Avaliação de Modelos de Preditivos de Regressão para Estimar Esforço de Software

Antonio G. L. Esteves[1], Leonardo M. Medeiros[1], Flavio M. Medeiros [1], Mirko Perkusich[2]

[1]Instituto Federal de Educação, Ciência e Tecnologia de Alagoas (IFAL) – Campus Maceió – Maceió, AL – Brasil
[2]Instituto Federal de Educação, Ciência e Tecnologia de Paraíba (IFPB) – Campus Monteiro – Monteiro, PB – Brasil

{flaviomotamedeiros, leonardomelomedeir, toni.esteves@gmail.com}@gmail.com

Abstract. *Effort estimation is a critical task in the software development life cycle. Inaccurate estimations might cause customer dissatisfaction and reduce product quality. In this paper, we evaluate the use of machine learning-based techniques to estimate effort for software tasks. We executed an empirical study based on the [Desharnais 1989] dataset and compared the predictions of three models: Linear Regression (LR), Support Vector Machine (SVM) and K-Nearest Neighbor (KNN) algorithms. The results of our study show that some software metrics are more important to estimate software effort than others. Also based on the quadratic error, which calculates how close the distance of a square regression line is to a set of points, we can successfully estimate 76% of the software effort for the dataset studied, where the predictive models created show only 3% difference between them.*

Keywords: *Machine Learning. Software Metrics. Predictive Model. Effort Estimation.*

Resumo. *A estimativa de esforço é uma tarefa crítica no ciclo de vida de desenvolvimento de software. Estimativas imprecisas podem causar insatisfação do cliente e reduzir a qualidade do produto. Neste artigo, avaliamos o uso de técnicas baseadas em aprendizado de máquina para estimar o esforço para tarefas de software. Executamos um estudo empírico baseado no conjunto de dados [Desharnais 1989] e comparamos as previsões de três modelos: Regressão Linear (LR), Support Vector Machine (SVM) e K-Nearest Neighbor (KNN). Os resultados do nosso estudo mostram que algumas métricas de software são mais importantes para estimar o esforço de software do que outras. Também com base no erro quadrático, que calcula quão próxima a distância de uma linha de regressão quadrada é de um conjunto de pontos, podemos estimar com sucesso 76% do esforço de software para o conjunto de dados estudado, onde os modelos preditivos criados mostram apenas 3% diferença entre eles.*

Palavras-chave: *Aprendizado de Máquina. Métricas de Software. Modelo Preditivo. Estimativa de Esforço.*

1. INTRODUÇÃO

O gerenciamento da complexidade no desenvolvimento de software por vezes é uma tarefa desafiadora. Assim, os profissionais e os gestores do projeto devem analisar as razoes do investimento, os benefícios esperados da iniciativa, os custos assumidos, os riscos e as oportunidades futuras que o projeto irá criar [Trendowicz and Jeffery 2014].

A estimativa do esforço necessário para a construção de um software é um dos problemas mais antigos e importantes no gerenciamento de projetos de software [Huijgens et al. 2017]; ser capaz de planejar corretamente é a base para todas as atividades de gerenciamento dos projetos. Nos últimos 40 anos pesquisadores propuseram uma serie de abordagens de estimativa de esforço de software como an alise de especialistas, uso de dados históricos, estimativas baseadas em analogias e modelos algorítmicos, que misturam diferentes métricas e diversos fatores de custo. Isto é, o objetivo principal é produzir estimativas precisas, seja empiricamente baseando-se na opinião de especialistas ou através de modelos matemáticos [Abran et al. 2015].

Existe um grande Número de modelos, cada um com diferentes pontos fortes e fracos e com diferentes características relacionadas tanto ao ambiente, quanto ao contexto ao qual se inserem. Modelos de esforço de software, por sua vez, consideram em sua criação tanto aspectos do projeto, quanto de seu ambiente [Trendowicz and Jeffery 2014]. Desta maneira, considerando o desafio de prever o esforço e o custo no desenvolvimento de software, é possível salientar que o processo de estimativa leva em consideração características presentes no conjunto de dados, bem como aspectos do ambiente em que o modelo de esforço esta inserido.

Neste estudo, nos concentramos em criar um modelo preditivo para estimativa de esforço de software utilizando o conjunto de dados Desharnais [Desharnais 1989]. Além disso, analisamos a importância de vários atributos presentes no conjunto de dados para criação desse modelo, bem como o quanto esses atributos estão correlacionados entre si. A correlação ou o coeficiente de correlação mede a tendência de duas variáveis mudarem em função de seu relacionamento. O coeficiente de correlação de Pearson produz um resultado entre -1 e 1. Um resultado de -1 significa que existe uma correlação negativa perfeita entre os dois valores, enquanto um resultado de 1 significa que ha uma correlação positiva perfeita entre as duas variáveis. Resultados entre 0,5 e 1,0 indicam alta correlação [Boslaugh and Watters 2008].

Portanto, diante da incerteza em estimar projetos de software com precisão, bem como a correta definição dos fatores que influênciam uma estimativa de esforço mais precisa, pretendemos responder a duas questões de pesquisa:

1. Quais são as métricas mais relevantes e sua correlação ao com o esforço de software no conjunto de dados estudado?
2. Quão preciso pode ser um modelo de estimativa de esforço de software para este conjunto de dados?

As próximas seções do artigo estão organizadas da seguinte forma: na seção 2, discutimos a estimativa de esforço e os trabalhos relacionados, a seção 3 descreve os procedimentos aplicados e os detalhes sobre como construímos os modelos estatísticos

para estimar o esforço de software, na seção 4 discutimos os resultados do nosso estudo, e por fim na seção 5 fizemos nossas considerações sobre este trabalho e apresentamos as propostas de estudo futuras.

2. ESTIMATIVA DE ESFORÇO

Uma das questões fundamentais em um projeto de software é saber, antes de executá-lo, o esforço necessário para desenvolve-lo [Wazlawick 2013].

Atualmente, profissionais da área de engenharia de software necessitam de estimativas de esforço para apoia-los em decisões de projeto, definição de prazos e custos. Por este motivo, identificar uma abordagem apropriada de estimativa é um ponto crucial no processo de mensurar o esforço necessário para um produto de software, em especial no contexto de fornecimento de software por meio de contratos de preço fixo. Logo, a qualidade do software, a eficiência do processo e a produtividade podem ser calculadas através de métricas pré-estabelecidas, e naturalmente todas essas informações podem ajudar uma organização a tomar decisões eficazes [Trendowicz and Jeffery 2014]. Exemplos de métricas ao tamanho da tarefa, complexidade da tarefa, e experiência da equipe [Dantas et al. 2018].

Tendo uma base de dados contendo os valores coletados das métricas e do esforço resultante, é possível criar modelos preditivos para estimar o esforço. A precisão desses modelos, e consequentemente das estimativas, depende diretamente da qualidade da informação na qual ela baseia a previsão. No entanto, no mundo real, ao se trabalhar com dados históricos é comum que essas informações possam ser ambíguas, incompletas, inconsistentes ou até mesmo contraditórias [Trendowicz and Jeffery 2014].

Uma maneira de abordar métricas de software é classifica-las através dos critérios usados para determinar os valores dessa métrica, isto é, as métricas de software podem ser classificadas em métricas objetivas e métricas subjetivas. Métricas objetivas são obtidas através de regras bem definidas e são a melhor maneira de permitir comparações subsequentes e consistentes. As métricas subjetivas por sua vez, dependem de quem está realizando a mensuração e discriminam as correlações e a reprodutibilidade das medidas [Trendowicz and Jeffery 2014]. Em projetos de software, a estimativa de esforço é um ponto crítico, onde o custo e o esforço comumente usam dados históricos para realizar estimativas, o que é útil durante as fases iniciais do projeto de software [Malhotra 2016]. No entanto para permitir a avaliação do software de forma sistemática, eficiente e barata, os valores obtidos através das métricas devem ser sempre os mesmos, independentemente do tempo ou condições individuais em que foram realizadas as medições [Trendowicz and Jeffery 2014].

Infelizmente, especialistas humanos nem sempre são bons em estimar o esforço necessário para um produto de software. Estimativas de custo e esforço em projetos de software são muitas vezes imprecisas, com uma superação média de cerca de 30% [Halkjelsvik and Jørgensen 2011]. Decisões sobre o método de estimativa particular e o seu uso sensato exigem perspicácia sobre as técnicas de estimativa de esforço [Trendowicz and Jeffery 2014].

Todavia, técnicas de aprendizado de máquina em modelos algorítmicos são frequentemente aplicadas, dada a sua capacidade de aprender continuamente e sua precisão. Modelos baseados em aprendizado de máquina apoiam-se em dados históricos e conhecimento especializado para aprender e realizar estimativas [Nayebi et al. 2015]. Uma aplicação essencial de um modelo algorítmico de regressão, por exemplo, é prever novas ou futuras observações *Y* correspondentes a um nível específico de uma variável *x* de regressão. O caso da regressão linear simples considera uma única variável *x* de regressão ou predição, e uma variável dependente ou variável alvo *Y* [Montgomery and Runger 2010].

Ao se utilizar modelos preditivos também se faz necessário utilizar dados históricos em que os resultados são conhecidos para que seja possível validar e testar a previsão do modelo em relação ao resultado real [Sigweni et al. 2016]. Por este motivo, a qualidade do conjunto de dados utilizados e dos subconjuntos de recursos selecionados na construção desses modelos é essencial, assim como o conhecimento dos especialistas [Nayebi et al. 2015].

2.1. TRABALHOS RELACIONADOS

Nesta seção, apresentamos estudos e detalhes relacionados ao campo de estimativa de esforço.

O trabalho desenvolvido por [Panda et al. 2015] se propôs a construir um modelo para melhorar a precisão da estimativa de esforço no processo ágil de software. Para isso, foram aplicadas diferentes redes neurais: *General Regression Neural Network* - GRNN, *Probabilistic Neural Network* - PNN, *Group Method of Data Handling* - GMDH, Redes Neurais Polinomiais e Redes Neurais de Correlação em Cascata. O conjunto de dados utilizado era composto por dados empíricos com informações de 21 projetos de software e considerou na construção do modelo, o número total de pontos de história e a velocidade desses projetos. A fim de validar o desempenho dos modelos, [Panda et al. 2015] calcula o erro quadrático médio - *MSE*, a magnitude media do erro relativo – *Mean Magnitude of Relative Error* – MMRE e a qualidade de previsão (*pred (n)*). Em seus resultados, [Panda et al. 2015] concluiu que o modelo de Rede Neural de Correlação em Cascata se destacou ao alcançar um erro quadrático de 93%.

O trabalho desenvolvido por Ayyıldız faz uso do conjunto de dados Desharnais [Desharnais 1989] para encontrar os atributos necessários que influênciam a estimativa de esforço de software, Além de analisar o real impacto desses atributos [Ercelebi Ayyıldız and Can Terzi 2017]. Neste estudo, Ayyıldız analisa a correlação de Pearson entre as métricas do conjunto de dados e o esforço de software, bem como a aplicabilidade da análise de regressão. Visando destacar expressamente as diferenças entre os valores reais e estimados da variável alvo, Ayyıldız produz ainda gráficos de dispersão e gráficos residuais. Em seus resultados alcançou um erro quadrático de 59.44%, bem como concluiu que a métrica *PointsNonAdjust* seria a mais determinante para a estimativa de esforço. *PointsNonAdjust é* o tamanho do projeto medido em pontos de função não ajustados e é calculado pela soma da métrica *Transactions* com a métrica *Entities*. Finalmente, a fim de avaliar o desempenho da previsão, Ayyıldız calcula a magnitude do erro relativo – *Mean Relative Error* – MRE, a magnitude media do erro relativo – *Mean Magnitude of Relative Error* – MMRE, a magnitude mediana do erro relativo – *Median of the Magnitude of*

Relative Error – MdMRE, o erro quadrático médio – *Mean Squared Error* – MSE e a qualidade de previsão (*pred (n)*).

O trabalho proposto por [Kocaguneli et al. 2010] constrói um modelo de estimativa de esforço para a divisão de desenvolvimento de software da subsidiária turca de um banco multinacional. Neste estudo, Kocaguneli construiu um modelo de estimativa de esforço baseado no aprendizado e validou o desempenho usando dados de projetos concluídos no conjunto de dados do banco, bem como conjuntos de dados públicos coletados de varias organizações. Kocaguneli também aplica várias técnicas de aprendizado de maquina para seleção de métricas, *clustering* e estimativas. Seus resultados mostram que combinações de técnicas de aprendizado de máquina proporcionam melhorias significativas na precisão das estimativas, e são validados através de testes estatísticos de forma a verificar a significância desses resultados. A aplicação de testes estatísticos mostrou que o *Linear Support Vector Machine é* o melhor entre os seis algoritmos aplicados em seu estudo, quando usado com a técnica de *clustering*, o desempenho do modelo aumenta significativamente, em media 25% para 68%, com relação a Pred(25) em três conjuntos de dados.

O trabalho de [Kitchenham et al. 2002] apresenta um conjunto de dados que permite investigar a precisão real das estimativas industriais e compara-las com estimativas produzidas a partir de vários modelos de estimativa de pontos de função. O estudo é baseado em dados de uma única empresa com um processo de estimativa específico (diretamente dependente das habilidades específicas da equipe de estimativas) e de um conjunto específico de clientes. O conjunto completo de dados corporativos, implementado no Microsoft Access com ferramentas adicionais no Word e no Excel, inclui mais de 145 projetos. No entanto, a análise dos projetos para os quais o esforço real, o esforço estimado e as contagens de pontos de função estavam disponíveis utilizou projetos datados entre os anos de 1994 e 1998. Em seus resultados [Kitchenham et al. 2002] constatou que as estimativas de esforço escolhidas como base para os orçamentos de projeto eram, em geral, razoavelmente boas, com 63% das estimativas estando dentro de 25% do valor real e um erro absoluto medio de 0,26. Essas estimativas foram significativamente melhores do que as estimativas de regressão baseadas em pontos de função ajustados, embora os modelos de pontos de função tenham sido baseados em um subconjunto homogêneo do conjunto de dados completo.

Dado que os dados coletados são apenas de uma empresa, a generalização dos resultados é limitada (i.e., ameaça a validade externa). Além disso, havia poucas evidencias de que a precisão das estimativas selecionadas se devesse ao fato de se tornarem os valores-alvo para os gerentes de projeto, assim, não se espera que nenhum dos modelos apresentados neste artigo se generalize automaticamente para outras situações de manutenção ou desenvolvimento.

Em nosso estudo, construímos e comparamos três diferentes modelos estatísticos, com o objetivo de analisar as estimativas alcançadas. Esses modelos tiveram seus resultados alinhados e comparados a fim de validar suposições sobre o conjunto de dados estudado. Os erros quadráticos foram analisados, bem como a correlação entre as métricas mais fortemente correlacionadas.

3. MATERIAIS E MÉTODOS

Para realizar este estudo, usamos o conjunto de dados Desharnais do PROMISE Software Engineering Repository. Este conjunto de dados consiste em 81 projetos de software de uma empresa de software canadense coletados por J. M. Desharnais [Desharnais 1989].

Primeiramente, analisamos a correlação entre cada atributo do conjunto de dados e o atributo de esforço, em seguida, aplicamos um algoritmo LR, seguido pelo algoritmo KNN e posteriormente o algoritmo SVM para investigar a relação entre os atributos mais fortemente relacionados pela correlação de Pearson com o esforço de software. Por fim, avaliamos nosso desempenho de previsão comparando o valor do erro quadrático dos algoritmos.

3.1. DATASET

O conjunto de dados Desharnais [Desharnais 1989] tem um total de 81 projetos desenvolvidos por uma empresa de software canadense em 1989. Cada projeto tem doze características que são descritas na Tabela 1.

Tabela 1. Definição de métricas para o conjunto de dados Desharnais

Métrica	**Descrição**
Project	ID do projeto iniciando em 1 e finalizando em 81.
TeamExp	Experiência do time mensurada em anos.
Manager Exp	Experiência do gerente de projetos mensurada em anos.
YearEnd	Ano de término do projeto.
Length	Duração do projeto em meses.
Effort	esforço atual mensurado em homens-hora.
Transactions	Total de transações lógicas no sistema.
Entities	Número de entidades de modelo do sistema.
PointsAdj	Tamanho do projeto mensurado em pontos de função não ajustados.
PointsNonAdjust	Tamanho do projeto mensurado em pontos de função ajustados.
Language	Linguagem utilizada no expressa categoricamente como 1, 2 ou 3.

Este conjunto de dados se tornou muito popular, visto seu uso frequente por muitos desenvolvedores a fim de treinar e avaliar modelos de estimativa de software como citado em [Kitchenham and Mendes 2009] e [Braga et al. 2007]. Em sua totalidade este conjunto de dados inclui nove atributos numéricos, entre os quais, no contexto desse estudo, oito são

chamados atributos independentes para a construção do modelo, são eles: “TeamExp”, “ManagerExp”, “YearEnd”, “Length”, “Transactions”, “Entities”, “PointsAdj”, e ”PointsNonAjust”. O atributo alvo “Effort”e mensurado em homens-hora.

3.2. PESQUISA REPRODUTÍVEL

A reprodutibilidade tem sido um dos princípios fundamentais do método científico e refere-se a capacidade de um teste ou experimento para ser reproduzido com precisão, ou replicado de forma independente. Em vez de apenas descrever os algoritmos desenvolvidos com precisão, nesse estudo nos damos aos leitores acesso a toda a informação (código, dados, esquemas, etc.) que foi usada para produzir os resultados apresentados como defendido por [Knuth 1984] e [Claerbout and Karrenbach 1992].

Nesse estudo utilizamos o conceito de pesquisa reprodutível porque um dos objetivos é fornecer aos leitores (e potencialmente usuários) versões da pesquisa que podem ser exploradas e os resultados reproduzidos no próprio computador do leitor.

Segundo [Donoho 2010], o trabalho de pesquisa diz-se reprodutível quando todos os seus artefatos de pesquisa, como texto, dados, figuras e código, estão disponíveis para pesquisadores independentes reproduzirem os resultados. Os dados compartilhados através de pesquisas reprodutíveis também fornecem melhor continuidade para pesquisas relacionadas e representam um ato de boa administração dos recursos públicos (financiamento de pesquisa), bem como a expansão do acesso a geração de conhecimento, como podemos ver em [Fomel 2015].

A reprodutibilidade de experimentos é essencial durante o processo de validação, portanto, todas as etapas do fluxo de trabalho analítico foram codificadas em Python e disponibilizadas para download em um repositório do Github - https://github.com/toniesteves/sw-effort-predictive-analysis. Uma descrição completa das etapas necessárias para baixar e executar o código para reproduzir essa análise na íntegra esta disponível no arquivo Readme na pagina inicial do repositório.

Além disso, reunir uma grande quantidade de dados para estudos empíricos pode ser uma tarefa complicada, propensa a introdução de erros não intencionais e, potencialmente, causar mais problemas do que resolver. A popularização e a ascensão do movimento de desenvolvimento de software de código aberto tem frequentemente disponibilizado dados que são uteis para fins de pesquisa. Consequentemente, podemos encontrar oportunidades na comunidade de pesquisa para obter dados para grandes amostras de projetos de software, e de forma integrada e estruturada, para que esses repositórios possam ser facilmente consultados para extrair informações.

Logo, em nosso estudo os dados brutos para a construção dos modelos são oriundos do PROMISE Software Engineering Repository, que é um repositório público de dados disponível a comunidade para a reprodutibilidade de estudos. Assim, como já colocado anteriormente, o link permanente para esses dados pode ser encontrado em: http://promise.site.uottawa.ca/SERepository.

Para a criação dos modelos utilizou-se a aplicação web de código aberto jupyter notebook. Essa aplicação é um ambiente de desenvolvimento voltado para programação em Python que permite leitores e pesquisadores interessados, criarem e avaliarem a

validade dos resultados sob diferentes pressupostos de dados e modelos. Desta forma, os resultados deste trabalho estão disponíveis não somente para pesquisadores experientes com um nível moderado de conhecimento na linguagem de computação Python, mas para qualquer leitor de interesse potencial, seja um estudante de pós-graduação, político ou proprietário na área. Assim, fornecemos todos os dados e resultados em: https://github.com/toniesteves/sw-effort-predictive-analysis.

3.3. SELEÇÃO DE CARACTERÍSTICAS

Como já citado, a correlação ou o coeficiente de correlação entre duas variáveis avalia a tendencia entre estas variáveis mudarem em função de seu relacionamento. Nosso estudo analisa quais atributos estão mais fortemente correlacionados e qual a influência destes atributos na estimativa do esforço de software para o conjunto de dados estudado. Como mencionado anteriormente em [Kocaguneli et al. 2010] e [Ercelebi Ayyıldız and Can Terzi 2017], podemos usar um conjunto de características ou métricas para validar e identificar essa correlação. Nesse contexto uma característica é uma propriedade mensurável individual do processo sob observação.

A métrica mais comum de correlação na estatística é o coeficiente de correlação de Pearson (ou Pearson Product Moment Correlation - PPMC) que mostra a relação linear entre duas variáveis [Rodgers and Nicewander 1988]. O coeficiente de correlação de Pearson é uma métrica estatística que mensura a força e direção de uma relação linear entre duas variáveis aleatórias [Rodgers and Nicewander 1988]. Tem sido comumente aplicado a vários índices estatísticos, como análise de dados e classificação [Ye 2014].

Os coeficientes de correlação de Pearson entre os atributos e o esforço de software mensurado para o conjunto de dados são mostrados na Figura 1.

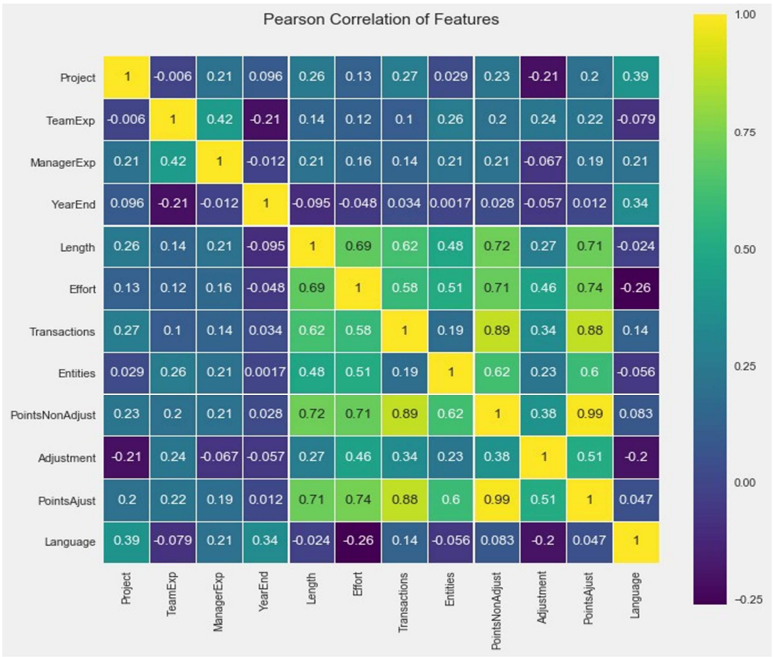


Figura 1. Correlação de Pearson no conjunto de dados Desharnais

Como pode ser visto na Figura 1, as métricas *Length*, *Transactions*, *Entities*, *PointsAdj* e *PointsNonAdjust* tem seus coeficientes de correlação dos atributos acima de 0.50. Como ja colocado anteriormente, quando os valores dos coeficientes de correlação são maiores que 0.50, há uma forte correlação entre variáveis dependentes e independentes. Dessa forma, podemos concluir que os atributos supracitados são estatisticamente os mais significantes para a construção do modelo.

3.4. CONSTRUÇÃO DOS MODELOS

Neste estudo, usamos os seguintes algoritmos: RL, SVM e KNN. Realizamos treinamento dos modelos de regressão utilizando 67% das instâncias do conjunto de dados, escolhidos aleatoriamente, e nas instâncias restantes, 33% do conjunto de dados que foram previamente isolados, utilizamos para testes do modelo. O treinamento dos modelos foi construído utilizando a linguagem Python, juntamente com as seguintes bibliotecas: Numpy, Pandas, Scikit-learn, Seaborn e Matplotlib. Durante o treinamento, foi necessário estimar o valor do parâmetros *random state* uma vez que o mesmo não era conhecido previamente.

A aplicação de uma análise de regressão, como já explicitado anteriormente, visa verificar a existência de uma relação funcional entre uma variável com uma ou mais variáveis, obtendo uma equação que explica a variação da variável dependente ou variável alvo *Y*, pela variação dos níveis das variáveis independentes. O treinamento do modelo utilizando Regressão Linear consiste em gerar uma regressão para a variável alvo *Y*.

Já o algoritmo *Linear Support Vector Machine é* utilizado amplamente no aprendizado de maquina. Comparado ao *Support Vector Machine*, que mapeia dados para um espaço de recurso dimensional mais alto, o *Linear Support Vector Machine* funciona diretamente no espaço de entrada original [Liu et al. 2017]. O método do *Support Vector Machine também* pode ser aplicado ao caso de regressão, mantendo todas as principais características que definem o algoritmo de margem máxima: uma máquina de aprendizado linear aprende uma função não linear em uma área de recurso induzida pelo kernel enquanto a capacidade do sistema é controlada por um parâmetro que não depende da dimensionalidade do espaço. Como no caso de classificação, o algoritmo de aprendizado minimiza um funcional convexo, e sua solução é esparsa [Nello Cristianini 2013].

Da mesma forma, o *K-Nearest Neighbor Regression* é um algoritmo simples que armazena todos os casos disponíveis e prediz o alvo numérico baseado em uma medida de similaridade, sendo aplicado na estimativa estatística e reconhecimento de padrões como técnica não paramétrica classificando corretamente pontos desconhecidos calculando distancia euclidiana entre esses pontos de dados [Buza et al. 2015].

Cabe ressaltar então, que nossa escolha pelo *K-Nearest Neighbor Regression* foi motivada dada a ausência de uma explicação detalhada sobre o valor do atributo de esforço no conjunto de dados Desharnais. No *K-Nearest Neighbor Regression*, optamos por especificar apenas três vizinhos para consultas de um número *k* que são ponderados igualmente.

4. RESULTADOS

Esta seção apresenta os resultados obtidos através dos métodos aplicados no conjunto de dados. O modelo de regressão linear, o modelo *Linear Support Vector Machine* e o modelo *K-Nearest Neighbor Regression* gerados tiveram seus desempenhos avaliados para demonstrar a precisão do modelo de regressão linear na estimativa do esforço de software.

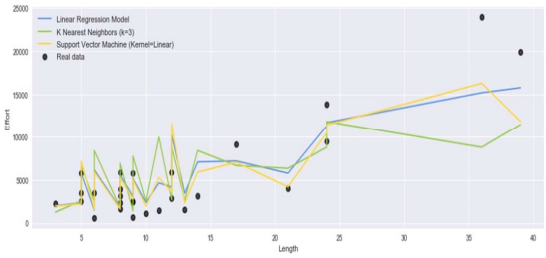
Com os modelos treinados, executamos os testes nas instancias de teste previamente isoladas. Assim, calculamos seus respectivos valores de R^2. A Tabela 2 mostra os coeficientes alcançados.

Tabela 2. Resultado dos modelos algorítmicos

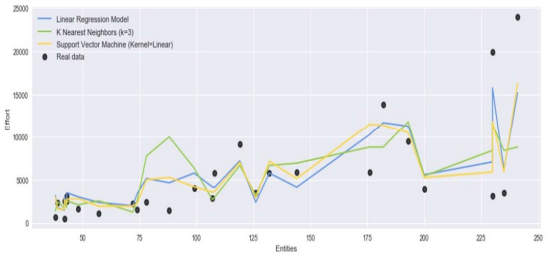
Modelos algorítmicos	**R^2 Score**
Modelo de Regressão Linear	0.7680074954440712
Support Vector Machine	0.735919788126071
K-Nearest Neighbor Regressor	0.7379861869550943

Como na regressão linear simples, o coeficiente de determinação (R^2 *Score*) deve ter valores entre 0 e 1, onde o valor 0 indica que a regressão é inexistente enquanto o valor 1 indica uma "perfeita" relação linear [Freund-Levi et al. 2006].

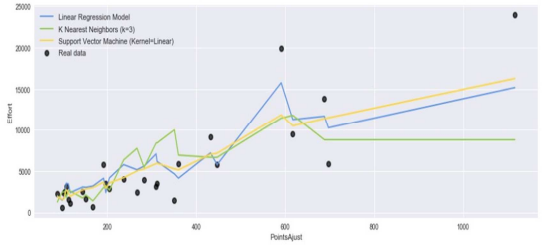
Ilustramos cada um dos recursos correlacionados na Figura 2. A figura mostra que a previsão do modelo linear (linha azul) é relativamente próxima da previsão de esforço do modelo KNN (linha verde) e também da previsão de esforço do modelo SVM (linha amarela), prevendo o objetivo numérico baseado em uma medida de similaridade.



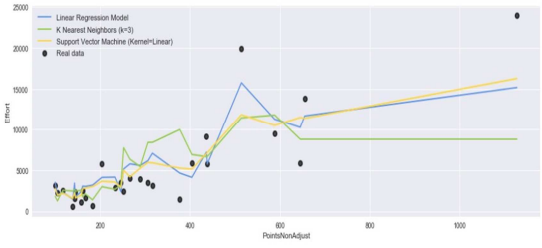
(a) *KNN* x *LR* x *SVR* - Length



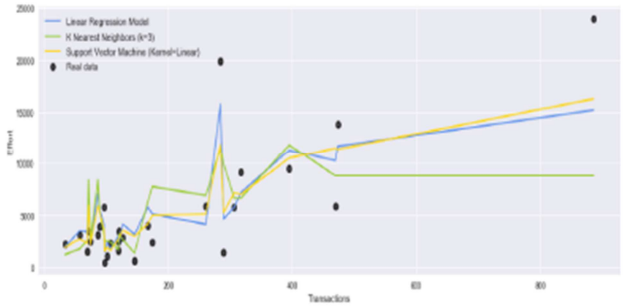
(b) *KNN* x *LR* x *SVR* – Entities



(c) *KNN* x *LR* x *SVR* – PointsAdjust



(d) *KNN* x *LR* x *SVR* – PointsNonAdjust



(e) *KNN* x *LR* x *SVR* – Transactions

Figura 2. Comparação dos erros quadráticos de K-neighbors Regression, Linear Regression e Linear Support Vector Machine

De acordo com o gráfico, observamos que o modelo de Regressão Linear (linha azul) apresenta um melhor desempenho. Embora a previsão de esforço do modelo de Regressão KNN (linha verde) e do modelo SVM (linha amarela) estejam relativamente próximas dos pontos de dados, o modelo Regressão Linear mostra um erro quadrático medio menor. É possível observar também que as arestas dos modelos apresentam uma leve tendência a subir nas imagens, o que justifica sua correlação com o aumento do esforço, além de *outliers* presentes na maioria das métricas.

5. CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho compara três possibilidades de modelos preditivos para a estimativa de esforço de software. Assim, suas estimativas são baseadas em dados de projetos existentes, mais precisamente sobre o conjunto de dados Desharmais. Nos atribuímos as contribuições deste trabalho ao o uso de três modelos algorítmicos que buscam tirar proveito das relações entre os valores-alvo do projeto. Esses métodos resultaram em modelos preditivos capazes de estimar os custos para as operações de estimativa de esforço de software.

A metodologia proposta consistiu na montagem dos modelos de Regressão Linear, *Linear Support Vector Machine* e *K-Nearest Neighbor Regression* apresentados neste trabalho. Assim os resultados obtidos com cada algoritmo foram comparados para avaliar a viabilidade de usar qualquer um desses modelos para estimar o esforço do software. Além desses modelos, também apresentamos a correlação entre suas métricas e dadas as métricas mais correlacionadas, como cada uma influência a variável alvo.

Nossos resultados obtiveram um erro quadrático superior a 70% e uma diferença de apenas 3% entre modelos construídos, enfatizando a viabilidade de se usar a regressão linear, entre outros, para prever o esforço de software. No entanto, para alcançar um resultado mais conciso e justo, é necessário reproduzir a mesma abordagem aplicando outros algoritmos disponíveis além de outras técnicas.

Como trabalho futuro, propomos o uso de uma base de projetos maior e mais recente de forma a diversificar e atribuir mais confiabilidade ao método. Explorar o uso de redes neurais para criação de um modelo preditivo também é um ponto merece destaque, dada sua capacidade adaptável e não paramétrica; Assim, os modelos criados sob essa abordagem poderiam se adaptar a um conjunto de dados específico. Outro ponto a

considerar é avaliar e comparar os modelos criados nesse estudo com a técnica de análise de pontos de função.

Referências

[Abran et al. 2015] Abran, A., Desharnais, J.-M., Zarour, M., and Demirors, O. (2015).¨ Productivity-based software estimation models and process improvement: an empirical study. *Int. J. Adv. Softw*, 8(1&2):103–114.

[Boslaugh and Watters 2008] Boslaugh, S. and Watters, P. (2008). *Statistics in a Nutshell: A Desktop Quick Reference*. In a Nutshell (O'Reilly). O'Reilly Media.

[Braga et al. 2007] Braga, P. L., Oliveira, A. L., and Meira, S. R. (2007). Software effort estimation using machine learning techniques with robust confidence intervals. In *Hybrid Intelligent Systems, 2007. HIS 2007. 7th International Conference on*, pages 352–357. IEEE.

[Buza et al. 2015] Buza, K., Nanopoulos, A., and Nagy, G. (2015). Nearest neighbor regression in the presence of bad hubs. *Knowledge-Based Systems*.

[Claerbout and Karrenbach 1992] Claerbout, J. F. and Karrenbach, M. (1992). Electronic documents give reproducible research a new meaning. In *SEG Technical Program Expanded Abstracts 1992*, pages 601–604. Society of Exploration Geophysicists.

[Dantas et al. 2018] Dantas, E., Perkusich, M., Dilorenzo, E., Santos, D. F., Almeida, H., and Perkusich, A. (2018). Effort estimation in agile software development: an updated review. pages 496–501.

[Desharnais 1989] Desharnais, J.-M. (1989). Analyse statistique de la productivitie des projets informatique a partie de la technique des point des fonction.

[Donoho 2010] Donoho, D. L. (2010). An invitation to reproducible computational research. *Biostatistics*, 11(3):385–388.

[Erc¸elebi Ayyıldız and Can Terzi 2017] Erc¸elebi Ayyıldız, T. and Can Terzi, H. (2017). Case study on software effort estimation. 7:103–107.

[Fomel 2015] Fomel, S. (2015). Reproducible research as a community effort: Lessons from the madagascar project. *Computing in Science & Engineering*, 17(1):20–26.

[Freund-Levi et al. 2006] Freund-Levi, Y., Eriksdotter-Jonhagen, M., Cederholm, T. E., Ba-¨ sun, H., Faxen-Irving, G., Garlind, A., Vedin, I., Vessby, B. O. H., Wahlund, L.-O., and Palmblad, J. (2006). Omega-3 fatty acid treatment in 174 patients with mild to moderate alzheimer disease: Omegad study: a randomized double-blind trial. *Archives of neurology*, 63 10:1402–8.

[Halkjelsvik and Jørgensen 2011] Halkjelsvik, T. and Jørgensen, M. (2011). From origami to software development: A review of studies on judgment-based predictions of performance time. 138:238–71.

[Huijgens et al. 2017] Huijgens, H., Van Deursen, A., Minku, L. L., and Lokan, C. (2017). Effort and cost in software engineering: A comparison of two industrial data sets. In *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*, pages 51–60. ACM.

[Kitchenham and Mendes 2009] Kitchenham, B. and Mendes, E. (2009). Why comparative effort prediction studies may be invalid. In *Proceedings of the 5th international Conference on Predictor Models in Software Engineering*, page 4. ACM.

[Kitchenham et al. 2002] Kitchenham, B., Pfleeger, S. L., McColl, B., and Eagan, S. (2002). An empirical study of maintenance and development estimation accuracy. *Journal of Systems and Software*, 64(1):57 – 77.

[Knuth 1984] Knuth, D. E. (1984). *The TEX Book*. Addison-Wesley, 15th edition.

[Kocaguneli et al. 2010] Kocaguneli, E., Tosun, A., and Bener, A. (2010). Ai-based models for software effort estimation. pages 323–326.

[Liu et al. 2017] Liu, D., Xie, S., Li, Y., Zhao, D., and El-Alfy, E. (2017). *Neural Information Processing: 24th International Conference, ICONIP 2017, Guangzhou, China, November 14-18, 2017, Proceedings*. Number pt. 1 in Lecture Notes in Computer Science. Springer International Publishing.

[Malhotra 2016] Malhotra, R. (2016). *Empirical Research in Software Engineering: Concepts, Analysis, and Applications*. CRC Press.

[Montgomery and Runger 2010] Montgomery, D. and Runger, G. (2010). *Applied Statistics and Probability for Engineers*. John Wiley & Sons.

[Nayebi et al. 2015] Nayebi, F., Abran, A., and Desharnais, J.-M. (2015). Automated selection of a software effort estimation model based on accuracy and uncertainty. *Artificial Intelligence Research*, 4(2):45.

[Nello Cristianini 2013] Nello Cristianini, J. S.-T. (2013). *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press.

[Panda et al. 2015] Panda, A., Satapathy, S. M., and Rath, S. K. (2015). Empirical validation of neural network models for agile software effort estimation based on story points. *Procedia Computer Science*, 57:772–781.

[Rodgers and Nicewander 1988] Rodgers, J. and Nicewander, W. (1988). Thirteen ways to look at the correlation coefficient. *The American Statistician*, 42(1):59–66.

[Sigweni et al. 2016] Sigweni, B., Shepperd, M., and Turchi, T. (2016). Realistic assessment of software effort estimation models. In *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering*, page 41. ACM.

[Trendowicz and Jeffery 2014] Trendowicz, A. and Jeffery, R. (2014). *Software Project Effort Estimation: Foundations and Best Practice Guidelines for Success*. Springer International Publishing.

[Wazlawick 2013] Wazlawick, R. (2013). *Engenharia de Software: Conceitos E Praticas*. Elsevier Editora Ltda.

[Ye 2014] Ye, J. (2014). Correlation coefficient of dual hesitant fuzzy sets and its application to multiple attribute decision making. 38:659–666.