UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO

Eça da Rocha Fernandes Neto

Uma Abordagem Baseada em Aprendizado de Máquina para Dimensionamento de Requisitos de Software

Recife

2016

Eça da Rocha Fernandes Neto

Uma Abordagem Baseada em Aprendizado de Máquina para Dimensionamento de Requisitos de Software

Monografia apresentada ao Curso de Bacharelado em Sistemas de Informação da Universidade Federal Rural de Pernambuco, como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação.

Universidade Federal Rural de Pernambuco – UFRPE

Departamento de Estatística e Informática

Curso de Bacharelado em Sistemas de Informação

Orientador: Rodrigo Soares

Recife

2016

Dados Internacionais de Catalogação na Publicação
Universidade Federal Rural de Pernambuco
Sistema Integrado de Bibliotecas
Gerada automaticamente, mediante os dados fornecidos pelo(a) autor(a)

F363Neto a

Fernandes Neto, Eça da Rocha
Uma Abordagem Baseada e Aprendizado de Máquina para Dimensionamento de Requisitos de Software / Eça da Rocha Fernandes Neto. - 2016.
48 f. : il.

Orientador: Rodrigo Soares.
Inclui referências.

Trabalho de Conclusão de Curso (Graduação) - Universidade Federal Rural de Pernambuco, Bacharelado em Sistemas da Informação, Recife, 2016.

1. dimensionamento de requisitos. 2. aprendizado de máquina. 3. machine learning. 4. estimativa. 5. requisitos de software. I. Soares, Rodrigo, orient. II. Título

CDD 004

EÇA DA ROCHA FERNANDES NETO

Uma Abordagem Baseada em Aprendizado de Máquina para Dimensionamento de Requisitos de Software

Monografia apresentada ao Curso de Bacharelado em Sistemas de Informação da Universidade Federal Rural de Pernambuco, como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação.

Aprovada em: 13 de dezembro de 2016.

BANCA EXAMINADORA

Rodrigo Soares (Orientador)
Departamento de Estatística e Informática
Universidade Federal Rural de Pernambuco

Cícero Garrozi (Membro da banca)
Departamento de Estatística e Informática
Universidade Federal Rural de Pernambuco

À Rafa e Rique.
Aos meus pais e minhas irmãs.
À toda a minha família.
Aos meus amigos, todos eles.
Obrigado pelo apoio e confiança sempre.

Agradecimentos

Agradeço à meus pais, minhas irmãs e toda a minha família por me proporcionarem as melhores oportunidades e por sempre me apoiarem em tudo. Amo vocês demais.

Rafa, você é meu porto seguro, obrigado por toda a força e amor que sempre me deu, precisasse eu ou não. Henrique, obrigado por tudo que me da (ou me pede) sempre, ser seu pai é um aprendizado diário e uma benção eterna. Amo vocês dois do fundo do meu coração.

Agradeço ao meu orientador Rodrigo, por me guiar durante todo esse trabalho, ele é tão seu quanto meu. Heron e Maurício, seus insights foram essenciais, não teria nem começado sem vocês. Ana, você sabe o quão importante foi na minha vida, nunca agradecerei o suficiente por ter acreditado em mim e me dado apoio nessa trajetória. Obrigado também a todos que compõem a UFRPE, professores, funcionários, alunos e animais.

Não poderia esquecer dos amigos que fiz pelo caminho e com certeza levarei pra onde for, Rê, Sandro, Flavi, Erick, Thays, Delando, Emeson e a todos da SWquality.

Um especial a Victor, Mari e Matheus, nem sei dizer o quanto me ajudaram sempre que precisei. Rafael, não poderia deixar você de fora, enorme apoio nesse finalzinho, tamo junto!

Vovó Deça e vovô Eça, vovó Mana e vô Luiz, amo vocês! Tia Béia, obrigado por ter me proporcionado o start nessa vida de TI, lá por 97 ou 98, nunca esquecerei.

Sei que são muitos, então, a todos que de alguma forma contribuíram e eu tenha esquecido de citar, obrigado.

”A vida é a arte do encontro, embora haja tanto desencontro pela vida.”
(Vinicius de Moraes)

Resumo

Este trabalho se propõe a realizar o dimensionamento automático de requisitos de software utilizando uma abordagem de aprendizado de máquina. A base de dados utilizada é real e foi obtida de uma empresa que trabalha com processo de desenvolvimento baseado no Scrum e estimativa Planning Poker. Durante os estudos foram utilizadas técnicas de pré processamento de dados, classificação e seleção de melhores atributos com os algorítimos termo–inverso da frequência nos documentos (tf-idf) e análise de componentes principais (PCA). O aprendizado de máquina e classificação automática se deu com o uso de Máquinas de Vetores de Suporte (SVM) baseado no histórico de dados disponível. Os testes finais foram realizados com e sem a seleção de atributos via PCA. Está demonstrado que a assertividade é maior quando é feita a seleção dos melhores atributos. A ferramenta fruto do trabalho consegue estimar o tamanho de histórias de usuário com uma generalização de até 91%. Os resultados foram considerados passíveis de serem utilizados em ambiente de produção sem prejuízo para a equipe de desenvolvimento.

Palavras-chave: dimensionamento, estimativa, aprendizado de máquina, inteligência artificial, tf-idf, pca, scrum, planning poker, automatização.

Abstract

This work proposes to perform the automatic sizing of software requirements using a machine learning approach. The database used is real and was obtained from a company that works with Scrum-based development process and Planning Poker estimation. During the studies, data pre-processing, classification and selection of best attributes were used along with the term frequency–inverse document frequency algorithm (tf-idf) and principal component analysis (PCA). Machine learning and automatic sorting occurred with the use of Support Vector Machines (SVM) based on available data history. The final tests were performed with and without attribute selection by PCA. It is demonstrated that the assertiveness is greater when the best attributes are selected. The final tool can estimate the size of user stories with a generalization of up to 91 %. The results were considered likely to be used in the production environment without any problems to the development team.

Keywords:sizing, estimate, machine learning, artificial intelligence, tf-idf, pca, scrum, planning poker, automation.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

TI	Tecnologia de Informação
PCA	Principal Component Analysis
NLTK	Natural Language Toolkit
SVM	Maquinas de Vetores de Suporte
TF-IDF	Term frequency–inverse document frequency
HU	Histórias de Usuário
PO	Product Owner
SM	Scrum Master
AM	Aprendizado de Máquina
IA	Inteligência Artificial
SGBD	Sistema Gerenciador de Banco de Dados

Sumário

1 Introdução

Em ambientes de desenvolvimento é sempre útil ter algum conceito que meça tamanho dos requisitos de um determinado *software*. Esse número é útil na avaliação de custos e prazos de mudanças de requisitos, tarefas de desenvolvimento, manutenções e outras medições de planejamento e acompanhamento de projeto (SWEBOK, 2004).

O planejamento é uma etapa crucial para o sucesso de qualquer projeto pois reduz as incertezas, desperdícios, perdas e o retrabalho. Em se tratando de Tecnologia da Informação (TI), fica ainda mais evidente a necessidade de se planejar bem, de maneira que os recursos necessários não faltem nem sejam subutilizados, pois na velocidade que as inovações surgem, se um produto não é lançado rápido o bastante, pode se tornar obsoleto ou simplesmente perder o *timing* de mercado.

Segundo o (SOFTEX, 2016), para uma projeto de *software* ter maior probabilidade de sucesso, deve-se prover informações sobre o andamento do mesmo que permitam a realização de correções quando houver desvios significativos no desempenho do projeto em relação ao que foi planejado. Para que estas informações sejam obtidas, as tarefas e os produtos de trabalho do projeto devem ser dimensionados utilizando-se métodos apropriados para cada situação.

Neste trabalho serão utilizados os dados de uma empresa de desenvolvimento de *software* que será chamada de Empresa X. A mesma utiliza adaptações do *framework* de desenvolvimento ágil *Scrum* para o gerenciamento de projetos e variações do *Planning Poker* para realizar suas estimativas.

As metodologias de desenvolvimento ágeis, especificamente o *Scrum*, tem sido extensamente utilizadas em projetos de desenvolvimento de *software*. Segundo pesquisa realizada pela (ALLIANCE, 2015) em 2015, o *Scrum* era a metodologia adotada por 95% das empresas participantes do estudo.

O *Scrum* trabalha com o conceito de Histórias de Usuário, sendo estas, a unidade básica de características de *software* a serem desenvolvidas e consequentemente, dimensionadas ou estimadas pelo Time, usualmente através de técnicas de opinião de especialistas, consenso e analogia a dados de outros projetos.

1.1 Motivação

Em engenharia de *software*, são utilizados modelos de estimativa para prever atributos importantes de projetos, como esforço em desenvolvimento, produtividade

de equipes e qualidade dos produtos entregues. Modelos que estimam o esforço de *software* têm motivado pesquisas consideráveis nos últimos anos. O procedimento de previsão destes modelos pode ser baseado em funções matemáticas ou outras técnicas, como analogia a projetos similares, redes neurais, árvores de regressão e modelos de indução de regra. A estimativa por analogia é uma das técnicas mais atraentes no campo de estimativa de esforço de *software* e consiste em comparar parâmetros de novos requisitos com dados históricos ou de outros projetos.(Khoshgoftaar,Taghi M., 2002)

Considerando que o dimensionamento de tarefas é uma etapa essencial e que serve de base para várias outras atividades de grande importância em projetos de *software*, como o monitoramento e controle através de medições e análises das mesmas, considerando a dificuldade que as empresas tem de treinar funcionários e mais ainda de manter uma baixa rotatividade da equipe, considerando que a atividade de dimensionar os requisitos de *software* é realizada através da análise das Histórias de Usuário (HU) pelos times *Scrum* e se trata de uma tarefa realizada com base na experiência e interpretação que a equipe tem do que está escrito nas HU, foi proposto que este processo poderia ser substituído por um sistema que utilizasse abordagens de Interpretação de Linguagem Natural (ILN), Aprendizado de Máquina (AM) e Inteligência Artificial (IA).

1.2 Objetivos

Este trabalho visa, no seu decorrer, atingir os seguintes objetivos principais:

- Desenvolver uma solução capaz de aprender com a base de dados de Histórias de Usuário estimadas anteriormente pelos especialistas (equipe) e dimensionar corretamente o tamanho de cada nova atividade enviada para a equipe de desenvolvimento;
- Analisar a eficácia da substituição das estimativas análogas feitas por especialistas, pela solução proposta no objetivo anterior, que aborda Aprendizado de Máquina (AM) e Inteligência Artificial (IA).

1.3 Contribuições

Com a automatização da predição de tamanho das atividades, os *Product Owners* das equipes *Scrum* economizarão um tempo precioso no planejamento de suas *Sprints*. O Time *Scrum* terá mais uma medida para validar suas estimativas e não será mais requisitado no meio da etapa de desenvolvimento para estimar demandas

não planejadas. As empresas terão menos prejuízo com a rotatividade de pessoas e poderão mensurar a produtividade das mesmas com facilidade, pois a estimativa continuará sendo feita à medida dos membros anteriores mais experientes, mesmo que estes deixem a instituição ou mudem de função. Consequentemente, o custo com mão de obra especializada tende a reduzir. Automatizar as estimativas das tarefas a serem realizadas também será cada vez mais necessário à medida que a tecnologia evolui, ao passo que, num futuro próximo, colocaremos os computadores para desenvolver a maioria dos nossos programas sozinhos.

As principais contribuições do trabalho são:

- Demonstrar os passos do dimensionamento automático de Histórias de Usuário através de técnicas de Aprendizado de Máquina [AM] e Inteligência Artificial [IA];
- Disponibilizar um sistema de dimensionamento automático de Histórias de Usuário para ser utilizado pela Empresa X;
- Fazer uma análise da eficácia das estimativas automáticas realizadas com o SVM.

1.4 Conteúdo

Além deste capítulo introdutório, o trabalho é composto por:

- Capítulo 2 - Trabalhos Relacionados: Apresenta uma visão geral sobre o que é estudado atualmente nas áreas relacionadas ao trabalho, Metodologias Ágeis, *Scrum*, Aprendizado de Máquina e Inteligência Artificial.
- Capítulo 3 - Estimativas de Tamanho de Software: Descreve detalhadamente o problema ao qual o trabalho visa resolver.
- Capítulo 4 - Método de Automação de Estimativa de Tamanho de Software: Descrição detalhada sobre os métodos utilizados na solução, do pré-processamento de dados ao teste do sistema com dados reais da Empresa X.
- Capítulo 5 - Experimentos: Cobre a análise dos parâmetros do algorítimo do SVM e os testes com dados reais da Empresa X.
- Capítulo 6 - Análise dos Resultados: Discussão a respeito dos resultados obtidos.
- Capítulo 7 - Conclusão: Conclusões a respeito da realização do trabalho como um todo e trabalhos futuros possíveis.

2 Trabalhos Relacionados

No final de 1997, Shepperd, M.; Schofield, C. validaram um método de estimativa análoga baseado na comparação de parâmetros entre projetos com auxílio da ferramenta ANGEL, onde argumentou que a estimativa análoga, baseada no conhecimento adquirido de outros projetos é uma técnica viável e pode ser útil à gerentes de projeto que necessitam de previsões de esforço o mais precisas possíveis, principalmente em estágios iniciais de projetos, onde não existem dados. O método foi analisado em 275 projetos de nove tipos diferentes.

Segundo Ken-ichi Matsumoto, há mais de dez anos o sucesso da estimativa de custos de software usando raciocínio baseado em análises de parâmetros de projetos tem sido notável. A precisão da estimativa depende fortemente de diferentes métodos heurísticos para selecionar os melhores subconjuntos de recursos e um conjunto adequado de projetos similares a partir do repositório. O trabalho realizado por Ken-ichi Matsumoto (2013) propôs um novo algoritmo adaptado para a estimativa de custos de software baseado em analogia, utilizando estrutura de computação *CUDA* permitindo a estimativa com grandes conjuntos de dados de projeto. Neste, foi demonstrado o uso do algoritmo distribuído, executado em unidades de processamento gráfico (GPU), arquitetura adequada para problemas de alto custo computacional. O método foi avaliado utilizando 11 conjuntos de dados do mundo real do repositório *PROMISE*. Os resultados mostraram que a abordagem *ABE-CUDA* foi capaz de produzir as melhores estimativas de custo do projeto através da determinação dos melhores subconjuntos de recursos e do número mais adequado de projetos análogos para estimativa, melhorando significativamente o tempo total de busca de recursos e a precisão de previsão para estimativa de custo de *software*. Mais importante ainda, o resultado de estimação otimizado pode ser usado como um *benchmark* de linha de base para comparar com outros métodos sofisticados baseados em analogia para estimativa de custo de software. O uso da plataforma *CUDA* mostra o quão custoso pode ser o dimensionamento de tarefas.

George Michael (2016) comparou o desempenho de métodos de estimativa análoga utilizados nos projetos de software de voo espacial da NASA, chegando a conclusão que se existem dados de projetos similares suficientes, a estimativa análoga paramétrica é a melhor maneira de estimar.

Grimstad, Stein e Jørgensen, Magne (2007) analisou, através de testes com diferentes especialistas, que informações irrelevantes da linguagem natural, bastante comuns em registros de requisitos de software a serem desenvolvidos, tendem a atra-

palhar o julgamento dos especialistas. Essa dificuldade se dá na distinção do que é ou não relevante na hora de estimar, causando confusões e discrepância nos resultados, especialistas que dispunham de textos mais simples e concisos tendiam a ser mais corretos em suas estimativas. A linguagem natural é quase sempre imprecisa e confusa, gerando várias interpretações diferentes. Quando transformamos os requisitos em linguagem de máquina, conseguimos maior precisão nas estimativas.

A maioria dos métodos de estimativa existentes e tentativas de automação dos mesmos, consistem em algorítimos sofisticados e custosos ou são de grande dependência de especialistas treinados nos métodos ou com larga experiência em projetos similares aos que serão executados. Um dos intuitos deste trabalho é justamente possibilitar à empresas de pequeno e médio porte, maneiras simples e baratas de realizar suas estimativas sem ocupar grande tempo de suas equipes.

Em se tratando da tecnologia utilizada para realizar o dimensionamento baseado em analogia, existe um trabalho referência, de Joachims (1998), no uso de máquinas de vetores de suporte (SVM) no aprendizado dos classificadores de texto a partir de exemplos. Ele analisou as propriedades particulares de aprendizagem com dados de texto e identificou por que SVM são apropriados para esta tarefa. Resultados empíricos apoiaram os achados teóricos. Ele concluiu que as SVM conseguem melhorias substanciais em relação aos métodos mais eficientes da época e se comportam de forma robusta em uma variedade de diferentes tarefas de aprendizado. Soares (2008) também obtém resultados consistentes utilizando SVM.

Como visto, estimativas baseadas em dados anteriores são bem viáveis e apresentam resultados consistentes. Levando em consideração que existem tais dados e existem tecnologias de AM capazes de aprender com eles e classificar novos dados de maneira igual, é considerado possível realizar o dimensionamento automático e correto de requisitos de software, principalmente em empresas com grandes bases de dados. Este trabalho pretende provar de maneira prática esta possibilidade.

3 Estimativas de Tamanho de Software

Eficiência, entrega do produto no prazo, dentro do orçamento e com um nível de qualidade desejado pelo cliente são características que influenciam no sucesso de organizações de desenvolvimento de software no globalizado e competitivo mercado atual de TI. Neste sentido, ressalta-se a importância de as empresas possuírem mecanismos de acompanhamento e a avaliação do progresso de processos, projetos e produtos.

Estes mecanismos, normalmente baseados em informações qualitativas e quantitativas, coletadas através de medições realizadas durante o projeto, são recursos essenciais na gestão de equipes de desenvolvimento de *software*. Para os gerentes de projetos, essas medidas, também conhecidas como métricas, permitem melhor entendimento do processo de produção e o controle do projeto de uma maneira geral. Sendo assim, as medições fornecem informações que permitem a determinação de pontos fortes e fracos dos processos e produtos, indicam ações corretivas e propiciam avaliações de impactos de tais ações. Por fim, colaboram com a garantia da qualidade dos processos e produtos existentes (E. Fenton, Norman ; L. Pfleeger, Shari, 1997; Rombach, H.Goal, 1994).

Dimensionar o tamanho do que será feito é fundamental no desenvolvimento de qualquer software. Os constantes avanços tecnológicos exigem que as empresas e equipes reajam rapidamente às mudanças de cenário e lancem seus produtos em prazos cada vez mais curtos, sob pena de ficarem obsoletos ou mesmo de algum produto concorrente ser lançado antes e tornar o projeto inviável em termos comerciais.

Para não perder o tempo de mercado, as equipes precisam ser eficientes e precisas em suas estimativas, sabendo rapidamente quanto tempo será necessário para lançarem pacotes de software, podendo assim, negociar com seus clientes quando cada nova funcionalidade será disponibilizada ou quando terá um produto mínimo viável para lançar no mercado, evitando gastos desnecessários e podendo direcionar a sua publicidade para o que for estar disponível primeiro.

Miranda (2014) constata que profissionais experientes são a melhor forma fazer estimativas de alto grau de complexidade. Estes profissionais possuem conhecimento especializado, que é adquirido ao longo de anos de prática em determinada atividade, observando as melhores e piores possibilidades de resolver cada problema. Esta habilidade é na maioria das vezes individual e cada pessoa a desenvolve de maneira diferente, sendo considerada difícil de ser transmitida.

A questão então, é que a estimativa do tamanho das atividades tem um custo

considerável e requer mão de obra qualificada, experiente e disponível nas fases de planejamento e, caso se trate de empresas que trabalham com desenvolvimento ágil, que prega responder a mudanças mais que seguir um plano[1], o dimensionamento se torna muito variável e necessita de atenção em praticamente todas as fases do desenvolvimento.

As capturas de tela das figuras 1 a 7 mostram tarefas registradas no Redmine da Empresa X, com Tamanhos distintos, estimados via *Planning Poker*, exemplificando o que cada um deles representa na realidade e mostrando as diferenças de escrita entre eles.

Figura 1 – Tarefa registrada no Redmine com tamanho 1 (campo Estimativa).

Pensando na economia de tempo, recursos e na redução de interrupções das equipes, este trabalho se propõe a substituir as estimativas humanas por uma máquina que colete as entradas dos usuários responsáveis por cadastrar atividades de desenvolvimento, processe e retorne um valor referente ao tamanho da atividade para que o gestor do projeto possa planejar quando a mesma entrará em desenvolvimento e qual a previsão de entrega ao cliente.

[1] http://www.manifestoagil.com.br/

História #10411

Editar | Tempo de trabalho | Observar | Copiar | Excluir

Melhorar a exibição das mensagens de retorno da SEFAZ na NFCe Migrate/Daruma

Adicionado por Eça F. menos de um minuto atrás.

Situação:	Novo	**Início:**	06/12/2016
Prioridade:	Normal	**Data prevista:**	
Atribuído para:	-	**% Terminado:**	0%
Versão:	-		
Setor Solicitante:	COLTIN		
Solicitante:		**Critério de Aprovação:**	20079
Escopo:		**Produto:**	
Estimativa:	2	**Módulo:**	N/A
Valor de Negócio:	40	**Análise:**	Não
		Item Cronograma Control:	35 N/A

Descrição Responder

Como: Operador do PDV
Quero: E exibição da descrição do retorno de rejeições da SEFAZ
Para: Que o motivo da rejeição possa ser identificado mais rapidamente **Critério de validação**

- Exibir para o usuário o motivo da rejeição da SEFAZ

Como implementar

- Concatenar a descrição do retorno da SEFAZ

Subtarefas Adicionar

Tarefas relacionadas Adicionar

Editar | Tempo de trabalho | Observar | Copiar | Excluir

Exportar para Atom | PDF

Figura 2 – Tarefa registrada no Redmine com tamanho 2 (campo Estimativa).

História #10412

Editar | Tempo de trabalho | Observar | Copiar | Excluir

não está sendo permitido fazer impressão da carta de correção eletrônica

Adicionado por Eça F. 1 minuto atrás. Atualizado menos de um minuto atrás.

Situação:	Novo	**Início:**	06/12/2016
Prioridade:	Normal	**Data prevista:**	
Atribuído para:	-	**% Terminado:**	0%
Versão:	-		
Setor Solicitante:	COLTIN		
Solicitante:		**Critério de Aprovação:**	19912
Escopo:		**Produto:**	
Estimativa:	3	**Módulo:**	N/A
Valor de Negócio:	60	**Análise:**	Não
		Item Cronograma Control:	35 N/A

Descrição Responder

1) Qual o comportamento observado?

Não é possível fazer a impressão da carta de correção.

2) Quais as etapas para reproduzir o comportamento observado?

. Fazer uma carta de correção: vendas balcão, procedimentos especiais, carta de correção eletrônica;
. Ao concluir as correções, clicar em F9 para enviar e logo após pedir a impressão;
. É apresentado a mensagem:" Erro ao imprimir a Carta de Correcao Eletronica: Caminho do arquivo de impressão do EVENTO não assinalado."

3) Qual o comportamento esperado?

Que seja permitido pedir a impressão da carta de correção.

Subtarefas Adicionar

Tarefas relacionadas Adicionar

Figura 3 – Tarefa registrada no Redmine com tamanho 3 (campo Estimativa).

História #10413

Editar | Tempo de trabalho | Observar | Copiar | Excluir

Refatorar o relatório de títulos baixados a receber

Adicionado por Eça F. menos de um minuto atrás.

Situação:	Novo	**Início:**	06/12/2016
Prioridade:	Normal	**Data prevista:**	
Atribuído para:	-	**% Terminado:**	0%
Versão:	-		
Setor Solicitante:	COLTIN		
Solicitante:		**Critério de Aprovação:**	19659
Escopo:		**Produto:**	
Estimativa:	5	**Módulo:**	N/A
Valor de Negócio:	30	**Análise:**	Não
		Item Cronograma Control:	35 N/A

Descrição Responder

Como: Analista de negócios
Quero: Que seja realizado a refatoração do relatório de títulos baixados a receber.
Para: Que seja adequado ao novo framework e ao novo banco.

Critérios de validação:

- Geração do relatório Títulos baixados a receber com os filtros sugeridos no requisito.

Como implementar:

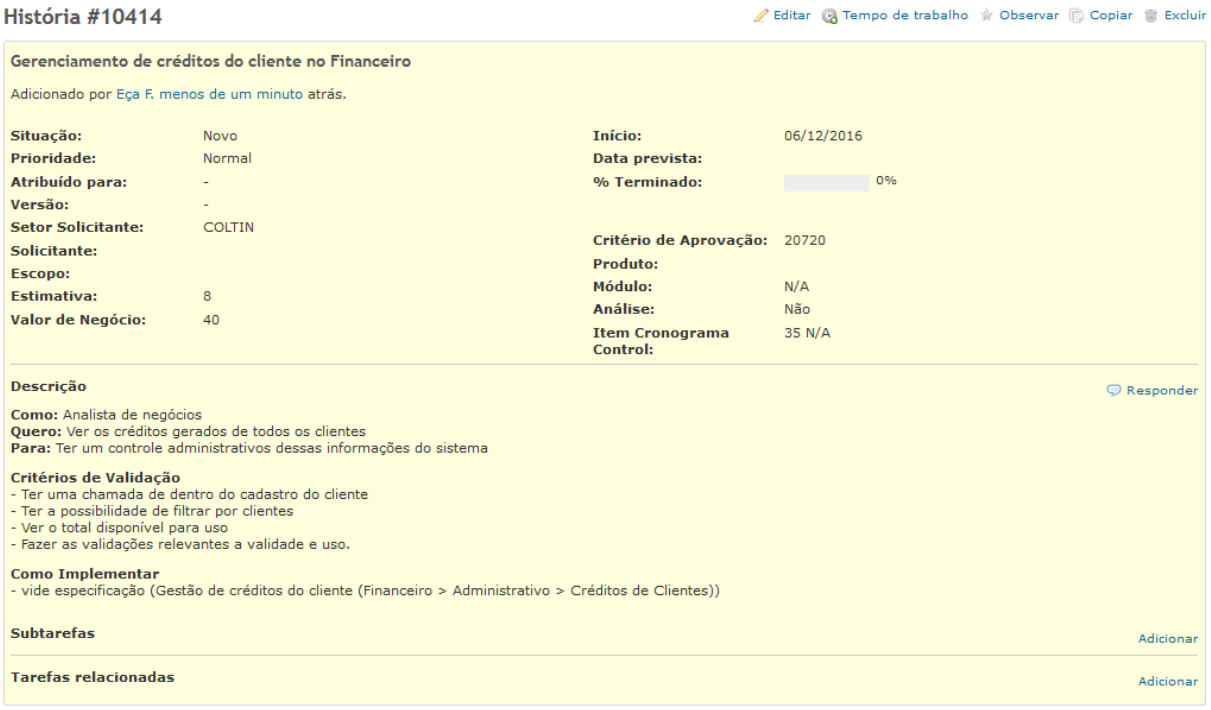
- Interface para a geração do relatório.
- Possibilidade de exportar em pdv, excel e visualização em tela.

Requisito:
Seguir o padrão já existente no sistema.

Subtarefas Adicionar

Tarefas relacionadas Adicionar

Figura 4 – Tarefa registrada no Redmine com tamanho 5 (campo Estimativa).



História #10414

Editar | Tempo de trabalho | Observar | Copiar | Excluir

Gerenciamento de créditos do cliente no Financeiro

Adicionado por Eça F. menos de um minuto atrás.

Situação:	Novo	**Início:**	06/12/2016
Prioridade:	Normal	**Data prevista:**	
Atribuído para:	-	**% Terminado:**	0%
Versão:	-		
Setor Solicitante:	COLTIN		
Solicitante:		**Critério de Aprovação:**	20720
Escopo:		**Produto:**	
Estimativa:	8	**Módulo:**	N/A
Valor de Negócio:	40	**Análise:**	Não
		Item Cronograma Control:	35 N/A

Descrição Responder

Como: Analista de negócios
Quero: Ver os créditos gerados de todos os clientes
Para: Ter um controle administrativos dessas informações do sistema

Critérios de Validação
- Ter uma chamada de dentro do cadastro do cliente
- Ter a possibilidade de filtrar por clientes
- Ver o total disponível para uso
- Fazer as validações relevantes a validade e uso.

Como Implementar
- vide especificação (Gestão de créditos do cliente (Financeiro > Administrativo > Créditos de Clientes))

Subtarefas Adicionar

Tarefas relacionadas Adicionar

Figura 5 – Tarefa registrada no Redmine com tamanho 8 (campo Estimativa).

Figura 6 – Tarefa registrada no Redmine com tamanho 13 (campo Estimativa).

Figura 7 – Tarefa registrada no Redmine com tamanho 20 (campo Estimativa).

4 Métodos de Estimativa de Tamanho de Software

Este Capítulo descreve as etapas do desenvolvimento do sistema de estimativa automática de tamanho de *software*. Serão detalhados todos os passos realizados para atingir os objetivos descritos no início do trabalho.

Alguns métodos de estimativa existentes na literatura:

- **Análise de Pontos de Função (APF)** (ALBRECHT, 1979) - Mede o tamanho em pontos de função, cada funcionalidade implementada do ponto de vista do usuário corresponde a uma pontuação;
- **Método Delphi** (PMI, 2013) - Baseado na experiência de especialistas, cada um deles faz uma ou mais estimativas e a média dos valores é tida como o resultado final;
- **Análise de Pontos por Caso de Uso (UCP)** (WIKIPEDIA, 2016) - Analisa os casos de uso de forma a estimar a complexidade do trabalho exigido para implementá-los;
- **Modelo de Custo Construtivo (COCOMO)** (PRESSMAN, 2006) - Estima o tamanho de um software baseado na quantidade de linhas de código e diversos outros parâmetros como função do programa, custo, avaliação do produto final, hardware, equipe e atributos do projeto;
- **Estimativa por Analogia** (PMI, 2013) - Consiste na utilização de dados históricos de outros projetos, avaliando as semelhanças e utilizando a comparação para definir as estimativas do novo projeto.

Como visto, o dimensionamento de requisitos de software pode ser feito através de diversas técnicas disponíveis na literatura. Dentre todas as diferenças existentes entre elas, há duas condições que sempre se repetem: os requisitos são descritos e registrados de alguma forma e recebem alguma classificação. Consideradas estas, será analisado como a Empresa X trabalha, como registra seus requisitos e realiza suas estimativas, como será proposta a automação desse processo e quais elementos são considerados necessários no processo de estimativa manual e automática.

4.1 Definição do Estudo de Caso

Como objeto de estudo e realização dos testes práticos, foi escolhida uma empresa de desenvolvimento de *software* de cerca de 50 funcionários do estado da Bahia, que chamaremos de Empresa X. Ela faz uso de metodologias ágeis, como *Scrum* e *Planning Poker* e produz softwares de gestão e automação comercial, tendo milhares de clientes ao longo do Brasil.

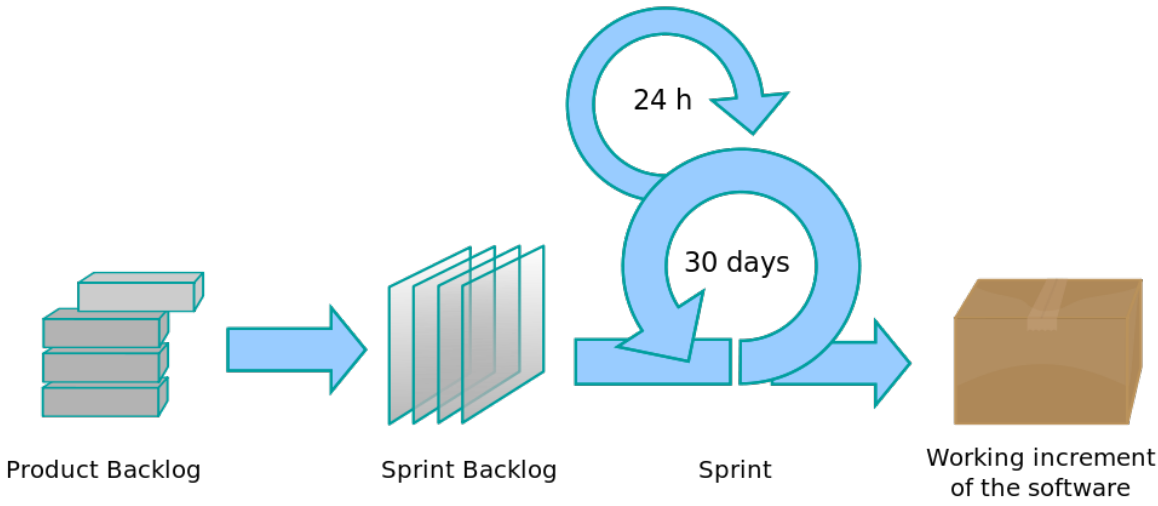
A empresa foi escolhida por ter uma base de dados acessível e de tamanho considerado bom, cerca de 2 mil Histórias de Usuário registradas e classificadas na ferramenta Redmine, seguindo o modelo da Figura 8.

Figura 8 – Exemplo de tarefa registrada no Redmine.

4.1.1 Scrum

Scrum é um *framework* para desenvolver e manter produtos complexos (Schwaber, Ken; Sutherland, Jeff, 2013), largamente utilizado ao redor do Brasil e do mundo (ALLIANCE, 2015). A Figura 9 ilustra o processo padrão do *Scrum*.

No processo básico do *Scrum*, o PO levanta as necessidades dos clientes e as cadastra no *Product Backlog*. A cada *Sprint*, itens do *Product Backlog* são priorizados pelo PO e Time para serem desenvolvidos. *Sprints* são iterações curtas de 2 a 4 semanas, onde o Time vai desenvolvendo as Histórias escolhidas no planejamento da *Sprint*. A cada dia são feitas reuniões diárias onde a equipe conversa rapidamente



Créditos: *Wikipedia*

Figura 9 – Processo padrão do *Scrum*

sobre o que foi feito, o que pretendem fazer e o que os está impedindo de ser mais produtivos. Ao fim da *Sprint*, o Time entrega as Histórias concluídas ao PO e o ciclo volta ao início.

O *Scrum* tem alguns conceitos:

1. Histórias de Usuário (HU): São as representações dos Requisitos de Software a serem dimensionados, desenvolvidos e inseridos em novas versões de Sistemas.
2. Papéis:
   a) *Product Owner* (PO): Coleta os requisitos junto aos Clientes, escreve as *HU*, as apresenta e posteriormente as recebe do Time.
   b) *Scrum Master* (SM): Responsável por manter a equipe livre de impedimentos e funcionando em alta performance.
   c) *Scrum Team* (Time): Responsável por estimar (junto como *PO*) e desenvolver as *HU* no decorrer das *Sprints*.
3. *Sprint*: *Time Box* de usualmente 15 dias onde o *Time* desenvolve as *HU* definidas pelo *PO*.
4. *Planning Poker*: Método de estimativa onde os especialistas (*Time e PO*) leem as *HU* e discutem o seu Tamanho, que só pode assumir um desses valores 1, 2, 3, 5, 8,13 e 20.
5. Cerimônias:

a) *Sprint Planning*: Cerimônia que ocorre no início de cada *Sprint* onde o *PO* apresenta as *HU* e o *Time* vai estimando uma a uma e aceitando ou recusando inclui-las na *Sprint*.

b) *Sprint Review/Retrospective*: Cerimônia que ocorre no fim de cada *Sprint* onde o *Time* apresenta as *HU* realizadas e o *PO* as aprova ou rejeita.

6. Tamanho: Unidade utilizada de medida utilizada no dimensionamento básico das *HU*.

7. *Product Backlog*: Conjunto de atividades que o PO levanta para serem desenvolvidas no momento definido por ele.

8. *Sprint Backlog*: Conjunto de atividades aceitas pelo Time para serem desenvolvidas na *Sprint*.

4.1.2 Estimativas da Empresa X

A Empresa X realiza estimativas através de uma adaptação do *Planning Poker*. Na prática, para cada nova HU apresentada ao Time, eles realizam de um a três rodadas de estimativas, onde cada membro sugere um Tamanho normalmente condizente com a Tabela 1, baseando-se em sua experiência e em uma tarefa considerada Padrão para ser comparada sempre. Este método é uma mistura de *Delphi* com Analogia e é muito comum em empresas ágeis devido à sua facilidade de aprendizado e praticidade de utilização.

Tamanho	Interpretação
1	Tarefas quase prontas
2	Tarefas muito pequenas
3	Tarefas pequenas
5, 8	Tarefas médias
13, 20	Tarefas grandes

Tabela 1 – Interpretação dos Tamanhos utilizada na Empresa X

O histórico de dados da Empresa X é composto por milhares de HU com Tamanho, tempo necessário para implementação, desenvolvedor responsável, além de todo um detalhamento de situações que possam ter vindo a dificultar ou facilitar a vida do desenvolvedor, conforme mostram as Figuras 8 e 10, nesta última é possível ver como o histórico de alterações é registrado. Neste estudo, são considerados apenas o título, descrição e o *Tamanho* de cada *HU*.

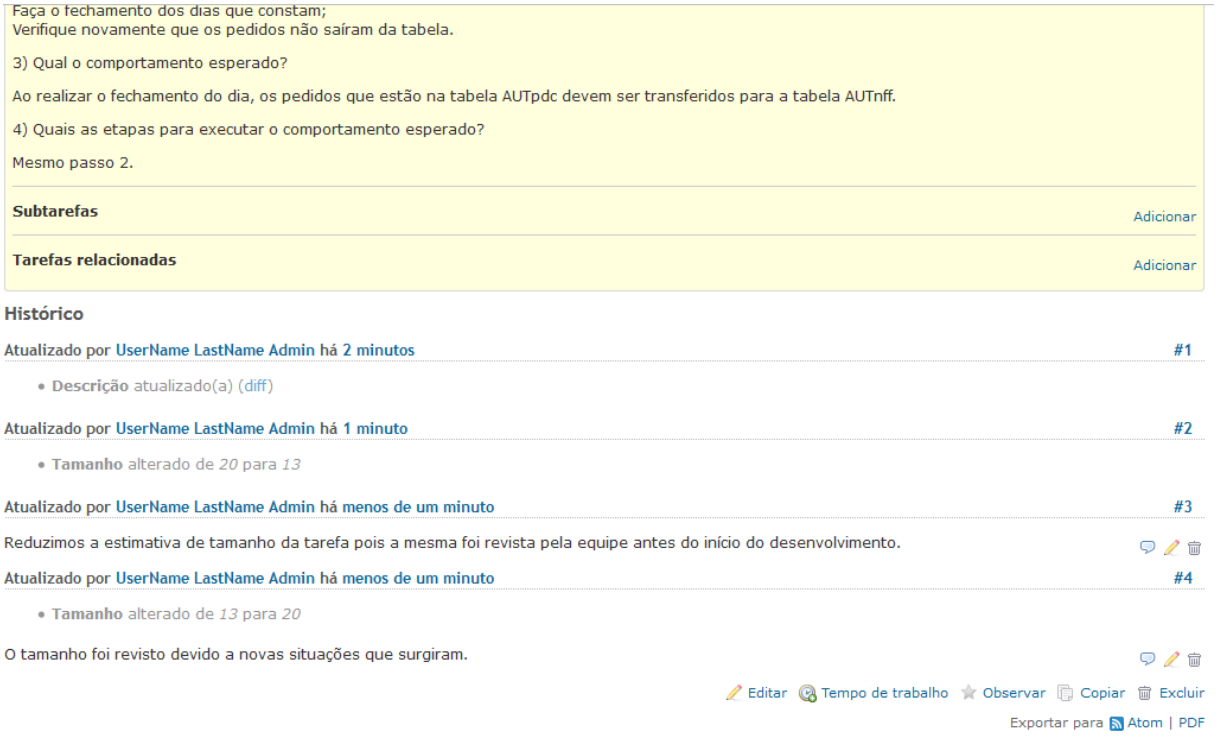


Figura 10 – Histórico registrado de uma História de Usuário cadastrada na ferramenta Redmine.

4.2 Ferramentas e Técnicas Utilizadas

A estratégia para implementar a solução é baseada em ler a base de dados disponível, montar um algorítimo que limpe e traduza os dados para linguagem de máquina.

Os dados estão registrados na ferramenta Redmine[1] sob o banco de dados MySQL[2].

A extração, limpeza e tradução dos dados para linguagem de máquina foi realizada com a a linguagem Python[3] por ser clara, concisa e simples de aprender além de ter inúmeras bibliotecas maduras para trabalhar com linguagem natural como o *Natural Language Toolkit (NLTK)*[4] que fornece várias facilidades no trabalho com textos. O Python e suas bibliotecas estão detalhados na seção *Pré Processamento de Dados*.

Em Celso A. A. Kaestner (2002) e Xin Chen (2006) notamos que a maioria das técnicas de seleção de características relevantes funcionava bem para problemas onde os textos eram demasiado longos e haviam diversas outras formas de relacionamento entre as palavras, como a similaridade com o título. Mas devido à restrições de tempo e ao fato das HU serem curtas a ponto de algumas terem menos de 100 caracte-

1 www.redmine.org
2 http://dev.mysql.com/doc/
3 www.python.org
4 http://www.nltk.org/

res, a técnica de seleção de características escolhida foi o termo–inverso da frequência nos documentos(TF-IDF) pois, ela calcula o peso de cada termo em relação à frequência do mesmo na sentença e ao inverso da frequência dele no documento. O TF-IDF será explicado em mais detalhes na seção *Seleção de Características Relevantes*.

Após a obtenção do peso de cada termo, é necessário analisar quais termos são realmente relevantes para a definição da sentença. O algorítimo selecionado para a realização desta tarefa foi o *Principal Component Analysis (PCA)* que será detalhado na seção *Seleção de Características Relevantes*.

Joachims (1998) explica muito bem de como realizar classificação de textos baseados em exemplos através de *Support Vector Machines (SVMs)*, e suas vantagens frente a outros algorítimos de IA. Soares (2008) também apresenta bons resultados na solução de problemas semelhantes. Devido ao fato do trabalho de JOACHIMS ser uma referência em categorização de texto e SVM, somando-se à escassez de tempo para a realização de testes com outros algorítimos, ficou definido SVM como a ferramenta de classificação a ser utilizada. O detalhamento compõe a seção 4.3.4 *Máquina de Vetores de Suporte*.

Para a execução dos algorítimos de *AM e IA*, foi utilizada a ferramenta Weka[5], uma suíte capaz de executar diversos algorítimos na área de IA.

4.3 Técnicas de Aprendizado de Máquina

Não é necessário conhecer a totalidade das formas de estimativa para perceber que, se ela estiver registrada corretamente, podemos utilizar uma máquina para aprender os padrões de escrita e baseado nisto, estimar novas tarefas que venham a surgir dentro de um contexto similar.

Sendo assim, este trabalho sugere que, se uma empresa tiver uma base de dados histórica bem definida, ela pode automatizar suas estimativas mantendo o padrão de classificação das tarefas anteriores.

Para tal, é sugerido o uso de técnicas e ferramentas de Aprendizado de Máquina (AM) e Inteligência Artificial (IA).

Etapas do aprendizado de máquina:

1. Compreensão e coleta dos dados;
2. Pré processamento;
3. Seleção de características relevantes;

[5] http://www.cs.waikato.ac.nz/ml/weka/

4. Maquina de Vetores de Suporte.

Abaixo são explanados alguns conceitos básicos de Aprendizado de Máquina (AM) (SOARES, 2008), que foram considerados na execução deste trabalho.

1. Instâncias: Tupla de valores de interesse, neste trabalho é o texto e o Tamanho estimado dele.
2. Característica, atributo ou variável: Descreve o objeto de interesse, no caso, atributos são as palavras contidas nos textos das HU.
3. Classe: Atributo que rotula o objeto de interesse, neste estudo a classe é o Tamanho das HU.
4. Base de dados: Conjunto de instâncias contendo tuplas de atributos, normalmente divididos como neste trabalho, em instâncias de treinamento e de teste. As de treinamento são utilizadas para aprender como relacionar os atributos às classes de valores e as de teste para validar o aprendizado.
5. Ruído: Sinal que torna os dados imperfeitos, pode ser proveniente do pré processamento, transformação ou instâncias classificadas incorretamente.

4.3.1 Compreensão e Coleta

A base de dados da Empresa X possui cerca e 2 mil tarefas realizadas pelos times de desenvolvimento. A enorme maioria delas é composta por título, descrição, tamanho e outras características que não serão utilizadas na classificação.

O banco de dados do *Redmine* da Empresa X é armazenado sob o sistema gerenciador de banco de dados (SGBD) MySQL. Existe uma tabela central chamada *issues*, ela guarda o título, descrição e o autor de cada tarefa, dentre outras informações. A tabela *issues* se relaciona com a tabela *custom_values*, onde estão armazenados os Tamanhos estimados das Histórias de Usuário. As tabelas básicas para a coleta de dados são estas, porém é necessário um cuidado especial ao buscar os valores em *custom_values* pois ela armazena vários valores de vários campos personalizados de cada *issue*. A Figura 11 mostra o código utilizado para a criação da View *db* que serve de base para a análise e coleta de dados.

Os textos das Histórias contém, como qualquer texto de linguagem natural, várias palavras que não significam nada sozinhas, como artigos e valores numéricos, que não adicionam nenhuma informação relevante para a máquina decidir como classificar.

No momento da análise dos dados foi percebido que haviam muitas HU que estavam escritas erradas, após uma limpeza inicial, retirando as tarefas sem Tamanho

```
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `db` AS
    SELECT
        `issues`.`id` AS `id`,
        `issues`.`subject` AS `titulo`,
        `issues`.`description` AS `descricao`,
        `custom_values`.`value` AS `tamanho`
    FROM
        (`issues`
        JOIN `custom_values`)
    WHERE
        ((`custom_values`.`customized_id` = `issues`.`id`)
            AND (`custom_values`.`custom_field_id` = 85)
            AND (`custom_values`.`value` > 0))
    ORDER BY `custom_values`.`value`
```

Figura 11 – view que serve de base para a coleta e análise de dados.

definido, sem título ou sem descrição, reduzimos manualmente a base de dados para cerca de 1600 Histórias e então para pouco mais de 600, pois haviam muitas HU mal escritas que dificultavam o aprendizado da máquina.

Em seguida foi feita a coleta utilizando um *Script* em *Python* que recebe as entradas do banco de dados, executa a limpeza em cada texto e o salva numa *List* contendo *ID*, *Tamanho*, *Titulo* e *Descrição* de cada História.

Um detalhe que também foi percebido na análise dos dados é que algumas HU precisavam do título para serem compreendidas, então o *script* considera o Título e a Descrição como um só campo, concatenando-os.

4.3.2 Pré Processamento

Entre a leitura dos dados do Redmine como na Figura 12 e a escrita dos mesmos na *list* há uma etapa muito importante para o desempenho do restante do sistema, a limpeza dos dados, nela são:

1. Lidos os dados no banco de dados do Redmine;
2. Retirados os espaços em excesso;
3. Removidos os números - pois os mesmos apenas confundem o SVM;
4. Removidas as pontuações - Pois não servem para o SVM e em diversas situações são utilizados de forma errada e acabam transformando uma palavra em duas ou inverso;

5. Removidas as *Stopwords* - palavras que não servem para classificação e apenas atrapalham os algorítimos;

6. Removidas as divisões de linha e tabulação para que o script possa salvar uma HU por item da *List*.

Figura 12 – Histórico registrado de uma História de Usuário de tamanho 5 cadastrada na ferramenta Redmine.

Ao fim do processo de limpeza, são geradas três saídas: a Figura 13 que contém a *ID*, o tamanho e as palavras em linguagem natural que restaram do processo, a Figura 14 que representa a matriz de frequência das palavras do texto e a Figura 15 que representa o calculo da medida TF-IDF do texto.

```
['20855', '5', 'migrate duplicidade numeração cupons local dincash pdv
venda versão identificado última versão 8k problema ocorre versões
anteriores pré condições acessar ambiente homologação utilizar migrate
erro sistema reutiliza numerações nota é rejeitada causando duplicidade
números cupom dincash sistema programado tratar duplicidade casos mesma
causou problemas amazon print algumas notas canceladas subiram sap sendo
autorizadas fato tentaram subir logo porém cancelada mesma numeração
possível subir nota gerando problema cliente situação desejada sistema
deve deixar reutilizar numerações comportando total nfce pulando
inutilizando numerações notas rejeitadas']
```

Figura 13 – História de usuário inserida na *list*, após os procedimentos de limpeza de dados.

```
[0, 1, 0, 0, 1, 0, 0, 0, 2, 0, 0, 0, 1, 3, 0, 1, 0, 1, 0, 3, 0, 2, 0, 2,
0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 2,
0, 1, 0, 0, 1, 0, 2, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1,
0, 2, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 2, 1, 0, 1,
1, 0, 1, 0, 1, 1, 0, 0, 3, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1,
0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 2, 2, 1, 0, 0, 0, 1, 0, 0, 1,
0, 0, 1, 0, 0, 1, 0]
```

Figura 14 – História de usuário transformada em matriz de frequência, para ser calculado o TF-IDF.

```
[ 0.         0.         0.16461527 0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.05117936 0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.08230763
  0.         0.         0.         0.         0.         0.         0.
  0.21079672 0.         0.         0.         0.         0.         0.
  0.         0.05434683 0.16461527 0.         0.         0.         0.
  0.         0.         0.         0.         0.08230763 0.         0.
  0.         0.         0.         0.         0.         0.         0.
  0.         0.07549343 0.16461527 0.         0.         0.         0.
  0.         0.         0.08230763 0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.         0.
  0.         0.         0.06322142 0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.32923053
  0.         0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.08230763
  0.         0.         0.         0.         0.         0.         0.
  0.         0.         0.17467054 0.         0.         0.08230763
  0.         0.         0.         0.         0.         0.         0.
```

Figura 15 – Cálculo do TF-IDF da mesma História de Usuário.

Todo o processo foi codificado em *Python* com a utilização da biblioteca NLTK que provê diversas classes de auxílio para trabalhar com linguagem natural.

Esta etapa foi inicialmente muito custosa devido à problemas de codificação de texto, e má escrita de histórias. Em trabalhos futuros relacionados a este assunto, é importante se certificar logo no início do desenvolvimento que todas as bases de dados estão trabalhando com a mesma codificação e que as HU são escritas seguindo um padrão mínimo de qualidade e informação.

4.3.3 Seleção de Características Relevantes

Primeiramente, para que o Weka (Figura 16) possa trabalhar com os dados, é necessário converter os dados obtidos no processo anterior para uma linguagem que a máquina consiga entender. Esta conversão foi feita criando uma matriz de frequência com calculo do TF-IDF, onde cada coluna é um termo do universo de termos de todos os textos e cada linha é composta pelo calculo TF-IDF de cada termo.

O resultado é uma matriz de *termos X textos* onde cada item da mesma é a

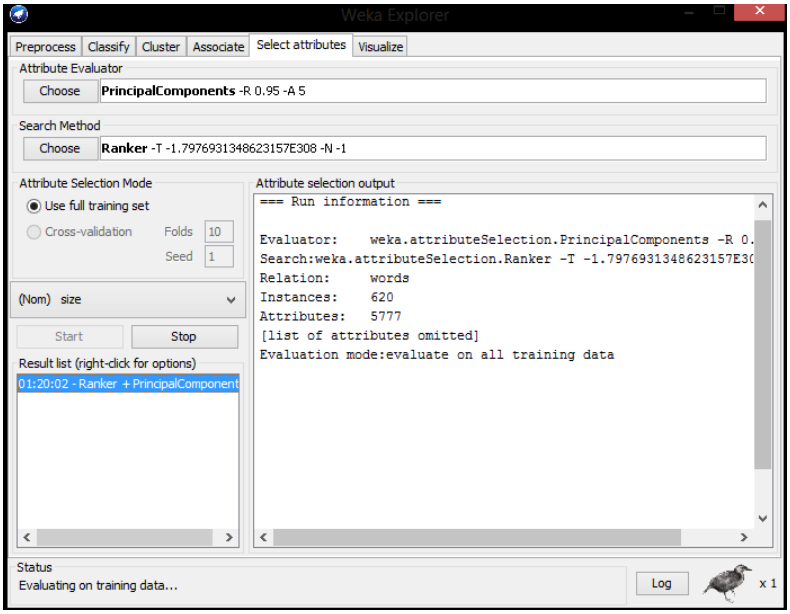


Figura 16 – Ferramenta Weka.

frequência de cada termo na linha.

Porém, até esta etapa ainda existem muitos termos ou atributos, exatamente 5.777 termos e 620 textos, esta grande quantidade acaba por tornar a execução do algorítimo SVM custosa e imprecisa. É necessário selecionar quais desses termos são de fato relevantes para a classificação dos Tamanhos. Então foi utilizado, conforme descrito em Celso A. A. Kaestner (2002) a medida estatística TF-IDF 4.3 que mede a importância de um termo 4.1 em relação a uma coleção de documentos 4.2, o que acaba por funcionar perfeitamente para a situação em questão.

A equação tf(t,d) é dada pela frequência do termo t no documento d.

$$\text{tf}(t, d) = f_{t,d} \tag{4.1}$$

A equação idf(t,D) é:

$$\text{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|} \tag{4.2}$$

onde N é o número de total de documentos observados e $|\{d \in D : t \in d\}|$ é a quantidade de documentos onde o termo t aparece.

E por fim, tfidf(t, d, D) de um termo t num documento d na coleção de documentos D:

$$\text{tfidf}(t, d, D) = \text{tf}(t, d) \cdot \text{idf}(t, D) \tag{4.3}$$

Ao final do calculo TF-IDF 4.3 para cada um dos termos em cada um dos textos, é gerada uma matriz com a importância de cada termo. O próximo passo é selecionar os principais termos que a partir de agora serão chamados de Atributos.

Para realizar essa seleção, foi utilizado o *PCA* (Figura 22) que realiza uma conversão ortogonal no conjunto de atributos transformando-os em um outro conjunto de variáveis linearmente não correlacionadas, de tamanho menor ou igual ao conjunto original, chamadas de *Componentes Principais*.

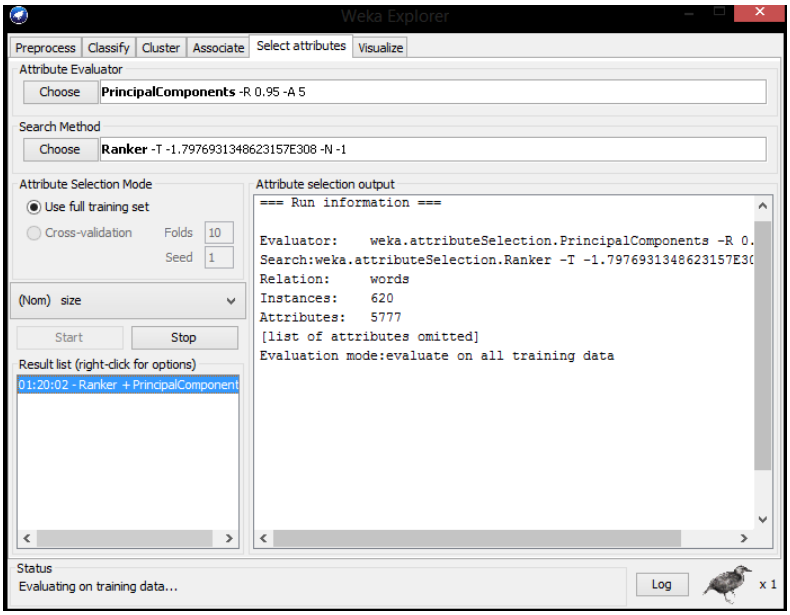


Figura 17 – Execução do PCA no Weka.

No experimento, ao ser realizado a seleção de atributos com o PCA, foi obtido um resultado bastante satisfatório, reduzindo o número de atributos de 5.777 para 311, uma redução de 94% tornando a execução da próxima etapa (SVM) muito mais rápida e eficaz.

A seção 5 *Experimentos* será executada com as duas bases, a inicial, com todos os atributos e a PCA, com apenas os Componentes Principais, para que possamos comparar os resultados.

Apesar dos bons resultados com o PCA, seria interessante comparar com outros algorítimos de seleção de atributos em trabalhos futuros no intuito de analisar a eficiência dos mesmos em diferentes situações.

4.3.4 Máquina de Vetores de Suporte

Uma Máquina de Vetores de Suporte (SVM) é um algorítimo de classificação que necessita de dados de treinamento para predizer se algo pertence ou não a uma determinada classe (KOWALCZYK, 2016).

A SVM divide os dados classificados em hiperplanos e os mais eficientes são os que tem maior margem para os limites dos dados.

A Figura 18 demonstra um conjunto de dados de exemplo.

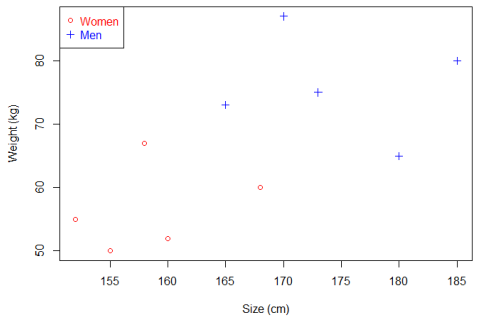


Figura 18 – Exemplo de dados de classificação
(KOWALCZYK, 2016)

A Figura 19 exemplifica um hiperplano traçado para separar as classes Homem e Mulher.

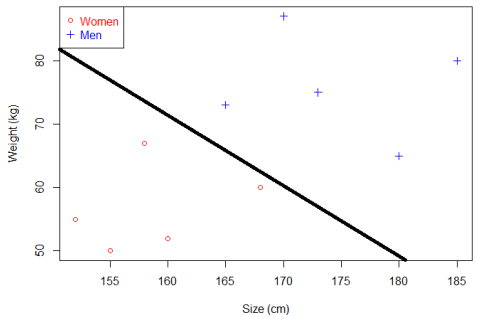


Figura 19 – Exemplo de dados de classificação separados por um hiperplano
(KOWALCZYK, 2016)

Porém, como veremos na Figura 20, não existe apenas um hiperplano possível para dividir os dados nas duas classes. O acerto nos parâmetros da equação do SVM deve ser feito de maneira a encontrar o melhor hiperplano como podemos ver na Figura 21.

Na prática, esta seleção dos melhores parâmetros foi feita testando vários diferentes na ferramenta Weka. Os dados são carregados, definimos uma *grid search* com parâmetros que queremos testar e analisamos um a um da seguinte maneira:

1. São separados dez conjuntos de dados distintos;
2. Dos dez, cada equação é treinada em nove deles;

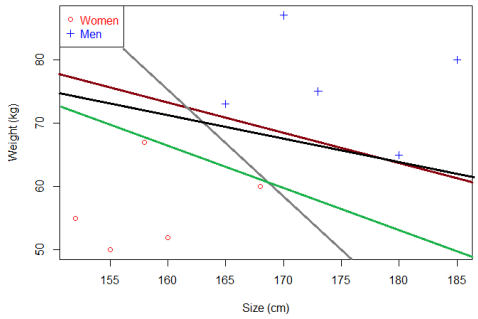


Figura 20 – Exemplo de diversos hiperplanos possíveis na instância (KOWALCZYK, 2016)

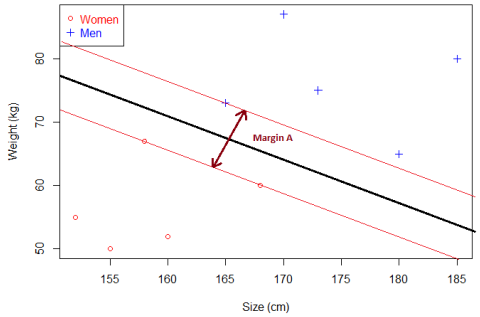


Figura 21 – Exemplo de margens de um hiperplano (KOWALCZYK, 2016)

3. Após o treinamento, a equação é validada no último deles;
4. Depois de validada, são regerados os dados utilizando a semente aleatória;
5. Em seguida os dados são testados e os resultados registrados para identificar qual das equações teve melhor desempenho.

5 Experimentos

A realização dos experimentos nas instâncias existentes deu-se em quatro fases: seleção dos parâmetros do algorítimo *epsilon-SVM*; execução do algorítimo; e análise dos resultados obtidos.

5.1 Seleção dos Parâmetros SVM

Na seleção dos melhores parâmetros, foi realizada uma *grid search* com validação cruzada onde foram testadas apenas funções kernel RBF, polinomial e sigmóide. Preliminarmente foi constatado que a kernel polinomial tinha um custo computacional muito maior, impossibilitando mais testes por limitações de tempo. O parâmetro *cost* C foi definido dentre $2^{-15}, 2^{-12}, 2^{-9}, 2^{-6}, 2^{-3}, 2^{0}, 2^{3}, 2^{6}, 2^{9}, 2^{12}, 2^{15}$, a largura do kernel *gamma* G entre $10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}, 10^{4}$ e o grau do polinômio do kernel polinomial D entre $1, 2, 3, 4$.

Após realizar testes com todas as variações de parâmetros, chegamos à Tabela 2 com os melhores parâmetros.

Parâmetro	Valor
gamma	3.0
C	1000.0
epsilon	0.001
seed	333333
kernel	RBF

Tabela 2 – Melhores parâmetros encontrados para a SVM

5.2 Execução do Algorítimo

Após ter a base de dados e os parâmetros do algorítimo *epsilon-SVM* (Figura 22) definidos na Tabela 2, foi iniciada a execução do sistema e analise dos resultados.

Foram realizados dois testes, um com todos os atributos encontrados inicialmente e outro contendo apenas os principais atributos que foram selecionados com o PCA. A Tabela 3 mostra a quantidade de atributos e instâncias de cada base e fica claro que, mesmo após serem retiradas as *stopwords*, números e códigos sem significado dos textos, ainda restavam muitas palavras desnecessárias para que o algorítimo de AM realizasse a classificação, ou seja, muitos termos, apesar de importantes para

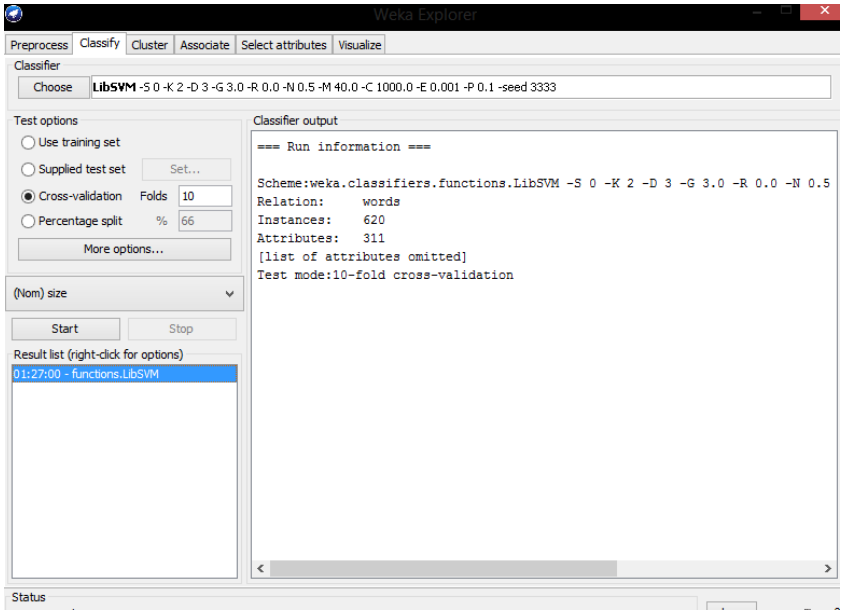


Figura 22 – Weka executando a SVM com os parâmetros escolhidos na tabela 2.

o entendimento humano das HU, não acrescentavam nada para que a estimativa de tamanho fosse realizada pela máquina.

A Tabela 3 mostra os resultados obtidos ao executar os testes com Todos os Atributos.

Base de Dados	Atributos	Instâncias
Todos os Atributos	5777	620
Componentes Principais	311	620

Tabela 3 – Tabela de atributos e instâncias das bases utilizadas

A Tabela 4 detalha o percentual de acertos dos testes realizados na base com Todos os Atributos.

Base de Dados	Atributos	Instâncias	Acertos	%Acertos
Todos os Atributos	5777	620	557	89.838%
Componentes Principais	311	620	565	91.129%

Tabela 4 – Resultados da execução do SVM

A matriz de confusão (Tabela 6) representa a quantidade exata de instâncias que foram classificadas erroneamente (horizontal e vertical) e corretamente (diagonal), inclusive mostrando para qual classe foi realizada a classificação, nos testes realizados na base com todos os atributos.

A Tabela 7 detalha a matriz de acerto dos resultados da base com os atributos selecionados pelo PCA.

Classe	TP Rate	FP Rate	Precision	Recall	F-Measure
1	0.868	0	1	0.868	0.929
2	0.857	0.01	0.958	0.857	0.905
3	0.723	0.006	0.952	0.723	0.822
5	0.993	0.156	0.829	0.993	0.903
8	0.927	0	1	0.927	0.962
13	0.758	0	1	0.758	0.862
20	0.889	0	1	0.889	0.941
Média Ponderada das Classes	0.898	0.07	0.911	0.898	0.897

Tabela 5 – Detalhamento dos resultados por Classe utilizando todos os atributos

a	b	c	d	e	f	g	<– classificado como
46	2	1	4	0	0	0	a = 1
0	114	1	18	0	0	0	b = 2
0	1	60	22	0	0	0	c = 3
0	1	1	266	0	0	0	d = 5
0	0	0	3	38	0	0	e = 8
0	1	0	7	0	25	0	f = 13
0	0	0	1	0	0	8	g = 20

Tabela 6 – Matriz confusão gerada utilizando todos os atributos

Classe	TP Rate	FP Rate	Precision	Recall	F-Measure
1	0.906	0.002	0.98	0.906	0.941
2	0.887	0.01	0.959	0.887	0.922
3	0.807	0.013	0.905	0.807	0.854
5	0.974	0.116	0.864	0.974	0.916
8	0.927	0	1	0.927	0.962
13	0.758	0.002	0.962	0.758	0.847
20	0.889	0	1	0.889	0.941
Média Ponderada das Classes	0.911	0.055	0.916	0.911	0.911

Tabela 7 – Detalhamento dos resultados por Classe utilizando apenas os atributos selecionados pelo PCA

A matriz de confusão (Tabela 8) representa a quantidade exata de instâncias que foram classificadas erroneamente (horizontal e vertical) e corretamente (diagonal), inclusive mostrando para qual classe foi realizada a classificação, nos testes realizados na base com os atributos selecionados pelo PCA.

a	b	c	d	e	f	g	<– classificado como
48	0	1	4	0	0	0	a = 1
0	118	1	14	0	0	0	b = 2
1	2	67	13	0	0	0	c = 3
0	2	4	261	0	1	0	d = 5
0	0	0	3	38	0	0	e = 8
0	0	1	7	0	25	0	f = 13
0	1	0	0	0	0	8	g = 20

Tabela 8 – Matriz de confusão gerada utilizando apenas os atributos selecionados pelo PCA

6 Análise dos Resultados

Após a realização dos testes e consequente obtenção dos resultados anteriores, algumas análises devem ser feitas para avaliarmos se o sistema criado pode de fato substituir a experiência que os especialistas adquirem após anos de vivência e observação.

Esta análise é importante para evitar a utilização de resultados distorcidos por classes desbalanceadas, que podem levar-nos a crer que uma média boa de acertos para uma classe se reflete em outras, quando na verdade esta afirmação só é válida em determinadas classes. Utilizando como exemplo o banco de dados da Empresa X, podemos notar que a classe Tamanho 5 tem quase a metade das instâncias testadas. Ao longo desse capítulo veremos que os resultados são consistentes para todas as classes existentes.

6.1 Análise da Seleção de Atributos

Inicialmente, deve ser observado que a seleção dos melhores atributos visa melhorar o desempenho preditivo, reduzindo a quantidade de variáveis utilizadas sem perdas na objetividade. Quando comparamos os resultados obtidos nas duas bases de dados, percebemos quando utilizamos o PCA, que além de ganho de desempenho, temos uma ligeira melhora no percentual de acertos médio, pulando de 89.838% para 91.129%. Quando analisamos classe a classe, temos resultados também muito próximos, com ligeira melhora na taxa de verdadeiros positivos de Tamanho 5 com PCA, como visto no gráfico da Figura 23 e melhora considerável nos falsos positivos também para os testes com PCA.

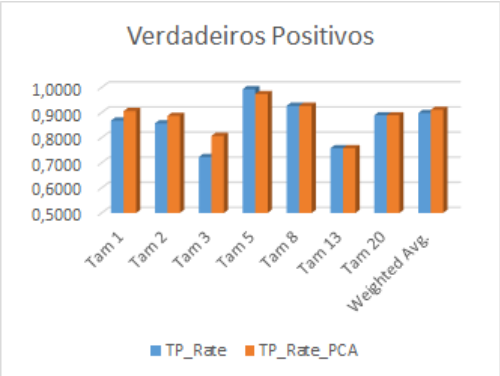


Figura 23 – Comparativo de Verdadeiros Positivos com e sem PCA, ligeira vantagem para o PCA no Tamanho 3.

A taxa de falsos positivos também se saiu melhor com o uso do PCA, a figura 24 mostra a diferença de 25% a menos chances de estimar uma tarefa em 5 sem que ela seja de fato deste tamanho. Devido à falta de tempo, não foram realizados testes com as classes balanceadas, porém seriam interessantes, pois nas bases existentes o tamanho 5 é muito mais representativo que os demais.

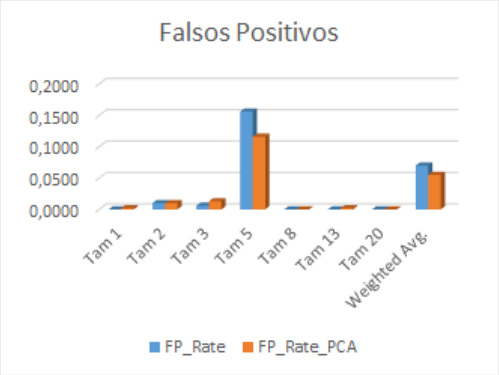


Figura 24 – Comparativo de falsos positivos com e sem PCA, vantagem para o PCA no Tamanho 5.

A Precisão, obtida através da divisão da quantidade de verdadeiros positivos (TP) pela soma dos verdadeiros positivos (TP) com os falsos positivos (FP) ($Precision = TP/(TP+FP)$) demonstra qual o percentual dos acertos do sistema é verdadeiro. Ou seja, de todas as classificações de determinada classe, quantos são de fato daquela classe.

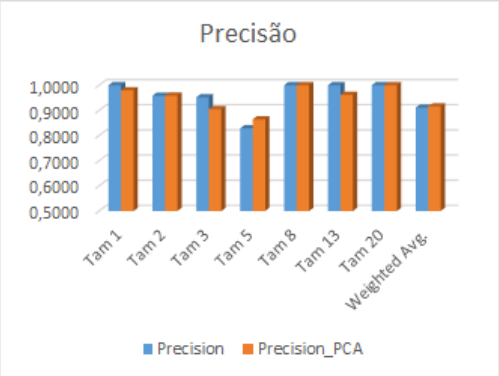


Figura 25 – Comparativo de Precisão com e sem PCA.

O Sensitividade ou Recall, é a porcentagem de classificações positivas corretas (TP) sobre o total de classificações positivas (TP+Falsos Negativos, FN), $Recall = TP/(TP+FN)$.

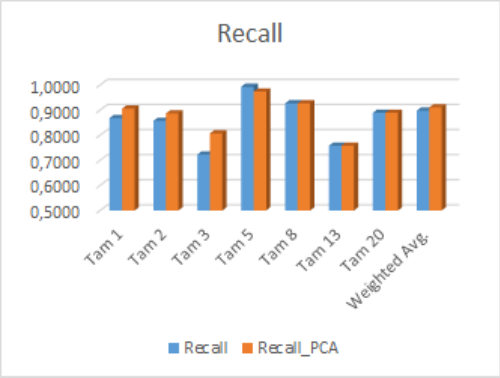


Figura 26 – Comparativo de Recall com e sem PCA.

F-Measure F_1 é a medição da fidelidade do teste, levando em consideração a precisão $precision$ e a sensitividade $recall$ e calculando a média harmônica entre ambos:

$$\mathrm{F}_1 = 2 \cdot \frac{1}{\frac{1}{\mathrm{recall}} + \frac{1}{\mathrm{precision}}} = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}$$

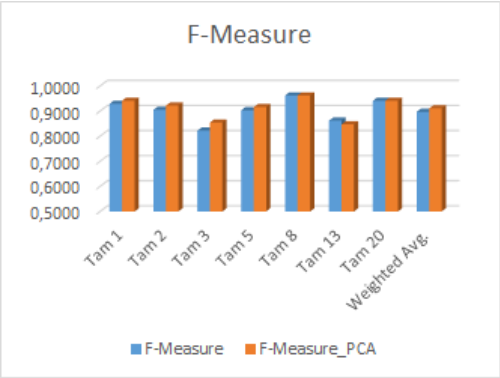


Figura 27 – Comparativo de F-Measure com e sem PCA.

Os comparativos de Precisão, Recall e F-Measure (Figuras 25 a 27) resumem uma situação sempre melhor ou igual no geral para os testes realizados após a seleção de atributos com PCA.

Os resultados obtidos com ambas as bases se mostram similares, com o PCA levando vantagem por identificar bem menos falsos positivos. Sendo assim, podemos concluir que a seleção de atributos pode e deve ser mais explorada, com a adoção de algorítimos mais modernos e eficientes. Para os objetivos em questão, o PCA se mostrou consistente e útil. Deixando claro que a maioria das palavras escritas em linguagem natural não colabora para a análise da máquina. Por exemplo, a palavra *Sistema*, presente em várias instâncias de todas as classes, não trás nenhum ganho à interpretação feita pela máquina, sendo desnecessária.

6.2 Resultados Por Classe (Tamanho)

No uso real, cada Classe é um Tamanho de uma História de Usuário (HU), como as *Sprints* são compostas por várias HU, o Tamanho das mesmas está diretamente relacionado com as expectativas do *Product Owner* e do Time a respeito das entregas. Basicamente, o Time tem uma capacidade de desenvolvimento por *Sprint* e essa capacidade vai ser preenchida com os Tamanhos das HU até que chegue ao limite.

De forma simples, a importância de ser assertivo na estimativa é que o Time pode ficar ocioso, caso as HU sejam estimadas acima da realidade ou que as entregas ao Cliente não sejam realizadas, por terem estimado as HU menores do que são na verdade.

Eis que nas Tabelas 5 e 7 temos as medidas Verdadeiros Positivos(*TP Rate*), Falsos Positivos (*FP Rate*), Precisão (*Precision*), *Recall* e *F-Measure* que nos interessam.

As matrizes de confusão das Tabelas 4 e 6 apresentam os erros de classificação no detalhe e nos possibilitam ver e medir quão distante do Tamanho correto as HU foram classificadas. Esta informação é bastante pertinente quando da discussão do quão custosos os erros podem ser para o projeto.

Apesar da média ponderada dos verdadeiros positivos de ambas as Tabelas 5 e 7 serem boas e bem próximas, podemos notar que os piores resultados são nos valores de Tamanho 3 e 13. O impacto desses erros pode ser visto ao relacionar essa informação com as matrizes de confusão. Temos que a maioria dos Tamanhos 13 assim como a maioria dos Tamanhos 3 que não foram identificados corretamente, foram considerados 5. Para a classe 13, o erro é normalmente fácil de ser identificado por qualquer integrante do Time pois a diferença é considerável. Porém, para o Tamanho 3, fica mais difícil de enxergar a diferença, por estar muito próximo de 5. Apesar da dificuldade, os problemas devido ao erro são poucos quando as equipes trabalham com *Sprints* curtas.

Os erros maiores, do tipo Tamanho 20 estimado como tamanho 1 ou vice versa, são todos facilmente identificados na fase inicial de desenvolvimento. A matriz de confusão da Tabela 8 mostra excelentes resultados, os Tamanhos consecutivos com maior quantidade de erros são 3 e 5, como mostra a Figura 28, apesar de não serem facilmente identificados, não geram grandes discrepâncias na estimativa de custos e prazos de uma *Sprint*.

a	b	c	d	e	f	g	<– classified as
48	0	1	4	0	0	0	a = 1
0	118	1	14	0	0	0	b = 2
1	2	67	13	0	0	0	c = 3
0	2	4	261	0	1	0	d = 5
0	0	0	3	38	0	0	e = 8
0	0	1	7	0	25	0	f = 13
0	1	0	0	0	0	8	g = 20

Figura 28 – Detalhe da Matriz de Confusão da Tabela 8

O percentual de falsos positivos de tamanhos grandes, 13 e 20, pode ser considerado muito bom, apenas o tamanho 13 dos testes com PCA resultou diferente de zero e mesmo assim foi de apenas $0,002\%$, ou seja, a possibilidade de deixar uma equipe ociosa por superestimar tarefas é quase nula, ainda mais quando visto que o erro veio de um Tamanho 5 que foi classificado como 13, diferença simples de ser notada e corrigida por um desenvolvedor.

Como o *Scrum* estimula entregas em curtos e constantes intervalos de tempo, pequenas divergências de dimensionamento não causam grandes problemas. Tarefas classificadas como tamanho 2 que na verdade são 3, por exemplo, pouco influenciam no resultado final de uma *Sprint* de tamanho total 100. Essas pequenas divergências (1 ponto) inclusive podem passar despercebidas em determinadas situações. No caso de erros grosseiros, acontece algo parecido pois fica fácil identificar e corrigir as diferenças.

Encerrando as observações sobre os resultados obtidos, temos uma ferramenta que apesar de ter espaço para diversas melhorias, pode ser utilizada em produção, poupando recursos sem causar problemas para a equipe de desenvolvimento.

7 Conclusões

Os objetivos deste trabalho foram desenvolver uma solução de estimativas automáticas de Histórias de Usuário (HU) baseado nos dados históricos das equipes de desenvolvimento e analisar a eficácia do mesmo.

Como apresentado no decorrer da leitura, ao final da etapa de experimentos foi constatado que é possível substituir o trabalho humano especializado por uma Máquina de Vetores de Suporte (SVM). A mesma foi criada com o auxílio de ferramentas e bibliotecas de código aberto e disponíveis gratuitamente no mercado, como *Python, NLTK*, *Scipy e Weka*, sendo o conjunto de artefatos gerados por elas de relativa fácil adaptação.

Os resultados obtidos foram bastante positivos, chegando à generalização de 91% quando utilizados atributos selecionados através do PCA. E isto comprova que nem todos os atributos levantados inicialmente na etapa de pré-processamento são relevantes para a obtenção da estimativa pela SVM.

O uso dos tamanhos pré definidos também se mostrou eficiente para a solução das estimativas, uma vez que é possível ter uma granularidade razoável, indispensável para medir custos e avanços nos projetos, e simplificar os resultados possíveis para a máquina. Com isso conseguimos ter um desempenho satisfatório de uma maneira geral.

Considerando todos os passos do trabalho, é possível economizar esforço de dimensionamento em contextos que tenham um número razoável de instâncias para serem treinadas.

7.1 Trabalhos futuros

Futuramente, seria interessante comparar a efetividade do método proposto em contextos diferentes, como empresas que estimem seus requisitos em Pontos de Função ou Horas. Isto colocaria em prova o uso de SVM com valores de classes contínuos.

Também vale salientar que a solução proposta pode variar. Caso haja tempo disponível, uma comparação entre o desempenho da solução TF-IDF, PCA e SVM com outras combinações de seletores de atributos e de classificadores, como o *Classifier Subset Evaluator*, *Naive-Bayes* e Redes Neurais Artificiais na tentativa de elevar ainda mais o percentual de acertos.

Vale salientar a importância de se ter um dimensionador automático preciso

para ser ser utilizado com outras soluções de automatização de processos da Engenharia de *Software*, como o Next Release Problem (MOURA, 2014).

Também foi verificada a possibilidade de, com um pouco mais de codificação e estudo de bibliotecas específicas, realizar todas as tarefas em Python, pois o mesmo dispõe de uma enorme variedade de ferramentas para este fim.

Referências

ALBRECHT, A. J. Medindo a produtividade do desenvolvimento de aplicativos. *Proc. Joint SHARE/GUIDE/IBM Application Development Symposium (October, 1979),*, IBM Corporation, 1979. P. 83 - 92. Citado na página 21.

ALLIANCE, S. The 2015 state of scrum report. Scrum Alliance, 2015. Citado 2 vezes nas páginas 11 e 22.

Celso A. A. Kaestner, J. . A. Automatic text summarization using a machine learning approach. *Pontifical Catholic University of Parana (PUCPR)*, Brazilian Symposium on Artificial Intelligence, 2002. Citado 2 vezes nas páginas 25 e 31.

E. Fenton, Norman ; L. Pfleeger, Shari. *Software Metrics: A Rigorous and Practical Approach*. [S.l.]: International Thomson Computer Press, 1997. Citado na página 16.

George Michael, J. . L. J. T. Improving and expanding nasa software cost estimation methods. *IEEE Conference Publications*, 2016 IEEE Aerospace Conference, 2016. Citado na página 14.

Grimstad, Stein; Jørgensen, Magne. The impact of irrelevant information on estimates of software development effort. IEEE Computer Society, 2007. Citado na página 14.

JOACHIMS, T. Text categorization with support vector machines: Learning with many relevant features. *Universit at Dortmund*, Universit at Dortmund, 1998. Citado 2 vezes nas páginas 15 e 26.

Ken-ichi Matsumoto, P. . Jacky Keungand. An empirical experiment on analogy-based software cost estimation with cuda framework. *IEEE Conference Publications*, 2013 22nd Australian Conference on Software Engineering, 2013. Citado na página 14.

Khoshgoftaar,Taghi M., I. A. Estimating software project effort by analogy based on linguistic values. IEEE Computer Society, 2002. Citado na página 12.

KOWALCZYK, A. *SVM-Tutorial*. [S.l.]: http://www.svm-tutorial.com/ acessado em 11/2016, 2016. Citado 3 vezes nas páginas 32, 33 e 34.

MIRANDA, E. Fermi questions to estimate software development projects. *Carnegie Mellon University*, Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement, 2014. Citado na página 16.

MOURA, M. Algoritmos de otimização para solução do problema do próximo release. *Universidade Federal Rural de Pernambuco*, Trabalho de Conclusão do Curso de Bacharelado em Sistemas de Informação, 2014. Citado na página 45.

PMI, P. M. I. *A Guide to the Project Management Body of Knowledge (PMBOK Guide)*. 5. ed. [S.l.]: Project Management Institute, 2013. Question Metric paradigm: p. 527-532. ISBN 978-1-935589-67-9. Citado na página 21.

PRESSMAN, R. S. *Engenharia de Software*. 6. ed. [S.l.]: McGrawHill, 2006. ISBN 8586804576. Citado na página 21.

Rombach, H.Goal, B. . C. . *Encyclopedia of Software Engineering*. 2. ed. [S.l.]: Pearson, 1994. Question Metric paradigm: p. 527-532. Citado na página 16.

Schwaber, Ken; Sutherland, Jeff. *Guia Scrum*. [S.l.]: Scrum.Org and ScrumInc, 2013. Citado na página 22.

Shepperd, M.; Schofield, C. Estimating software project effort using analogies. *IEEE Journals and Magazines*, IEEE Transactions on Software Engineering, v. 23, 1997. Citado na página 14.

SOARES, R. Uso de meta-aprendizado para a seleção e ordenação de algoritmos de agrupamento aplicados a dados de expressão genica. *Universidade Federal de Pernambuco*, Dissertação de Mestrado, 2008. Citado 3 vezes nas páginas 15, 26 e 27.

SOFTEX. *MPS.BR Guia Geral Software*. [S.l.]: SOFTEX, 2016. Processo: Gerência de Projetos GPR. Citado na página 11.

SWEBOK. *Software Engineering Body of Knowledge*. [S.l.]: École de technologie supérieure (ÉTS), 2004. P. 1-14. Citado na página 11.

WIKIPEDIA. *Use Case Points — Wikipedia, The Free Encyclopedia*. 2016. [Online; accessed 20-October-2016]. Disponível em: <https://en.wikipedia.org/w/index.php?title=Use_Case_Points&oldid=745276545>. Citado na página 21.

Xin Chen, Y. . Q. R. Finding nuggets in documents: A machine learning approach. *New Jersey Institute of Technology*, JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE AND TECHNOLOGY, 2006. Citado na página 25.