Revista Brasileira de Computação Aplicada, Novembro, 2019

DOI: 10.5335/rbca.v11i3.9230
Vol. 11, Nº 3, pp. 88–98
Homepage: seer.upf.br/index.php/rbca/index

ISSN 2176-6649

ARTIGO ORIGINAL

Aplicando aprendizado de máquina para identificação do gosto musical de um indivíduo

Applying machine learning to identify musical taste

Julio Cesar Lemos[1], Marcelo Carlos Benitez dos Santos[1], Plínio Roberto Souza Vilela[1], Marcelo Novaes de Rezende[2]

[1]Faculdade de Technologia, Universidade Estadual de Campinas, [2]Linked Education

*juliodelemos@gmail.com; uemgmarcelo@gmail.com; prvilela@unicamp.br; mrezende@linkededucation.com.br

Recebido: 26/03/2019. **Revisado:** 14/08/2019. **Aceito:** 13/09/2019.

Resumo

Descobrir o gosto musical de uma pessoa tem aplicação óbvia nos mecanismos de recomendação de provedores de serviços de música. Estamos interessados em uma aplicação menos óbvia, relacionada ao ambiente de trabalho de um desenvolvedor de software. Neste trabalho, comparamos dois classificadores usados na mineração de dados. O *Support Vector Machine* (SVM) e o *k-Nearest Neighbor* (k-NN) são avaliados como preditores do gosto musical de um usuário. Utilizamos um banco de dados de músicas classificadas previamente com um rótulo indicando se o usuário gosta ou não de cada música. O banco de dados inclui características das músicas; cada classificador usa as mesmas combinações de características no processo de aprendizado e depois classifica novas instâncias de músicas de acordo com o gosto previsto para o usuário. Este estudo inicial indicou o SVM como um melhor preditor do que o k-NN. Investigações futuras pretendem avaliar o usuário em um ambiente síncrono; nossa hipótese é que seja possível entender mais do que o cenário de gostar / não gostar e expandir para o que o usuário deseja ouvir em um determinado momento, capturando seu humor. Eventualmente, correlacionando o humor de um desenvolvedor de software com a propensão a falhas do código escrito.

Palavras-Chave: aprendizado de máquina; support vector machine; k-nearest neighbor.

Abstract

Discovering the musical taste of a person has an obvious application in recommendation mechanisms used by music service providers. We are interested in a less obvious application, related to the work environment of a software developer. In this work we compare two algorithms used in data mining as classifiers. The goal is to compare Support Vector Machine (SVM) and k-Nearest Neighbor (k-NN) as predictors of the musical taste of a user. We use a database of songs previously classified with a label indicating whether the user likes or dislikes each song. The database includes features of the song; each classifier uses the same combinations of features in the learning process and then classifies new instances of songs according to the user's predicted taste. This initial study indicated SVM as a better predictor than k-NN for this particular context. Future investigations intend to evaluate the user in a synchronous environment, our hypothesis is that it might be possible to understand more than the like / dislike scenario and expand to what the user wants to hear at a particular moment, capturing her mood. Eventually correlate the mood of a software developer to the fault proneness of the code she has written.

Keywords: machine learning; support vector machine; k-nearest neighbor.

1 Introdução

Este artigo trata do problema de identificação do gosto musical de um indivíduo, cuja aplicação pode variar entre a definição de um sistema de recomendação mais eficaz até o uso em Engenharia de Software, por exemplo, indicando músicas que possam reduzir o estresse, aumentar a produtividade ou reduzir a ocorrência de defeitos no software. As aplicações em si não serão tratadas no escopo deste trabalho, apenas a capacidade de inferir se um usuário vai ou não gostar de uma música com base em classificações feitas anteriormente será considerada; nas conclusões apresentamos ideias de trabalhos futuros que podem considerar aplicações em Engenharia de Software.

A motivação original para o estudo desse problema surgiu com a mudança nos mecanismos de disponibilização de músicas para o público, basicamente com a mudança da mídia de distribuição. Inicialmente a mídia era composta de discos, fitas, ou até chips de memória, todas com a característica de limitar a disponibilidade das músicas para um conjunto previamente selecionado pelo usuário. Com o surgimento do serviço de *streaming* de músicas esse paradigma se altera completamente, a partir daí a disponibilidade de músicas passa a ser praticamente ilimitada e o problema então se volta a como sugerir as músicas para o usuário desses serviços. Ainda é possível selecionar previamente as músicas preferidas e criar *play lists*, mas agora é possível também sugerir outras músicas que o usuário possa gostar.

Uma das diversas plataformas que oferecem recomendação é o Spotify e a forma de implementar esse serviço é através de técnicas como a Mineração de Dados e Aprendizado de Máquina (Su et al., 2010, Wang and Wang, 2014).

Técnicas de identificação do gosto musical podem também ter aplicações em outras áreas, como a Engenharia de Software, uma possível aplicação é buscar identificar o estado emocional de um programador através da análise das músicas que ele escolhe escutar em um determinado momento. Questões sobre se seria possível influenciar esse estado emocional através da seleção de músicas que o afetem positiva e gradativamente também precisam ser estudadas.

A Mineração de Dados é uma tecnologia que vem sendo utilizada de maneira bem sucedida na atualidade. Um dos fatores deste sucesso é o fato de dezenas, e muitas vezes centenas de milhões de reais serem gastos pelas companhias na coleta dos dados e, no entanto, nenhuma informação útil ser identificada (Larose, 2004). Em seu trabalho, Han et al. (2011) refere-se a essa situação como "rico em dados, pobre em informação". Além da iniciativa privada, o setor público e o terceiro setor (ONGs) também podem se beneficiar com a Mineração de Dados (Wang et al., 2008).

Uma técnica que vem recebendo crescente atenção da comunidade de Aprendizado de Máquina ou *Machine Learning* (ML) é a SVM-Máquinas de Vetores de Suporte (*Support Vector Machines*). Os resultados da aplicação dessa técnica são comparáveis e muitas vezes superiores aos obtidos por outros algoritmos de aprendizado, como as Redes Neurais Artificiais (RNAs). Exemplos de aplicações de sucesso podem ser encontrados em diversos domínios, como na categorização de textos, na análise de imagens e em Bioinformática (Lorena and de Carvalho, 2007). As SVMs são embasadas pela teoria de aprendizado estatístico, desenvolvida por Vapnik (1995). Essa teoria estabelece uma série de princípios que devem ser seguidos na obtenção de classificadores com boa generalização, definida como a sua capacidade de prever corretamente a classe de novos dados do mesmo domínio em que o aprendizado ocorreu (Lorena and de Carvalho, 2007).

Outro método de Machine Learning é o *k-nn* do inglês *"k-Nearest Neighbor"* (k-ésimo vizinho mais próximo). Tal método tem como objetivo classificar um novo elemento, atribuindo a ele o rótulo representado mais frequentemente dentre as *k* amostras mais próximas e utilizando um esquema de votação. O que determina essa "proximidade entre vizinhos" é uma distância calculada entre pontos em um espaço euclidiano (instâncias), cujas fórmulas mais comuns são a distância Euclidiana e a distância Manhattan (Han et al., 2011). A Distância Euclidiana (Eq. (1)) é definida como a soma da raiz quadrada da diferença entre x e y em suas respectivas dimensões. Já a Distância Manhattan (Eq. (2)) tem uma definição mais simples, sendo apenas a soma das diferenças entre x e y em cada dimensão [1].

$$d = \sqrt{x^2 + y^2} \tag{1}$$

$$d = | x_1 - x_2 | + | y_1 - y_2 | \tag{2}$$

A proposta desse trabalho é utilizar uma base real de um usuário em particular e realizar comparações entre dois classificadores, o SVM e o k-nn a fim de verificar qual deles oferece o modelo de ML com a melhor acurácia. A base de dados utilizada neste estudo é constituída de várias características (features) como o andamento rítmico das músicas, índice de energia de uma música, artista ou banda da música, entre outras, e um rótulo informando se o usuário gostou ou não gostou da música (like/unlike). O melhor algoritmo será aquele que consiga predizer o gosto musical (gosta/não gosta), a partir das características apresentadas, a exceção do rótulo, obviamente. Ao longo do trabalho serão apresentadas técnicas de pré-processamento e classificação, essenciais para um bom resultado. Alguns classificadores não trabalham muito bem com dados categóricos, para isso será necessário transformá-los em dados numéricos, essa técnica também será apresentada nesse trabalho. A base de dados foi obtida através da plataforma Spotify e os algoritmos utilizados foram implementados utilizando bibliotecas do Python.

[1] De maneira informal podemos imaginar a diferença entre essas duas distâncias como sendo o deslocamento de um carro e um helicóptero para ir de um ponto **A** a um ponto **B**, o carro percorre a distância Manhattan pois tem que seguir o caminho das ruas e o helicóptero a distância Euclidiana pois pode seguir em uma linha reta entre **A** e **B**.

2 Principais conceitos

Muitas pesquisas têm sido direcionadas para o desenvolvimento de técnicas com objetivo de extrair informações a partir de um grande volume de dados e transformar estas informações em conhecimento útil. Esta área é conhecida como *KDD* (*Knowledge Discovery in Databases*). KDD é o "Processo, não trivial, de extração de informações implícitas, previamente desconhecidas e potencialmente úteis, a partir dos dados armazenados em um banco de dados" (Fayyad et al., 1996).

No caso de uma aplicação em um ambiente comercial, esse processo pode, por exemplo, identificar padrões e descobrir informações relevantes que auxiliam um comerciante no processo de formação de preços, nas estratégias de marketing, no comportamento de clientes em relação às compras, entre outras coisas.

A fase de mineração de dados é onde realmente se extrai as informações através de algoritmos que executam uma determinada tarefa, consequentemente gerando um padrão entre itens em uma base de dados.

2.1 Técnicas de pré-processamento de dados

De acordo com Han et al. (2011), o pré-processamento pode melhorar a qualidade dos dados, melhorando assim a acurácia e eficiência dos processos de mineração subsequentes, sendo uma etapa fundamental para a mineração de dados.

Sua aplicação é necessária, pois as bases de dados, em geral, são muito grandes (gigabytes ou mais) e contém registros que comprometem a qualidade dos dados, como por exemplo, registros inconsistentes, falta de informação (registros faltantes), registros duplicados, outliers (valores discrepantes), assimetria, transformação entre outros.

As técnicas de pré-processamento são divididas em:

- Limpeza dos dados (*data cleaning*)
- Integração de dados (*data integration*)
- Transformação de dados (*data transformation*)
- Redução de dados (*data reduction*)

2.2 Classificação

A classificação reconhece modelos que melhor descrevem o grupo ao qual o item pertence por meio do exame dos itens já classificados e pela inferência de um conjunto de regras (Fayyad et al., 1996). Usualmente, os modelos de classificação de dados são obtidos com base em um processo de aprendizado supervisionado. Neste tipo de aprendizado o modelo é treinado a partir de uma base de dados com as classes conhecidas previamente (base de treinamento). Além da base de dados de treinamento, normalmente é utilizada uma segunda base de dados durante o processo de criação do modelo, sendo conhecida como base de teste. A base de treinamento é utilizada na criação do modelo, durante a fase de obtenção das regras de classificação, já a base de teste não é utilizada no modelo e sim como um parâmetro para a avaliação do treinamento sendo que o modelo é aplicado para realizar predições no conjunto de dados de teste, momento no qual é feita uma avaliação da qualidade das predições. Esta avaliação é realizada através da classificação de novas observações que não foram apresentadas ao modelo durante a fase de definição das regras.

Existem fatores que precisam ser considerados para a construção de modelos de classificação confiáveis. O primeiro fator está relacionado ao desbalanceamento das classes nas bases de treinamento e teste.

Deve ser observada a importância de manter a proporção entre as classes para os conjuntos de treinamento e de teste. O conjunto de treinamento, com uma quantidade muito maior de exemplos de uma classe em relação às demais, faz com que o aprendizado favoreça os exemplos da maior classe, atribuindo menor importância para a classe com menos exemplos. O conjunto de teste com uma distribuição de classes balanceada favorece uma análise estatística mais confiável dos resultados obtidos. Para solucionar o problema de desbalanceamento entre as classes algumas medidas, vistas a seguir, podem ser tomadas.

Partição pela menor classe ou redução de classes: dados da classe com maior número de exemplos podem ser eliminados aleatoriamente para construção do conjunto de treinamento com igual número de classes. Acréscimo de dados com ruídos: a técnica de redução de classes não pode ser aplicada quando o conjunto de dados final se tornar muito reduzido. Este problema pode ser solucionado através da inclusão de uma taxa de ruído nos dados originais da menor classe, gerando assim, novos padrões. Também podem ser replicados exemplos com o objetivo de aumento do número total de exemplos (Zaki and Jr, 2014).

Utilização da técnica conhecida por validação cruzada (*cross-validation*). Nesta técnica propõe-se a divisão do conjunto total de dados classificados em n bases menores; cada base resultante desta divisão conterá a mesma quantidade de dados de mesma classe. Por n vezes haverá um rodízio no papel desempenhado por cada uma das bases, ou seja, ora uma das bases será a base de dados de treino e ora será a base de dados de teste. Os erros de cada rodada são então somados, obtendo-se com isso o erro médio (Han et al., 2011).

2.3 Predição

Assim como na classificação de dados, o apelo dos modelos de predição é explicar uma ou várias variáveis de interesse em função de outras variáveis. A diferença em relação ao modelo de classificação é que as saídas do modelo são valores contínuos e não valores discretos (classes). Portanto, podemos considerar a classificação como um caso particular da predição onde o valor de saída do modelo é discretizado e pertence a um conjunto finito de classes

Existe uma infinidade de utilizações para os modelos de predição, podendo ser empregados para estimar, por exemplo: probabilidades, dimensões, valores financeiros e temperaturas.

A avaliação do modelo de predição normalmente é realizada com base no erro quadrático médio, ou *mean square error* (MSE), que consiste na diferença quadrática

média entre o resultado correto e o resultado previsto pelo modelo (Zaki and Jr, 2014).

2.4 Algoritmos de Machine Learning

De acordo com De Castro (2016), vários são os algoritmos existentes em *Machine Learning* utilizados para resolver problemas de classificação, dentre eles podemos citar:

a *Decision Tree* (árvore de decisão) – trata-se do número mínimo de perguntas que devem ser respondidas para avaliar a probabilidade de tomar uma decisão correta, na maioria das vezes. Como um método, permite-lhe abordar o problema de uma forma estruturada e sistemática para chegar a uma conclusão lógica;
b Classificação *Naïve Bayes* - Os classificadores *Naïve Bayes* são uma família de classificadores probabilísticos simples com base na aplicação do teorema de *Bayes* com forte independência entre as características;
c Regressão logística - A regressão logística é uma poderosa forma estatística de modelar um resultado binomial com uma ou mais variáveis explicativas. Ela mede a relação entre a variável dependente categórica e uma ou mais variáveis independentes, estimando as probabilidades usando uma função logística, que é a distribuição logística cumulativa;
d SVM (*Support Vector Machine*) – O SVM é um algoritmo binário da classificação. Dado um conjunto de pontos de 2 tipos em espaço N dimensional, SVM gera um hiperplano ($N-1$) dimensional para separar esses pontos em 2 grupos. Digamos que você tem alguns pontos de 2 tipos em um papel que são linearmente separáveis. SVM encontrará uma linha reta que separa esses pontos em 2 tipos e situados o mais longe possível de todos esses pontos;
e KNN – (*k-nearest neighbors* ou K-vizinhos mais próximos) – KNN é um algoritmo simples que prevê pontos de dados desconhecidos com os seus vizinhos mais próximos. O valor de k é um fator crítico quanto à precisão da predição. Ele determina o mais próximo ao calcular a distância usando funções básicas de distância como a Euclidiana.

Este trabalho faz uso dos dois últimos algoritmos citados, o SVM e o KNN.

3 Trabalhos correlatos

Identificar uma música específica para um usuário é classificado dentro do estudo de Sistemas de Recomendação Musical (MRS – *Music Recommender Systems*) o objetivo final dessa recomendação pode ser apenas para apresentar músicas diferentes para o usuário, mas também seguir uma estratégia de modificação do estado de espírito do ouvinte. De qualquer forma os mecanismos utilizados para se chegar na recomendação são os mesmos.

Pesquisas em sistemas de recomendação musical têm despertado grande interesse tanto no meio acadêmico quanto na indústria. Graças a serviços de streaming de música como o Spotify, Pandora, ou a Apple Music, entusiastas de música tem acesso a dezenas de milhões de peças musicais. Esse abundante universo musical, limita o usuário devido à sobrecarga de escolha. Nesse sentido os MRSs são frequentemente muito bem-vindos para sugerir músicas que se encaixam às preferências de seus usuários. No entanto, tais sistemas ainda estão longe de serem perfeitos e frequentemente produzem recomendações insatisfatórios. Isto é em parte devido ao fato de que os gostos dos usuários e necessidades musicais são altamente dependentes de uma multiplicidade de fatores, que não são considerados em profundidade suficiente em abordagens MRS atuais. Fatores intrínsecos e extrínsecos influenciam o gosto musical do ouvinte. Como fator intrínseco pode-se citar a personalidade e o estado emocional do ouvinte, já as condições meteorológicas, o ambiente social ou lugares de interesse, são exemplos de fatores extrínsecos.

A recomendação de música possui ainda algumas particularidades que se diferenciam de outros itens de recomendação, como filmes ou livros. Pode-se citar como itens de diferenciação a: (1) Duração de itens: A duração de uma música varia entre 3 a 5 minutos, enquanto que a de um filme possui duração típica de 90 minutos. Na recomendação de livros o tempo de consumo é bem maior; (2) Magnitude de itens: O tamanho dos catálogos de músicas é na faixa de dezenas de milhões de peças, enquanto que o catálogo de filmes tem de lidar com tamanhos bem menores (milhares até dezenas de milhares de filmes e séries). Portanto a escalabilidade é uma questão muito mais importante na recomendação de música do que na de filmes (Schedl et al., 2018).

3.1 Recomendação baseada em conteúdo

A abordagem baseada em conteúdo para recomendação tem suas raízes na comunidade de recuperação de informações (IR) e emprega muitas das mesmas técnicas. Quando uma página para um usuário foi escolhida, ela pode ser mostrada a eles e o feedback de algum tipo é eliciado. Se o usuário gostou de uma página, pesos para as palavras extraídas podem ser adicionados aos pesos para as palavras correspondentes no perfil do usuário. Esse processo é conhecido como feedback de relevância (Han et al., 2011).

3.2 Recomendação Colaborativa

A abordagem colaborativa à recomendação é muito diferente: em vez de recomendar itens por serem semelhantes aos itens que um usuário gostou no passado, recomendamos itens que outros usuários semelhantes tenham gostado. Em vez de calcular a similaridade dos itens, calculamos a similaridade dos usuários. Normalmente, para cada usuário, um conjunto de usuários de "vizinhos mais próximos" é encontrado, com os quais a classificação mais forte é a correlação mais forte. As

pontuações para itens não vistos são previstas com base em uma combinação das pontuações conhecidas dos vizinhos mais próximos (Balabanović and Shoham, 1997).

3.3 O problema da abordagem de Recomendação

A abordagem de Recomendação introduz alguns problemas próprios. Se um novo item aparecer no banco de dados, não há como recomendá-lo a um usuário até que mais informações sobre ele sejam obtidas através de outro usuário, seja classificando-o ou especificando com quais outros itens ele é semelhante. Assim, se o número de usuários for pequeno em relação ao volume de informações no sistema (porque há um banco de dados muito grande ou que muda rapidamente), então há o risco da cobertura de classificações se tornar muito escassa, diminuindo a coleta de itens recomendáveis (Balabanović and Shoham, 1997).

Um dos maiores problemas enfrentados em sistemas de recomendações, seja de música ou de propósito geral, é o problema de *Cold Start*. Isso acontece quando um novo usuário se registra no sistema ou um novo item é adicionado ao catálogo e o sistema não tem dados suficientes associados a esses itens e (ou) usuários. Nesse caso, o sistema não pode recomendar adequadamente itens existentes a um novo usuário ou recomendar um novo item para os usuários existentes (Schedl et al., 2018). Um segundo problema é simplesmente que, para um usuário cujos gostos são incomuns em comparação com o resto da população, não haverá outros usuários que sejam particularmente semelhantes, levando a recomendações ruins (Balabanović and Shoham, 1997).

3.4 *Discover Weekly* do Spotify

Um exemplo real do uso de Sistemas de Recomendação pode ser encontrado na plataforma do Spotify, que não usa realmente um único modelo de recomendação. Em vez disso, eles combinam algumas das melhores estratégias usadas por outros serviços para criar seu próprio mecanismo de descoberta excepcionalmente poderoso (Ciocca, 2018). Para criar o *Discover Weekly*, existem três tipos principais de modelos de recomendação que o Spotify emprega:

- Modelos de Filtragem Colaborativa, que analisam tanto o seu comportamento quanto os comportamentos dos outros.
- Modelos de Processamento de Linguagem Natural (PNL), que analisam o texto.
- Modelos de áudio, que analisam as faixas de áudio brutas em si.

4 Metodologia

Para a realização desse estudo, a tarefa de classificação foi vista sob o ponto de vista de aprendizagem supervisionada de máquina e os classificadores foram criados a partir de um processo de aprendizagem em uma base de dados real.

A base de dados estudada foi obtida através da plataforma do Spotify e contempla 15 características em 2017 observações de um único usuário, conforme a abordagem de Recomendação Baseada em Conteúdo, apresentada na seção anterior. O rótulo **target**, representado por 0 indica que o usuário não gostou da música e 1 indica que ele gostou. Os itens a seguir apresentam uma descrição das características presentes na base de dados (a Fig. 1 ilustra um fragmento dessa base):

Acusticness (tipo float): Índice de confiança se a música é acústica ou não. Valores entre 1 e 0 onde um valor 1 é uma música totalmente acústica e um valor 0 ou próximo diz que a música não tem elementos de música acústica.

Danceability (tipo float): Danceability descreve como uma faixa é adequada para a dança baseada em uma combinação de elementos musicais, incluindo tempo, estabilidade de ritmo, força de batida e regularidade geral. Um valor de 0.0 é menos dançante e 1.0 é mais dançante;

Duration_ms (tipo int): A duração da faixa em milissegundos;

Energy (tipo float): Energia é uma medida de 0,0 a 1,0 e representa uma medida perceptual de intensidade e atividade. Normalmente, faixas energéticas são rápidas e barulhentas;

Instrumentalness (tipo float): Prevê se uma faixa não contém vocais. Os sons "Ooh" e "aah" são tratados como instrumentais neste contexto. Faixas de rap ou palavra falada são claramente "vocais". Quanto mais próximo o valor de instrumental for de 1,0, maior a probabilidade de a faixa não conter conteúdo vocal;

Key (tipo int): O tom na qual a faixa se encontra. Os inteiros mapeiam para campos de notas usando a notação de classe de "pitch" padrão . Por exemplo, 0 = C, 1 = C♯ / D♭, 2 = D e assim por diante;

Liveness (tipo float): Detecta a presença de um público na gravação. Valores mais altos de atividade representam uma probabilidade maior de que a faixa foi executada ao vivo. Um valor acima de 0,8 fornece uma forte probabilidade de que a faixa esteja ativa;

Loudness (tipo float): O volume total de uma faixa em decibéis (dB). Os valores de sonoridade são calculados ao longo de toda a faixa e são úteis para comparar a intensidade relativa das faixas. Valores típicos variam entre -60 e 0 db;

Mode (tipo int): Mode indica a modalidade (maior ou menor) de uma faixa, o tipo de escala a partir da qual seu conteúdo melódico é derivado. Maior é representado por 1 e menor é 0;

Speechiness (tipo float): A fonação detecta a presença de palavras faladas em uma faixa. Quanto mais exclusivamente discursiva a gravação (por exemplo, talk show, livro de áudio, poesia), quanto mais próximo de 1,0 o valor do atributo. Valores acima de 0,66 descrevem faixas que provavelmente são feitas inteiramente de palavras faladas. Valores entre 0,33 e 0,66 descrevem faixas que podem conter música e fala, seja em seções ou em camadas, incluindo casos como música rap. Valores abaixo de 0,33 provavelmente representam músicas e outras faixas não relacionadas à fala;

acousticness	danceability	duration_ms	energy	instrumentalness	key	liveness	loudness	mode	speechiness	tempo	time_signature	valence	target	song_title
0.010200	0.833	204600	0.434	0.021900	2	0.1650	-8.795	1	0.4310	150.062	4.0	0.286	1	Mask Off
0.199000	0.743	326933	0.359	0.006110	1	0.1370	-10.401	1	0.0794	160.083	4.0	0.588	1	Redbone
0.034400	0.838	185707	0.412	0.000234	2	0.1590	-7.148	1	0.2890	75.044	4.0	0.173	1	Xanny Family
0.604000	0.494	199413	0.338	0.510000	5	0.0922	-15.236	1	0.0261	86.468	4.0	0.230	1	Master Of None
0.180000	0.678	392893	0.561	0.512000	5	0.4390	-11.648	0	0.0694	174.004	4.0	0.904	1	Parallel Lines
0.004790	0.804	251333	0.560	0.000000	8	0.1640	-6.682	1	0.1850	85.023	4.0	0.264	1	Sneakin'
0.014500	0.739	241400	0.472	0.000007	1	0.2070	-11.204	1	0.1560	80.030	4.0	0.308	1	Childs Play
0.020200	0.266	349667	0.348	0.664000	10	0.1600	-11.609	0	0.0371	144.154	4.0	0.393	1	Gyöngyhajú lány

Figura 1: Fragmento da base de dados

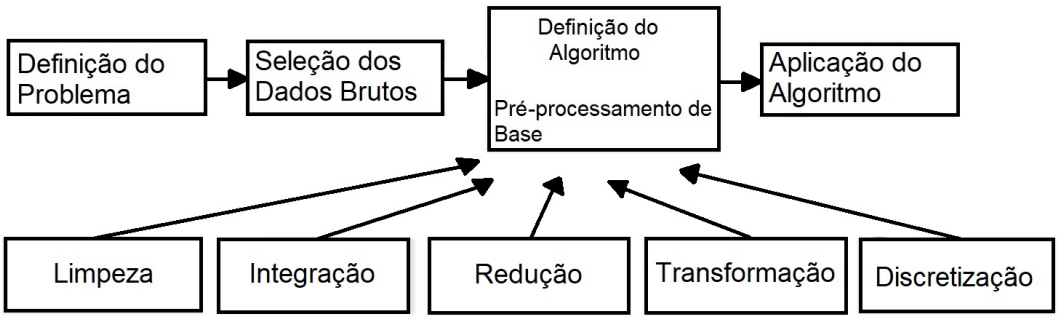

Figura 2: Etapas do Pré-processamento dos dados

Tempo (tipo float): O tempo total estimado de uma faixa em batidas por minuto (BPM). Na terminologia musical, tempo é a velocidade ou ritmo de uma determinada peça e deriva diretamente da duração média da batida;

Time_signature (tipo int): Um tempo estimado global de assinatura de uma faixa. A assinatura de tempo (metro) é uma convenção de notação para especificar quantas batidas estão em cada barra (ou medida);

Valence (tipo float): Uma medida de 0,0 a 1,0 descrevendo a positividade musical transmitida por uma faixa. Faixas com alta valência soam mais positivas (por exemplo, felizes, alegres, eufóricas), enquanto as faixas com baixa valência soam mais negativas (por exemplo, triste, deprimido, zangado).

Target (tipo int): valor 0 indica que o usuário não gostou da música, valor 1 indica que gostou.

4.1 Condução do experimento

Para a condução do experimento foi utilizado o método *holdout*, portanto a base de dados foi dividida em dados de treinamento e em dados de teste, na proporção de 70% e 30%, respectivamente. O conjunto de dados (treinamento e teste) passou pelas etapas de pré-processamento dos dados, conforme apresentado na seção III deste documento. A Fig. 2 ilustra as etapas do pré-processamento (De Castro, 2016).

Não foram identificados valores faltantes (*Missing Values*), referentes à etapa de "Limpeza", conforme mostrado na Tabela 1, onde o valor zero representa a quantidade de observações faltantes associada a cada característica. A característica "song_title" mostrou-se irrelevante à análise e foi eliminada, na etapa de "Redução" dos dados.

Dataset.isnull().sum()	
Unnamed: 0	0
Acousticness	0
Danceability	0
Duration-ms	0
Energy	0
Instrumentalness	0
Key	0
Liveness	0
Loudness	0
Mode	0
Speechiness	0
Tempo	0
Time_signature	0
Valence	0
Target	0
Song_title	0
Artist	0
Dtype: int64	0

Tabela 1: Itens Nulos do Conjunto de Dados

```
#objeto to tipo LabelEncoder
enc = LabelEncoder()

#criar uma variavel que ira receber os dados
    convertidos para inteiro:
inteiros = enc.fit_transform(dataset['artist'])

#adicionar uma coluna no dataset chamada
    artist_inteiros
dataset['artist_inteiros'] = inteiros

remove_features(['artist'])
```

Listagem 1: Transformação dos Dados Categóricos

O tipo da coluna 'artist' (omitida na Fig. 1) é categórica e como os modelos abordados neste estudo (SVM e Knn) não trabalham com dados categóricos, foram utilizadas técnicas para transformá-la em uma coluna numérica (etapa Transformação dos dados). O problema em transformar a coluna categórica em numérica é a possibilidade de enviesar o modelo, que pode "entender" os números transformados em inteiros como dados hierárquicos, ou seja, o modelo pode entender que o valor 4 é maior que o valor 3 e isso pode enviesá-lo, o que é um erro! Não existe na coluna 'artist' um artista que tenha algum tipo de prioridade sobre o outro. O código da Listagem 1 apresenta a transformação dos dados categóricos em inteiros.

Partindo dessa premissa, uma outra técnica foi avaliada. Tal técnica propõe converter valores categóricos em vetores binários. O resultado dessa técnica é uma matriz de presença onde as colunas são os dados categóricos e as linhas a presença - Listagem 2. Essa estratégia foi descartada pelo fato da coluna em questão possuir 1343 linhas únicas, que seriam transformadas em 1343 colunas, que se somadas às que já existiam causariam um problema de dimensionalidade, a Fig. 3 apresenta o resultado se a técnica fosse aplicada.

```
#Aplicando o One Hot Encoder

#Criando um objeto do tipo OneHotEncoder
ohe = OneHotEncoder()
dataset_array = dataset.values
inteiros = inteiros.reshape(len(inteiros),1)
novas_features = ohe.fit_transform(inteiros)
dataset_array = np.concatenate([dataset_array,
    novas_features.toarray()], axis=1)
```

Listagem 2: Criando a Matriz de Presença

Como se pode observar, o algoritmo foi fortemente penalizado pela alta dimensionalidade, confirmando ser a estratégia de transformação do tipo da coluna de categórica em inteiro a melhor alternativa. Sendo assim, prosseguimos utilizando a estratégia de transformação da coluna categórica em numérica (tipo inteiro), mesmo correndo o risco de enviezar o modelo

5 Resultados e Discussão

Após realizar a primeira execução do algoritmo de classificação (SVM, utilizando o *Radial basis funcion kernel*) exibido na Listagem 3, verificou-se uma acurácia de 53% (conforme mostrado na Tabela 2. Tal média pode ser confirmada através da Matriz Confusão apresentada na Tabela 3, bastando para isso aplicar a Eq. (3) – (Considere: PM = Precisão Média, VP = Verdadeiro Positivo, FP = Falso Positivo, VN = Verdadeiro Negativo, FN = Falso Negativo).

$$PM = \frac{\frac{VP}{(VP+FP)} + \frac{VP}{VN+FN}}{2} \quad (3)$$

Como forma de melhorar os resultados, aplicamos a estratégia de normalização, que segundo De Castro (2016) é um processo de transformação dos dados que objetiva torná-los mais apropriados à aplicação de algum algoritmo de mineração. Tal estratégia mostrou-se eficiente, melhorando a acurácia para 73% após implementação da normalização conforme exibida na Listagem 4. Os resultados podem ser confirmados nas Tabelas 4 e 5.

```
## Divisao treino-teste
df_feat = dataset
df_target = classes
from sklearn.model_selection import
    train_test_split

X_train, X_test, y_train, y_test =
    train_test_split(df_feat, np.ravel(df_target),
     test_size=0.30, random_state=101)

# Treinando o Support Vector Classifier
from sklearn.svm import SVC
model = SVC()
model.fit(X_train,y_train)
predictions = model.predict(X_test)
from sklearn.metrics import classification_report,
    confusion_matrix

param_grid = {'C': [0.1,1, 10, 100, 1000], 'gamma'
    : [1,0.1,0.01,0.001,0.0001], 'kernel': ['rbf'
    ]}

from sklearn.model_selection import GridSearchCV
grid = GridSearchCV(SVC(),param_grid,refit=True,
    verbose=3)
grid.fit(X_train,y_train)
grid.best_params_
grid.best_estimator_
grid_predictions = grid.predict(X_test)
print('\n')
print('-SVM SEM NORMALIZACAO-')
print('\n')
print('-MATRIZ CONFUSAO - TABELA III')
print('\n')
print(confusion_matrix(y_test,grid_predictions))
print('\n')
print('-RELATORIO DE CLASSSIFICACAO - TABELA II')
print('\n')
print(classification_report(y_test,
    grid_predictions))
```

Listagem 3: Aplicando SVM sem normalização

	0	1	2	3	4	5	6	7	8	9	...	1348	1349	1350	1351	1352	1353	1354	1355	1356	1357
0	0.0	0.0102	0.833	204600.0	0.434	0.021900	2.0	0.1650	-8.795	1.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	1.0	0.1990	0.743	326933.0	0.359	0.006110	1.0	0.1370	-10.401	1.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	2.0	0.0344	0.838	185707.0	0.412	0.000234	2.0	0.1590	-7.148	1.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	3.0	0.6040	0.494	199413.0	0.338	0.510000	5.0	0.0922	-15.236	1.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	4.0	0.1800	0.678	392893.0	0.561	0.512000	5.0	0.4390	-11.648	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Figura 3: Conjunto de dados com 2017 linhas x 1357 colunas

	Precision	Recall	F1-score	Support
0	0.55	0.27	0.36	303
1	0.52	0.78	0.62	303
avg/ total	0.53	0.52	0.49	606

Tabela 2: Resultado SVM sem normalização

```
## Divisao treino-teste
df_target = classes
from sklearn.model_selection import
    train_test_split
X_train, X_test, y_train, y_test =
    train_test_split(df_feat, np.ravel(df_target),
     test_size=0.30, random_state=101)

# Treinando o Support Vector Classifier
from sklearn.svm import SVC
model = SVC()
model.fit(X_train,y_train)
predictions = model.predict(X_test)
from sklearn.metrics import classification_report,
    confusion_matrix

param_grid = {'C': [0.1,1, 10, 100, 1000], 'gamma'
    : [1,0.1,0.01,0.001,0.0001], 'kernel': ['rbf'
    ]}
from sklearn.model_selection import GridSearchCV
grid = GridSearchCV(SVC(),param_grid,refit=True,
    verbose=3)
grid.fit(X_train,y_train)
grid.best_params_
grid.best_estimator_
grid_predictions = grid.predict(X_test)

print('\n')
print('-SVM COM NORMALIZACAO-')
print('\n')
print('-MATRIZ CONFUSAO - TABELA V')
print('\n')
print(confusion_matrix(y_test,grid_predictions))
print('\n')
print('-RELATORIO DE CLASSSIFICACAO - TABLE IV')
print('\n')
print(classification_report(y_test,
    grid_predictions))
```

Listagem 4: Usando dados normalizados

	0	1
0	81	222
1	67	236

Tabela 3: Matriz confusão – SVM Sem normalização

Com a base normalizada, seguiu-se o estudo para verificar qual classificador representaria o melhor desempenho, o Support Vector Machine (SVM) ou o k-nearest neighbors (k-nn), em classificar corretamente as preferências musicais (Gosta/Não Gosta) de um usuário em particular. Os modelos de classificação foram criados utilizando o conjunto de treinamento e avaliados através do conjunto de teste (técnica holdout). As medidas utilizadas para a avaliação do modelo foram as de: (1) "Precision", chamada de valor preditivo positivo, que é a fração de instâncias relevantes entre as instâncias recuperadas; (2) "Recall", conhecida como sensibilidade é a fração de instâncias relevantes que foram recuperadas sobre o total quantidade de instâncias relevantes; (3) "F1-score" ou F-score ou F-measure, medida da acurácia do teste.

O algoritmo SVM, apresentou melhor desempenho em relação ao K-nn. O modelo criado através do SVM apresentou 73%, 73% e 73% contra 65%, 65% e 64% do modelo K-nn, nas métricas de Precision, Recall e F1-score, respectivamente. Os resultados podem ser vistos nas Tabelas 6 e 7.

Na tentativa de melhorar o modelo gerado pelo algoritmo K-nn, foi empregado o método do cotovelo, tal método consiste em testar a variância dos dados em relação ao número de clusters a fim de encontrar o melhor valor de k conforme mostrado na Fig. 4.

	Precision	Recall	F1-score	Support
0	0.71	0.77	0.74	303
1	0.75	0.69	0.72	303
avg/ total	0.73	0.73	0.73	606

Tabela 4: Resultado SVM com normalização

	0	1
0	233	70
1	95	208

Tabela 5: Matriz confusão – SVM Com normalização

		Precision	Recall	F1-score	Support
KNN	0	0.63	0.70	0.66	303
	1	0.66	0.59	0.63	303
	avg	0.65	0.65	0.64	606
		Precision	Recall	F1-score	Support
SVM	0	0.71	0.77	0.74	303
	1	0.75	0.69	0.72	303
	avg	0.73	0.73	0.73	606

Tabela 6: Resultado K-NN Versus SVM

5.1 Aplicações em Engenharia de Software

Nesta seção listamos alguns trabalhos relacionados à aplicação desse tipo de estudo à Engenharia de Software como base para trabalhos futuros nessa área.

Em um trabalho mais voltado para o desenvolvimento de software Kuutila et al. (2018), conduziu um experimento baseado em um questionário diário para avaliar o bem-estar dos desenvolvedores com perguntas sobre raiva, interrupções, estresse, insônia, desenvolvimento de software ineficiente e outras variáveis referentes à autonomia e independência. Esse questionário foi aplicado a 8 desenvolvedores por 8 meses de trabalho. Os resultados mostram que foram encontradas correlações positivas entre todas as variáveis. A associação mais forte foi entre interrupções e ineficiência no desenvolvimento de software (0,522). A segunda associação mais forte foi entre raiva e interrupções (0,513). Foram encontradas fortes associações entre demanda de trabalho e interrupções, interrupções e desenvolvimento de software ineficiente, raiva e estresse e estresse e insônia (Kuutila et al., 2018).

Um estudo empírico realizado em Graziotin D. (2013) mostra o impacto dos estados afetivos na performance dos desenvolvedores de software enquanto estão programando. O intuito desse trabalho foi o de verificar se existe correlação entre o estado afetivo do desenvolvedor e sua avaliação de produtividade. Analisar se emoções, humor e sentimentos tem impacto no trabalho, mais precisamente em atividades cognitivas dos desenvolvedores. Os resultados mostram que os estados afetivos dos desenvolvedores de software são positivamente correlacionados com sua produtividade auto avaliada.

Estudos realizados na Engenharia de Software, mais especificamente por Müller (2015) utilizam técnicas para identificar o estado emocional de um usuário, técnicas como analise de batimento cardíaco e rastreabilidade da íris, estas técnicas apresentam bons resultados e buscam confirmar o estado emocional frente ao desenvolvimento de software.

A literatura sugere que proporcionar a felicidade dos desenvolvedores, mais especificamente no ambiente de trabalho, pode diminuir a infelicidade dos mesmos proporcionando mais produtividade. A metodologia utilizada foi colher as informações através de um questionário com perguntas sobre quais são a distribuição da felicidade e Infelicidade dos desenvolvedores de software e quais foram as experiências que causaram infelicidade nos desenvolvedores de software enquanto desenvolvem. Estar preso na resolução do problema, foi o que mais causa infelicidade em um desenvolvedor de software (Graziotin et al., 2017b).

KNN	0	1
0	211	92
1	123	180
SVM	0	1
0	233	70
1	95	208

Tabela 7: matriz confusão - K-nn Versus SVM

```
knn = KNeighborsClassifier(n_neighbors=17)
knn.fit(X_train,y_train)
pred = knn.predict(X_test)
print('\n')
print('-COM K=17-')
print('\n')
print('-MATRIZ CONFUSAO')
print('\n')
print(confusion_matrix(y_test,pred))
print('\n')
print('-RELATORIO DE CLASSIFICACAO - TABELA VIII')
print('\n')
print(classification_report(y_test,pred))
print('\n')
print('-------')
```

Listagem 5: Atualizando o valor de k

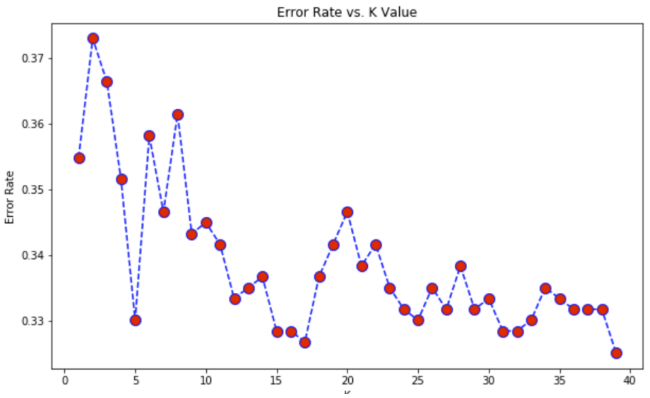

Figura 4: Melhor valor para k – (K-NN)

O resultado, após a atualização do valor de k=17, melhorou a precisão do modelo de 65% para 68% - Tabela 8.

Mesmo com o aumento de sua precisão, o modelo *k-nn* ainda se mostrou inferior às métricas apresentadas pelo modelo *SVM*.

	Precision	Recall	F1-score	Support
0	0.64	0.80	0.71	303
1	0.73	0.50	0.63	303
avg/ total	0.68	0.67	0.67	606

Tabela 8: Modelo k-nn após ajuste de k

6 Conclusão

Este trabalho apresentou a comparação entre dois classificadores sob o ponto de vista de Machine Learning (ML). O tipo de aprendizado utilizado foi o Supervisionado; onde durante o aprendizado, o modelo tem acesso às saídas reais. Os classificadores selecionados para o estudo foram o *k-Nearest Neighbor* (K-NN) e o *Support Vector Machine* (SVM). O objeto de estudo empregado foi uma base de dados extraída da plataforma Spotify com 15 características (*features*) e 2017 observações de um usuário em particular.

O objetivo desse experimento foi a verificação da melhor predição, dado dois algoritmos de classificação distintos, aprendendo sobre a preferência musical ("Gosto/Não Gosto", rótulo do conjunto de dados), de um usuário com base nas observações disponíveis. O classificador vencedor foi o SVM, apresentando uma acurácia de 73%. Outros estudos podem ser feitos para determinar o melhor conjunto de características dos dados que traria a melhor classificação. Esses estudos são possíveis trabalhos futuros relacionados diretamente à este estudo.

Ainda como conclusão deste trabalho identificamos a motivação de aplicar esse tipo de análise à Engenharia de Software. A identificação do gosto musical de um indivíduo pode ser ampliada para identificar o seu estado emocional em um determinado momento. Mesmo gostando de uma música podemos decidir não ouví-la por ela não estar adequada a como estamos nos sentindo naquela hora. Saber o estado emocional de um indivíduo pode ser particularmente útil na Engenharia de Software. Por exemplo, como o estado emocional de um programador afeta a densidade de defeitos do código escrito por ele. Pesquisas recentes na área de psicologia investigam a relação entre o afeto e a mudança de emprego (Graziotin et al., 2017a), investigações mais aprofundadas na área de Engenharia de Software ainda são necessárias.

Pretendemos avançar em dois estudos relacionados, um para analisar se é possível utilizar a música que o indivíduo escolhe ouvir como um fator identificador de seu estado emocional. E também se é possível influenciar o estado emocional de um indivíduo através da seleção ordenada de um conjunto de músicas que ele deve ouvir. Esses estudos servirão de base para outros estudos relacionando o estado emocional de programadores com a densidade de defeitos gerada no código produzido por eles.

Referências

Balabanović, M. and Shoham, Y. (1997). Fab: Content-based, collaborative recommendation, *Commun. ACM* **40**(3): 66–72. http://doi.acm.org/10.1145/245108.245124.

Ciocca, S. (2018). How does spotify know you so well?, Online. Disponível em https://bit.ly/2H954zz.

De Castro, L. (2016). *Introduçao A Mineraçao De Dados - Conceitos: BASICOS, ALGORITMOS E APLICAÇOES*, SARAIVA EDITORA. Disponível em https://books.google.com.br/books?id=7HxSvgAACAAJ.

Fayyad, U., Piatetsky-shapiro, G. and Smyth, P. (1996). From data mining to knowledge discovery in databases, *AI Magazine* **17**: 37–54.

Graziotin, D., Fagerholm, F., Wang, X. and Abrahamsson, P. (2017a). Consequences of unhappiness while developing software, *Proceedings of the 2nd International Workshop on Emotion Awareness in Software Engineering*, SEmotion '17, IEEE Press, Piscataway, NJ, USA, pp. 42–47. https://doi.org/10.1109/SEmotion.2017...5.

Graziotin, D., Fagerholm, F., Wang, X. and Abrahamsson, P. (2017b). On the unhappiness of software developers, *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*, EASE'17, ACM, New York, NY, USA, pp. 324–333. http://doi.acm.org/10.1145/3084226.3084242.

Graziotin D., Wang X., A. P. (2013). Are happy developers more productive?, *Heidrich J., Oivo M., Jedlitschka A., Baldassarre M.T. (eds) Product-Focused Software Process Improvement*, PROFES 2013, Springer, Heidelberg, Berlin, pp. 42–47. https://doi.org/10.1007/978-3-642-39259-7_7.

Han, J., Kamber, M. and Pei, J. (2011). *Data Mining: Concepts and Techniques*, 3rd edn, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Kuutila, M., Mäntylä, M. V., Claes, M. and Elovainio, M. (2018). Daily questionnaire to assess self-reported well-being during a software development project, *Proceedings of the 3rd International Workshop on Emotion Awareness in Software Engineering*, SEmotion '18, ACM, New York, NY, USA, pp. 39–43. http://doi.acm.org/10.1145/3194932.3194942.

Larose, D. T. (2004). *Discovering Knowledge in Data: An Introduction to Data Mining*, Wiley-Interscience, New York, NY, USA.

Lorena, A. C. and de Carvalho, A. C. P. L. F. (2007). Uma introdução às support vector machines, *Rev. Informática Teórica e Apl.* **14**(2): 43–67.

Müller, S. C. (2015). Measuring software developers' perceived difficulty with biometric sensors, *Proceedings of the 37th International Conference on Software Engineering - Volume 2*, ICSE '15, IEEE Press, Piscataway, NJ, USA, pp. 887–890. http://dl.acm.org/citation.cfm?id=2819009.2819206.

Schedl, M., Zamani, H., Chen, C.-W., Deldjoo, Y. and Elahi, M. (2018). Current challenges and visions in

music recommender systems research, *International Journal of Multimedia Information Retrieval* **7**. http://dx.doi.org/10.1007/s13735-018-0154-2.

Su, J.-H., Yeh, H.-H., Yu, P. S. and Tseng, V. S. (2010). Music recommendation using content and context information mining, *IEEE Intelligent Systems* **25**(1): 16–26. http://dx.doi.org/10.1109/MIS.2010.23.

Vapnik, V. N. (1995). *The nature of Statistical learning theory*, Vol. 22 of *22*, Springer-Verlag, New York.

Wang, J., Hu, X. and Zhu, D. (2008). *Data Mining in Public Administration*, Vol. 4 of *10*, 3 edn, IGI Global, The address. http://dx.doi.org/10.4018/978-1-59904-857-4.ch051.

Wang, X. and Wang, Y. (2014). Improving content-based and hybrid music recommendation using deep learning, *Proceedings of the 22Nd ACM International Conference on Multimedia*, MM '14, ACM, New York, NY, USA, pp. 627–636. http://doi.acm.org/10.1145/2647868.2654940.

Zaki, M. J. and Jr, W. M. (2014). *Data Mining and Analysis: Fundamental Concepts and Algorithms*, Cambridge University Press, New York, NY, USA.