Rating Prediction of Google Play Store Apps with Application of Data Mining Techniques

R. G. da Silva, J. de O. L Magalhães, I. R. R. Silva, R. A. de A. Fagundes, E. A. de O. Lima and M. A. MACIEL

***Abstract*—Software development is based on implementation standards. In the case of selling and accepting software by customers, it has been a challenge to develop applications for marketplaces. App stores have features such as the number of downloads, comments, and ratings on ratings. From this, the difficulties were the fields (previously listed) and their way of analyzing the problem, thus resulting in characteristics that define the pattern of success in apps. Based on this scenario, this work aimed to create two inference engines from the KNN and Random Forest algorithms and, with that, the features that determine the best correlation for the rating of the applications were investigated, besides to compute and evaluate regression metrics using a Google Play Store database. The work is structured as follows: in section II, the theoretical framework will be presented, where the main themes relevant to this work will be explained. Section III will discuss the materials and methods, where the step by step will be described according to the CRISP-DM to obtain knowledge of the database. In section IV, the results obtained with the execution of the algorithms will be structured and in section V, it will be the conclusion, which will be scored what was possible to understand with this research.**

I. Introdução

O desenvolvimento de *software* é baseado em padrões de implementação. No caso da venda e aceitação de software pelos clientes têm sido um desafio para o desenvolvimento de aplicativos para *marketplaces*. As lojas de aplicativos possuem características, como quantidade de *downloads*, comentários e avaliações sobre os *ratings*. A partir disso, as dificuldades foram os campos (anteriormente listados) e sua forma de analisar o problema, resultando assim, em características que definissem o padrão de sucesso nos *apps*.

Apesar da importância de verificar estas estatísticas, a análise humana destes dados é bastante custosa para as empresas obterem resultados em tempo real, devida a alta quantidade de *softwares* disponibilizados. Em 2018, a Google Play contabilizou mais de 2.6 milhões de aplicações ativas [1]. Para viabilização dessa automação, técnicas de inteligência artificial apresentam-se como uma alternativa, utilizando conceitos de regressão e mineração de dados. Modelos de aprendizado requerem que hajam bases de dados de volume considerável (acima de mil registros) [2]. Para uma compreensão mais apurada dos dados foram utilizados pelos algoritmos KNN (*K-Nearest Neighbors*) [3] e *Random Forest* [4], onde a pesquisa foi direcionada por metodologias clássicas dentro do contexto de *Data Mining*, como o CRISP-DM (*Cross Industry Standard Process for Data Mining*), cujo processo defende uma estrutura padrão para a realização de projetos de mineração de dados de maneira agnóstica, ou seja, sendo mais generalistas possível havendo assim maior abrangência, independente da tecnologia aplicada. A mineração de dados será utilizada como parte do processo de descoberta de conhecimento, cujo objetivo foi selecionar as técnicas que obtiveram uma melhor correspondência ao contexto do problema [5].

Baseado neste cenário, este trabalho teve como objetivo criar dois motores de inferência a partir dos algoritmos KNN e Random Forest e com isso, foi investigado as *features* que determinam a melhor correlação para o *rating* das aplicações, além de computar e avaliar as métricas de regressão, utilizando uma base de dados da Google Play Store. No ponto de vista estatístico, foi realizado um comparativo através de testes de hipótese para identificar a melhor para solução desta análise. A lógica de Wilcoxon [6] será útil para determinar qual o algoritmo mais robusto para a predição. O resultado desta pesquisa poderá ser bastante útil no mercado de desenvolvimento de *software*, sendo possível compreender as reais diretrizes que afetam o interesse ou a falta do mesmo por parte dos clientes.

O trabalho está estruturado da seguinte forma: na seção II será apresentado o referencial teórico, onde serão explanados os principais temas pertinentes a este trabalho. A seção III discutirá os materiais e métodos, onde serão descritos o passo a passo de acordo com o CRISP-DM para obtenção do conhecimento da base de dados. Na seção IV, serão estruturados os resultados obtidos com a execução dos algoritmos e na seção V, será a conclusão, na qual será pontuado o que foi possível compreender com esta pesquisa.

II. Referencial Teórico

Esta seção apresenta os conceitos fundamentais para entendimento desse trabalho.

A. Trabalhos Relacionados

De acordo com a literatura sobre análise de *rating* de aplicativos da Google Play [7], foi implementada a predição baseada em análise de sentimento. Esta análise é feita a partir dos comentários dos usuários, utilizando técnicas de mineração textual. Existe uma limitação para este problema, que é o fato de o usuário avaliar o *app* sem informar textualmente o seu *feedback* [7]. É possível compreender o comportamento

R. G. da Silva, Universidade de Pernambuco, Brasil, rgs@ecomp.poli.br
J. de O. L. Magalhães, Universidade de Pernambuco, Brasil, jolm@ecomp.poli.br
I. R. R. Silva, Universidade de Pernambuco, Brasil, irrs@ecomp.poli.br
R. A. de A. Fagundes, Universidade de Pernambuco, Brasil roberta.fagundes@upe.br
E. A. de O. Lima, Universidade de Pernambuco, Brasil eal@poli.br
M. A. MACIEL, Universidade de Pernambuco, Brasil amam@ecomp.poli.br

das avaliações sem utilizar nenhum modelo de inferência, pesquisa esta já feita em outro artigo. As desvantagens dessa abordagem são o aumento no tempo para coleta de resultados, além de gerar possíveis falhas de análise [8]. Para garantir que o resultado desse estudo não possua ruídos, é importante aplicar um processo estruturado de pré-processamento dos dados. É bastante comum que os usuários determinem uma pontuação descontextualizada com o seu ponto de vista real [9] ou simplesmente ignoram o *feedback* da classificação [10]. Se após a etapa de limpeza, ainda não for possível chegar em um modelo estruturado, torna-se necessário aplicar uma outra iteração de coleta de dados [9].

B. CRISP-DM

A mineração de dados é um processo que exige a existência de diversas técnicas, ferramentas e pessoas [11]. Para garantir o sucesso em projetos de dados, torna-se necessário aplicar uma metodologia validada pela indústria, como também pela comunidade científica [11]. O CRISP-DM (*Cross-Industry Standard Process of Data Mining*) provê uma implementação que atende toda a cadeia de um projeto de mineração de dados [12]. As fases do projeto, suas respectivas tarefas e seus resultados, são documentadas nesta metodologia. [11]. Apesar do CRISP-DM ter uma estrutura bem genérica, seu objetivo não é capturar todas as possibilidades do processo de mineração de dados, pois a metodologia tenderia a se tornar muito complexa [11].

C. K-Nearest Neighbors (KNN)

O K-Nearest Neighbors é uma técnica de aprendizagem de máquina, que tem o objetivo de classificar o rótulo frequentemente dentre as k amostras [13]. O KNN é formado por vetores multidimensionais e seus elementos representam pontos no espaço. O principal objetivo do algoritmo é rotular a classificação, tendo como fonte uma amostra baseada em amostras vizinhas, prescindidas de um treinamento amostral [14]. Para definir a classe, que não pertença ao grupo de treinamento, será necessário verificar a distância em relação ao elemento desconhecido. Para determinar a proximidade entre vizinhos (onde x é cada ponto e a iteração irá incrementando os pontos) é utilizada a distância Euclidiana, demostrado na Equação 1.

$$e = \sqrt{\sum_{j=0}^{n} (x_{i,j} - x_{i+n,j})^2} \tag{1}$$

Os principais pontos do KNN são: o cálculo da distância e o valor de cada K vizinho. A execução do algoritmo tem os seguintes passos: o cálculo da distância com um conjunto de treinamento, identificando os K vizinhos mais próximos e o rótulo da classe escolhido com mais representantes

D. Random Forest (RF)

O Random Forest é formado por uma combinação de preditores em árvore, de maneira que cada árvore seja dependente dos valores de cada vetor aleatório [15].

A partir do treinamento constrói uma grande quantidade de árvores de decisão. O treinamento utiliza *bagging* (meta-algoritmo para melhorar a classificação ou regressão).[15]. O uso de *bagging* no treinamento, reduz a variância, evitando *over-fitting*. Este procedimento extrai aleatoriamente a partir de conjunto de dados de treinamentos originais e os conjuntos usados para construir uma das árvores de decisão [15].

Cada árvore classificadora funciona como um preditor. Através da contagem de votos é a realizado a decisão. A primeira fase consiste no treinamento da árvore utilizando a estratégia de *bagging* aleatória. Os dados são selecionados pelos maiores votados. Onde para verificar sua precisão pode se tentar usando um terço da árvore [16]. A árvore é formada de forma inteligente, onde de acordo com a classificação dos valores irá descer através de um nó outro, tomando como ponto de parada a folha [16].

A atualização de árvores de dados, dependendo da atualização se fará necessário uma nova execução do que a atualização, pois tende a gerar inconsistências, em questão de importância de uma camada para outra, onde as respostas normalmente estão nas folhas e as informações mais genéricas estão mais próximos da raiz [16].

E. Google Play

O Google Play é uma loja virtual de aplicativos, na qual permite que um cliente consiga baixar rapidamente um *software* do seu interesse, além de avaliá-los e comentá-los [17]. Os desenvolvedores e empresas que criam essas aplicações, utilizam um ambiente de monitoramento disponibilizada pela própria Google Play, para analisar os dados da loja informados pelos consumidores. Esses dados são relevantes para otimizar as funcionalidades disponibilizadas ou executar correções críticas e não críticas. A automação do comportamento da avaliação de um *app* pode ser útil para que os mantenedores possam compreender preditivamente, como melhorar os seus produtos.

A Base de dados da Google Play [18] possui 10.841 instâncias e 13 features. Todas variáveis são categóricas, exceto a coluna '*Rating*', cujo tipo é *float*. A base de dados contém as informações da Tabela I. A base pode ser acessada em: Kaggle (https://www.kaggle.com/lava18/google-play-store-apps), no arquivo *googleplaystore.csv*.

F. Pré-processamento dos Dados

O pré-processamento é uma técnica de mineração de dados que permite avaliar o comportamento de cada *feature* [19]. A base do Google Play, possui algumas variáveis que precisavam ser tratadas para otimizar a performance do modelo e com isso, evitar resultados incoerentes. Por exemplo, os valores quando baixados do arquivo .csv, eles estão todos no formato de texto e para ser mais rápido sua manipulação é necessário converter os valores de texto para inteiro, para *float* ou mesmo trabalhar com valores booleanos (verdadeiro e falso). Para isso, se faz necessário o pré-processamento. No pré-processamento é feita uma limpeza dos dados, como por exemplo, converter textos para tipos numéricos [19]. No pré-processamento o *rating* pode ser visualizado com mais ênfase segundo a Fig. 1 e tendo os outros valores com menos registros. As linhas da Fig. 1,

TABLE I
ENTENDIMENTO DOS DADOS

Coluna	Tipo	Descrição
App	Categórica	Nome da aplicação.
Category	Categórica	Categoria a qual a aplicação pertence.
Rating	Discreta	Avaliação atribuída pelo usuário.
Reviews	Contínua	Número de usuários que avaliaram o aplicativo.
Size	Contínua	Tamanho do aplicativo.
Installs	Contínua	Número de downloads.
Type	Categórica	Pago ou grátis.
Price	Contínua	Preço do aplicativo.
Content Rating	Categórica	Faixa etária do aplicativo.
Genres	Categórica	Gênero do aplicativo.
Last Updated	Categórica	Data de atualização do aplicativo.
Current ver	Discreta	Versão do aplicativo na loja.
Android ver	Discreta	Versão do android compatível.

principalmente na coluna de *rating* demostram os valores nulos que foram tratados.

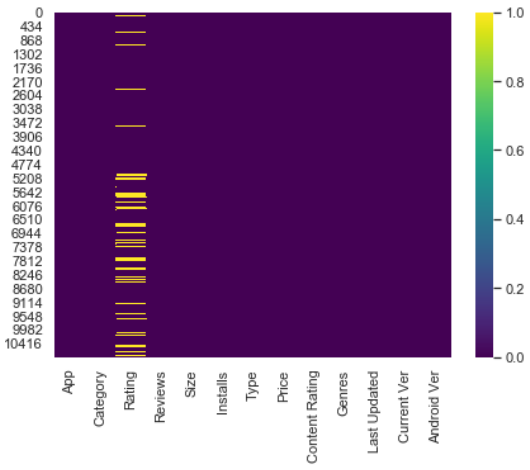


Fig. 1. Contabilização de *Rating*.

Parece que estão faltando valores "*Rating*", "*Type*", "*Content Rating*" e "Android Ver". Há duas estratégias para manipular dados ausentes, removendo registros com esses valores ausentes ou substituindo valores ausentes por um valor específico como média, mediana ou moda. É óbvio que o primeiro valor deste registro está faltando (nome do aplicativo) e todos os outros valores são propagados, respectivamente, começando de "*Category*" para "Current Ver"; e a última coluna, que é "Android Ver", é deixada em branco. É melhor remover esta instância em vez de considerar estes valores durante a limpeza de cada coluna.

III. MATERIAIS E MÉTODOS

1) Preparação da base de dados: Ao realizar a análise da base de dados utilizada no projeto, foram encontrados alguns problemas, como dados faltosos, atributos contendo *strings*, alta amplitude dos valores e alta quantidade de atributos. Para a preparação dos dados, foram utilizadas diversos *softwares* e plataformas, tais quais: Python, Excel e R, cada um utilizado para uma finalidade específica. A resolução dos problemas citados será discutida nos pontos a seguir.

2) Codificação de dados categóricos: Muitos algoritmos de aprendizado de máquina podem suportar valores categóricos sem manipulação adicional, mas há muito mais algoritmos que não. Foram disponibilizados todos os dados para o modelo, portanto, foram convertidas variáveis categóricas (variáveis armazenadas como valores de texto) em variáveis numéricas. A Tabela II demostra algumas modificações que foram necessárias para limpeza dos dados.

TABLE II
CONVERSÃO DAS VARIÁVEIS CATEGÓRICAS EM NUMERICAS

Coluna	Ação
Rating	Executar mediana para os campos nulos.
Current version	Remover os caracteres especiais.
Price	Remover o $
Installs	Remover o +
Last updated	Colocar a data em um sequencial numérico (ddMMaaaa)
Size	Remover o M e o k
Demais campos	Colocar 0 em valores Null

É descartado a coluna de referência e apenas mantém apenas uma das duas colunas, pois reter essa coluna extra não adiciona nenhuma informação nova para o processo de modelagem, esta linha é exatamente a mesma que define o parâmetro dropfirst como *True*.

3) Outras técnicas de pré-processamento: Além da transformação de dados categóricos em numéricos e preenchimento de dados faltosos, podem ser uteis também a aplicação da seleção de atributos, a partir de uso do PCA (*Principal Component Analysis*) e o balanceamento de classes. Quando uma determinada base possui muitos atributos, os modelos geralmente sofrem na execução devido ao custo computacional que deverá ser aplicado, para isso torna-se indispensável o uso de alguma de técnica que realize o mapeamento das melhores correlações. O *ranking* de uma correlação é determinado entre o intervalo de 0 e 1. Quanto mais próximo, melhor é a correlação.

O balanceamento de classes é aplicado geralmente em problemas de classificação, mas que podem ser contextualizados em casos de regressão. Esta ferramenta sugere que as classes da variável de análise das instâncias tenham uma divisão proporcional. Por exemplo, para uma base de dados de ressonâncias de tumores, a classificação de que há evidências de alguma doença no total das imagens, seja de 50% e o caso contrário, possua o mesmo valor. Quando a estrutura não corresponde a este cenário, é necessário aplicar estas técnicas para que o algoritmo não seja treinado e testado de maneira equivocada.

4) Análise de correlação entre as variáveis: Para critério de análise dos atributos mais relevantes, foi aplicada a técnica de correlação de *Spearman*. A técnica de *Spearman* realiza a correlação entre duas variáveis para determinar as features mais relevantes da base de dados [20]. A execução de um algoritmo *feature selection* é de suma importância no processo de pré-processamento, pois é possível remover atributos que

não trarão bons resultados na acurácia, além de diminuir o custo computacional para os algoritmos de classificação de dados. A acurácia foi calculada a partir da função fit da biblioteca do *scikit-learn*. O modelo é ajustado usando X como dados de treinamento e y como valores-alvo.

A Fig. 2 apresenta a maneira como as *features* estão correlacionadas. Neste exemplo, quanto mais claros forem os quadrados, maior a importância para o modelo. As colunas que mais influenciam o resultado são: *rating*, *type*, *content rating*, *genres*, *last updated*, *current ver*. É possível parametrizar a função de plot do mapa, para que as colunas com maior significância sejam caracterizadas por uma cor mais evidente [21]. O menor valor para uma determinada variável é de -1 e o maior de 1 [22].

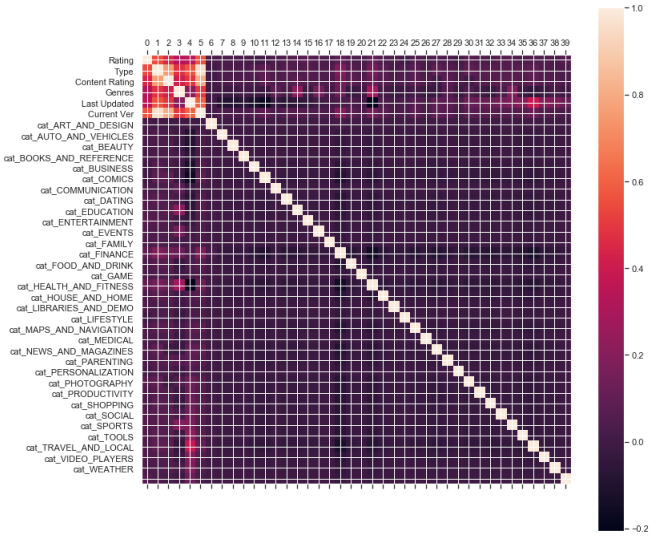


Fig. 2. Mapa de correlação.

A. Modelagem

Nesta seção, foram mostrados como o KNN e o *Random Forest* podem ser usados para prever as avaliações de aplicativos com base nas outras matrizes. Primeiro, o conjunto de dados deve se separar em variáveis dependentes e independentes (ou recursos e rótulos). Então, essas variáveis precisam ser divididas em um conjunto de treinamento e teste. Durante o estágio de treinamento, foram dados ao modelo os recursos e os rótulos para que ele possa aprender a classificar os pontos com base nos recursos. Este script divide o conjunto de dados em 75% de dados de treino e 25% de dados de teste.

1) K-Nearest Neighbors (KNN): O algoritmo de k vizinhos mais próximos baseia-se na ideia simples de prever valores desconhecidos, combinando-os com os valores conhecidos mais semelhantes. Construir o modelo consiste apenas em armazenar o conjunto de dados de treinamento. Para fazer uma previsão para um novo ponto de dados, o algoritmo encontra os pontos de dados mais próximos no conjunto de dados de treinamento - seus "vizinhos mais próximos". Esta técnica foi escolhida por ser amplamente utilizada no mercado de *Data Science* para solução de problemas de regressão, além de ser fácil de interpretar os *outputs*, permitir um baixo custo computacional e possui um alto poder preditivo [23].

2) Random Forest: O *Random Forest* é um algoritmo comumente utilizado em problemas de regressão e classificação. Ele implementa o conceito de árvore de decisão, porém de maneira randômica, criando múltiplas estruturas e juntando-as para adquirir uma maior eficácia na previsibilidade [24]. O *Random Forest* foi selecionado para predição dos dados do Google Play, pois o mesmo possui um histórico de modelos com alta acurácia, além de implementar um controle para capturar dados faltosos que trarão uma inconsistência nos resultados.

Outro fator bastante interessante é o alto poder de processamento independente da escalabilidade dos dados [24]. Este modelo foi implementado com a classe RandomForestRegressor da biblioteca sklearn.ensemble. O parâmetro mais importante da classe RandomForestRegressor é o parâmetro *n estimators*, pois ele define o número de árvores na floresta aleatória [25].

B. Avaliação

O projeto utiliza a biblioteca do Scikit Learn (biblioteca Python) para realizar regressão. As funções uteis para esta pesquisa são a raiz do erro médio quadrático, a variância explicada e o R^2. A raiz do erro médio quadrático é a raiz do erro médio quadrático da diferença entre a predição e o valor real, segundo a Equação 2.

$$RMSE = \sqrt{\frac{\sum(y_{pred} - y_{ref})^2}{N}} \tag{2}$$

A variância explicada é calculada através da Equação 3, a variância da regressão.

$$Var(y, \widehat{y}) = 1 - \frac{Var(y - \widehat{y})}{Var(y)} \tag{3}$$

O melhor possível valor é 1, menor que esse valor, se torna um resultado pior. O R^2 e a variância aceitam um outro parâmetro, que seria a variância de peso. O R^2 calcula o coeficiente de determinação, segundo a Equação 4, que calcula um melhor exemplo para predizer o modelo, cujo melhor valor seria 1.0.

$$R^2(y, \widehat{y}) = 1 - \frac{\sum_{i=0}^{n-1}(y_i - \bar{y}_i)^2}{\sum_{i=0}^{ns-1}(y_i - \bar{y}_i)^2} \tag{4}$$

IV. Resultados

A. Resultados dos Experimentos

A *Table II* apresenta os resultados obtidos com a aplicação do KNN e do Random Forest.

TABLE III
RESULTADO DOS MODELOS

	KNN	RANDOM FOREST
RMSE	0.19963233762406599	0.1620077279185534
R^2	0.9234911882926484	0.9379107788833106
Variação explicada	0.9235585231599096	0.9380437155359854
Acurácia (10 K-fold)	92.35%	93.8%
Desvio Padrão	0.03%	0.02%

Os algoritmos K-NN e *Random Forest* foram treinados com 75% da base e testados com 25%, para evitar o *overfitting* e *underfitting* [26]. As amostras foram aleatorizadas em 10 grupos através do *k-fold Cross Validation*. O K-NN obteve um desvio padrão de 0.03% e o *Random Forest* em 0.02%. A partir dessa métrica de desvio padrão, é possível identificar que o K-NN é mais disperso e afastado da média em comparação ao *Random Forest*.

A raiz do erro médio quadrático foi utilizada (RMSE), pois é a medida mais comumente aplicada para avaliar a qualidade do ajuste em modelos de regressão. O R^2 foi utilizado para identificar o ajuste dos modelos de regressão.

O Random Forest possui um valor de RMSE menor que o K-NN. Quanto menor o valor obtido, menor também é a taxa de erro em relação à predição e o valor real. Então é possível afirmar que o Random Forest tende a realizar uma predição de melhor qualidade.

O Random Forest possui o R^2 maior que o K-NN. Isso significa que quanto maior esse valor, mas eficiente é o modelo.

Baseado nestas duas métricas, é possível afirmar que o Random Forest é melhor recomendado que o K-NN, pois o mesmo obteve um melhor desempenho na raiz do erro médio quadrático, como também no R^2.

B. Testes Estatísticos Realizados

O teste estatístico de Wilcoxon foi utilizado para aplicar um teste de hipótese [6]. O teste contemplou a eficiência entre os modelos K-NN e o *Random Forest*. A hipótese nula descreveu que as duas amostras apresentaram valores aproximadamente iguais. A hipótese alternativa, apresentou que uma amostra obteve valores maiores que a outra amostra. A formulação desta hipótese foi descrita segundo a Equação 5.

$$\begin{cases} H_0 : U_1 = U_2 \\ H_1 : U_1 > U_2 \end{cases} \tag{5}$$

Devido ao fato de o *Random Forest* ter obtido em média valor maior que o K-NN, assume-se que se deseja provar que de fato o primeiro algoritmo obteve maior rendimento de acurácia e RMSE que o segundo algoritmo. Desta forma, de acordo com a hipótese formulada acima, assume-se que u1 seja a média obtida com os experimentos com o *Random Forest* e u2 seja a média obtida com os experimentos com o K-NN. Para isso, são necessárias as execuções de dois testes de hipótese, um para a acurácia e o outro para o RMSE. Os resultados do teste de hipótese são apresentados na Tabela IV, onde a segunda e terceira colunas apresentam os valores de W e p-value retornados a partir do teste.

TABLE IV
RESULTADOS OBTIDOS COM OS TESTES DE HIPÓTESE REALIZADOS EM CADA CONJUNTO DE DADOS (MÉTRICAS).

Métrica	W	p-value
Acurácia	0	4.320464057827488e-08
RMSE	0	4.320464057827488e-08

Com os resultados dos testes de hipótese, pode-se afirmar, em ambas as métricas, que a hipótese nula pode ser refutada, e que há evidências suficientes no conjunto de dados apresentados de que o *Random Forest* apresenta o melhor resultado de regressão para *ratings* de apps da Google Play Store.

V. CONCLUSÃO

Este trabalho apresentou uma abordagem utilizando técnicas de regressão a partir do aprendizado de máquina para extração de conhecimento em uma base dados, com a finalidade de prever o *rating* de pouco mais de 10000 aplicativos da loja da Google Play.

Após a execução dos algoritmos, foram coletadas as métricas de regressão, como por exemplo, a variância explicitada e o RMSE, pois apenas o score não determina se o modelo é capaz de prever corretamente as entradas. A *feature* de melhor correlação foi o *Last Updated*. A partir desta informação é possível extrair diversos conhecimentos, como o fato de que as últimas versões tendem a ter melhor compatibilidade com o sistema operacional nativo, melhor segurança, usabilidade, além de prover barreiras a ataques cibernéticos.

Foi possível perceber também que as *features* dinamicamente criadas a partir da distinção identificada da *feature Category*, não trouxeram significância para os resultados dos modelos, pois de acordo com o mapa de correlação, elas possuíam com a variável categórica do eixo y (*rating*). Alguns aspectos fortes desta pesquisa são o fato do KNN e do Random Forest atenderem bem a necessidade de processamento e avaliação da base de dados estudada. Uma oportunidade para este artigo é de permitir a aplicação em outros cenários para provar que os modelos possuem arquiteturas genéricas.

Uma possível ameaça ao projeto, são os dados fornecidos, pois não há como garantir que os consumidores avaliem corretamente os aplicativos da Google Play. Um outro fator a ser analisado, é a possível necessidade de utilizar outros algoritmos caso o tamanho precise ser escalável. Como trabalhos futuros pretende-se aplicar o *Deep Learning* a partir do *framework* Tensorflow, desenvolvido pela Google. O uso do *Deep Learning* permitirá testes com bases mais robustas.

AGRADECIMENTOS

Os autores agradecem a Coordenação de Aperfeiçoamento de Pessoa de Nível Superior (CAPES) pelo financiamento de bolsa de pesquisa.

REFERENCES

[1] "Google play store: number of apps 2018 — statista," https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/, (Accessed on 12/21/2018).

[2] "Basic concepts in machine learning," https://machinelearningmastery.com/basic-concepts-in-machine-learning/, (Accessed on 12/21/2018).

[3] Z. Zhang, "Introduction to machine learning: k-nearest neighbors," *Annals of translational medicine*, vol. 4, no. 11, 2016.

[4] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[5] G. Kesavaraj and S. Sukumaran, "A study on classification techniques in data mining," in *Computing, Communications and Networking Technologies (ICCCNT), 2013 Fourth International Conference on*. IEEE, 2013, pp. 1–7.

[6] E. A. Gehan, "A generalized wilcoxon test for comparing arbitrarily singly-censored samples," *Biometrika*, vol. 52, no. 1-2, pp. 203–224, 1965.

[7] M. R. Islam, "Numeric rating of apps on google play store by sentiment analysis on user reviews," in *2014 International Conference on Electrical Engineering and Information & Communication Technology*. IEEE, apr 2014. [Online]. Available: https://doi.org/10.1109/iceeict.2014.6919058

[8] Y. Tian, M. Nagappan, D. Lo, and A. E. Hassan, "What are the characteristics of high-rated apps? a case study on free android applications," in *Software Maintenance and Evolution (ICSME), 2015 IEEE International Conference on*. IEEE, 2015, pp. 301–310.

[9] X. Amatriain, J. M. Pujol, N. Tintarev, and N. Oliver, "Rate it again: increasing recommendation accuracy by user re-rating," in *Proceedings of the third ACM conference on Recommender systems*. ACM, 2009, pp. 173–180.

[10] S. Bidmon, R. Terlutter, and J. Röttl, "What explains usage of mobile physician-rating apps? results from a web-based questionnaire," *Journal of medical Internet research*, vol. 16, no. 6, 2014.

[11] R. Wirth and J. Hipp, "Crisp-dm: Towards a standard process model for data mining," *Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining*, 01 2000.

[12] K. R. Santos, I. R. R. Silva, and R. A. A. Fagundes, "Classifiers comparison for attack detection in computer networks," *IEEE Latin America Transactions*, vol. 15, no. 1, pp. 87–96, 2017.

[13] "Classificação usando knn – monolito nimbus," https://www.monolitonimbus.com.br/classificacao-usando-knn/, (Accessed on 12/15/2018).

[14] J. M. d. C. de Sá, I. R. R. Silva, R. G. da Silva, L. G. A. Souto, and P. G. S. Silva, "Análise de crédito utilizando uma abordagem de mineração de dados," *Revista de Engenharia e Pesquisa Aplicada*, vol. 3, no. 3, 2018.

[15] A. Cutler, D. Cutler, and J. Stevens, *Random Forests*, 01 2011, vol. 45, pp. 157–176.

[16] C. Procópio, "Aplicacao do algoritmo random forest como classificador de padroes de falhas em rolamentos de motores de inducao."

[17] N. Zhong and F. Michahelles, "Google play is not a long tail market: an empirical analysis of app adoption on the google play app market," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*. ACM, 2013, pp. 499–504.

[18] "Google play store apps — kaggle," https://www.kaggle.com/lava18/google-play-store-apps, (Accessed on 01/07/2019).

[19] N. Mohd Nawi, W. Atomi, and S. M. Rehman Gillani, "The effect of data pre-processing on optimized training of artificial neural networks," vol. 11, 06 2013.

[20] "Correlation (pearson, kendall, spearman) - statistics solutions," https://www.statisticssolutions.com/correlation-pearson-kendall-spearman, (Accessed on 12/17/2018).

[21] "seaborn.heatmap — seaborn 0.9.0 documentation," https://seaborn.pydata.org/generated/seaborn.heatmap.html, (Accessed on 12/21/2018).

[22] C. J. Burke, W. R. Walter, S. Gaddam, H. Pham, J. S. Babb, J. Sanger, and F. Ponzo, "Correlation of benign incidental findings seen on whole-body pet-ct with knee mri: patterns of 18f-fdg avidity, intra-articular pathology, and bone marrow edema lesions," *Skeletal Radiology*, vol. 47, no. 12, pp. 1651–1660, Dec 2018. [Online]. Available: https://doi.org/10.1007/s00256-018-3001-x

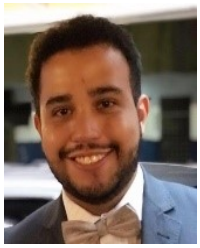
[23] Z.-H. Zhou and M. Li, "Semi-supervised regression with co-training." in *IJCAI*, vol. 5, 2005, pp. 908–913.

[24] A. Liaw, M. Wiener *et al.*, "Classification and regression by randomforest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.

[25] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[26] ——, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

Raniel Gomes da Silva possui graduação em Sistemas de Informação Bacharelado pela Faculdade Joaquim Nabuco (2014), pós-graduação em Engenharia de Software pela Universidade de Pernambuco (2016). Atualmente é mestrando em Engenharia da Computação pela Universidade de Pernambuco. Tem experiência na área de Engenharia de Software, Engenharia de Requisitos, Redes de Computadores e Desenvolvimento de Softwares Desktop, Web e Mobile.



Jailson de Oliveira Liberato Magalhães possui graduação em Ciência da Computação (2013) pela Universidade Católica de Pernambuco e Especialização em Desenvolvimento de Dispositivos Móveis – TECDAM (2018) pelo Centro de Estudos e Sistemas Avançados do Recife/Faculdade dos Guararapes. Atualmente é mestrando em Engenharia da Computação pela Universidade de Pernambuco. Possui experiência em desenvolvimento de sistemas web.

Iago Richard Rodrigues Silva é graduado (2016) em Sistemas de Informação pela Universidade de Pernambuco (UPE). Atualmente, é mestrando em Engenharia da Computação na UPE – Escola Politécnica de Pernambuco.

Roberta Andrade de Araujo Fagundes é Pós-Doutora (2015) em Estatística pela Universidade Federal de Pernambuco (UFPE), Mestre (2006) e Doutora (2013) em Ciência da Computação pela UFPE e Graduada (2002) em Tecnologia em Telemática pelo Centro Federal de Educação Tecnológica da Paraíba (CEFET-PB). Atualmente é Professora Adjunta (2007) da UPE.

Emerson Alexandre de Oliveira Lima é Doutor em Matemática Pura e Aplicada (2003) com Pós-Doutorados em Matemática Computacional (2005) e Geofísica Computacional (2007) na UNICAMP (SP). Sua áreas de atuação incluem aplicações de matemática à Ciência da Computação em particular ao Processamento de Sinais e Computação Inteligente. Atualmente atua na Universidade de Pernambuco como Professor Adjunto.

Alexandre Magno Andrade Maciel Possui graduação em Ciência da Computação pela Universidade Católica de Pernambuco (2003) e mestrado e doutorado em Ciência da Computação pela Universidade Federal de Pernambuco (2007 e 2012). Atualmente é Coordenador Geral de Inovação da Universidade de Pernambuco, pela qual acumula o cargo de Cientista-chefe do Instituto de Inovação Tecnológica.