UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
UNIDADE ACADÊMICA DE SISTEMAS E COMPUTAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

LEONARDO LIMA FELIX DA SILVA

USO DE *AUTOMATED MACHINE LEARNING* (AUTO ML) EM SISTEMAS DE RECOMENDAÇÃO

CAMPINA GRANDE - PB

2021

LEONARDO LIMA FELIX DA SILVA

USO DE *AUTOMATED MACHINE LEARNING* (AUTO ML) EM SISTEMAS DE RECOMENDAÇÃO

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Professor Dr. Claudio Elízio Calazans Campelo.

CAMPINA GRANDE - PB

2021

S586u Silva, Leonardo Lima Felix da.
Uso de Automated Machine Learning (Auto ML) em sistemas de recomendação. / Leonardo Lima Felix da Silva. – 2021.

10 f.

Orientador: Prof. Dr. Claudio Elízio Calazans Campelo.
Trabalho de Conclusão de Curso - Artigo (Curso de Bacharelado em Ciência da Computação) - Universidade Federal de Campina Grande; Centro de Engenharia Elétrica e Informática.

1. Sistema de recomendação. 2. AutoML. 3. Automated Machine Learning. 4. Predição do próximo ítem. 5. AutoSklearn. I. Campelo, Claudio Elízio Calazans. II. Título.

CDU:004(045)

Elaboração da Ficha Catalográfica:

Johnny Rodrigues Barbosa
Bibliotecário-Documentalista
CRB-15/626

LEONARDO LIMA FELIX DA SILVA

USO DE *AUTOMATED MACHINE LEARNING* (AUTO ML) EM SISTEMAS DE RECOMENDAÇÃO

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA:

Professor Dr. Claudio Elízio Calazans Campelo
Orientador – UASC/CEEI/UFCG

Professor Dr. Eanes Torres Pereira
Examinador – UASC/CEEI/UFCG

Professor Dr. Tiago Lima Massoni
Disciplina TCC – UASC/CEEI/UFCG

Trabalho aprovado em: 20 de outubro de 2021.

CAMPINA GRANDE – PB

ABSTRACT

Recommendation Systems (RS) are responsible for filtering content and making recommendations based on users' preferences, these intelligent models are being used by several companies and in different areas. Due to the advancement of machine learning models, there is an increasing number of researches or tools that seek to automate the task performed by them, these tools are also known as AutoML. AutoML has been a tool to suggest the best model and the best hyperparameters for a dataset. Therefore, this work proposes to discuss and compare, among some AutoML tools, if they can be suitable for use in recommendation systems scenarios.

Uso de Automated Machine Learning (Auto ML) em Sistemas de Recomendação

Trabalho de Conclusão de Curso

Leonardo Lima Felix da Silva (Aluno), Cláudio Campelo (Orientador)
Departamento de Sistemas e Computação
Universidade Federal de Campina Grande
Campina Grande, Paraíba - Brasil

RESUMO

Os Sistemas de recomendação (SR) são responsáveis por filtrar conteúdos e realizar recomendações baseadas nas preferências dos usuários, esses modelos inteligentes estão sendo utilizados por diversas empresas e em diferentes áreas. Com o avanço dos modelos de aprendizado de máquina, está cada vez maior o número de pesquisas ou ferramentas que buscam automatizar a tarefa realizada por elas, essas ferramentas também são conhecidas como AutoML. AutoML vem sendo uma ferramenta para sugerir o melhor modelo e os melhores hiperparâmetros para um conjunto de dados. Dessa forma, este trabalho propõe discutir e comparar, dentre algumas ferramentas de AutoML, se podem ser adequadas para o uso em cenários de sistemas de recomendação.

PALAVRAS-CHAVE

Sistema de Recomendação, AutoML, predição do próximo item, TPOT, FLAML, AutoSklearn

1 INTRODUÇÃO

Os Sistemas de Recomendação (SR) são modelos inteligentes conhecidos por auxiliar os usuários a tomar decisões levando em consideração seus interesses, gostos pessoais e restrições. Os SR são modelados utilizando as opiniões e *feedbacks* de usuários ou as similaridades entre os itens consumidos. Atualmente existem duas abordagens muito utilizadas e conhecidas, a filtragem por conteúdo e a filtragem colaborativa. Na filtragem por conteúdo, o modelo recomenda itens que possuem similaridades, podendo ser feito através de *tags* que descrevem o item ou metadados que informem o criador e gênero. A filtragem colaborativa é construída a partir das interações entre usuários e conteúdos, recomendando interesses similares de outros usuários de forma coletiva.

Grandes empresas como Google, Facebook e Uber estão desenvolvendo ferramentas para automação de projetos de aprendizado de máquina. Os AutoML [16], estão sendo amplamente discutidos como uma ferramenta capaz de realizar automação em todos os estágios iniciais de um projeto, como limpeza dos dados, modelagem, treinamento, *pipeline* dos dados entre outros. Uma ferramenta de AutoML também tem como objetivo escolher o melhor modelo com os melhores hiperparâmetros, realizando assim um trabalho de engenharia dos dados de maneira rápida e barata.

Entretanto, ao usar AutoML, a escolha de *features* realizada de forma automática pode não ser tão eficaz, visto que, para problemas de aprendizado de máquina, o estudo do conhecimento de domínio e sua melhor utilização, podem ajudar o modelo a acertar mais. Outro ponto, é que métodos de automação devem ser utilizados com cautela por projetistas de modelos de aprendizado, para que não as utilizem sem que antes possuam conhecimento sobre algoritmos de aprendizado de máquina e suas sutilezas. Além disso, as decisões de modelagem tomadas pelas ferramentas não são totalmente transparentes. Por exemplo, algumas vezes, não são claros os critérios adotados para escolha de alguns algoritmos de aprendizagem durante a etapa de treinamento dos modelos, em detrimento a outros. Em geral, isso acontece pela escassez de informação em algumas documentações de ferramentas de AutoML.

Diante deste cenário, o trabalho tem como objetivo comparar o uso de diferentes ferramentas de AutoML no contexto de sistemas de recomendação, utilizando uma base de dados de filmes e usuários reais [9]. Para tal, a estratégia utilizada para recomendação consiste na predição de um próximo item a ser consumido pelo usuário, seguindo a abordagem de filtragem colaborativa. As ferramentas foram avaliadas em termos de eficácia e eficiência. No que tange as métricas de eficácia, os modelos foram avaliados em termos de acurácia e *Normalized Discounted Cumulative Gain* (NDCG). Por sua vez, para a avaliação da eficiência dos modelos, foram utilizadas as métricas de tempo de processamento, consumo de memória e consumo de CPU.

Nos experimentos realizados, nenhuma das ferramentas conseguiu atingir resultados consideráveis, ou seja, obtiveram valores de acurácia e NDCG muito próximos a zero. Isso se dá pelo fato de que a recomendação, baseada na predição de um próximo item, em uma base com milhões de filmes, é uma tarefa muito difícil para um modelo de classificação.

Todavia, mesmo com os baixos resultados, o presente trabalho é útil para que possa ser uma base de experimentos realizados com AutoML em um contexto de sistemas de recomendação. Tal informação é válida para que outras pesquisas surjam com a finalidade de utilizar AutoML para o mesmo fim, fazendo uso de outras ferramentas de AutoML, ou até mesmo, outras estratégias de recomendação.

2 FUNDAMENTAÇÃO E TRABALHOS RELACIONADOS

Ferramentas de AutoML são úteis para ajudar as pessoas com pouco ou nenhum conhecimento de aprendizado de máquina para criar modelos personalizados e de alta qualidade, uma vez que possuem um *pipeline* consistente. Esse *pipeline* é composto por preparação dos dados, engenharia de recursos, geração e avaliação de modelos. É interessante notar que alguns modelos de AutoML trabalham tanto com algoritmos de aprendizado convencionais como *k-nearest neighbors* (KNN), *Support Vector Machine* (SVM), como também redes neurais. Os AutoML também podem auxiliar na busca dos melhores hiperparâmetros para cada modelo de aprendizado, ou, no caso de redes neurais, valores ótimos de taxa de aprendizado, *dropout*, números de camadas e neurônios. [16].

Ferramentas de AutoML são experimentadas em diversas áreas ou natureza dos dados. O estudo realizado por Brandle et al. [12] propôs o uso de ferramentas de AutoML em dados textuais (não estruturados). Entretanto, este uso de AutoML mostrou um desempenho inferior se comparado a técnicas específicas para problemas de aprendizado de máquina em dados textuais, como *bag of words* ou *word2vec*.

O uso de ferramentas de AutoML também é feito na área de saúde, como no caso de tarefas de classificação de imagens médicas [5], ou até mesmo na predição de diabetes de algum indivíduo, mostrando que essas ferramentas podem ser exploradas em diversos cenários e tipos de dados.

Dado que as ferramentas de AutoML vêm sendo utilizadas em diferentes áreas, este trabalho busca comparar tais ferramentas em um contexto de sistemas de recomendação. Pieter Gijsbers et al. [10] reportaram um *Benchmark* onde são utilizadas ferramentas de AutoML em diferentes conjuntos de dados, alguns dos modelos usados nesse artigo foram: Auto-WEKA [7], AutoSklearn [8], TPOT [14] e H2O AutoML [2]. Com base nos resultados de acurácia do artigo, selecionamos 2 modelos para explorar em nosso estudo comparativo: AutoSklearn e TPOT. Além dessas, também utilizamos o FLAML [4], uma recém lançada ferramenta de AutoML desenvolvida pela Microsoft, totalizando 3 ferramentas a serem comparadas. O critério de escolha foi baseado em seu desempenho e viabilidade de instalação.

Yang et al. construíram uma ferramenta de AutoML específica para sistemas de recomendação, baseando-se na técnica de filtragem colaborativa. A ferramenta, chamada de OBOE *Collaborative Filtering for AutoML Model Selection*, difere do presente trabalho, visto que a nossa proposta é comparar as ferramentas de AutoML (de propósito geral) já criadas e bem avaliadas pela comunidade, em um contexto de sistemas de recomendação de filmes. O "Auto-CaseRec" [13], similar ao OBOE, é um *framework* exclusivo para tarefas de sistemas de recomendação, construído utilizando ferramentas de AutoML. O *framework* apresentou uma melhor eficácia quando se é utilizado para o desenvolvimento de sistemas de recomendação.

3 METODOLOGIA

A metodologia aplicada neste trabalho consiste nas etapas de coleta de dados, formatação dos dados, escolha das ferramentas de AutoML e coleta de resultados. Primeiramente, a base de dados do groupLens, chamada MovieLens [9], uma base correlacionada entre filmes e usuários muito utilizada em diversos problemas de sistemas de recomendação. Segundo, para melhor adaptação dos modelos de predição para este problema de recomendação de filmes, os dados então, foram formatados de tal forma que, para uma sequência de tamanho *n-1* de filmes vistos pelo usuário, cada sequência possua um próximo filme *n* a ser visto. A predição do próximo filme no qual o usuário poderá assistir é baseada nos seus gostos e de usuários com gostos similares, visto que o modelo irá buscar similaridades entre as sequências de filmes dos usuários. Terceiro, 3 ferramentas de AutoML foram utilizadas para realização dos experimentos propostos: FLAML, TPOT e AutoSklearn. Quarto, os resultados foram avaliados com métricas de acurácia e NDCG, além de obter informações a respeito do tempo de processamento para cada modelo de AutoML.

3.1 Base de Dados

A base de dados foi formulada e fornecida gratuitamente pela equipe do GroupLens, que é um laboratório de pesquisa de inteligência artificial e um dos primeiros a estudar e fornecer base de dados para sistemas de recomendação. A base MovieLens é composta por diversos dados a respeito dos filmes [9], como, por exemplo, os *ratings* do usuário para um determinado filme e *tags* que descrevem o filme. Entretanto, o foco deste trabalho é usar os dados correspondentes às interações dos usuários com os filmes, onde, cada registro indica que o usuário assistiu um filme específico. O tamanho original dos dados do *movielens* é de 15 milhões de avaliações de filmes, para este trabalho, por motivos de custo computacional, optamos selecionar um subconjunto contendo 1 milhão de registros.

3.2 Predição do próximo item

Nos sistemas de recomendação que utilizam a abordagem de filtragem colaborativa, é comumente visto uma implementação que utiliza fatoração de matrizes para codificar a informação relativa ao comportamento do usuário em relação aos itens vistos, ou seja, uma matriz composta por item x usuário, onde o valor associado nessa relação é o *rating* do usuário para o filme específico. Nossa estratégia difere dessa abordagem, visto que consiste em obter uma sequência temporal dos itens vistos pelo usuário, e assim realizar a predição do próximo item que poderá ser consumido.

Para realizar a implementação dessa estratégia, obtivemos toda a sequência dos itens vistos pelo usuário e segmentamos até *n-1* itens para ser nosso conjunto de *features*, ou seja, a sequência de itens que o modelo irá utilizar para aprender sobre os interesses do usuário. Portanto, o n-ésimo item, que é o próximo item visto pelo usuário, é utilizado como rótulo para a sequência de entrada do modelo. As ferramentas de AutoML utilizadas, têm como requisito que os dados de entrada para realizar o treinamento do modelo possuam o mesmo tamanho. Entretanto, os registros dos usuários presentes nos dados do MovieLens possuem diferentes quantidades de filmes assistidos, para solucionar isso, os dados foram transformados em sequências de filmes de tamanhos iguais através do uso da implementação de *padding sequences* do *Keras*, que realiza um corte de um tamanho fixo da sequência de entrada, o tamanho do corte escolhido foi de 50 filmes para todos os usuários.

3.3 FLAML

FLAML [4] é uma ferramenta recém lançada e desenvolvida pela equipe da Microsoft Research que propõe a idéia de AutoML. A ferramenta contém algoritmos de última geração para otimização do espaço de busca de melhores hiperparâmetros, promovendo menor custo computacional e melhor desempenho para descoberta do modelo de aprendizado. Os métodos utilizados pela equipe da Microsoft no desenvolvimento do FLAML foram os *Cost-Frugal Optimization* (CFO) e as *BlendSearch*.

O CFO é um método que realiza um processo de pesquisa de hiperparâmetros de forma consciente em relação aos custos, ou seja, o processo de pesquisa começa a partir de valores iniciais de baixo custo e gradualmente alcança uma região de custo mais elevado enquanto otimiza o modelo objetivo através dos valores de precisão. Por sua vez, as *BlendSearch* é uma extensão do CFO que inicia a pesquisa de hiperparâmetros a partir de um ponto inicial de baixo custo como entrada. No entanto, ao contrário do CFO, o *BlendSearch* não espera que os valores de hiperparâmetros convirjam totalmente antes de tentar novos pontos de partida.

Por esse motivo, o CFO e o *BlendSearch* conseguem informações estruturais úteis sobre os hiperparâmetros no espaço de custo, sendo capazes de reduzir o custo gasto ao longo do caminho, sem afetar a convergência para a solução ideal.

3.4 TPOT

O TPOT [14] é uma ferramenta de AutoML implementada na linguagem Python que otimiza *pipelines* de aprendizado de máquina usando programação genética. O TPOT funciona usando uma estrutura baseada em árvore para representar um *pipeline* de modelo para um problema de modelagem preditiva. O TPOT também faz uso da biblioteca de aprendizado de máquina *scikit-learn* [11] para transformações de dados e algoritmos de aprendizado de máquina.

A ideia proposta pelo TPOT para otimizar a escolha do modelo e dos hiperparâmetros, é encontrar a melhor estrutura em árvore através da programação genética, onde é realizada uma otimização global estocástica. Algoritmos genéticos são inspirados no processo de seleção natural, o seu uso na área inteligência artificial é para prover soluções para problemas de otimização e busca. Os algoritmos genéticos geralmente possuem três propriedades: a seleção, cruzamento e mutação. A seleção é uma etapa onde uma população de possíveis soluções é avaliada em várias iterações através de uma função de ajuste. No cruzamento são selecionados os candidatos mais aptos, onde é realizado um cruzamento para criar uma nova população. Por fim, na mutação, o resultado do cruzamento é escolhido e transformado com alguma aleatoriedade. Tal procedimento é repetido até obter a solução mais adequada.

3.5 AutoSklean

O Auto-Sklearn [8] também é uma ferramenta de automação que fornece um *pipeline* contendo várias etapas de processamento dos dados, cobrindo as etapas de transformação de dados, seleção do melhor modelo que se adeque aos dados e meios para otimização de hiperparâmetros. O Auto-Sklearn é composto principalmente por modelos implementados pelo próprio *scikit-learn*, sendo 15 classificadores e 14 métodos de pré-processamento.

Diferente do modelo TPOT que utiliza algoritmos genéticos para identificar o melhor modelo e hiperparâmetros que se adequem aos dados de treinamento do modelo, esta ferramenta utiliza uma otimização bayesiana, sendo formada por três etapas importantes: meta-aprendizagem, otimização bayesiana e seleção de conjuntos.

Na etapa de meta-aprendizagem, o objetivo é reduzir o espaço de busca, aprendendo com modelos que tiveram um bom desempenho em conjuntos de dados semelhantes. Durante a fase de otimização bayesiana, o espaço de busca criado na etapa de meta-aprendizado é utilizado para criar modelos bayesianos que tem como objetivo encontrar a melhor configuração para o *pipeline*. Por fim, a etapa de seleção de conjunto é criada utilizando os modelos mais bem avaliados e precisos encontrados na etapa de otimização bayesiana

3.6 Métricas

As métricas escolhidas para a avaliação dos resultados de cada ferramenta de AutoML foram separadas entre métricas de eficácia e de eficiência. As métricas de eficácia são a acurácia e o *Normalized Discounted Cumulative Gain* (NDCG). Como métricas de eficiência são observados o tempo de processamento, consumo de memória e consumo de CPU.

3.6.1 Acurácia.

A métrica de acurácia mede a taxa de assertividade do modelo, ou seja, a proporção entre o número de instâncias classificadas corretamente e o número total de instâncias.

Dado que estamos realizando classificações do tipo *multilabel*, a acurácia será calculada levando em consideração a razão entre as observações de verdadeiros positivos e verdadeiros negativos de cada classe individualmente para todas as observações, conforme observado na equação 1. Vale ressaltar que verdadeiro positivo (TP) ocorre quando temos uma classe positiva e o modelo prevê como positivo e verdadeiro negativo (FP) ocorre quando temos uma classe negativa e o modelo prevê como negativo. Para o nosso trabalho, os positivos são os próximos filmes que o usuário assistiu, já os negativos são os que ele não assistiu. A biblioteca do *scikit-learn* foi utilizada para obter a implementação dessa métrica.

$$Acurácia = \frac{tp_i + tn_i}{tp_i + fn_i + fp_i + tn_i} \tag{1}$$

3.6.2 NDCG.

O NDCG é uma métrica de qualidade de classificação que avalia o quão bem foram as recuperações de documentos, ou seja, no contexto desse trabalho, o NDCG irá avaliar em relação às recomendações dos filmes [1]. Para entender melhor como essa métrica funciona, é preciso entender primeiramente o *Cumulative Gain* (CG). O CG é uma métrica onde cada recomendação tem uma pontuação de relevância associada a ela, no nosso contexto, essa pontuação de relevância podem ser os valores de probabilidade para que um item seja recomendado. O ganho cumulativo é a soma de todas as pontuações de relevância em um conjunto de recomendações, conforme observado na equação 2.

$$CumulativeGain(CG) = \sum_{i=1}^{n} relevance_i \tag{2}$$

Para melhor exemplificar como é calculada essa métrica, suponhamos que *SetA* 3 seja o conjunto das pontuações para cada

recomendação, no nosso contexto, essa pontuação nada mais é do que as probabilidades em que cada filme pode ser recomendado. Sendo assim, *CG* é calculado como sendo o somatório de *SetA* 4.

$$SetA = [2, 3, 3, 1, 2] \tag{3}$$

$$CGA = 2 + 3 + 3 + 1 + 2 = 11 \tag{4}$$

Todavia, essa métrica possui problemas relacionados à ordem na qual os itens são recomendados, visto que, se os itens mais relevantes são recomendados em qualquer ordem, não há penalidade. Para resolver esse problema, a métrica de *Discounted Cumulative Gain* (DCG), surge como uma modificação da métrica original, onde o cálculo envolve o desconto da pontuação de relevância como sendo a razão entre o valor da pontuação de relevância e o log da posição correspondente, conforme observado na equação 5.

$$DCG = \sum_{i=1}^{n} \frac{relevance_i}{log_2(i+1)} \tag{5}$$

Por fim, o NDCG é a normalização do valor calculado pelo DCG. Esse valor varia de 0 a 1 e, quanto mais próximo a 1, mais as recomendações estão sendo próximas ao real padrão de consumo do usuário. A normalização é feita de maneira simples, onde apenas precisamos dividir o valor de DCG obtido do conjunto de recomendações pelo valor ideal de DCG.

$$NDCG = \frac{DCG}{iDCG}$$

3.6.3 Eficiência.
Para executar as ferramentas foi utilizada uma CPU com todos os núcleos disponíveis e nenhuma ferramenta teve restrição de memória RAM. As ferramentas foram submetidas em um mesmo cenário, onde possuíam apenas 1 hora de *budget* para finalizar os experimentos. As medições de eficiência foram realizadas em uma máquina que possui uma configuração descrita na Tabela 1. Para capturar as observações do uso de memória e consumo de CPU utilizamos tanto o sistema de monitoramento de recursos do Ubuntu quanto uma biblioteca do Jupyter chamada de jupyter-source [6]. Para capturar o tempo de execução da ferramenta utilizamos a biblioteca Execute Time [3], também utilizada no ambiente *Jupyter*.

Tabela 1: Configurações da máquina

Componente	Descrição
Memória Ram	16gb(2x8Gb) 3000Mhz DDR4 CL16
CPU	Ryzen 3200G 4 Núcleos e 4 Threads

4 RESULTADOS

Esta seção apresenta os resultados obtidos em cada uma das ferramenta de AutoML discutidas anteriormente. Inicialmente é apresentado o cenário de teste, igualmente utilizado por todas as ferramentas, as métricas de eficácia obtidas nos dados de teste, bem como os modelos escolhidos por cada ferramenta de AutoML, além do consumo de recursos ao longo do tempo por cada ferramenta. Por fim, as métricas de eficácia e eficiência obtidas em cada ferramenta foram sumarizados na Tabela 5.

As ferramentas foram submetidas a um mesmo cenário de testes. O sistema operacional ubuntu na sua versão 20.04 foi utilizado com recursos de 16Gb de memória RAM 3000MHZ DDR4 e um processador Ryzen 3200G com 4 núcleos e 4 threads. É importante salientar que todos os núcleos foram usados pelas ferramentas. Além disso, para se ter um julgamento justo em relação a eficiência, as ferramentas foram configuradas para que tivessem o mesmo tempo de execução, portanto, um *time budget* de 3600 segundos (1 hora), foi utilizado. As informações de consumo de CPU e consumo de memória RAM presentes na Tabela 5, são as máximas apresentadas por cada uma das métrica.

4.1 FLAML

Para o FLAML, a eficácia obtida foi de 0 para métrica de acurácia, e 0.0012 para a métrica de NDCG. Os resultados foram extremamente baixos para a estratégia de predição de um próximo filme.

Por sua vez, para as métricas de eficiência, as Figuras 1 e 2 mostram o consumo de CPU e memória RAM respectivamente ao longo do tempo de execução da ferramenta. Os valores máximos observados foram: 100% do uso da CPU e 800mb de memória RAM.

Figura 1: Consumo de CPU (3600 Segundos).

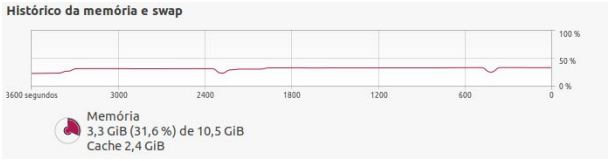

Figura 2: Memória RAM (3600 Segundos).

O modelo escolhido pelo FLAML foi o *Light Gradient Boosting Machine* (LGBM) Classifier, e os hiperparâmetros encontrados foram sumarizados na Tabela 2.

Tabela 2: LGBM (FLAML)

Hiperparâmetro	Valor
Taxa de aprendizado	0.7260
Número de estimadores	12
Número de folhas	4

4.2 TPOT

Para o TPOT, a eficácia obtida foi de 0 para métrica de acurácia, e 0.000565 para a métrica de NDCG. Os resultados foram extremamente baixos para a estratégia de predição de um próximo filme.

Por sua vez, para as métricas de eficiência, as Figuras 1 e 2 mostram o consumo de CPU e memória RAM respectivamente ao longo do tempo de execução da ferramenta. Os valores máximos observados foram: 100% do uso da CPU e 1.4Gb de memória RAM.

Figura 3: Consumo de CPU (3600 Segundos).

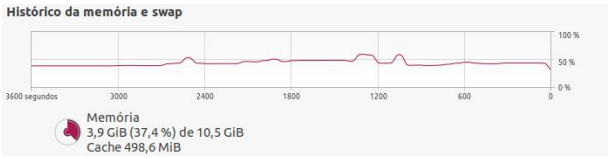

Figura 4: Memória RAM (3600 Segundos).

O modelo escolhido pelo TPOT foi o *Gaussian Naive Bayes* (GaussianNB) e os hiperparâmetros utilizados foram sumarizados na Tabela 3. Dado que o TPOT é baseado em programação genética, o modelo *Gaussian* foi escolhido utilizando: 4 gerações e um tamanho de 30 populações.

Tabela 3: GaussianNB (TPOT)

Hiperparâmetro	Valor
Alpha	10

4.3 AutoSklearn

Para o AutoSklearn, a eficácia obtida foi de 0.001312 para métrica de acurácia, e 0.000809 para a métrica de NDCG. Os resultados foram extremamente baixos para a estratégia de predição de um próximo filme.

Por sua vez, para as métricas de eficiência, as Figuras 5 e 6 mostram o consumo de CPU e memória RAM respectivamente ao longo do tempo de execução da ferramenta. Os valores máximos observados foram: 100% do uso da CPU e 4.2Gb de memória RAM.

Figura 5: Consumo de CPU (3600 Segundos).

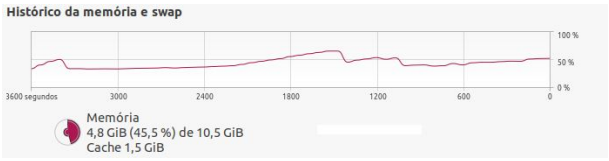

Figura 6: Memória RAM (3600 Segundos).

Tabela 4: Passive Aggressive (AutoSklearn)

Hiperparâmetro	Valor
C	0.00072
Loss	hinge

O modelo escolhido pelo FLAML foi o Passive Aggressive Classifier, e os hiperparâmetros utilizados foram sumarizados na Tabela 4.

Uma ferramenta muito interessante do AutoSklearn, é que é possível visualizar todos os modelos treinados e suas respectivas acurácias, em uma visualização 7.

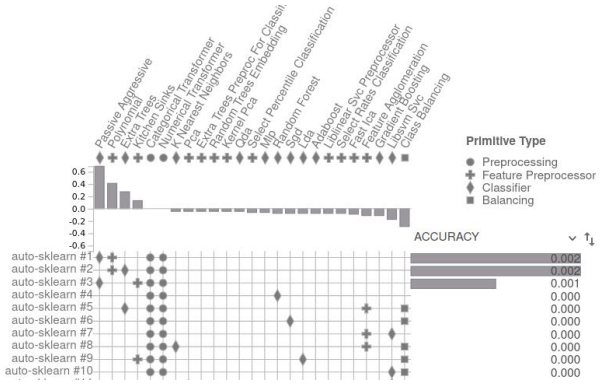

Figura 7: Modelos treinados no AutoSklearn.

	FLAML	TPOT	AutoSklearn
Acurácia	0	0	0.001312
NDCG	0.0012	0.000565	0.000809
Consumo de Memória	800Mb	1.4Gb	4.2Gb
Consumo de CPU	100%	100%	100%
Tempo de Exec.	3600s	3600s	3600s

Tabela 5: Resultados de cada ferramenta

4.4 Análise geral dos resultados

Dado que as ferramentas obtiveram resultados de eficácia insatisfatória, essa análise dos resultados propõe avaliar de maneira geral possíveis causas ou motivos para esse resultado.

A acurácia baixa deve-se ao fato de estarmos predizendo em relação a milhares de rótulos, ou seja, se torna muito mais difícil acertar

exatamente o próximo filme correspondente a uma determinada sequência vista pelo usuário.

A métrica de NDCG, que é muito utilizada para avaliar um conjunto de resultados de mecanismo de recuperação de documentos, no nosso caso, recomendação de filmes, é útil para entender se o modelo está buscando itens que são próximos aos itens que deveriam de fato serem recuperados, ou seja, indica se os documentos que foram recuperados e que são relevantes, aparecem a frente na lista de resultados do mecanismo de pesquisa. Sabendo disto, e que a métrica de NDCG para as ferramentas foram próximas a zero, é provável que o próximo filme predito pelas ferramentas não tiveram relevância para o usuário segundo a sequência anteriormente vista pelo mesmo.

Em relação às métricas de eficiência, o modelo que menos consumiu de memória foi FLAML, enquanto que, o modelo que utilizou mais memória foi o AutoSklearn. As ferramentas foram submetidas a uma hora de processamento e todas utilizaram CPU, todavia, não parece ser interessante investir nessa direção, visto que, ainda que melhore, não será um bom modelo com boas acurácias.

5 TRABALHOS FUTUROS

Apesar dos resultados terem mostrado baixos valores de eficácia, estudos com diferentes estratégias para recomendação de filmes podem ser conduzidos em trabalhos futuros, de forma a prover melhores índices de eficácia e viabilizar uma análise comparativa mais interessante. Para poder confirmar se os baixos resultados obtidos com as ferramentas de AutoML são causados pela estratégia de predição de um próximo item, trabalhos futuros poderão atuar no uso de modelos de recomendação estado da arte em conjunto com a estratégia de predição do próximo item, a fim de verificar se a suspeita de que a estratégia utilizada é a causa do baixa eficácia.

Uma outra possível alternativa é, ao invés de usar predição de um próximo item, realizar a predição de "top-k"itens, ou seja, de forma que o modelo seja avaliado não apenas em relação a sua capacidade de acertar exatamente o próximo item consumido por um usuário, mas de identificar se o item recomendado faz parte de um conjunto de k próximos itens consumidos pelo usuário. Uma outra estratégia é predizer a avaliação do usuário para um determinado filme. Isso se torna possível visto que os dados do MovieLens possuem informação dos usuários a respeito das suas avaliações dos filmes. Portanto, as ferramentas de AutoML podem predizer as avaliações dos filmes para cada usuário e, por fim, realizar a recomendação dos itens de maior avaliação.

6 AGRADECIMENTOS

Gostaria de agradecer ao meu orientador Cláudio Campelo, que estava aberto a discutir sobre pesquisas relacionadas ao uso de ferramentas de AutoML em sistemas de recomendação, além de mostrar como eu deveria implementar e escrever o presente trabalho. Agradeço também à Universidade Federal de Campina Grande, mais especificamente à Unidade Acadêmica de Sistemas e Computação, onde tive a oportunidade de aprender com excelentes professores e colegas. Gostaria de dedicar este trabalho aos meus pais, Aldo e Rosemary, que sempre me apoiaram e me ajudaram em todas minhas decisões. Aos meus amigos Tibério e Raquel, que me incentivaram durante todo o processo de pesquisa deste artigo. Agradeço também a cada um dos meus amigos e colegas, que sempre se dispuseram a ensinar e esclarecer minhas dúvidas, em especial a Ricardo Santos de Oliveira pela sua revisão neste trabalho e Ítalo de Pontes Oliveira, que me indicou o tema e me direcionou em pesquisas na área, além de me ajudar a revisar o trabalho.

7 CONCLUSÃO

Com esse trabalho pudemos comparar o uso das ferramentas de AutoML no contexto de sistemas de recomendação. Vimos que a eficácia das ferramentas não foram satisfatórias ou expressivas, alçando valores de acurácia e NDCG baixíssimos, provavelmente por causa da estratégia de predição de um próximo item, dado que temos milhões de rótulos a serem levados em consideração pelo nosso modelo no momento da predição. Para o desempenho de eficiência, os resultados são inconclusivos, visto que foram configuradas para utilizarem todos os núcleos da CPU e sem restrição de memória. Entretanto, o avanço das ferramentas de AutoML existentes, bem como o surgimento de novas ferramentas de automação exclusivas para uso em sistemas de recomendação, podem prover melhores resultados. Por fim, o estudo realizado neste trabalho pôde demonstrar o uso das ferramentas de AutoML no contexto de sistemas de recomendação, e refletir a respeito de possíveis adaptações que possam ser feitas para que se obtenham melhores modelos automatizados para sistemas de recomendação. O código e os dados utilizados podem ser encontrados no github [15].

REFERÊNCIAS

[1] Pranay Chandekar. 2020. Evaluate your Recommendation Engine using NDCG. (2020). https://towardsdatascience.com/evaluate-your-recommendation-engine-using-ndcg-759a851452d1
[2] S. Poirier E. LeDell. 2020. H2O AutoML: Scalable Automatic Machine Learning. (2020). https://www.automl.org/wp-content/uploads/2020/07/AutoML_2020_paper_61.pdf
[3] Execute-Time. 2015. Execute-Time. (2015). https://jupyter-contrib-nbextensions.readthedocs.io/en/latest/nbextensions/execute_time/readme.html
[4] FLAML. 2021. FLAML. (2021). https://github.com/microsoft/FLAML
[5] Bingbing Ni Jiancheng Yang, Rui Shi. 2021. MEDMNIST CLASSIFICATION DECATHLON: A LIGHTWEIGHT AUTOML BENCHMARK FOR MEDICAL IMAGE ANALYSIS. (2021). https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9434062
[6] Jupyter-Resource. 2019. Jupyter-Resource. (2019). https://github.com/jupyter-server/jupyter-resource-usage
[7] Holger H. Frank Hutter Kevin Leyton-Brown Lars Kotthoff, Chris Thornton. 2016. Auto-WEKA 2.0: Automatic model selection and hyperparameter optimization in WEKA. (2016). https://www.cs.ubc.ca/labs/beta/Projects/autoweka/papers/16-599.pdf
[8] Katharina Eggensperger Jost Tobias Springenberg Manuel Blum Frank Hutter Matthias Feurer, Aaron Klein. 2019. Efficient and Robust Automated Machine Learning. (2019). https://proceedings.neurips.cc/paper/2015/file/11d0e6287202fced83f79975ec59a3a6-Paper.pdf
[9] MovieLens. 2021. MovieLens. (2021). https://grouplens.org/datasets/movielens/
[10] Janek Thomas Sebastien Poirier Bernd Bischl Joaquin Vanschoren1 Pieter Gijsbers, Erin LeDell. 2019. An Open Source AutoML Benchmark. (2019). https://arxiv.org/pdf/1907.00909.pdf
[11] Scikit-Learn. 2007. Scikit-Learn. (2007). https://scikit-learn.org/
[12] Matthias Blohm Maximilien Kintz Sebastian Brandle, Marc Hanussek. 2021. Evaluation of Representation Models for Text Classification with AutoML Tools . (2021). https://arxiv.org/pdf/2106.12798.pdf
[13] Joeran Beel Srijan Gupta. 2020. Auto-CaseRec: Automatically Selecting and Optimizing Recommendation-Systems Algorithms. (2020). https://osf.io/4znmd
[14] TPOT. 2021. TPOT. (2021). http://proceedings.mlr.press/v64/olson_tpot_2016.pdf
[15] tsleolima. 2021. Github: AutoML Recommender System. (2021). https://github.com/tsleolima/automl-recommender-system
[16] Xiaowen Chu Xin He, Kaiyong Zhao. 2021. AutoML: A Survey of the State-of-the-Art. (2021). https://arxiv.org/pdf/1908.00709.pdf