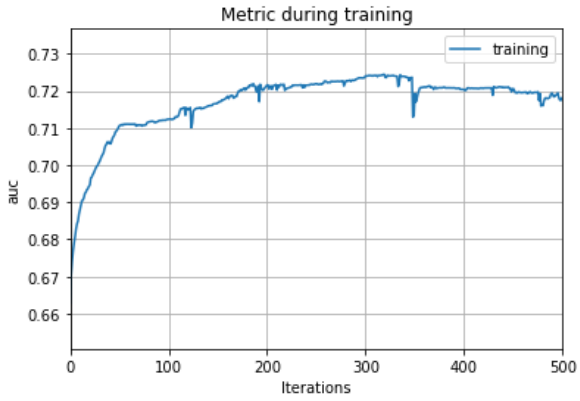
Listing 2. First LGBM Model

```
import numpy as np
import pandas as pd
import lightgbm as lgb
import matplotlib.pyplot as plt

# True for validation or False to generate predictions
is_valid = False
evals_result= {}
learn_start=1.2
times=500

for learn in range(3):

    #parameters for LightGBM
    def train_lgb(dtrain, val_sets, n_round):
        params = {
            'boosting_type': 'gbdt',
            'objective': 'binary',
            'metric': 'auc',
            'learning_rate': learn_start,
            'lambda_l1': 0.1,
            'random_state': 123,
            'verbosity': 1}

        model = lgb.train(params,
                          dtrain,
                          num_boost_round = n_round,
                          valid_sets = val_sets,
                          verbose_eval=100,
                          evals_result=evals_result,
                         early_stopping_rounds = 300)
        return model

    train = pd.read_csv(path+'input/train.csv')
    test = pd.read_csv(path+'input/test.csv')
    # to record eval results for plotting

    if (is_valid == True):

        train, valid = split_validation(train)


        y_train = train['is_listened']
        y_valid = valid['is_listened']

        del train['is_listened'], valid['is_listened']
        d_train = lgb.Dataset(train, y_train)
        d_valid = lgb.Dataset(valid, y_valid)

        model = train_lgb(d_train, val_sets=[d_train, d_valid], n_round
   =times)

    else:

        y_train = train['is_listened']
        del train['is_listened']
```

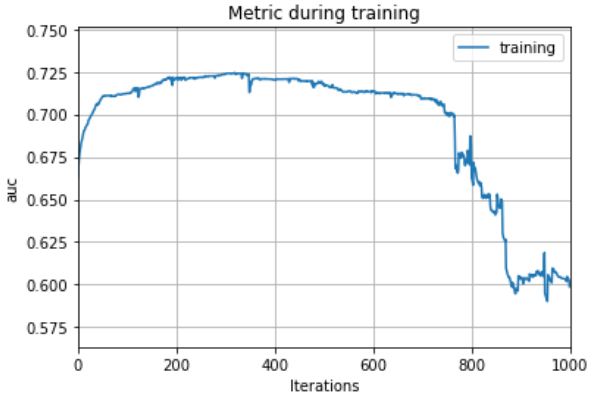
```
56
57          features = train.columns
58          d_train = lgb.Dataset(train, y_train)
59
60          model = train_lgb(d_train, val_sets=[d_train], n_round=times)
61          preds = model.predict(test[features])
62
63          sub = pd.DataFrame({'sample_id': test['sample_id'], '
    is_listened': preds})
64          sub.to_csv(str(learn_start)+'-'+str(times)+'sub.csv', index=
    False)
65
66      learn_start=learn_start/5
67
68      print('Plot metrics during training with Learning rate:',
    learn_start)
69      ax = lgb.plot_metric(evals_result, metric='auc')
70      plt.show()
71  train.profile_report()
```

Listing 3. LGBM Model with Validation case

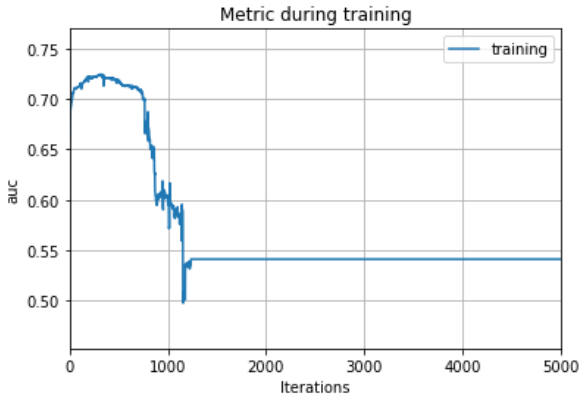
Figura 4. Plots of models from 3 during training.



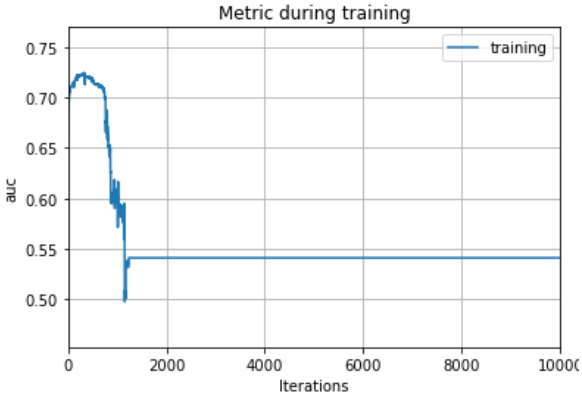
(a) Model 1.



(b) Model 4.

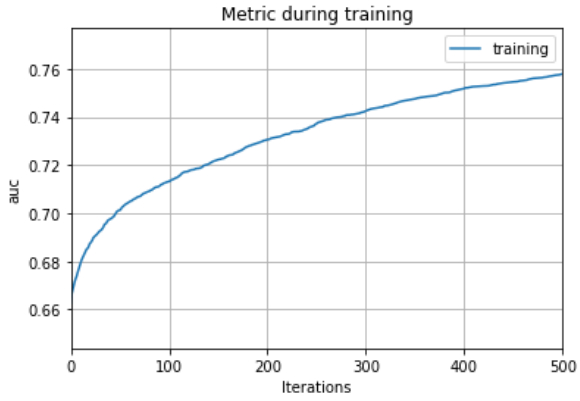


(c) Model 7.

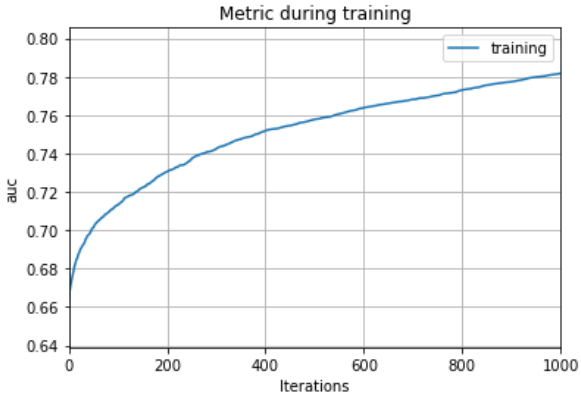


(d) Model 10.

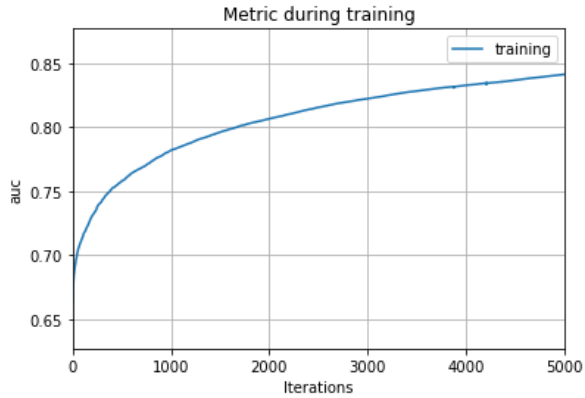
Figura 5. Plots of models from 4 during training.



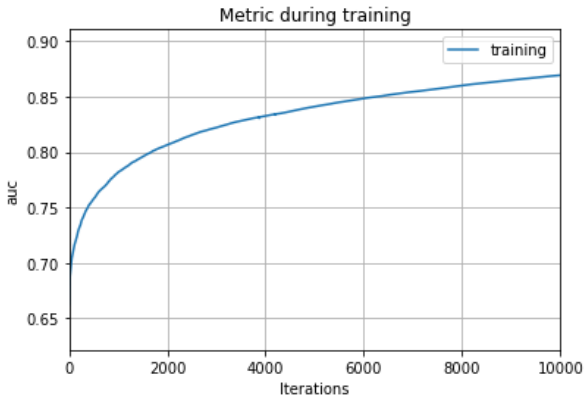
(a) Model 2.



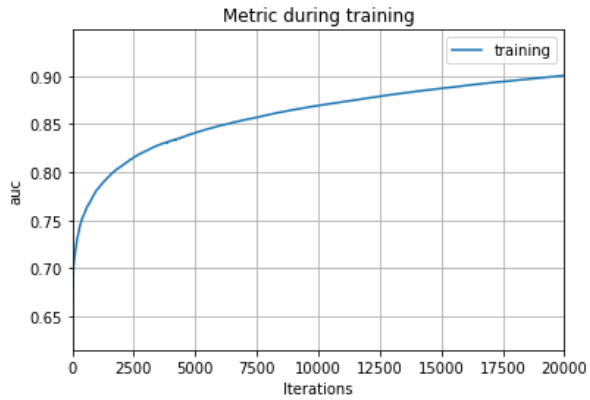
(b) Model 5.



(c) Model 8.

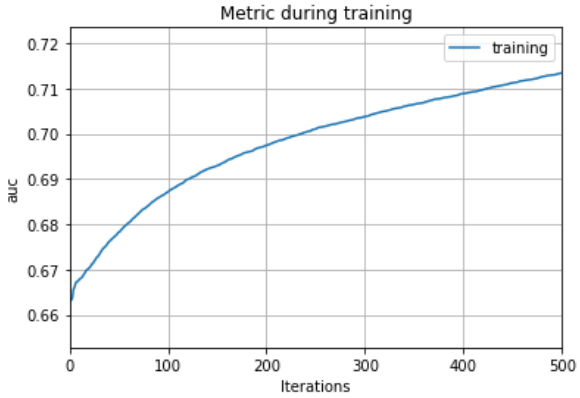


(d) Model 11.

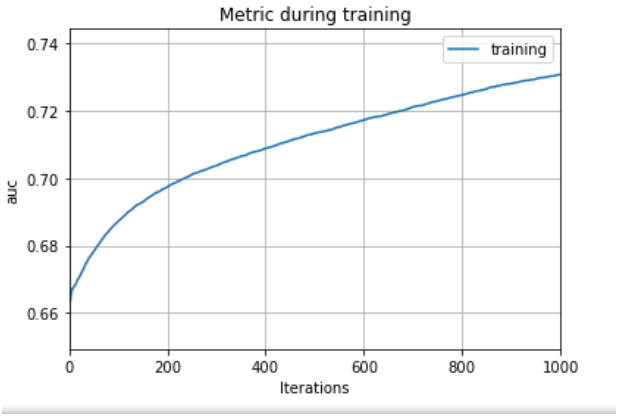


(e) Model 13.

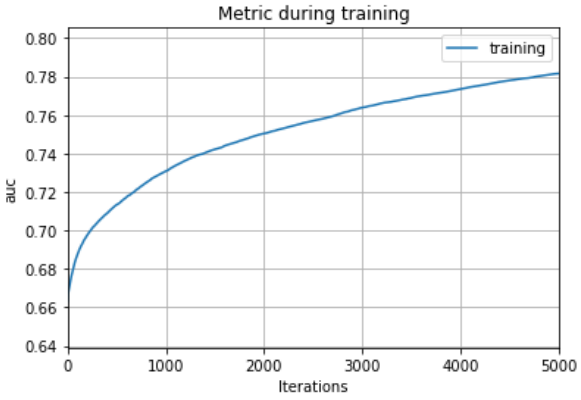
Figura 6. Plots of models from 5 during training.



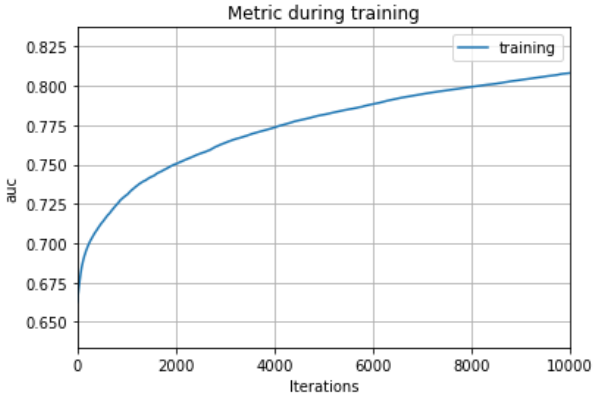
(a) Model 3.



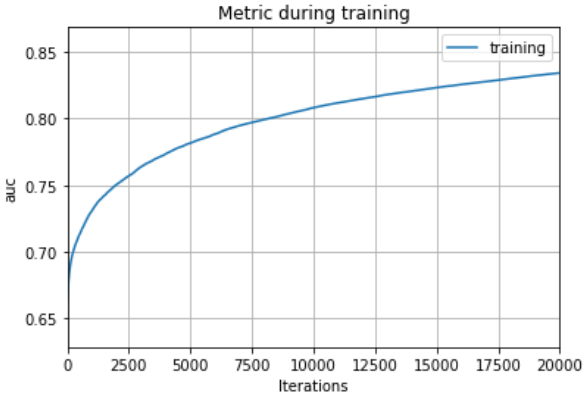
(b) Model6.



(c) Model 9.



(d) Model 12.



(e) Model 14.

Referências

Abou Omar, K. B. (2018). Xgboost and lgbm for porto seguro's kaggle challenge: A comparison. *Preprint Semester Project*.

Celma, O. (2010). Music recommendation. In *Music recommendation and discovery*, pages 43–85. Springer.

Freitas Junior, M. A. d. (2019). Classificador lgbm aplicado na resolução do desafio plasticc do telescópio lsst.

Logan, B. (2004). Music recommendation from song sets. In *ISMIR*, pages 425–428.

MEDRI, W. (2011). Análise exploratória de dados. *http://www. uel. br/pos/estatisticaeducacao/textos_d idaticos/especializacao_estatistica. pdf Acesso em*, 15:05–13.