UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
UNIDADE ACADÊMICA DE SISTEMAS E COMPUTAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

ANTONIO PEDRO DE ABREU NETO

ESTIMATIVA DE ESFORÇO PARA TAREFAS EM PROJETOS DE DESENVOLVIMENTO DE SOFTWARE

CAMPINA GRANDE - PB

2019

ANTONIO PEDRO DE ABREU NETO

ESTIMATIVA DE ESFORÇO PARA TAREFAS EM PROJETOS DE DESENVOLVIMENTO DE SOFTWARE

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Professor Dr. Claudio Elízio Calazans Campelo.

CAMPINA GRANDE - PB

2019

A162e Abreu Neto, Antonio Pedro de.
Estimativa de esforço para tarefas em projetos de desenvolvimento de software. / Antonio Pedro de Abreu Neto. – 2019.

9 f.

Orientador: Prof. Dr. Claudio Elízio Calazans Campelo.
Trabalho de Conclusão de Curso - Artigo (Curso de Bacharelado em Ciência da Computação) - Universidade Federal de Campina Grande; Centro de Engenharia Elétrica e Informática.

1. Desenvolvimento de software. 2. Previsão de esforço. 3. Duração de tasks. 4. Tarefas de desenvolvimento de software - planejamento. 5. Aprendizagem de máquina. 6. Floresta aleatória. 7. Regressão linear. 8. Árvore de regressão. I. Campelo, Claudio Elízio Calazans. II. Título.

CDU:004(045)

Elaboração da Ficha Catalográfica:

Johnny Rodrigues Barbosa
Bibliotecário-Documentalista
CRB-15/626

ANTONIO PEDRO DE ABREU NETO

ESTIMATIVA DE ESFORÇO PARA TAREFAS EM PROJETOS DE DESENVOLVIMENTO DE SOFTWARE

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA:

Professor Dr. Claudio Elízio Calazans Campelo
Orientador – UASC/CEEI/UFCG

Professor Dr. Carlos Wilson Dantas Almeida
Examinador – UASC/CEEI/UFCG

Professor Dr. Tiago Lima Massoni
Disciplina TCC – UASC/CEEI/UFCG

Trabalho aprovado em: 25 de novembro 2019.

CAMPINA GRANDE - PB

Estimativa de esforço para tarefas em projetos de desenvolvimento de *software*

Trabalho de Conclusão de Curso

Antônio Pedro de Abreu Neto (Aluno), Cláudio Campelo (Orientador)
Departamento de Sistemas e Computação
Universidade Federal de Campina Grande
Campina Grande, Paraíba - Brasil

RESUMO

Uma das dificuldades encontradas no desenvolvimento de *software* é conseguir predizer o tempo necessário para se concluir uma tarefa (*task*). Esta dificuldade pode gerar diversos problemas, tais como ciclos de desenvolvimento (*sprints*) sobrecarregados, com excesso de tarefas complexas para a equipe de desenvolvimento, ou o contrário, uma sprint subdimensionada. O objetivo desta pesquisa foi avaliar a viabilidade de utilizar modelos baseados aprendizagem de máquina para estimar automaticamente o tempo necessário para realizar tarefas de desenvolvimento de *software*. Foram realizados experimentos utilizando modelos baseados em Floresta Aleatória, Regressão Linear e Árvore de Regressão, utilizando dados reais de projetos de desenvolvimento de *software*. Os resultados obtidos foram encorajadores, especialmente com o modelo de regressão com Floresta Aleatória, que apresentou erros de previsão de aproximadamente uma hora.

KEYWORDS

Desenvolvimento de *software*, previsão de esforço, planejamento, duração de *tasks*.

1 INTRODUÇÃO

A estimativa é uma necessidade cada vez mais recorrente em várias áreas. Atualmente, há sistemas preditivos operando, por exemplo, para avaliação de desempenho de alunos [6] e no mercado de seguros, de forma a mensurar automaticamente um valor adequado de seguro com base em determinadas características de uma pessoa que podem estar relacionadas ao risco de utilização do seguro [15]. A vantagem no uso de métodos preditivos está em mostrar uma visão mais critica de uma determinada situação ou valor que podemos encontrar no futuro, apoiando o processo de tomada de decisão. Na área de engenharia de *software*, modelos baseados em Inteligência Artificial (IA) já foram propostos para diversas finalidades [14], como, por exemplo, para recomendação de requisitos não funcionais em User Stories [11] ou para aprimoramento dos processos de verificação e teste [7].

No processo de desenvolvimento de *software*, uma parte crucial são as entregas de artefatos que irão compor o produto final. Em geral, a produção de artefatos entregáveis consiste na execução de várias atividades distintas, de diversas naturezas (e.g., análise, modelagem, implementação) envolvendo mais de um membro da equipe. Porém, uma dificuldade frequente em times de desenvolvimento de *software* consiste em estimar o esforço (em tempo de desenvolvimento) para completar uma tarefa *task*.

A falha na mensuração pode ocorrer por diversos fatores externos e internos da equipe, tais como a falta de conhecimento do domínio do problema, a capacidade técnica dos membros da equipe, a negligenciação dos requisitos de qualidade ou a falta de entrosamento dos membros. Quando estes fatores são bem administrados e avaliados, a equipe tem uma alta chance de sucesso na entrega dos artefatos.

O apontamento de horas para *tasks* é um processo fundamental para o sucesso de um projeto de *software*, uma vez que fornece subsídios para inferir diversos dados sobre o projeto e sobre a equipe de desenvolvimento, tais como: produtividade do time, maturidade com as tecnologias, domínio do escopo do projeto, entre outros. Estimar o esforço das tarefas serve como porta de entrada para priorizar requisitos, definir velocidade do time e controlar do cronograma. Erros relacionados a esforço na etapa de criação das tarefas de desenvolvimento têm sido motivo do atraso na entrega ou encerramento de projetos [10].
[1]

As empresas estão cada vez mais interessadas em armazenar dados sobre seus projetos e seu time de desenvolvimento, o que deve permitir um avanço significativo nas técnicas baseadas em IA aplicadas à Engenharia de *software*, tais como modelos preditivos baseados em Aprendizagem de Máquina (AM). Alguns autores já têm relatado uma tendência em soluções baseadas em IA e AM para apoiar estimativas de esforço em projetos de *software* [5, 14]. Além disso, sabe-se que algumas empresas já possuem bases de dados de seus programadores e suas *tasks*, produzidas durante anos de desenvolvimento. Estas bases contêm dados extremamente úteis para o desenvolvimento de modelos preditivos. Dentre estes dados, citam-se o tempo estimado para concluir a tarefa, a tecnologia utilizada, a lista de programadores alocados para a tarefa. No entanto, o acesso a esses dados para realização de pesquisa científica ainda é muito restrito, tornando o cenário mais desafiador.

Este artigo apresenta o resultado de uma investigação para avaliar a viabilidade de se utilizar modelos baseados em AM para estimar automaticamente o tempo necessário para realizar tarefas de desenvolvimento de *software*. Foram realizados experimentos utilizando modelos baseados em Floresta Aleatória, Regressão Linear e Árvore de Regressão, utilizando dados reais de projetos de desenvolvimento de *software*.

O restante do artigo está estruturado da seguinte maneira. A próxima seção introduz alguns conceitos relevantes para o entendimento do trabalho. Em seguida, a Seção 3 descreve a metodologia utilizada para condução desta pesquisa. A Seção 4 descreve os resultados obtidos e, por fim, a Seção 5 conclui o artigo e aponta para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Esta seção apresenta a fundamentação teórica para o entendimento deste trabalho. Primeiro apresentamos informações sobre a estimativa de esforço para *tasks* em times de desenvolvimento utilizando Scrum [13]. Em seguida, será apresentada uma breve descrição dos algoritmos de AM utilizados.

2.1 Estimativa de Esforço para *Tasks*

Os times de desenvolvimento de *software* são geralmente guiados por uma determinada metodologia de desenvolvimento. Dentre as várias utilizadas, o Scrum [13] é uma metodologia usada para a gestão dinâmica de projetos, possuindo vários papéis que, se executados de forma eficiente, permite a entrega de um artefato com muita qualidade [3]. As várias etapas do Scrum permitem que o time de desenvolvimento tenha uma ideia aproximada de quanto tempo um determinado artefato levará para ser concluído, porém, esse processo pode falhar e acarretar um possível atraso na entrega, devido à natureza dinâmica do processo.

As *Tasks* podem ser vistas como pequenas partes de uma *User Story* (US). Estas *tasks* são definidas pela equipe de desenvolvimento ao início de uma *sprint*, sendo acompanhadas de uma estimativa de tempo para sua execução, que pode variar dependendo do escopo, da experiência do desenvolvedor, dentre outros fatores.

Na falta de uma base histórica, empresas podem utilizar alguns dos métodos ágeis citados acima, como o Julgamento de Especialista, que consiste no ato dos gerentes de projetos estimarem os valores para os projetos baseando-se em suas próprias experiências passadas [2]. O *Planning Poker*, por sua vez, é um método diferente por se tratar de um trabalho coletivo da equipe de desenvolvimento, que se apoia em uma definição consensual entre os membros da equipe de desenvolvimento. O processo é conduzido em grupo, de forma que a experiência coletiva possa ajudar a identificar detalhes que um indivíduo não identificaria sozinho, conforme discutido por Mahnic e Hovelja [9].

2.2 Algoritmos de AM utilizados

Os seguintes algoritmos foram considerados na implementação dos modelos utilizados nesta pesquisa: Regressão Linear, Árvore de Regressão e Floresta Aleatória. Estes algoritmos são descritos brevemente a seguir.

Regressão Linear: a regressão linear utiliza apenas a relação entre uma variável preditora (X) e uma variável alvo (Y). Porém, nos dados dos experimentos realizados, há mais de uma variável (Complexidade e Experiência), então nossa regressão linear é chamada de múltipla. Neste caso, ao invés de uma reta, tem-se uma curva. Conceitualmente, seria equivalente a fazer um gráfico de uma linha de regressão num espaço n-dimensional, ao invés de 2-dimensões. Sua resposta é uma função que tenta chegar próximo dos valores de Y. A regressão linear múltipla faz o uso de coeficientes de uma função para tentar explicar os valores de uma variável alvo (Y) (Equação 1).

$$y = b0 + b1x1 + b2x2 + ... + bkxk + e \quad (1)$$

Árvore de Regressão: uma árvore de regressão é idêntica a uma árvore de decisão porque também é formada por um conjunto de nós de decisão (i.e., perguntas). Porém, o resultado, ao invés de uma categoria, é um escalar. Ou seja, a variável resposta (Y) que a árvore de regressão trata é uma variável quantitativa (Equação 2).

$$Y \in \mathbb{R} \quad (2)$$

Floresta Aleatória: Floresta Aleatória é um algoritmo que trabalha com várias árvores culminando o resultado delas em um único resultado. Pode ser usado tanto para a classificação quanto para a regressão. O algoritmo de floresta aleatória adiciona aleatoriedade extra ao modelo, quando está criando as árvores. Ao invés de procurar pela melhor característica ao fazer a partição de nodos, ele busca a melhor característica em um subconjunto aleatório das características. Este processo cria uma grande diversidade, o que geralmente leva a geração de modelos melhores.

3 METODOLOGIA

Para a realização desta pesquisa, utilizamos uma base de dados estruturada de projetos de *software* e equipes de desenvolvimento do Laboratório de Sistemas Embarcados e Computação Pervasiva da Universidade Federal de Campina Grande. Os dados foram gentilmente cedidos pela organização. A base possui informações como: tarefas desenvolvidas, tempo total para desenvolver a tarefa, desenvolvedor que finalizou a tarefa, quantas tarefas um determinado desenvolvedor realizou, a pontuação média de suas tarefas terminadas, quantos projetos um desenvolvedor participou e papéis do Scrum que ele desempenhou. A base de dados é uma extensão da base de dados utilizada por Dantas et. al. [4], ilustrada na Figura 1.

A extração dos dados da base do laboratório foi feita em três etapas. Primeiro foi solicitado acesso ao banco de dados do sistema de gerenciamento de projetos utilizado pelo laboratório. Após a concessão do acesso, iniciamos a segunda etapa, que consistiu em uma inspeção no banco de dados visando compreender como os dados estavam estruturados e se era possível recuperar esses dados de forma automática. Após essa análise, percebemos que alguns dados (tais como o número de *commits* enviados para os repositórios dos projetos) não eram possíveis de serem recuperados de forma automática, pois o banco de dados não continha informações sobre os dados do GitLab (Sistema para versionamento de código utilizado pelo laboratório). Partimos então para a terceira etapa, que consistiu na extração dos dados. O banco de dados utilizado foi o MySQL 5.7, portanto a extração foi realizada através de um conjunto de consultas SQL. Os dados obtidos foram armanezados em formato CSV, conforme ilustrado na Figura 2.

Após o processo de extração, realizamos algumas transformações nos dados, de forma a produzir 4 atributos que serão utilizados pelos modelos de classificação, descritos a seguir:

- **Tarefa Mapeada**, que consiste na descrição da tarefa;
- **Esforço**, que é o atributo alvo da predição, mensurado em horas;

Tarefa_mapeada	Esforco_estimado	Esforco_real	Responsavel	Experiência
Criar campos Login e Senha	2	3	Desenvolvedor 1	Pleno
Fazer o roteamento da URL	2	3	Desenvolvedor 2	Pleno
Criar entidade no banco de dados	2	3	Desenvolvedor 3	Pleno
Persistir no banco os dados de acesso	2	2	Desenvolvedor 4	Senior
Criar tela para cadastro de nova senha	2	4	Desenvolvedor 5	Pleno
Fazer o roteamento da URL	2	4	Desenvolvedor 6	Pleno
Fazer o roteamento da URL	4	5	Desenvolvedor 7	Pleno
Criptografar senha no banco	1	1	Desenvolvedor 8	Senior
Criar entidade no banco de dados	1	1	Desenvolvedor 9	Senior
Criar tela para cadastro de nova senha	1	1	Desenvolvedor 10	Pleno
Fazer o roteamento da URL	1	1	Desenvolvedor 11	Pleno
Fazer o roteamento da URL	2	3	Desenvolvedor 12	Pleno
Criar tela alteração	2	2	Desenvolvedor 13	Pleno
Criar rota de recuperação de senha	2	4	Desenvolvedor 14	Pleno
Criar serviço alteração	1	1	Desenvolvedor 15	Pleno
Alterar tela de login	2	1	Desenvolvedor 16	Pleno
Criar tela para cadastro de nova senha	2	3	Desenvolvedor 17	Pleno
Criar rota de recuperação de senha	4	4	Desenvolvedor 18	Senior
Configurar API	3	3.5	Desenvolvedor 19	Pleno
Criar serviço de notificação	1	2	Desenvolvedor 20	Pleno
Criar rota de recuperação de senha	2	2	Desenvolvedor 21	Pleno
Criar tela inserção	4	8	Desenvolvedor 22	Pleno
Criar tela inserção	4	3	Desenvolvedor 23	Pleno
Criar entidade no banco de dados	1	1	Desenvolvedor 24	Pleno
Criar rota de inserção de dados	1	1	Desenvolvedor 25	Pleno
Criar rota de inserção de dados	1	1	Desenvolvedor 26	Pleno
Criar serviço alteração	4	6	Desenvolvedor 27	Pleno
Criar tela alteração	4	4	Desenvolvedor 28	Pleno
Modificar rota de inserção de dados	2	2	Desenvolvedor 29	Pleno
Criar rotas para remoção de dados	1	1	Desenvolvedor 30	Pleno
Criar tela listagem	4	5	Desenvolvedor 31	Pleno
Modificar tela inserção	1	1.1	Desenvolvedor 32	Pleno
Criar rotas para listagem		1	Desenvolvedor 33	Pleno

Figura 1: Exemplo de uma parte da base de dados produzida por Dantas et. al. [4], que contém as informações das *tasks*.

Recurso Humano	Qtde-projetos	Qtde-tarefas	Qtde-tarefas-DONE	Média-pontos-tarefas	Scrum_Master
Desenvolvedor 1	2	45	45	0.48	0
Desenvolvedor 2	1	51	46	0.13	0
Desenvolvedor 3	1	39	35	0.2	0
Desenvolvedor 4	3	64	59	0	1
Desenvolvedor 5	1	8	6	0	0
Desenvolvedor 6	2	101	99	0.9	1
Desenvolvedor 7	3	102	45	1.03	0
Desenvolvedor 8	1	37	37	1.4	0
Desenvolvedor 9	1	52	49	1.18	1
Desenvolvedor 10	1	51	50	1.08	0
Desenvolvedor 11	7	5	5	1	0
Desenvolvedor 12	2	148	144	1.05	1
Desenvolvedor 13	5	9	9	0	0
Desenvolvedor 14	1	NA	NA	NA	0
Desenvolvedor 15	3	34	34	0	0
Desenvolvedor 16	2	70	68	1.33	0
Desenvolvedor 17	3	71	32	1.42	0
Desenvolvedor 18	2	72	60	1.25	1
Desenvolvedor 19	2	121	118	1.33	1
Desenvolvedor 20	3	22	16	2.22	1
Desenvolvedor 21	1	NA	NA	NA	0
Desenvolvedor 22	2	2	2	2.5	0
Desenvolvedor 23	NA	NA	NA	NA	0
Desenvolvedor 24	1	NA	NA	NA	0
Desenvolvedor 25	1	NA	NA	NA	0

Figura 2: Exemplo de uma parte da base de dados que contém as informações dos desenvolvedores

- **Experiência**, que consiste em um valor de 0 a 10, gerado através de vários atributos do desenvolvedor (descritos mais tarde nesta seção);
- **Complexidade da *Task***, que foi gerada através da avaliação dos próprios programadores ao finalizarem a *task*.

A experiência foi gerada a partir dos seguintes atributos originais:

- Quantidade de projetos que o desenvolvedor já participou;
- Quantidade de tarefas já realizdas;
- Médias dos pontos das tarefas realizadas;
- Papéis desempenhados no *Scrum*.

Entre os dados extraídos do sistema do laboratório, temos a pontuação das *tasks*, que representa um consenso da equipe sobre o grau de dificuldade para se completar uma determinada *task*. Através dela, inferimos características sobre o programador. Por exemplo, programadores com uma média de pontos de *tasks* alta, provavelmente são programadores experientes. Além disso, conseguimos também extrair a quantidade de projetos em que os programadores participaram. Com esse dado, conseguimos chegar a algumas conclusões positivas sobre o programador, ou seja, quanto mais projetos um determinado programador participou, provavelmente mais experiente ele é. Como o laboratório trabalha com a metodologia *SCRUM*, verificamos ainda os papéis que o programador realizou em sua equipe. Neste caso, se o programador foi *Scrum Master* durante sua jornada na equipe de desenvolvimento, ele foi considerado mais experiente.

Com o objetivo de realizar um mapeamento adequado, foi realizada uma investigação sobre quais medidas poderiam ser computadas para definir esses *Cost Drivers* e diminuir o nível de abstração. Várias abordagens de avaliação de medidas de desempenho da equipe de desenvolvimento são apresentadas na literatura, como as propostas por kitchenham [8] e Schneidewind [12]. Neste trabalho seguimos a abordagem de Antinyan et al. [1] que consiste em um método empírico complementar que se baseia no princípio de pesquisa ação que, nada mais é do que é uma forma de investigação baseada em uma autorreflexão coletiva, e deve ser usado em combinação com métodos de validação teórica. A escolha desta abordagem se deu pela sua maior simplicidade em relação a [8, 12] e já possuímos os especialistas (gerentes de projetos) disponiveis para fazer a validação. Assim, para identificar quais medidas são importantes para as variáveis de interesse, realizamos uma validação empírica por ciclos de pesquisa ação.

A coluna Experiência é gerada a partir da soma dos resultados de cada um dos limiares abaixo, definidos através do processo adotado para avaliação de medidas de desempenho.

A nota relacionada à quantidade (q) de projetos que o desenvolvedor já participou é definida da seguinte forma:

- Se $q > 5$, atribui-se nota 3;
- Se $2 \leqslant q \leqslant 4$, atribui-se nota 2.
- Se $q = 1$, atribui-se nota 1.
- Se nao participou de nenhum projeto, atribui-se nota 0.

A nota relacionada à quantidade (q) de tarefas já realizadas é definida da seguinte forma:

- Se $q > 80$, atribui-se nota 3.
- Se $30 \leqslant q \leqslant 80$, atribui-se nota 2;
- Se $q < 30$, atribui-se nota 1.

A nota relacionada à média (m) dos pontos das tarefas realizadas é definida da seguinte maneira:

- Se $m > 2$, atribui-se nota 2;
- Se $1 \leqslant m \leqslant 2$, atribui-se nota 1;
- Se $m < 1$, atribui-se nota 1.

Para os papéis desempenhados no *Scrum*:

- Se entre os papéis desempenhados no *Scrum*, há algum período como *Scrum Master*, atribui-se nota 2.

A coluna da Complexidade, por sua vez, foi gerada através da análise dos próprios programadores, sobre a dimensão da *task* que ele foi alocado, levando em consideração:

- A quantidade de classes criadas;
- A comunicação com outro sistema;
- Os fatores externos ao código.

Após a criação da base, foi necessário realizar um processo de ocultação de dados dos nomes de desenvolvedores, por questões de privacidade, e chegamos a um *data frame*, ou seja, um arquivo estruturado como uma matriz com colunas que representam os dados. Este é o formato de entrada de dados requerido pela biblioteca utilizada para treinamento dos algoritmos de predição, a Scikit-learn[2]. É importante mencionar ainda que todo o código produzido nesta pesquisa foi escrito em linguagem de programação Python 3, utilizando a plataforma compartilhada do Google Colab[3].

Antes de chegarmos no *data frame* final, mostrado na Figura 3, foi necessário fazer uma varredura pelo data frame mostrado na Figura 2, procurando por valores errados ou faltantes, para serem corrigidos ou desconsiderados, assim não prejudicando o processo de predição. Foram encontrados valores faltantes de alguns desenvolvedores e, como não eram possíveis de serem recuperados, foram descartados do *data frame*.

O *data frame* gerado após o processamento descrito possui um total de 871 *tasks*, realizadas em projetos Web com tecnologias para *front-end* e *back-end* como AngularJs, Angular2+, React, NodeJs e SpringBoot. As tasks estão datadas entre os anos de 2016 e 2019. Durante esses anos, vários projetos foram finalizados e outros foram iniciados em sequência, mas sempre utilizando as tecnologias citadas. Portanto, podemos dizer que nossa base para a estimativa de esforço de *tasks* é voltada para *tasks* do domínio Web com tecnologias como angularJs, angular2+, React, nodeJs e SpringBoot. Ou seja, é possível que nossos modelos sejam mais eficazes para predizer esforço em projetos de desenvolvimento desta natureza.

Para iniciar o processo de treinamento, dividimos o *data frame* (Figura 3) em dois conjuntos, um para treino e um para teste. Como se trata de uma base de *tasks* que não possuem data de criação ou qualquer outro parâmetro relevante para decidir se uma *task* ficaria no lado do treino ou no lado do teste, dividimos de forma aleatória. A base foi repartida em $\frac{3}{4}$ para o treino e $\frac{1}{4}$ para o teste, ou seja, 653 *tasks* para o treino e 218 *tasks* para os testes. Depois da divisão, utilizamos os algoritmos da biblioteca do scikit-learn para criação dos modelos preditivos.

Como o atributo alvo (Esforço) é um atributo numérico e contínuo, não podemos utilizar mecanismos de classificação como uma árvore de decisão. Portanto, optamos por usar métodos de regressão. Os dados serão submetidos a modelos preditivos gerados por cada um destes algoritmos:

- Regressão Linear.
- Árvore de Regressão.
- Floresta Aleatória.

No processo de avaliação experimental, avaliamos os modelos com o objetivo de identificar o que possui o menor erro associado, visando uma melhor explicação dos resultados.

[2]Scikit-learn: Machine Learning in Python. https://scikit-learn.org/stable/
[3]Google Colab - https://colab.research.google.com

Tarefa_mapeada	Esforco	Experiência	Complexidade
Criar campos Login e Senha	3	6	3
Fazer o roteamento da URL	3	5	3
Criar entidade no banco de dados	3	4	3
Persistir no banco os dados de acesso	2	8	3
Criar tela para cadastro de nova senha	4	3	3
Fazer o roteamento da URL	4	3	3
Fazer o roteamento da URL	5	7	3
Criptografar senha no banco	1	8	3
Criar entidade no banco de dados	1	8	3
Criar tela para cadastro de nova senha	2	7	3
Fazer o roteamento da URL	1	5	3
Fazer o roteamento da URL	3	4	3
Criar tela alteração	2	7	7
Criar rota de recuperação de senha	4	4	7
Criar serviço alteração	1	5	4
Alterar tela de login	1	6	3
Criar tela para cadastro de nova senha	3	6	3
Criar rota de recuperação de senha	4	8	7
Configurar API	3.5	7	5
Criar serviço de notificação	2	5	5
Criar rota de recuperação de senha	2	4	7
Criar tela inserção	8	3	7
Criar tela inserção	3	6	4

Figura 3: Exemplo de uma parte do data frame que contém os dados da experiência e complexidade

4 RESULTADOS

Esta seção apresenta os resultados da comparação entre os modelos preditivos de cada um dos algoritmos (i.e., Árvore de Regressão, Regressão Linear, Floresta Aleatória). Avaliamos a qualidade dos modelos em termos de três diferentes métricas para identificação do erro associado: *Mean Squared Error* (MSE), *Mean Absolute Error* (MAE) e *Root Mean Squared Error* (RMSE), que são métricas condideradas adequadas para modelos de predição por regressão. Ou seja, através dessas métricas, conseguimos compreender como nossos modelos se comportam de acordo com o *data frame* ao qual ele foi submetido.

Para um entendimento mais completo sobre os resultados obtidos, é importante explicar brevemente cada uma dessas métricas:

Mean Squared Error (MSE): Esta métrica calcula a média dos erros do modelo ao quadrado, conforme definido na Equação s3). Ou seja, diferenças menores recebem menos importância, enquanto diferenças maiores recebem mais peso no cálculo desta métrica.

$$MSE = \frac{1}{n}\sum_{t=1}^{n} e_t^2 \tag{3}$$

Root Mean Squared Error (RMSE): Esta métrica é uma variação do MSE citado a cima. Esta representa a raiz quadrada do MSE, conforme definido nas Equações 4 e 5). Neste caso, o erro volta a ter as unidades de medida originais da variável dependentes. Desta forma, torna-se a explicação do erro relacionado ao modelo muito mais simples, por se tratar de um valor na mesma escala da variável alvo.

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2} \tag{4}$$

ou

$$RMSE = \sqrt{MSE} \tag{5}$$

Mean Absolute Error (MAE): Esta é uma métrica parecida com a MSE, porém, em vez de elevar a diferença entre a previsão do modelo, e o valor real, ao quadrado, ela toma o valor absoluto, conforme definido na Equação 6). Neste caso, em vez de atribuir um peso de acordo com a magnitude da diferença, ele atribui o mesmo peso a todas as diferenças, de maneira linear.

$$MAE = \frac{1}{n}\sum_{t=1}^{n}|e_t| \tag{6}$$

A métrica que mais facilita a explicação dos resultados é a *Root Mean Squared Error* (RMSE), por se tratar de um valor que está na mesma escala dos dados que serão preditos. Por exemplo, em um modelo de predição de preços de imóveis com escala de mil reais, um erro de 2 significa que o modelo consegue predizer valores de imóveis com um erro associado de 2 mil reais para mais ou para menos em relação ao valor predito. No contexto desta pesquisa, onde o objetivo é prever o esforço (i.e., o tempo) necessário para completar uma *task*, o valor avaliado é em horas.

O resultado da execução dos modelos preditivos estão sumarizados nos gráficos das Figuras 4, 5 e 6. Este gráficos representam, respectivamente, os valores de erro obtidos pelas métricas RMSE, MSE e MAE.

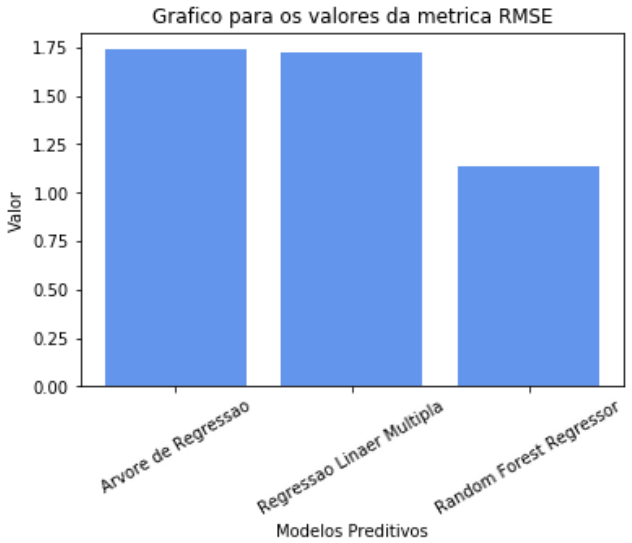

Figura 4: Gráfico da métrica RMSE para os modelos de predição Regressão Linear, Árvore de Regressão e Floresta Aleatória.

Os resultados indicam que a Floresta Aleatória é o modelo mais adequado para este tipo de previsao. Para a base de dados utilizada neste trabalho, este modelo resultou no menor erro associado, ou seja, os valores previstos para duração das *tasks* de desenvolvimento

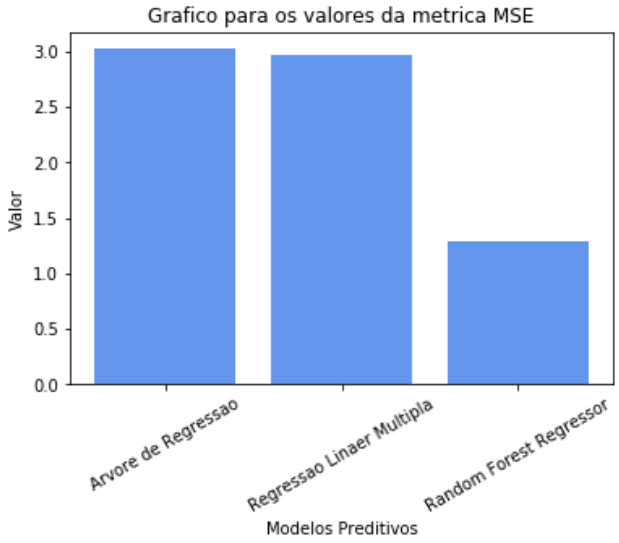

Figura 5: gráfico da métrica MSE para os modelos de predição Regressão Linear, Árvore de Regressão e Floresta Aleatória.

de *software* tiveram um erro médio associado de aproximadamente uma hora para mais ou uma hora para menos.

Por outro lado, os outros dois modelos avaliados (regressão linear e árvore de regressão) produziram valores de predição bem próximos entre si, porém muito distantes do modelo de Floresta Aleatória. De fato, este algoritmo tem se mostrado bastante poderoso em problemas similares. A Floresta Aleatória trabalha com várias árvores e, ao final de cada uma das árvores, o resultado ser somado em um novo resultado, com uma aleatoriedade extra adicionada pelo próprio modelo, aumentando as chances de obtenção de bons resultados de previsão. No nosso estudo utilizamos um parâmetro do scikit-learn chamado de n estimators com o valor de 100, o que significa que foi criado uma combinação aleatória de 100 árvores.

5 CONCLUSÃO E TRABALHOS FUTUROS

Este artigo apresentou uma abordagem para predição de duração de *tasks* em projetos de desenvolvimento de *software*. Foram executados experimentos envolvendo 3 modelos diferentes de regressão, utilizando dados de projetos reais utilizando tecnologias de desenvolvimento de sistemas para a Web.

Pode-se concluir que, mesmo os algoritmos de Árvore de Regressão e Regressão Linear tendo apresentado valores de predição distantes do algoritmo de Floresta Aleatória, ainda assim tiveram um rendimento aceitável, com erros de aproximadamente duas horas (para ou ou para menos), segundo a métrica RMSE, conforme exibido no gráfico da Figura 4. Podemos atribuir esse erro à falta de acesso à dados que poderiam ser adicionados ao *data frame*, mas que por limitações no acesso aos dados, ou pela indisponibilidade desses, não conseguimos levar esse estudo a um nível de predição mais apurado.

Um tipo de dado de extrema importância que pode ser considerado em trabalhos futuros é relacionado às tecnologias utilizadas na realização das tarefas. Por exemplo, na base de dados utilizada nesta

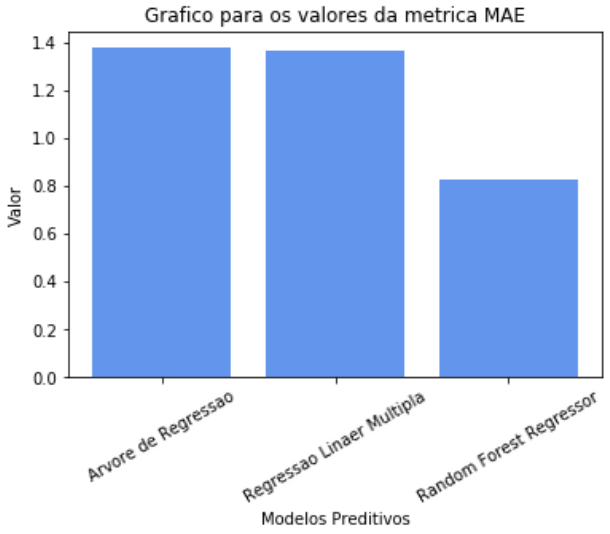

Figura 6: gráfico da métrica MAE para os modelos de predição Regressão Linear, Árvore de Regressão e Floresta Aleatória.

pesquisa, há tarefas que são praticamente iguais em sua essência, porém, executada com frameworks diferentes ou banco de dados diferentes. Esses fatores podem fazer muita diferença na velocidade de execução de uma tarefa e, portanto, podem ser decisivos para a acurácia de um preditor. Outras características como a experiência prévia do programador (antes de entrar na empresa) podem ser relevantes, especialmente para os programadores com pouco tempo na empresa e, portanto, com pouco dados associados a eles no banco de dados.

Um aspecto positivo neste trabalho é que conseguimos refinar bastante o entendimento da experiência dos programadores, através do levantamento de informações sobre fatores técnicos associados a eles. Adicionalmente, apesar das limitações dos dados adquiridos, conseguimos um nível de erro reduzido (de aproximadamente uma hora), considerando o algoritmo da Floresta Aleatória.

Embora possamos visualizar a possibilidade de explorar uma grande quantidade de outros tipos de dados para melhorar nossos modelos preditivos, percebe-se que há grande dificuldade de se conseguir bases de dados com tal nível de detalhamento, pois muitas dessas informações ainda não foram estruturadas em massa para disponibilização em domínio público.

REFERÊNCIAS

[1] Vard Antinyan, Miroslaw Staron, Anna Sandberg, and Jörgen Hansson. 2016. Validating software measures using action research a method and industrial experiences. In *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering*. ACM, 23.

[2] Monalessa Perini Barcellos, SM Figueiredo, ARC Rocha, and GH Travassos. 2003. Utilização de Métodos Paramétricos, Analogias, Julgamento de Especialistas e Conhecimento Organizacional no Planejamento de Tempo e Custos de Projetos de Software. *Anais do II Simpósio Brasileiro de Qualidade de Software* (2003), 17–31.

[3] Bernardo Vasconcelos de Carvalho and Carlos Henrique Pereira Mello. 2012. Aplicação do método ágil scrum no desenvolvimento de produtos de software em uma pequenaempresa de base tecnológica. *Gestão Produção* 19 (00 2012), 557 – 573.

[4] Emanuel Dantas, Alexandre Costa, Marcus Vinicius, Mirko Perkusich, Hyggo Almeida, and Angelo Perkusich. 2019. An Effort Estimation Support Tool for Agile Software Development: An Empirical Evaluation. 82–87. https://doi.org/10.18293/SEKE2019-141

[5] Emanuel Dantas, Mirko Perkusich, Ednaldo Dilorenzo, Danilo Santos, Hyggo Almeida, and Angelo Perkusich. 2018. Effort Estimation in Agile Software Development: An Updated Review. *International Journal of Software Engineering and Knowledge Engineering* 28 (11 2018), 1811–1831. https://doi.org/10.1142/S0218194018400302

[6] Reginaldo Gotardo e Paulo Roberto Cereda e Estevam Rafael Hruschka Junior. 2013. Predição do Desempenho do Aluno usando Sistemas de Recomendação e Acoplamento de Classificadores. *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação - SBIE)* 24, 1 (2013), 657.

[7] Frederico Alexandre Ferreira. 2017. *Inteligência artificial na verificação e teste de software para desenvolvimento ágil.* Ph.D. Dissertation. Instituto Superior de Engenharia de Lisboa.

[8] Barbara Kitchenham, Shari Lawrence Pfleeger, and Norman Fenton. 1995. Towards a framework for software measurement validation. *IEEE Transactions on software Engineering* 21, 12 (1995), 929–944.

[9] Viljan Mahnič and Tomaž Hovelja. 2012. On using planning poker for estimating user stories. *Journal of Systems and Software* 85, 9 (2012), 2086 – 2095. https://doi.org/10.1016/j.jss.2012.04.005 Selected papers from the 2011 Joint Working IEEE/IFIP Conference on Software Architecture (WICSA 2011).

[10] Bett Obadia. 2018. Why Projects Fail. *Project Management Journal* Vol 1 (07 2018), 7.

[11] Felipe Ramos, Antonio Pedro, Marcos Cesar, Alexandre Costa, Mirko Perkusich, Hyggo Almeida, and Angelo Perkusich. 2019. Evaluating Software Developers' Acceptance of a Tool for Supporting Agile Non-Functional Requirement Elicitation. 26–31. https://doi.org/10.18293/SEKE2019-107

[12] Norman F. Schneidewind. 1992. Methodology for validating software metrics. *IEEE Transactions on software engineering* 18, 5 (1992), 410–422.

[13] Ken Schwaber and Mike Beedle. 2002. *Agile software development with Scrum.* Vol. 1. Prentice Hall Upper Saddle River.

[14] Renato Afonso Cota Silva. 2005. Inteligência artificial aplicada a ambientes de engenharia de software: Uma visão geral. *INFOCOMP Journal of Computer Science* 4, 4 (2005), 27–37.

[15] Yi Yang, Wei Qian, and Hui Zou. 2018. Insurance premium prediction via gradient tree-boosted Tweedie compound Poisson models. *Journal of Business & Economic Statistics* 36, 3 (2018), 456–470.