Product Recommendation Using Classification Algorithms

Dalton L. Vargas[1], Jones Granatyr[1], Jeferson Knop[1], Cleber de Almeida[1]

[1]Universidade do Contestado (UNC)

R. Joaquim Nabuco, 314 - Cidade Nova, Porto União - SC, 89400-000

daltonluizvargas@hotmail.com, jones@unc.br, jefersonknop@gmail.com, cleber@unc.br

Abstract. *Recommender systems have become an extremely attractive topic, mainly because of the growth of the Internet, the use of services on mobile devices and also the growth of e-commerce. Therefore, it is necessary to filter, prioritize and provide relevant information in order to effectively address the problem of information overload. Based on that, the aim of this work is to present scientific contributions to the Artificial Intelligence field in order to demonstrate how to recomend products for users that do not have specific technical knowledge about the products, using classification techniques. Our approach is evaluated using the recommendation accuracy obtained by the application of machine learning algorithms applied to training datasets.*

Resumo. *Com a expansão da internet pelo mundo, o início do consumo em dispositivos móveis e a propagação do e-commerce, sistemas de recomendação e mineração de dados, tornaram-se um tema extremamente atrativo. Portanto, é necessário filtrar, priorizar e fornecer informações relevantes, a fim de enfrentar com eficácia o problema da sobrecarga de informações. Assim, o objetivo deste trabalho é trazer contribuições científicas para a área de Inteligência Artificial, no sentido de demonstrar como recomendar produtos, para usuários que não possuem conhecimento técnico específico, através de técnicas de classificação. Nossa proposta é avaliada pela precisão das recomendações obtidas através dos algoritmos de classificação aplicados as bases de dados de treinamento.*

1. Introdução

De modo geral, a classificação é a ação de atribuição a um objeto para uma categoria de acordo com as características do objeto. Na mineração de dados, a classificação refere-se à tarefa de analisar um conjunto de dados pré-classificados para aprender um modelo (ou uma função), que pode ser usado para classificar um dado invisível em uma das várias classes predefinidas. A mineração de modelos de classificação em bases de dados é um processo composto por duas fases: aprendizado e teste.

O interesse em pesquisar recomendação de produtos por meio de algoritmos de classificação surgiu de um problema que tem sido verificado no cenário de vendas de produtos, mais especificamente em um contexto de compra on-line de notebooks, a maioria dos usuários que compram no comércio eletrônico nem sempre possuem conhecimento sobre as informações técnicas do aparelho é melhor para sua necessidade, motivo pelo qual a mineração de dados pode ser útil, possibilitando a coleta de dados básicos da utilização do notebook e inferir o melhor produto para as necessidades do usuário [Almeida et al., 2014].

Neste contexto, este trabalho tem o intuito de demonstrar a indivíduos que não tem muito conhecimento técnico sobre o produto que estão adquirindo, podem contar com o auxílio de uma ferramenta de inteligência artificial que vai indicar a mercadoria mais recomendada para usar. Para isso, foram utilizadas duas bases de dados históricas, a primeira base de dados foi disponibilizada pelo site *Kaggle*, a segunda base de dados foi desenvolvida baseando-se na primeira base de dados, porém esta com um nível de detalhamento maior. Ambas as bases contêm os principais modelos de notebooks e suas características técnicas de hardware. Estes dados foram utilizados como fonte de treinamento para a criação de cada modelo de classificação. A acurácia de cada modelo de classificação criado foi avaliada e os modelos que tiveram o maior percentual de acerto na tarefa de classificação de notebooks, foram utilizados na versão final do exemplo prático do sistema de recomendação.

As próximas seções abordam o processo para aplicação das técnicas de classificação de dados e recomendação de produtos. No capítulo 2 apresenta alguns trabalhos relacionados com o tema. O capítulo 3 apresenta os materiais e métodos utilizados no processo de categorização e seleção dos dados, treinamento e avaliação dos algoritmos, a criação dos modelos de classificação baseado na análise dos resultados obtidos em cada base de dados e por fim o desenvolvimento do exemplo prático de um sistema de recomendação que utiliza os modelos de classificação criados.

2. Trabalhos Relacionados

Esta seção tem o objetivo de apresentar alguns trabalhos relacionados da área de aprendizagem de máquina, em específico, a utilização de algoritmos de classificação para a recomendação de produtos.

No trabalho de Miyahara e Pazzani (2000), os autores exploram a combinação sistemas de recomendação que utilizam a filtragem colaborativa com a técnica de classificação Bayesiana. Ao usar a classificação Bayesiana foi possível evitar um problema típico na filtragem colaborativa, onde é criado um modelo global de similaridade entre usuários, em vez de modelos separados por classes de classificações (por exemplo, classificação positiva vs. classificação negativa). Uma vez que o modelo de classificação Bayesiana trata cada classe de classificações separadamente, capturando a previsão dos usuários com mais precisão (1998), relatam uma variedade de modificações na técnica de filtragem colaborativa típica acima mencionada e o uso de agrupamento Bayesiano e uma rede bayesiana.

Similarmente, Puntheeranurak e Pitakpaisarnsin (2013) utilizam em um sistema de recomendação a técnica de classificação Bayesiana em conjunto com o método de filtragem colaborativa, porém com o objetivo de aprimorar o sistema de recomendação reduzindo o tempo que usuários devem aguardar para receber a recomendação, através da classificação Bayesiana nas informações dos usuários e nas informações dos itens. Para isso, o algoritmo de aprendizagem de máquina Bayesiano foi aplicado em uma base de dados menor do que aquela utilizada no sistema de recomendação completo, ou seja, foi utilizado um modelo de dados menor somente para o treinamento e criação do modelo de classificação.

O trabalho de Zhang e Iyengar (2002), é comparado um sistema de recomendação que utiliza a classificação linear com o modelo de classificação por árvore de decisão, com métodos baseados em memória usando dados de vários

domínios. Os resultados experimentais demonstraram que modelos lineares são mais adequados para esta aplicação, pois superam, os métodos baseados em memória, na precisão e na relação entre requisitos computacionais.

Por sua vez, Jadhav e Channe (2016), apresentam o uso dos algoritmos de classificação por árvore de decisão C4.5 e C5.0, com a abordagem da recomendação colaborativa para recomendação de livros. O desempenho dos classificadores é determinado pela precisão e tempo de execução. Os resultados demonstraram que o algoritmo C5.0 teve mais precisão e menor o tempo de execução em todos os testes realizados em relação ao algoritmo C4.5.

O trabalho de Xia, Dong e Xong (2006), é proposto o método heurístico para melhorar o desempenho da precisão preditiva da classificação por Máquinas de Vetores de Suporte, corrigindo repentinamente os valores faltantes na matriz de usuário - item. Um método direto para preencher esses elementos vazios na matriz de usuário - item é utilizar um padrão de valores, zeros ou a média de avaliações dos usuários, por exemplo. No entanto, esse o método pode enganar a máquina de aprendizado porque um usuário não avaliou um item ou pode ser o caso de o usuário não estar interessado no item ou o caso em que o usuário está interessado, mas tem ainda não comprou o item. Com o método heurístico foi possível estimar os elementos faltantes na matriz de acordo com os resultados obtidos a partir da iteração anterior.

No trabalho de Amatrian, Jaimes, Oliver e Pujol (2011) sobre métodos de mineração de dados para sistemas de recomendação, onde são abordadas as técnicas de classificação, foi concluído que embora o modelo de classificação KNN seja a abordagem preferida entre os desenvolvedores de software, os modelos de classificação por Árvore de Decisão, Baseado a Regras, Redes Neurais Artificiais e Máquina de Vetor de Suporte podem ser aplicados em situações diferentes.

3. Materiais e Métodos

Com a finalidade de construir um exemplo prático de um sistema para recomendação de produtos, foram aplicadas técnicas de classificação sobre as duas bases de dados detalhadas que contém modelos de notebooks, objetivando extrair modelos de classificação. A primeira base de dados foi obtida do site *Kaggle*, esta base de dados possui 16 atributos nominais e 1 atributo do tipo numérico, com um total de 1304 registros. A segunda base de dados de notebooks foi criada baseando-se na primeira base de dados, sendo esta base muito mais detalhada, contendo 25 atributos nominais e no total de 170 registros. Cada modelo de notebook registrado nas duas bases de dados possui 1 categoria de uso.

Os modelos de classificação extraídos passarão por algumas avaliações de acurácia, para determinar quais modelos tiveram a melhor adaptação à base de dados e consequentemente maior percentual de acerto nos testes de classificação.

Para a realização dos testes, foi utilizada a linguagem de programação Java, juntamente com a biblioteca Weka, onde é possível acessar as funções de cada algoritmo de classificação da ferramenta WEKA. Cada base de dados foi testada 30 vezes para cada algoritmo utilizando a validação cruzada com 10 *folds*, sendo esta uma técnica de avaliação padrão, uma maneira sistemática de executar repetições percentuais, onde a base de dados de treinamento é dividida em 10 subconjuntos de dados, sendo que todos os subconjuntos de dados da base serão utilizados para

treinamento e teste dos algoritmos. Em cada um dos 30 testes a semente geradora variou de 1 e 30, assim constituindo em uma estiva média dos acertos e erros de cada base.

Foi utilizada a ferramenta Waikato Environment for Knowledge Analysis (WEKA), para a criação dos modelos de classificação, pois esta ferramenta já disponibiliza a implementação dos algoritmos utilizados nesta pesquisa afim de testar cada paradigma de aprendizagem de máquina: probabilístico (NaiveBayes), regras (OneR, JRip), árvore (J48, RandomForest), instâncias (IBK), redes neurais (MultilayerPerceptron) e máquinas de vetores de suporte (LibSVM). Também foi utilizada a linguagem de programação R, para realizar a análise estatística dos resultados. Por fim, foi utilizada a linguagem de programação Java para o desenvolvimento do sistema de recomendação.

A seguir, na próxima seção, serão apresentadas as atividades desenvolvidas na categorização e seleção dos dados, treinamento e avaliação dos algoritmos, criação dos modelos de classificação que foram utilizados na versão final do exemplo prático de um sistema desenvolvido na linguagem de programação Java para a recomendação de notebooks.

3.1. Categorização e Seleção dos Dados

Primeiramente, foi realizada a classificação de cada modelo de notebook registrado nas duas bases de dados, ou seja, a categorização de uso para cada modelo. Para isso buscou-se conhecimentos quanto à maneira para a categorização de uso de notebooks, sendo realizada uma pesquisa em websites que já possuem um sistema de pesquisa de produtos por categoria de uso, ou que sugerem uma categorização do produto de acordo com as características de hardware do mesmo. No total foram identificadas 6 categorias mais comumente usadas, sendo estas a categoria de uso *basic*, *input*, *advanced*, *gaming*, *premium* e *hybrid*.

Na categoria de uso *basic*, são notebooks utilizados para acesso à web, leitura de e-mails e digitação. Já a categoria de uso *input*, são notebooks utilizados para o dia a dia, como para estudar, trabalhar, navegar na internet e multimídia. A categoria de uso *advanced*, são notebooks voltados a profissionais de design e fotografia, programas de engenharia, AutoCAD, arquitetura, ou que precisem de processamento matemático mais rápido. Para a categoria de uso *gamer*, um notebook precisa de requisitos especiais para definir que seja efetivamente *gamer*, detalhes nos acabamentos mais robustos e com maior circulação de ar, além de costumar trazer o nome “Gamer” na descrição do produto, nesta categoria os jogos devem rodar com mais fluidez e qualidade gráfica, devido ao alto poder de processamento. Na categoria *premium*, são modelos de notebooks compactos, design diferenciado e telas com bordas mais finas ou alta resolução para uso com trabalhos tradicionais, corporativos, com tamanho e acabamento refinado. Por fim a categoria de uso *hybrid (2 em 1)*, são notebooks que contém telas touchscreen, que se abrem em 360º ou destacáveis do teclado, alguns modelos de notebooks das categorias de uso *basic* e *advanced* podem trazer esta função de *2 em 1*.

A Tabela 1 apresenta a descrição dos atributos em cada base de dados, sendo no total 17 atributos preditivos na Base 1 e 24 atributos preditivos na Base 2. Ambas as bases de dados possuem os mesmos atributos classe.

Tabela 1. Descrição dos atributos em cada base de dados.

ATRIBUTOS	Base 1	Base 2
Classe	*basic, input, advanced, gaming, premium, hybrid*	
Previsores	Company, Model, Serie_Edition, Cpu, Cpu_Serie, Cpu_Edition, Cpu_Frequency, Ram, Hdd, Hdd_SSD, Screen_Type, Screem_Resolution, Inches, Gpu_Company, Gpu_Serie_Edition, OpSys, Weight	Company, Model, Serie_Edition, Cpu, Cpu_Model, Frequency, Cores, Threads, Cache, So, So_Version, Ram, Hdd, Hdd_Rpm, SSD, Graphics_Company, Graphics_Prefix, Graphics_Generation_Suffixx, Graphics_Memory, Graphics_Type, Usb, Display, Batery, Weight

3.2. Treinamento e Avaliação dos algoritmos

Primeiramente cada base de dados foi exportada e convertida para o formato .arff, para que, posteriormente, os resultados dos testes realizados em Java pudessem ser comparados com mais facilidade com os resultados dos testes realizados com a ferramenta WEKA, assegurando uma maior confiabilidade nos resultados.

Após carregada a base de dados no WEKA, iniciou-se a fase de aprendizado dos algoritmos conforme exposto na Seção 1. No intuito de obter a melhor versão de cada modelo de classificação, ou seja, modelos de classificação com o maior percentual de precisão em relação a cada base de dados, foram feitas diversas configurações nos parâmetros modificáveis de cada algoritmo. Para os algoritmos NaiveBayes e OneR foi utilizada a parametrização padrão, no algoritmo MultilayerPerceptron o parâmetro *trainingTime* variou somente de 1 a 10 e o parâmetro *learningRate* variou somente entre 0.1 a 0.8, devido ao limite computacional da máquina utilizada para a realização dos testes. A parametrização utilizada em cada algoritmo é apresentada na Tabela 2.

Tabela 2. Parâmetros dos algoritmos.

ALGORITMO	Base 1	Base 2
NaiveBayes	*batchSize = 100, debug = False, displayModelInOldFormat = False, doNotCheckCapabilities = False, numDecimalPlaces = 2, useKernelEstimator = False e useSupervisedDiscretization = False*	
J48	*batchSize = 100, binarySplits = False, collapseTree = True, confidenceFactor = 0.25, debug = False, doNotCheckCapabilities = False, doNotMakeSplitPointActualValue = False, minNumObj = 2, numDecimalPlaces = 2, numFolds = 3, reducedErrorPruning = False, saveInstanceData = False, seed = 1, subtreeRaising = True, unpruned = False, useLaplace = False e useMDLcorrection = True*	
RandomForest	*bagSizePercent = 100, batchSize = 100, breakTiesRandomly = False, calcOutOfBag = False, computeAttributeImportance = False, debug = False, doNotCheckCapabilities = False, maxDepth = 0, numDecimalPlaces = 2, numExecutionSlots = 1, numFeatures = 0,* ***numIterations = 200****, outputOutOfBagComplexityStatistics = False, printClassifiers = False, seed = 1 e storeOutOfBagPredictions = False*	*bagSizePercent = 100, batchSize = 100, breakTiesRandomly = False, calcOutOfBag = False, computeAttributeImportance = False, debug = False, doNotCheckCapabilities = False, maxDepth = 0, numDecimalPlaces = 2, numExecutionSlots = 1, numFeatures = 0,* ***numIterations = 1000****, outputOutOfBagComplexityStatistics = False, printClassifiers = False, seed = 1 e storeOutOfBagPredictions = False*
OneR	*batchSize = 100, debug = False, doNotCheckCapabilities = False, minBucketSize = 6 e numDecimalPlaces = 2*	
Jrip	*batchSize = 100, checkErrorRate = True, debug = False, doNotCheckCapabilities = False, folds*	*batchSize = 100, checkErrorRate = True, debug = False, doNotCheckCapabilities = False, folds*

	= 3, minNo = 2.0, numDecimalPlaces = 2, optimizations = 2, seed = 1 e ***usePruning = True***	*= 3, minNo = 2.0, numDecimalPlaces = 2, optimizations = 2, seed = 1 e* ***usePruning = False***
IBK	***KNN = 1****, batchSize = 100, crossValidate = False, debug = False, distanceWeighting = No Distance weighting, doNotCheckCapabilities = False, meanSquared = False, nearestNeighbourSearchAlgorithm = LinearNNSearch, numDecimalPlaces = 2 e windowSize = 0*	***KNN = 3****, batchSize = 100, crossValidate = False, debug = False, distanceWeighting = No Distance weighting, doNotCheckCapabilities = False, meanSquared = False, nearestNeighbourSearchAlgorithm = LinearNNSearch, numDecimalPlaces = 2 e windowSize = 0*
LibSVM	*SVMType = C-SVC, batchSize = 100, cacheSize = 40, coef0 = 0,* ***cost = 5000****, debug = False, degree = 3, doNotCheckCapabilities = False, doNotRaplaceMissingValues = False, eps = 0.001, gamma = 0,* ***KernelType = Polynomial****, loss = 0.1, normalize = False, nu = 0.5, numDecimalPlaces = 2, probabilityEstimates = False, seed = 1 e shrinking = True*	*SVMType = C-SVC, batchSize = 100, cacheSize = 40, coef0 = 0,* ***cost = 100000****, debug = False, degree = 3, doNotCheckCapabilities = False, doNotRaplaceMissingValues = False, eps = 0.001, gamma = 0,* ***KernelType = Polynomial****, loss = 0.1, normalize = False, nu = 0.5, numDecimalPlaces = 2, probabilityEstimates = False, seed = 1 e shrinking = True*
MultilayerPerceptron	*GUI = False, autoBuild = True, batchSize = 100, debug = False, decay = False, doNotCheckCapabilities = False, hiddenLayers = a,* ***learningRate = 0.8****, momentum = 0.2, nominalToBinaryFilter = True, normalizeAttributes = True, normalizeNumericClass = True, numDecimalPlaces = 2, reset = True, seed = 0,* ***trainingTime = 2****, validationSetSize = 0 e validationThreshlod = 20*	*GUI = False, autoBuild = True, batchSize = 100, debug = False, decay = False, doNotCheckCapabilities = False, hiddenLayers = a,* ***learningRate = 0.2****, momentum = 0.2, nominalToBinaryFilter = True, normalizeAttributes = True, normalizeNumericClass = True, numDecimalPlaces = 2, reset = True, seed = 0,* ***trainingTime = 10****, validationSetSize = 0 e validationThreshlod = 20*

3.2.1. Resultados obtidos em cada base

Na Tabela 3 são apresentados os valores médios de precisão, resultantes em cada algoritmo e base de dados de notebooks. Pode-se notar que dentre todos os algoritmos testados, os valores da Base 1 são em média 25,6406% melhores em relação aos valores da Base 2. No que diz respeito a classificação, pode-se observar que nas duas bases de dados, os algoritmos NaiveBayes, RandomForest, IBK e LibSVM tiveram a melhor média de acertos nos testes de classificação.

Tabela 3. Resultados dos algoritmos em cada base de dados.

ALGORITMO	Base 1	Base 2
NaiveBayes	**85,8634**	**64,0784**
J48	83,4024	59,1765
RandomForest	**87,2039**	**63,6471**
OneR	66,4134	29,3333
Jrip	85,6485	52,8627
IBK	**86,6053**	**62,4902**
LibSVM	**90,7444**	**62,8627**
MultilayerPerceptron	62,0875	52,7059

Para determinar a existência ou não de diferença estatisticamente significativa entre os algoritmos testados em cada base de dados, foram utilizados os testes de Friedman e Nemenyi. A Figura 1 apresenta os resultados em relação a Base 1, na qual pode-se observar que, pelo ranqueamento, dentre os 8 algoritmos testados os algoritmos RandomForest, LibSVM e IBK tiveram os melhores valores de precisão.

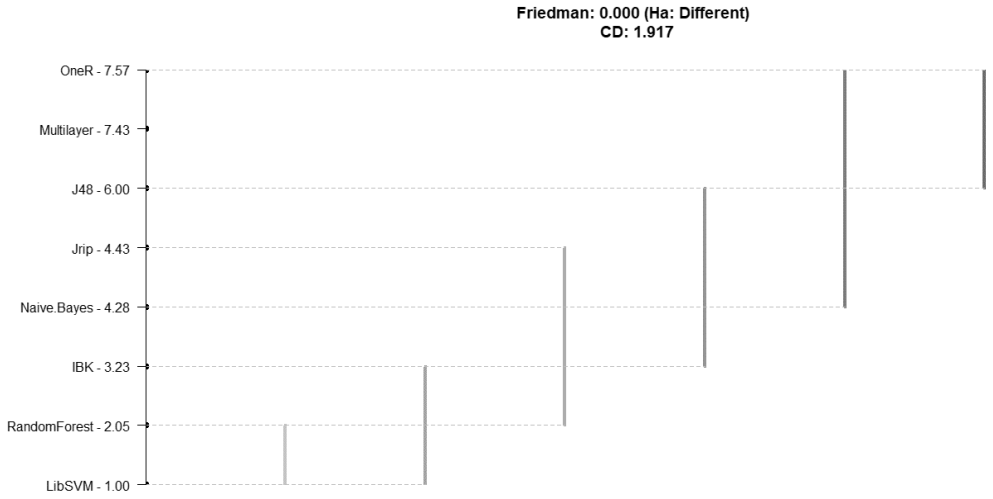

Figura 1. Teste de Nemenyi para cada algoritmo de classificação na Base 1.

A distância crítica (CD – *Critical Distance*) foi definida como 1.917, o que indica que as distâncias entre os algoritmos devem ser maiores do que este valor para caracterizar como diferença significativa. Pode-se concluir que se comparado o valor do ranking do algoritmo LibSVM com o valor do ranking do algoritmo IBK, há diferença estatística entre eles, pois o resultado do cálculo é maior do que o valor CD, porém de forma inversa, antes comparando o valor do ranking do algoritmo IBK com o valor do ranking do algoritmo LibSVM não há diferença estatística significativa entre eles, pois o resultado é menor que o CD definido.

Similarmente a Figura 2 apresenta os resultados dos testes de Friedman e Nemenyi realizados a partir do ranqueamento dos resultados realizados na Base 2, na qual pode-se observar que pelo ranqueamento, dentre os 8 algoritmos testados os algoritmos NaiveBayes, RandomForest, LibSVM e IBK tiveram os melhores valores de precisão.

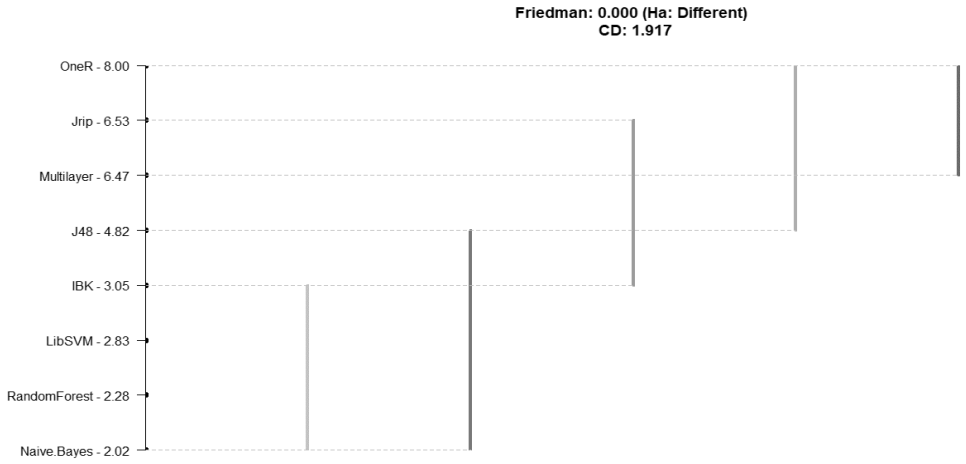

Figura 2. Teste de Nemenyi para cada algoritmo de classificação na Base 2.

A CD dos algoritmos para os testes realizados na Base 2 foi de 1.917, comparando o valor do ranking dos algoritmos NaiveBayes, RandomForest, LibSVM e IBK, pode-se concluir que não há diferença estatística significativa entre eles.

A seguir, na Seção 3.3, serão apresentados os métodos utilizados para a criação dos modelos de classificação no WEKA, que posteriormente foram utilizados no exemplo prático de sistema de recomendação. Foi utilizada a Base 1 devido a quantidade de registros ser muito maior comparada a Base 2, juntamente com os algoritmos LibSVM, RandomForest e IBK, pois conforme foi exposto na Tabela 3, estes algoritmos apresentaram uma melhor adaptatividade a esta base de dados.

3.3. Criação dos modelos de classificação

Para a criação dos modelos de classificação no WEKA foi utilizada a Base 1 e o método *Use Training set*. Este método utiliza toda a base de dados para treinamento e teste e deve ser utilizado somente para a criação dos modelos de classificação, pois na fase de avaliação dos algoritmos, os dados de treinamento devem ser diferentes dos dados de teste. Foram criados 3 modelos de classificação utilizando os algoritmos LibSVM, RandomForest e IBK, o qual foram selecionados com base nas avaliações descritas na Seção 3.2. A parametrização destes 3 algoritmos foi configurada conforme descrito na Tabela 2.

Após cada algoritmo, mencionado no parágrafo anterior, ser configurado e treinado, os modelos de classificação gerados foram salvos e utilizados no sistema para realizar a recomendação de uso de notebooks de acordo com suas características de hardware. A Seção 3.4 apresenta o desenvolvimento e as funcionalidades do exemplo prático do sistema para recomendação de notebooks, não será dado foco sobre os detalhes técnicos de cada componente utilizado, mas em como o sistema faz a recomendação de notebooks utilizando os modelos de classificação criados.

3.4. O Sistema para recomendação de notebooks

O sistema foi desenvolvido na linguagem de programação Java, o que facilitou no uso dos modelos de classificação criados no WEKA. O check-list das funcionalidades do sistema é apresentado abaixo:

1. Receber as características do notebook
2. Inserir as características como uma nova instância
3. Classificar a nova instância utilizando os modelos de classificação
4. Combinação e rejeição dos classificadores
5. Exibir o resultado da classificação

3.4.1. Recebendo as características do notebook

Na tela principal do sistema foram adicionados 17 componentes do tipo ComboBox editáveis. Contudo, algumas características de notebook comuns foram pré-carregadas nesses componentes na inicialização do sistema, como a marca do notebook, a marca do processador, etc., para que atribuísse mais praticidade ao usuário no momento em que fosse inserir as características do notebook, o botão **LOAD* faz esta função. O botão

**REMOVE* faz a limpeza de todos as informações preenchidas. A Figura 3 apresenta um exemplo de preenchimento das características de um notebook no sistema.

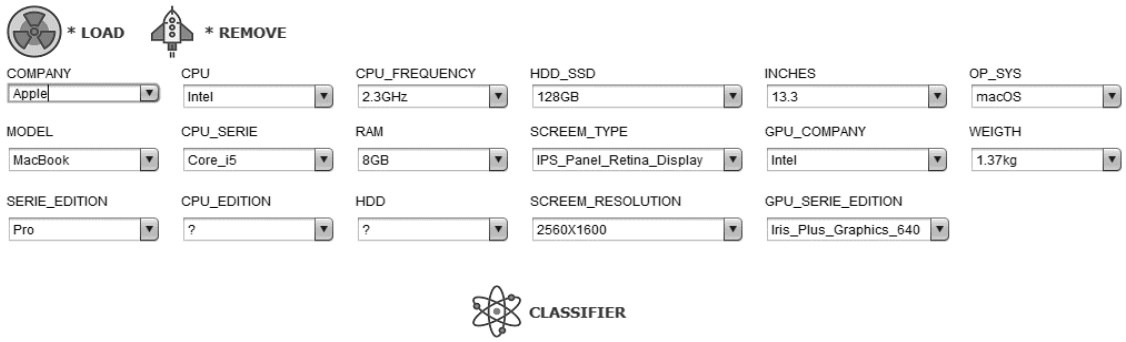

Figura 3. Exemplo de preenchimento de características no sistema.

3.4.2. Inserindo as características como uma nova instância

Após o usuário inserir todas a características do notebook, é necessário inserir as informações preenchidas como novas instâncias a serem classificadas. Esta função está implementada no botão *CLASSIFIER*, cada característica preenchida nos componentes ComboBox é capturada, convertida para o tipo *String* e atribuída à uma variável do tipo *Instance*, assim adicionando 17 novas informações a suas instâncias.

3.4.3. Classificando a nova instância utilizando os modelos de classificação

Ainda nas funções do botão *CLASSIFIER*, após carregar as novas informações em cada instância, é necessário carregar os modelos de classificação criados com os algoritmos LibSVM, RandomForest e IBK para a classificação das informações. Para isso, foram criadas três variáveis do tipo *ObjectInputStream*, nestas variáveis foram carregados os modelos de classificação de cada algoritmo. Também foi criada uma variável para cada tipo de classificador utilizado, ou seja, para o classificador IBK foi criada uma variável do tipo *IBK* e atribuída a esta variável seu modelo de classificação, o mesmo processo foi aplicado nos classificadores LibSVM e RandomForest. A Figura 4 apresenta um exemplo de classificação, onde é possível visualizar o resultado da classificação em cada algoritmo.

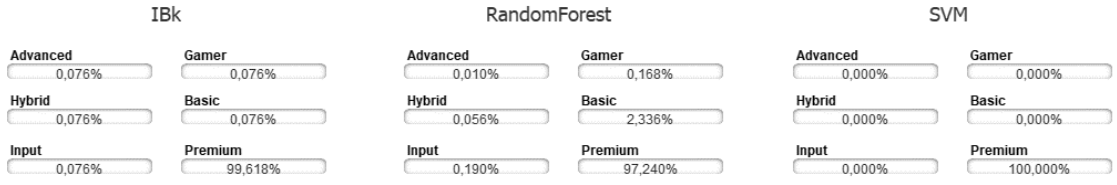

Figura 4. Resultado da classificação dos 3 algoritmos.

Para o exemplo acima, foram utilizados os mesmos dados de exemplo apresentados na Figura 3 da Seção 3.4.1. Neste exemplo é possível determinar que, a categoria de uso recomendada é a *premium*, pois esta é a categoria que possui a maior porcentagem de classificação em relação as demais categorias em cada algoritmo.

3.4.4. Combinação e rejeição dos classificadores

No intuito de obter uma maior confiabilidade, precisão e melhorar a visualização dos resultados de classificação dos algoritmos LibSVM, RandomForest e IBK, uma

estratégia de combinação de classificadores foi implementada.

A estratégia de combinação de classificadores utilizada foi a de pluralidade, onde o resultado da classificação será baseado na classe com o maior número de votos. Nesta estratégia não há a necessidade de que a classe ganhadora receba a maioria dos absoluta dos votos, mas apenas que possua mais votos que as demais classes. Para isso foram criadas 6 variáveis do tipo *Int*, uma para cada categoria de uso de notebook. Em cada algoritmo, os resultados de classificação das 6 categorias são comparados entre si, a categoria com o maior percentual de acerto em relação as demais categorias, receberá um voto, atribuindo no valor da variável correspondente a esta categoria.

Também, para determinar se o resultado de classificação dos algoritmos é aceitável ou não, foi criada uma variável do tipo *Int*, sendo atribuída a esta variável o valor de 0.8, ou seja, para que o resultado de classificação de cada algoritmo seja aceito, este deverá ser maior que 80%, do contrário a instância a que se deseja classificar será rejeitada.

3.4.5. Exibindo o resultado da classificação

A Figura 5 apresenta uma visão geral do sistema, juntamente com o resultado final da classificação utilizando os mesmos dados de exemplo apresentados na Figura 3 da Seção 3.4.1. Conforme apresentado na Seção 3.4.3, este exemplo foi classificado para a categoria *premium*, sendo também esta uma instância aceitável, pois teve o percentual de classificação em todos os algoritmos maior que 80%, valor de rejeição definido e apresentado na Seção 3.4.4.

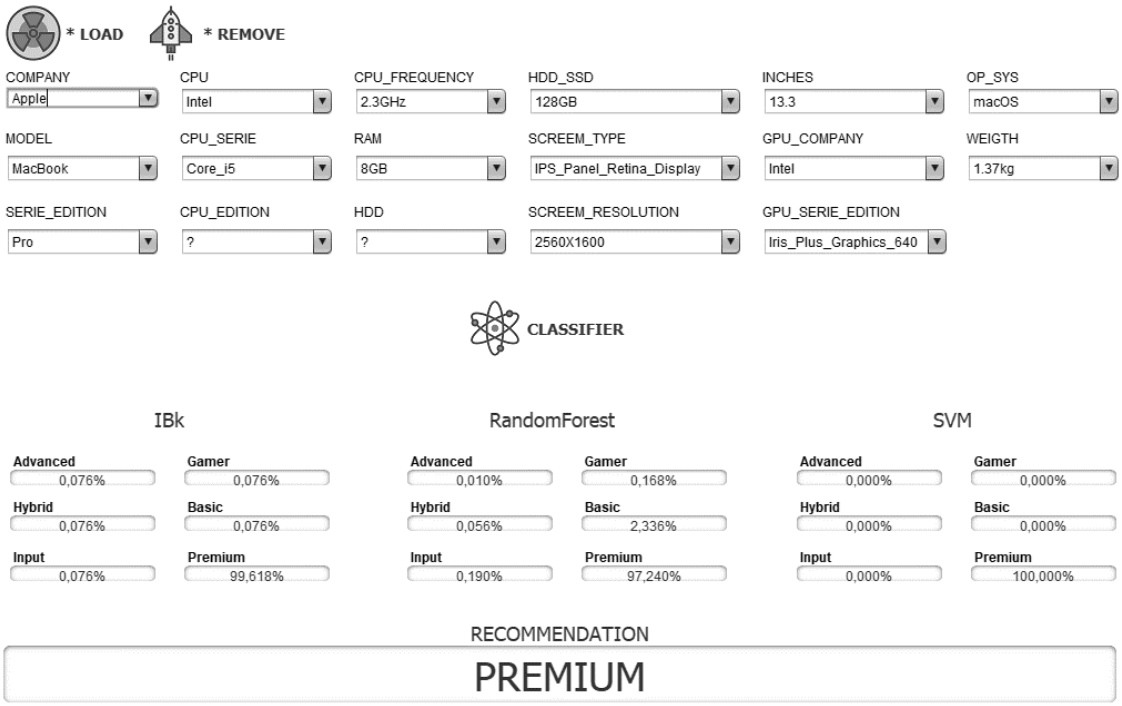

Figura 5. Visão geral do sistema e resultado da classificação.

4. Conclusão

A fim de atender aos requisitos de manuseio eficiente do grande volume de informações, os sistemas de recomendação são usados para entregar recomendações

significativas.

Este trabalho apresenta uma abordagem baseada em modelos de classificação para recomendar produtos para usuários que não possuem conhecimento técnico específico no produto. Foram explorados no total 8 algoritmos de classificação. Com o objetivo de determinar quais algoritmos tiveram a melhor adaptação a base de dados, ou seja, algoritmos com o maior percentual de acerto nos testes de classificação e consequentemente seleciona-los para a criação dos modelos de classificação, foram realizados alguns testes de acurácia.

Dentre os 8 algoritmos utilizados, os algoritmos LibSVM, IBK, RandomForest se destacaram, por apresentarem os maiores valores de precisão. Cada modelo de classificação criado, foi utilizado na versão final do exemplo prático de um sistema para recomendação de produtos que utiliza modelos de classificação.

Referências

R. Santos. (2007) "Conceitos de Mineração de Dados na Web". http://www.lac.inpe.br/~rafael.santos/Docs/WebMedia/2009/webmedia2009.pdf. Janeiro.

Benevenuto, Fabrício; Almeida, Jussara M.; Silva, Altigran S. (2008) "Coleta e Análise de Grandes Bases de Dados de Redes Sociais Online", In IEEE Multimedia Computing and Networking (MMCN).

Turney, P. D. (1999) "Learning to Extract Keyphrases from Text". National Research Council, Institute for Information Technology. Technical Report ERB-1057. W. Buntine; Fayyad, G. (1996) "Graphical Models For Discovering Knowledge". In Research Institute for Advanced Computing Sciences, Computational Sciences Division, NASA Ames Research Center, eds. U.M.

Cabena, P; Hadjinian, P; Stadler, R; Jaapverhees; Zanasi, A. (1998) "Discovering Data Mining: From Concept to Implementation". Prentice Hall.

Scombatti, G. (2017) "Mineração De Dados, Data Warehouse, Data Mining, BI e OLAP". https://www.devmedia.com.br/mineracao-de-dados-data-warehouse-data-mining-bi-e-olap-revista-clubedelphi-146/26537. Novembro.

Breese, J., Heckerman, D., And Kadie, C. (1998) "Empirical Analysis of Predictive Algorithms for Collaborative Filtering". In Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence, Madison, WI, Morgan Kaufmann.

Takahashi, Marcos M. (2015) "Estudo comparativo de Algoritmos de Recomendação". Universidade de São Paulo.

King, D. (2017) "Numerical Machine Learning". https://www.cc.gatech.edu/kingd/datatime/datatime.html., Janeiro.

Decker, K; Focardi, S. (2017) "Technological Overview: A Reporto on Data Mining". ftp://ftp.cscs.ch/pub/CSCS/techreports/1995/CSCS-TR-95-02.ps.gz. Outubro

Larose, D. T. (2005) "Discovering Knowledge in Data: An Introduction to Data Mining". John Wiley and Sons, Inc.

Hamilton, Bruce; Guer, Craig. (2017) "Data Mining Concepts". https://docs.microsoft.com/en-us/sql/analysis-services/data-mining/data-mining-

concepts. Outubro.

Moura, W. F., de Lima Francisco, L., & Gonçalves, C. A. (2016). The Dyad Physical/Virtual Stores in the New Markets. *International Journal of Business Administration*, *7*(6), 10.

Almeida, M. R. (2014). O varejo virtual na realidade do consumidor e lojas físicas no Brasil. *NEGÓCIOS EM PROJEÇÃO*, *5*(2), 01-19.

Gartner Data & Analytics Summit 2018. 5. (2018). "Scale the Value of Data and Analytics". Grapevine, TX.

AN, A. (2006) "Classification Methods". York University, Canada.

Rezende, S. O. (2005) "Mineração de Dados". XXV Congresso da Sociedade Brasileira de Computação.

Barros, P. (2017) "Aprendizagem de Maquina: Supervisionada ou Não Supervisionada?". https://medium.com/opensanca/aprendizagem-de-maquina-supervisionada-ou-n%C3%A3o-supervisionada-7d01f78cd80a. Novembro.

Onoda, M. (2001) "Estudo sobre um algoritmo de árvore de decisão acoplado a um sistema de banco de dados relacional". Rio de Janeiro.

Candiago, L. (2017) "Algoritmo de Classificação Naive Bayes".https://www.organicadigital.com/seeds/algoritmo-de-classificacao-naive-bayes/.

Rish, I. (2001) "An empirical study of the naive Bayes classifier". Proceedings of IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence.

Nurnberger, A.; Pedrycz, W.; Kruse, R. (2002) "Neural network approaches". In Klosgen & Zytkow (Eds.), Handbook of data mining and knowledge Discovery (pp. 304-317). Oxford University Press.

Portugal, Ivens; Alencar, Paulo; Cowan, Donald. (2015) "The Use of Machine Learning Algorithms in Recommender Systems: A Systematic Review", Canada.

Amatriain, Xavier; Jaimes, Alejandro; Oliver, Nuria; Pujo, Josep M. (2011) "Data Mining Methods for Recommender Systems", Barcelona.

Miyahara, Koji; Pazzani, Michael J. (2000) "Improvement of Collaborative Filtering with the Simple Bayesian Classifier", California.

Zhang, Tong; Iyengar, Vijay S. (2002) "Recommender Systems Using Linear Classifiers", U.S.A.

Xia, Zhonghang; Dong, Yulin; Xing, Guangming. (2006) "Support Vector Machines For Collaborative Filtering", Florida.

Jadhav, Sayali D.; Channe, H. P. (2016) "Efficient Recommendation System Using Decision Tree Classifier and Collaborative Filtering, Maharashtra.

Puntheeranurak, Sutheera; Pitakpaisarnsin, Pongpan. (2013) "Time-aware Recommender System Using Naïve Bayes Classifier Weighting Technique", Bangkok.