Comparação de Modelos e Otimização de Hiperparâmetros em um Problema de Classificação

Guilherme M. da Silva, Gustavo H. Zenke, Henrique A. de Oliveira, Wesley L. de Araujo

Centro de Engenharia, Modelagem e Ciências Sociais Aplicadas (CECS) – Fundação Universidade Federal do ABC (UFABC)
Santo André – SP – Brazil

{guilherme.mafra, gustavo.zenke, henrique.oliveira, lima.wesley}@aluno.ufabc.edu.br

Abstract. *The objective of this paper is to study K Nearest Neighbor (KNN) and Decision-Tree classifier algorithms for a classification problem that uses a dataset with games that have been rated by the Entertainment Software Rating Board (ESRB) and compare which one presents higher accuracy. Besides working with different algorithms, we also explored the best hyperparameters for these algorithms when applied to this classification problem. After doing these explorations we could realize the importance of optimizing the hyperparameters and diversifying the algorithms during the exploration of a problem. After the tests, it was noticed that the Decision Tree algorithm obtained better accuracy than the KNN.*

Resumo. *O objetivo deste artigo é estudar algoritmos classificadores K Nearest Neighbor (KNN) e Decision-Tree para um problema de classificação que utiliza um dataset com jogos que foram classificados pela Entertainment Software Rating Board (ESRB) e comparar qual apresenta maior acurácia. Além de trabalharmos com diferentes algoritmos, fizemos também explorações em busca dos melhores hiperparâmetros para esses algoritmos quando aplicados a esse problema de classificação. Após fazer essas explorações pudemos perceber a importância de otimizar os hiperparâmetros e diversificar os algoritmos durante a exploração de um problema. Após os testes, foi notado que o algoritmo de Árvore de Decisão obteve melhor acurácia em relação ao KNN.*

1. Introdução

Nos últimos anos, conforme as tecnologias avançam, cada vez mais ouvimos falar no termo “machine learning”. Aprendizado de máquina (machine learning) é um termo geral utilizado para definir uma série de algoritmos sem ser necessário definir um modelo matemático específico e que extraem informação a partir de um dataset, este dataset pode ser imagens, números e tudo que essa tecnologia possa identificar. A partir de um conjunto de dados de treinamento, estes algoritmos buscam um padrão relacionando entradas e saídas, permitindo utilizar este padrão para realizar predições. Dependendo da forma como estes dados são fornecidos, os algoritmos são classificados em diferentes categorias sendo uma delas a aprendizagem supervisionada [1]. Os algoritmos de aprendizagem supervisionada relacionam uma saída com uma entrada com base em dados rotulados. Neste caso, o usuário alimenta ao algoritmo pares de entradas e saídas conhecidos, normalmente na forma de vetores. Para cada saída é atribuído um rótulo, que pode ser um valor numérico ou uma classe. O algoritmo determina uma forma de prever qual o rótulo de saída com base em uma entrada informada. Por exemplo, neste projeto utilizaremos um dataset [2] que contém 494

amostras para treino e 1895 amostras para teste de jogos e de conteúdos presentes neles representados através de features binárias (1 para presente e 0 para não presente), cada coluna trata de um conteúdo, exemplo "conteúdo sexual = 0". O algoritmo pode então ser treinado com estes dados, sendo capaz de prever a classificação. No final o algoritmo fará a classificação parental de um jogo com base nos conteúdos presentes nele, a classificação parental é o target. Ela possui quatro classes, E (Everyone), ET (Everyone plus 10), T (Teen) e M (Mature). Por fim, será feita a comparação de dois algoritmos classificadores, o K Nearest Neighbor (KNN) e Decision-Tree e analisaremos qual destes obteve melhor acurácia. Por fim, Para cada algoritmo de classificação também faremos uma exploração para otimização dos hiperparâmetros.

2. Metodologia e Processos

Foi utilizado um dataset que contém informações se o jogo possui características como: referência a álcool, violência, nudez, conteúdo sexual e drogas. Dependendo da característica que cada jogo possuir, ele entrará nas seguintes classificações:

- E (Everyone): livre;
- ET (Everyone 10+): para maiores de 10 anos;
- T (Teen): adolescentes;
- M (Mature): maiores de idade.

Inicialmente, para automatizar parte do projeto, criamos uma função de pipeline que é capaz de treinar os modelos usando k-folds e devolver o modelo com maior acurácia. Dentro do pipeline, foi realizado um K-Fold com cinco partes (20%) do dataset. Antes de ser realizada a divisão, os dados foram embaralhados aleatoriamente com uma seed bem definida para reprodutibilidade do experimento. Para cada fold fizemos um treinamento considerando o respectivo fold como a parte de teste do modelo. Foi acumulado em uma lista os valores de acurácia alcançados no modelo treinado com os dados de teste e treino. Em seguida foi devolvida uma lista com os valores de acurácia, a matriz de confusão e os índices de treino e teste que resultaram no modelo de maior acurácia.

Outro pipeline que foi criado para automatizar mais uma parte do projeto é o de KNN. Nesse pipeline foi realizada a interação de número de vizinhos para o modelo passado. A criação desse pipeline é possível devido a já estar definido qual o range de vizinhos que buscamos, de 1 a 15. Esse pipeline devolveu todas as informações que o pipeline_treino_teste retorna, porém, vai devolver como vários dicionários para cada tipo de elemento onde a chave do dicionário é a quantidade de vizinhos. Isso facilitou análises futuras, pois uniu em uma única estrutura todos os resultados.

2.1 KNN com distância de Manhattan

Após executar os pipelines citados no item 2, foi executado o algoritmo KNN com distância de Manhattan e com os resultados de acurácia obtidos foi feito um gráfico da variação da acurácia em relação ao número de vizinhos:

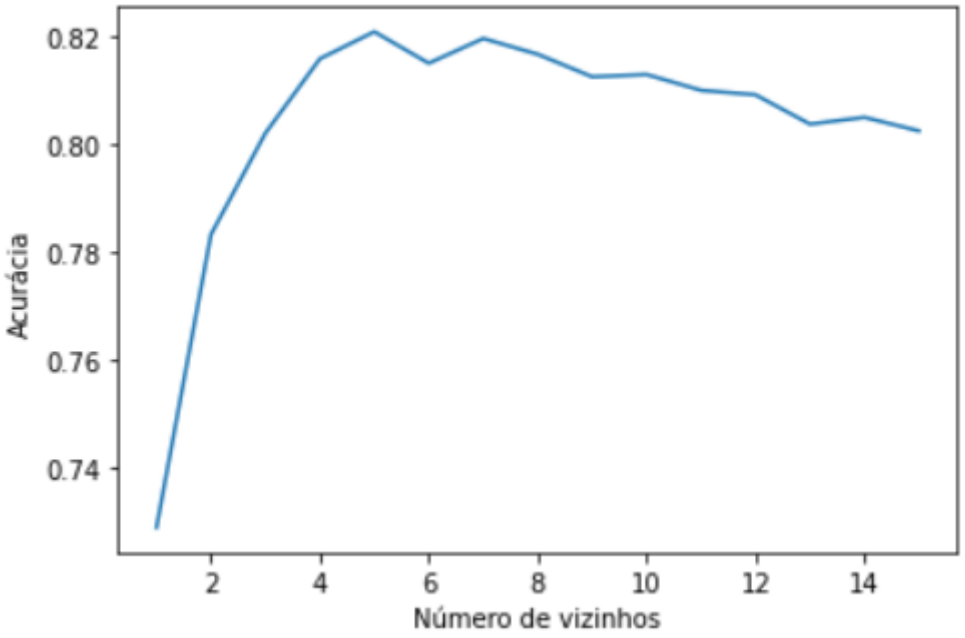

Figura 1. Gráfico da variação da acurácia em relação ao número de vizinhos para o algoritmo KNN com distância de Manhattan.

Pode-se observar que em relação ao número de vizinhos a acurácia média dos modelos começa muito baixa e cresce até atingir um pico com 5 vizinhos e depois começa uma queda estável. Com isso, foi feita uma matriz de confusão para observar o modelo com 5 vizinhos:

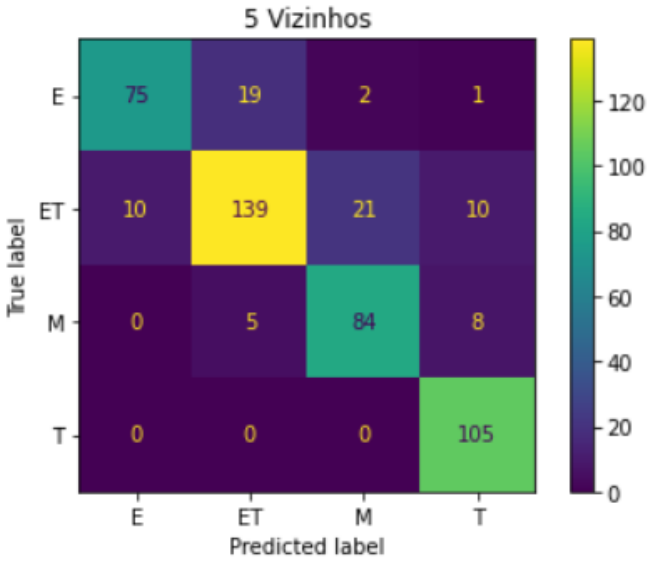

Figura 2. Matriz de confusão para 5 vizinhos para o algoritmo KNN com distância de Manhattan.

2.2 KNN com distância Euclidiana

Foi executado o pipeline de KNN citado no item 2, porém, alterando o parâmetro para o modelo de distância euclidiana. Pode-se identificar que mesmo utilizando o KNN com cálculo de distância euclidiana, o resultado obtido para as acurácias é exatamente o mesmo:

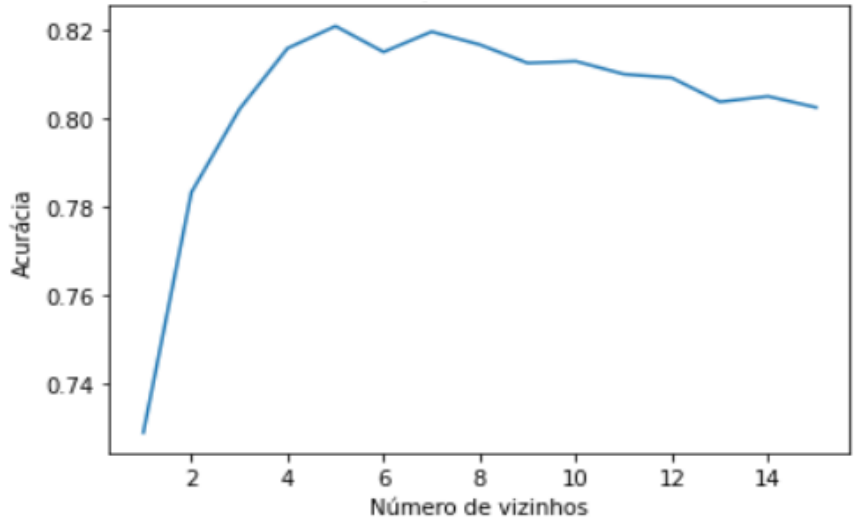

Figura 3. Gráfico da variação da acurácia em relação ao número de vizinhos com o algoritmo KNN com distância Euclidiana.

A mesma observação feita para o KNN com distância de Manhattan pode ser feita para o KNN com distância Euclidiana com relação ao número de vizinhos. A acurácia média dos modelos começa muito baixa e cresce até atingir um pico com 5 vizinhos e depois começa uma queda estável. Assim, a matriz de confusão para o caso de pico, será:

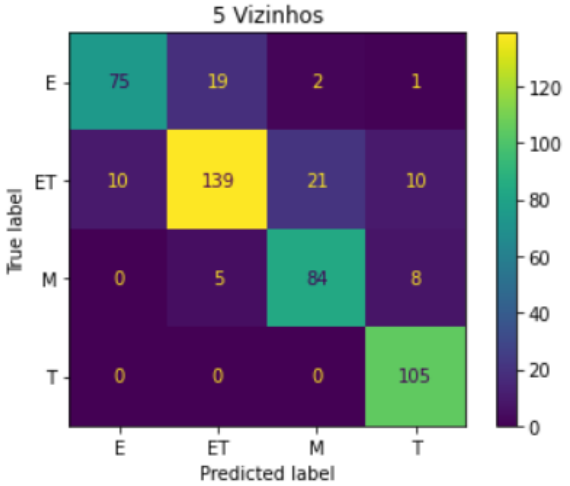

Figura 4. Matriz de confusão para 5 vizinhos para o algoritmo KNN com distância euclidiana

2.3 KNN com distância Chebyshev

De forma análoga aos itens 2.1 e 2.2, foi realizado o mesmo processo, porém, alterando o parâmetro do pipeline para o algoritmo KNN com distância de Chebyshev. Diferentemente dos dois KNNs utilizados nas etapas anteriores, pode-se identificar que o KNN com cálculo de distância de Chebyshev resulta em acurácias diferentes. As acurácias obtidas para o KNN com distância de Chebyshev foram menores para todos os números de vizinhos:

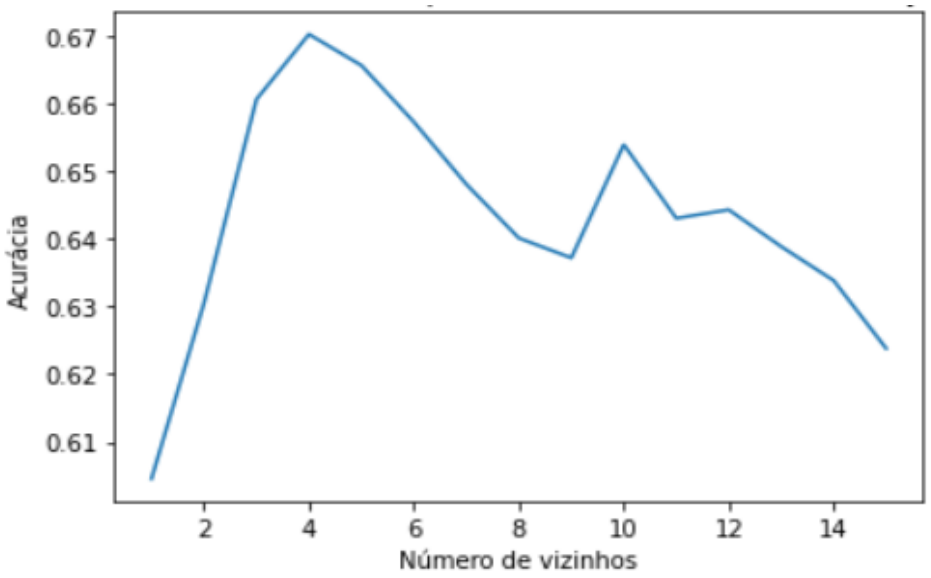

Figura 5. Gráfico da variação da acurácia em relação ao número de vizinhos com o algoritmo KNN com distância de Chebyshev.

Pode-se observar que em relação ao número de vizinhos a acurácia média dos modelos começa muito baixa, assim como nos dois modelos analisados nas etapas anteriores. Conforme o número de vizinhos cresce, a acurácia também aumenta consideravelmente, porém, conforme o KNN utilizou mais de 4 vizinhos, a acurácia caiu novamente. Com 10 vizinhos subiu um pouco, porém em seguida continuou diminuindo. Dessa forma, foi criada a matriz de confusão para o valor de pico de acurácia (4 vizinhos):

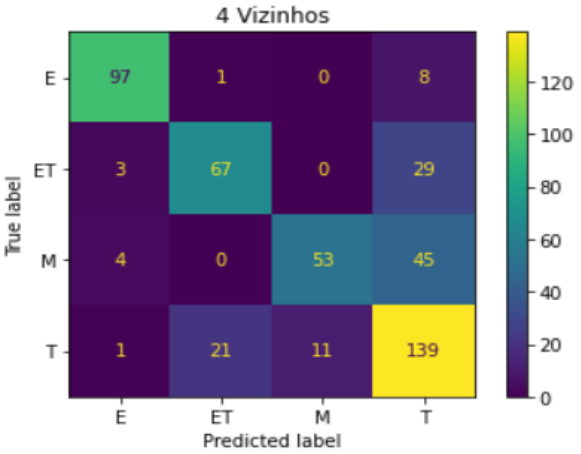

Figura 6. Matriz de confusão para 4 vizinhos para o algoritmo KNN com distância de Chebyshev.

2.4 Sobre hiperparâmetros

Antes de falarmos sobre os algoritmos de árvore de decisão, é necessário falar sobre hiperparâmetros. Hiperparâmetros são atributos que controlam o treinamento do modelo de machine learning: com eles podemos tornar o modelo mais preparado para resolver um determinado problema da vida real. Eles previnem o modelo de aprender apenas com os dados mostrados (overfitting e underfitting), tornando-o capaz de generalizar para outras situações possíveis. Os frameworks deixam os seus valores em defaults nas suas funções e por isso, geralmente, não são alterados [3]. Um exemplo de hiperparâmetro que já usamos nesse projeto é o número de vizinhos do KNN, a otimização desse parâmetro trata-se de tentarmos encontrar o valor de vizinhos N que leva o modelo a ter maior acurácia.

Obtidos os resultados de acurácia para o algoritmo KNN, foi feito um pipeline para o algoritmo de Árvore de Decisão. No KNN temos um hiperparâmetro numérico que temos interesse de variar, o número de vizinhos. Já na árvore de decisão também temos interesse em variar alguns hiperparâmetros numéricos, três exemplos seriam o número mínimo de amostras por folha, a profundidade máxima da árvore e o número mínimo de amostras para realizar um split em ramos.Assim, construímos um pipeline semelhante ao de KNN, mas que fez uma iteração nesses três parâmetros, o número mínimo de amostras por folha (1 a 30), a profundidade máxima da árvore (1 a 20) e o número mínimo de amostras para o split (2 a 30).

Veja que aqui temos um complicador que não enfrentamos nos testes de KNN, temos que variar mais de um parâmetro ao longo dos treinamentos e testes. Para fazer essa combinação de parâmetros usaremos a ferramenta GridSearchCV do scikit-learn. Esse é um método que realiza uma busca exaustiva com parâmetros passados em busca de otimizar uma métrica também passada como parâmetro. No nosso caso usaremos a acurácia, como nos outros modelos treinados até agora.

2.5 Árvore de Decisão com critério de Gini

Ao executar o pipeline da árvore de decisão com critério de Gini, foi retornado um objeto que possui os resultados de cada possibilidade de combinação entre hiperparâmetros na forma de um dicionário. Ao converter esse dicionário em um data frame para visualizar os resultados, observamos 18000 linhas (20X30X30), cada linha representando um modelo com uma configuração de hiperparâmetros e uma combinação diferente das colunas *"max depth"* (profundidade máxima da árvore), "*min samples leaf*" (número mínimo de amostras por folha) e "*min samples split"* (número mínimo de amostras para o split), os parâmetros que configuramos em nossa função de pipeline.

Além do resultado da acurácia, também temos outros valores referentes ao treino do modelo, como as colunas "*split acurácia"* que vão de 0 até 4, uma para cada fold (total de 5 folds). Além disso, também temos a média da acurácia para esses 5 treinos.

Para ter uma noção melhor de quais foram os melhores modelos, vamos ordenar o nosso dataframe pela coluna *rank* que já traz a posição do modelo da linha em relação a todos quando falamos da métrica usada. Depois mostramos as 10 primeiras combinações com melhor média de acurácia:

max depth	min samples leaf	min samples split	split0 acurácia	split1 acurácia	split2 acurácia	split3 acurácia	split4 acurácia	média	rank
20	1	5	839.248	862.213	860.125	841.336	853.862	851.357	1
19	1	3	843.424	860.125	855.950	845.511	845.511	850.104	2
20	1	10	843.424	860.125	868.476	839.248	839.248	850.104	2
20	1	3	845.511	860.125	855.950	843.424	843.424	849.687	4
20	1	13	835.073	860.125	866.388	839.248	845.511	849.269	5
18	1	13	837.161	855.950	870.564	841.336	841.336	849.269	5
20	1	11	839.248	860.125	866.388	839.248	841.336	849.269	5
18	1	7	835.073	858.038	868.476	841.336	843.424	849.269	5
19	1	5	837.161	858.038	862.213	837.161	849.687	848.852	9
20	1	15	843.424	858.038	866.388	830.898	843.424	848.434	10

Tabela 1. As 10 melhores variações de hiperparâmetros de melhor acurácia para a árvore de decisão com critério de Gini.

Na tabela 1 é possível ver que todos os 10 modelos com melhor acurácia possuem "min samples leaf" = 1 e "max depth" entre 18 e 20, além disso, não parece existir uma regra muito bem definida para o parâmetro "samples split". O que isso pode nos dizer é que se quisermos fazer mais explorações para otimização de hiperparâmetros já temos uma boa indicação de qual região alguns bons valores de alguns desses parâmetros estão.

2.5 Árvore de Decisão com critério de entropia

Repetimos o mesmo experimento do item 2.4 porém usando a Árvore de Decisão com critério de entropia para decisão das regras da árvore. Assim, os 10 modelos com melhor acurácia são:

max depth	min samples leaf	min samples split	split0 acurácia	split1 acurácia	split2 acurácia	split3 acurácia	split4 acurácia	média	rank	max depth
19	1	5	837161	870564	858038	841336	853862	852192	11986	1
20	1	5	835073	864301	855950	841336	851775	849687	10397	2
20	1	4	837161	864301	853862	839248	851775	849269	10003	3
20	1	3	832985	864301	858038	841336	849687	849269	11235	3
20	1	11	841336	862213	858038	837161	847599	849269	9558	3
19	1	3	841336	855950	853862	845511	847599	848852	5380	6
19	1	11	839248	862213	858038	839248	845511	848852	9576	6
19	1	4	832985	866388	851775	839248	853862	848852	11706	6
20	1	12	839248	860125	860125	839248	845511	848852	9485	6
20	1	2	832985	862213	855950	843424	847599	848434	10107	10

Tabela 2. As 10 melhores variações de hiperparâmetros de melhor acurácia para a árvore de decisão com critério de entropia.

Com o critério de entropia a região de parâmetros onde temos os modelos com melhor acurácia média é praticamente a mesma que a região apontada pelo critério de Gini. Na verdade, o melhor modelo apresentado com o critério de entropia (0,851357) teve desempenho inferior ao melhor modelo com critério de Gini (0,852192). Porém, o que pode-se dizer com esses resultados é que eles estão alinhados com os apresentados pelo critério de Gini.

3. Resultados

Para o KNN, com o uso de distância de Manhattan e distância Euclidiana, os maiores valores de acurácia foram iguais nos dois modelos, atingindo uma média de 0.82 com 5 vizinhos. Para valores de 1 a 5 vizinhos, à medida que o número de vizinhos crescia, a média da acurácia aumentava proporcionalmente. No entanto, a partir de 6 vizinhos, o valor médio da acurácia passou a cair.

Com a métrica de distância de Chebyshev, o valor médio de acurácia foi bem menor do que aqueles usando outras métricas, alcançando um valor máximo de, aproximadamente, 0.67 com 4 vizinhos. O mesmo comportamento foi observado em comparação aos modelos com as duas outras métricas, no qual a medida que aumentava

o número de vizinhos, a acurácia média aumentava, mas, ao atingir um determinado número de vizinhos esse mesmo valor de acurácia começou a diminuir quando o número de vizinhos aumentava.

No caso da Árvore de Decisão, que foram variados o número mínimo de amostras por folha, a profundidade máxima da árvore e o número mínimo de amostras para realizar um split, foram obtidos os seguintes resultados. As melhores acurácias possuíam um valor mínimo de amostras por folha igual a 1, uma profundidade máxima de 18 a 20, mas nenhuma conclusão chegou quanto ao número mínimo para realizar o split devido à grande diversidade de valores dessa métrica nos modelos com maior acurácia. Quanto a acurácia, tanto a construção da Árvore de Decisão por critério de Gini quanto por critério de Entropia apresentaram valores de acurácia próximos, por mais que os modelos por critério de Gini apresentaram valores um pouco melhores.

4. Conclusão

Visto os resultados obtidos, pode-se concluir que, os considerados melhores classificadores com KNN e Árvore de Decisão apresentaram maior valor de acurácia próximo à 0.85, sendo valores bem próximos uns dos outros. No entanto, alguns pontos devem ser ressaltados.

Com base nos resultados analisados, foi visto que após uma exploração pelos classificadores com melhor acurácia, considerando Árvores de Decisão e KNN, ambos os classificadores apresentaram valores de acurácia próximos à 0.85. Mas, acima de tudo, é importante observar como a mudança nos valores de hiperparâmetros para ambos os algoritmos de treinamento gerava modelos com uma alta variação de acurácias, por isso a otimização de hiperparâmetros é uma atividade importante de ser feita quando buscamos aplicar um modelo de machine learning a algum problema.

Muitas definições de Machine Learning Engineering dizem que a principal tarefa dessa área é criar métodos de comparar e testar o maior número de modelos possível de maneira ágil, para que assim o melhor modelo para um determinado problema possa ser encontrado e usado de maneira adequada. Acreditamos que, seguimos de maneira correta esse princípio ao buscarmos essa automatização de alguns processos no nosso projeto. E foi justamente a busca pela otimização de hiperparâmetros dos algoritmos que nos estimulou a criar pipelines e outras estruturas de treinamento e avaliação de modelos que trouxessem mais agilidade ao nosso processo já que o projeto baseava-se realmente nessa intensa criação de modelos e teste dos mesmos para comparação.

Como próximos passos pode-se definir um aumento no escopo de exploração do problema para otimizar ainda mais os modelos para o problema, poderíamos replicar a nossa metodologia considerando diferentes algoritmos de classificação como SVM ou Naive-Bayes, ou aumentado o range de variação de alguns hiperparâmetros, ou usando métodos de redução dimensão. As possibilidades de continuação do projeto são inúmeras, pois o que sempre se busca é a otimização dos resultados para que tenhamos o modelo com as melhores características possíveis dentro daquilo que é desejado.

5. Referências

[1] Prof. Éliton Fontana, **Introdução aos Algoritmos de Aprendizagem Supervisionada**, 2020. Disponível em: <https://fontana.paginas.ufsc.br/files/2018/03/apostila_ML_pt2.pdf>. Acesso em: 01 mai. 2022.

[2] Mohammed Alhamad, **Video Games Rating By 'ESRB'**, 2021. Disponível em:<https://www.kaggle.com/datasets/imohtn/video-games-rating-by-esrb?select=test_esrb.csv>. Acesso em: 29 abr. 2022.

[3] Samuel Hericlis, **Hiperparâmetros - por quê são importantes**. 20 jan. 2022. Disponível em: <https://blog.dsbrigade.com/hiperparametros-por-que-sao-importantes/>. Acesso em: 05 mai. 2022.