TIAGO GERALDO FERREIRA

NICESIM: UM SIMULADOR INTERATIVO BASEADO EM TÉCNICAS DE APRENDIZADO DE MÁQUINA PARA AVALIAÇÃO DE RECÉM-NASCIDOS PREMATUROS EM UTI NEONATAL

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, para obtenção do título de *Magister Scientiae*.

VIÇOSA
MINAS GERAIS-BRASIL
2014

Ficha catalográfica preparada pela Seção de Catalogação e Classificação da Biblioteca Central da UFV

T
F383n
2014

Ferreira, Tiago Geraldo, 1989-
NICeSim : um simulador interativo baseado em técnicas de aprendizado de máquina para avaliação de recém-nascidos prematuros em UTI neonatal / Tiago Geraldo Ferreira. – Viçosa, MG, 2014.
xi, 58f. : il. (algumas color.) ; 29 cm.

Inclui anexo.
Orientador: Fábio Ribeiro Cerqueira.
Dissertação (mestrado) - Universidade Federal de Viçosa.
Referências bibliográficas: f.34-36.

1. Aprendizado do computador. 2. Inteligência artificial. 3. Bioinformática. 4. Simulação computacional. I. Universidade Federal de Viçosa. Departamento de Informática. Programa de Pós-graduação em Ciência da Computação. II. Título.

CDD 22. ed. 006.31

TIAGO GERALDO FERREIRA

NICESIM: UM SIMULADOR INTERATIVO BASEADO EM TÉCNICAS DE APRENDIZADO DE MÁQUINA PARA AVALIAÇÃO DE RECÉM-NASCIDOS PREMATUROS EM UTI NEONATAL

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, para obtenção do título de *Magister Scientiae*.

APROVADA: 27 de fevereiro de 2014.

Alcione de Paiva Oliveira
(Coorientador)

Rodrigo Siqueira Batista

Fábio Ribeiro Cerqueira
(Orientador)

À mulher da minha vida Neliza
Aos meus amigos e família

"A sabedoria começa na reflexão."
(Sócrates)

Agradecimentos

Agradeço a minha mulher, Neliza, pelo apoio incondicional e compreensão.

Agradeço meus amigos, pelas alegrias, tristezas e dores compartilhas. Com vocês, as pausas entre um parágrafo e outro de produção tornam melhor tudo o que tenho produzido na vida.

Agradeço meus pais, irmãos e família, que com muito carinho e apoio, não mediram esforços para que eu chegasse até esta etapa de minha vida.

Ao professor Fábio, pela paciência, incentivo e sabedoria na orientação. Sem seus conselhos este trabalho não seria possível.

Obrigado a todas as pessoas que contribuíram para meu sucesso e para meu crescimento como pessoa. Sou o resultado da confiança e da força de cada um de vocês.

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

FERREIRA, Tiago Geraldo, M.Sc., Universidade Federal de Viçosa, fevereiro de 2014. **NICeSim: um simulador interativo baseado em técnicas de aprendizado de máquina para avaliação de recém-nascidos prematuros em UTI neonatal.** Orientador: Fábio Ribeiro Cerqueira. Co-Orientador: Alcione de Paiva Oliveira.

Este trabalho descreve o NICeSim, um simulador disponível gratuitamente e de código aberto que usa técnicas de aprendizagem de máquina para auxiliar os profissionais de saúde na obtenção de uma melhor avaliação, em termos de prognóstico, de recém-nascidos prematuros em unidades de terapia intensiva neonatal. A aplicação foi desenvolvida e testada usando um banco de dados coletado em um hospital público de ensino localizado na cidade de Viçosa, em Minas Gerais. Os dados disponíveis foram utilizados para alimentar um pipeline de aprendizagem de máquina que foi projetado para criar um simulador capaz de prever a probabilidade de morte para recém-nascidos internados em unidades de terapia intensiva neonatal. Aqui, apresentamos as técnicas que são a base do simulador, algoritmos de redes neurais e máquinas de vetores suporte foram utilizados como motor e aprendizagem para a ferramenta desenvolvida. Aprensentaremos também alguns resultados obtidos com o conjunto de dados mencionado acima. Nossos experimentos estatísticos mostraram que o modelo resultante para predição de óbito alcançou uma precisão de 86,7% nos melhores casos. Esta exatidão significativa do NICeSim demonstra que o mesmo pode ser utilizado para testes de hipóteses. De fato, em um experimento realizado por dois médicos, três atributos principais foram avaliados para compreender como eles afetam o risco de morte. Os resultados mostraram que o modelo fornece previsões que estão em boa concordância com a literatura, demonstrando que NICeSim pode ser uma importante ferramenta de apoio à tomada de decisão na prática clínica. Mais do que isso, o método pode ser utilizado como um molde para a criação de soluções computacionais semelhantes para

outros cenários de interesse que existam em problemas de domínios diferentes, desde que dados adequados sejam providos.

Abstract

FERREIRA, Tiago Geraldo, M.Sc., Universidade Federal de Viçosa, February of 2014. **NICeSim: an interactive simulator based on machine learning techniques to evaluate newborns in neonatal ICU.** Adviser: Fábio Ribeiro Cerqueira. Co-Adviser: Alcione de Paiva Oliveira.

This work describes the NICeSim, a freely available and open source simulator that uses machine learning techniques (ML) to assist health professionals in obtaining a better assessment in terms of prognosis of preterm infants in neonatal intensive care units (neonatal ICUs). The application was developed and tested using a database collected in a public teaching hospital located in the city of Viçosa, Minas Gerais. The available data were used to feed a pipeline of ML that was designed to create a simulator able to predict the probability of death for newborns admitted in neonatal ICUs. Here, we present the techniques used as base for the development of the simulator, artificial neural network and support vector machines algorithms were used as the machine learn engines of the application. We'll also discuss some results from statistical and practical tests using the data set mentioned above. Our statistical experiments showed that the resulting model to predict death achieved an accuracy of 86.7% in best case scenarios. This significant accuracy demonstrates that NICeSim can be used for hypothesis testing. In fact, in an experiment conducted by two doctors, three main attributes were evaluated to understand how they affect the risk of death. The results showed that the model provides predictions that are in good agreement with the literature, showing that NICeSim can be an important tool to support decision making in clinical practice. More than that, the method can be used as a template for the creation of similar solutions to other computing scenarios of interest, even if those belong to different problem domains, given that appropriate data is provided.

Capítulo 1

Introdução

1.1 Mineração de dados e Aprendizagem de Máquina

Mineração de dados é a tarefa de procurar por padrões existentes dentro de um dado conjunto de dados. Esta tarefa é parte de um processo maior chamado Extração de Conhecimento (ou KDD, do inglês *knowledge-discovery in databases*). Não existe nada de novo em procurar por padrões, mas a mineração de dados, no entanto, lida com a automação deste processo através do uso de computadores, tornando o armazenamento de dados e descoberta de informações uma tarefa computacional. Algoritmos de aprendizagem são utilizados como parte deste processo a fim de treinar computadores na tarefa de reconhecimento de padrões [Bishop, 2006].

A mineração de dados tem sido utilizada em diversos campos de conhecimento, tais como: sistemas de apoio a decisão que fazem parte do dia a dia das empresas [Davenport & Harris, 2007], algoritmos de aprendizagem que utilizam-se de técnicas de mineração para aprimorar a inteligência artificial de agentes computacionais em jogos [Weber & Mateas, 2009], meteorologistas que também fazem amplo uso da mineração de dados para analisar o histórico climático de uma região a fim de obter previsões [Cofiño et al., 2003 e na medicina em que a mineração de dados é utilizada para prever, entre outras coisas, o prognóstico de pacientes, reações a medicamentos e chances de rejeição em transplantes [Nakayama et al., 2012][Kang & Auinger, 2012][Hollander et al., 2012].

1.2 O problema

Uma das ferramentas de mineração de dados mais utilizada no meio acadêmico é a suíte Weka [Piatetsky-Shapiro, 2014]. Esta suíte agrupa diversos algoritmos de mineração de dados e tarefas afins. Usuários desta ferramenta podem analisar grandes massas de dados com diversos tipos de algoritmos, podendo-se obter informações sobre seus atributos, relações entre as instâncias existentes e predições para entradas futuras do conjunto de dados.

Apesar de fornecer todas estas possibilidades de análise, a suíte Weka não contém um módulo que permita ao usuário ter uma interação direta e interativa com seus algoritmos. De modo a realizar facilmente mudanças nos parâmetros de entrada de um problema a fim de observar variações nos valores de saída.

1.3 Solução Proposta e Objetivo Geral

Esta dissertação descreve o desenvolvimento e teste do aplicativo NICeSim, um simulador capaz de fornecer respostas interativamente, reagindo a modificações realizadas pelo usuário em instâncias de um problema previamente modelado. Essa ferramenta é voltada para o auxílio na compreensão de problemas, objetivando auxiliar na tomada de decisões não sobre instâncias específicas, mas sim sobre a maneira de lidar com determinado problema como um todo.

Neste trabalho utilizamos para testes um conjunto de dados médicos de pacientes recém nascidos que foram internados em Unidades de Tratamento Intensivo (UTI) neonatais. Trabalhos prévios nesta área visavam estudar o uso de sistemas de apoio à decisão dentro de UTIs neonatais [Monique Frize, 2000], [Monique Frize, 2001] e [Christine L Tsien, 2000]. Diferente disto, a simulação provida pelo aplicativo NICeSim objetiva fornecer uma ferramenta para que médicos e instituições médicas e governamentais possam estudar o impacto de diversos fatores envolvidos no tratamento de recém-nascidos em UTIs a fim de melhorar suas estruturas e processos. A aplicação NICeSim foi desenvolvida como um trabalho colaborativo entre o Departamento de Informática e o Departamento de Medicina e Enfermagem da Universidade Federal de Viçosa. Todos os dados de teste utilizados durante o desenvolvimento da aplicação foram coletados na UTI neonatal do hospital público São Sebastião, no município de viçosa, Minas gerais. Os dados foram coletados durante os anos de 2008 e 2010.

Durante o trabalho apresentaremos avaliações estatísticas da ferramenta e também uma avaliação do uso prático da mesma.

1.4 Objetivos Específicos

- Estruturar e desenvolver um aplicativo que faça uso da biblioteca Weka para prover um simulador interativo de uso genérico;
- Modelar o problema de tratamento de recém nascidos em UTIs neonatais para realização de testes práticos e estatísticos do simulador desenvolvido.

1.5 Aspectos Éticos

Esta pesquisa envolve seres humanos como participantes (dados coletados na UTI neonatal). Assim sendo, o projeto foi submetido para análise e aprovação pelo Comitê de Ética para Pesquisa (CEP) da Universidade Federal de Viçosa, de acordo com a Resolução 196/1996 - já revogada - e a Resolução 466/2012 do Conselho Nacional de Saúde do Brasil.

1.6 Estrutura e Organização

Este trabalho foi dividido em 5 seções principais. A primeira trata-se desta seção, onde são apresentadas informações introdutórias sobre o assunto abordado e os objetivos gerais e específicos deste trabalho. A segunda seção contém a revisão bibliográfica das metodologias e ferramentas utilizadas como base para este trabalho. Na seção 3 o desenvolvimento do trabalho é apresentado. E esta seção contém mais informações sobre o processo de obtenção do conjunto final de dados utilizado, os algoritmos de mineração de dados utilizados, diagramas de classe e sequência detalhando a implementação da aplicação e uma série de imagens e explicações sobre o uso do simulador. Na quarta seção serão apresentados os resultados obtidos durante os testes realizados com o simulador e serão discutidas possíveis melhorias ao aplicativo. A seção de número 5 apresenta as conclusões do trabalho e disserta sobre o alcance dos objetivos inicialmente estipulados. Após as conclusões temos o apêndice de anexos, onde o artigo publicado baseado no trabalho aqui descrito pode ser visto.

Capítulo 2

Revisão Bibliográfica

2.1 Mineração de Dados e Extração de Conhecimento

Com o passar dos anos, os dispositivos computacionais tornaram-se comuns como ferramentas de trabalho, o que levou a uma maior disponibilização de dados em quase todos os campos da ciência e dos negócios. Segundo Reddy 2011 a Extração de Conhecimento (KDD, do inglês: *Knowledge Discovery in Databases*) pode ser definida como o processo que recebe dados como entrada e provê como saída informações úteis.

A mineração de dados é o passo específico responsável pela análise de dados do KDD. E embora não seja um sinônimo, o termo tem sido usado muitas vezes para fazer referência a todo o processo de KDD e referências futuras a mineração de dados neste trabalho irão fazer o mesmo.

Para entender melhor o que a mineração de dados é capaz de fazer, é preciso analisá-la a partir de um escopo de alto nível. Abaixo estão descritos os principais conceitos de mineração de dados que, de alguma forma, relaciona-se com o trabalho aqui realizado.

Classificação e Regressão

Neste trabalho utilizamos uma categoria de mineração de dados conhecida como classificação e regressão. Tarefas que se encaixam nesta categoria utilizam dados disponíveis para construir um modelo capaz de prever o valor de uma nova instância do mesmo tipo. Usando este tipo de algoritmo podemos prever, por exemplo, qual será o prognóstico de um paciente com base nas informações disponíveis sobre casos passados com

os mesmos sintomas [Hollander et al., 2012]. Este é o tipo de mineração utilizada pelo aplicativo NICeSim a fim de obter estimativas de saída durante a simulação. Mas existem diversos outros tipo de mineração de dados que não abordaremos neste trabalho como, por exemplo: regras de associação, análise de cluster, mineração de texto e análise de conexões.

O processo de KDD

O processo de KDD é interativo e iterativo, ou seja, é divido em uma série de etapas que podem ter seus resultados consideravelmente melhorados através da supervisão e da tomada de decisões a serem feita pelo usuário. Por conseguinte, o sucesso do processo de KDD é altamente dependente da compreensão do domínio do problema e das etapas que compõe o processo de extração do conhecimento [Fayyad et al., 1996].

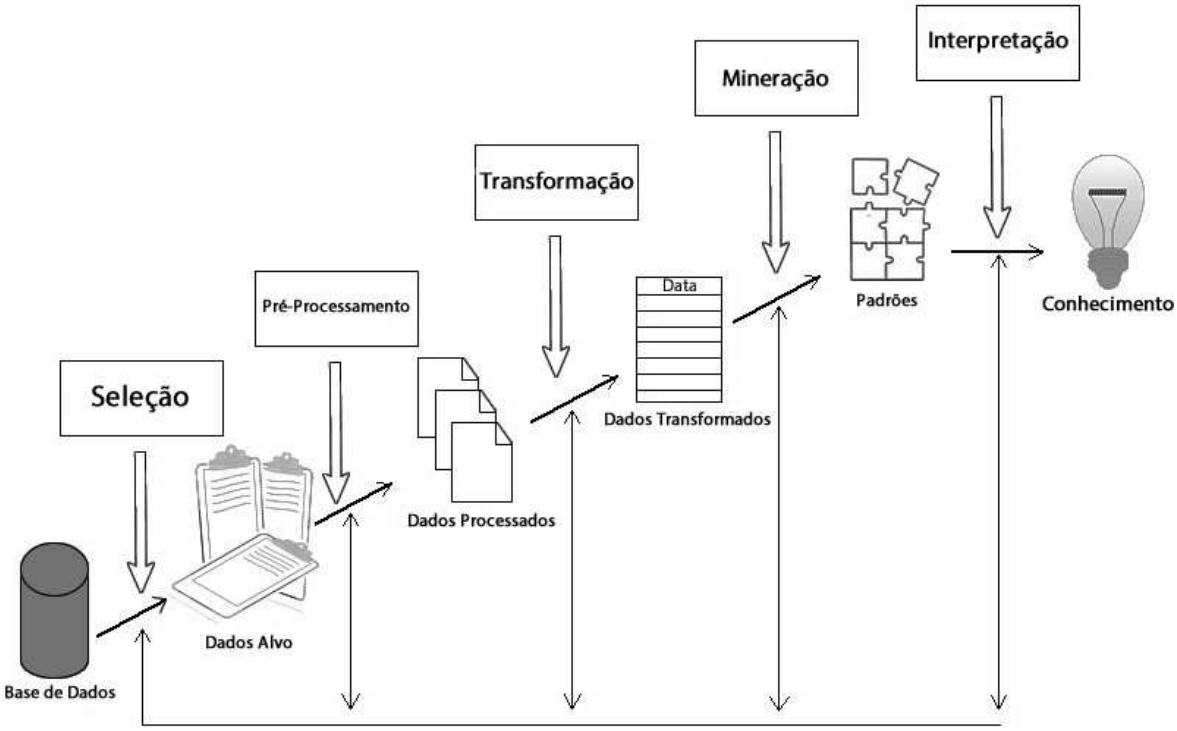

Figura 2.1. O processo de KDD

O KDD pode ser definido nos seguintes estágios:

- Seleção
- Pré-Processamento
- Transformação

- Mineração de Dados
- Interpretação / Avaliação

Como podemos ver na Figura 1, para iniciar o processo KDD precisamos primeiro selecionar os dados relevantes de todos os dados disponíveis. O conjunto de dados selecionado é frequentemente denominada conjunto alvo. Em seguida é preciso pré-processar o conjunto de dados alvo. Esta etapa inclui tarefas como manejamento de dados faltantes, remoção de possíveis ruídos presentes no conjunto de dados e correção de valores inconsistentes que podem estar presentes. A escolha de um bom conjunto de atributos, que não seja redundante, é também uma tarefa importante neste passo, e será explicada em mais detalhes numa seção futura.

Depois de adquirir os dados alvo e pré processá-los, é preciso transformar esses dados. Mais especificamente, precisamos transformar os atributos para representar melhor os dados que estamos trabalhando e para auxiliar o algoritmo de mineração de dados na tarefa de descoberta de padrões. Alguns exemplos de transformações realizadas neste passo são as normalizações dos valores de atributos e a discretização das características numéricas presentes na base de dados.

Depois de toda a preparação de dados realizado nas etapas anteriores agora chegamos à fase de mineração de dados. Nesta fase um ou mais algoritmos de mineração de dados serão escolhidos para analisar os dados com o objetivo de encontrar padrões e informações úteis. Esta etapa é responsável por executar as tarefas de classificação e predição citadas anteriormente. A execução das etapas anteriores corretamente pode melhorar significativamente o desempenho de mineração de dados. Características redundantes, ruídos ou qualquer outra inconsistência de dados podem, sem dúvida, interferir na tarefa de detecção de informações relevantes [Witten & Frank, 2005]. Algumas opções comuns de algoritmos de mineração de dados, no que se refere a classificação, são: redes bayesianas, árvores de decisão, redes neurais e máquinas de vetor de suporte [Han, 2005]. Esses dois últimos foram utilizados neste trabalho e explicados em mais detalhes em seções posteriores.

A última etapa do processo de KDD é fortemente dependente da interação do usuário. É na fase de interpretação que o usuário avalia os padrões obtidos a fim de decidir se houve de fato obtenção de conhecimento. Nesta etapa o usuário decide se uma análise mais aprofundada será necessária ou se o conhecimento que se almejava alcançar já foi descoberto. É importante notar que todo o processo de KDD apoia e incentiva feedback. A qualquer momento o usuário poderá voltar uma ou mais etapas, a fim de tentar alcançar melhores resultados. Neste trabalho o processo de preparação

dos dados foi reiniciado diversas vezes utilizando a cada nova iteração os conhecimentos obtidos anteriormente.

2.2 Seleção de Atributos

Durante o desenvolvimento e teste da aplicação NICeSim a seleção de atributos foi um dos passos mais importantes. Como será visto em maiores detalhes na seção de Materiais e Métodos nossa base de dados inicial continha 114 atributos, e destes somente 4 estavam presentes no conjunto de dados final após todo o processamento realizado.

Dentro da área de mineração de dados, a seleção de atributos é um problema de otimização combinatória. Neste problema, dado um conjunto de atributos que descrevem o domínio de uma base de dados, deve-se selecionar o subconjunto ótimo dos atributos mais relevantes para a construção de um modelo de classificação.

Como visto em Guyon & Elisseeff 2003, a eliminação de atributos irrelevantes pode gerar melhora em diversos aspectos do processo de construção e utilização de um modelo de classificação. Dentre estas melhoras destacam-se a geração de modelos de classificação mais compreensíveis, menor tempo computacional necessário para geração e execução dos modelos, abrandamento dos efeitos da maldição da alta dimensionalidade (*curse of dimensionality*) e aumento da capacidade de generalização do problema.

Os algoritmos presentes na literatura para resolução do problema de seleção de atributos, na etapa anterior à geração do modelo de classificação, se dividem em dois tipos, wrappers e filtros. Algoritmos do tipo wrapper presam ao máximo pela acurácia da função de avaliação dos subconjuntos analisados. Para cada subconjunto de atributos que se deseja avaliar os algoritmos wrappers constroem um modelo classificatório e testam o desempenho do mesmo. Uma técnica wrapper realiza a contagem do número de erros ao utilizar um conjunto determinado de atributos presentes na base e então utiliza estes erros para atribuir uma pontuação a estes atributos, e através da repetição deste processo gerar um ranking de desempenho para diferentes conjuntos. Esta abordagem gera resultados precisos na função de avaliação, mas são muito ineficientes em termo de escalabilidade. Na medida em que o espaço de soluções aumenta, abordagens wrappers rapidamente se tornam inviáveis devido ao seu elevado custo computacional.

Outra categoria existente de algoritmos de seleção de atributos são os algoritmos de Filtro. Técnicas de filtro não fazem uso do modelo de predição, estes algoritmos analisam diretamente determinadas características e relacionamentos entre os atributos existentes, sem a necessidade de executar o algoritmo de aprendizado para pontuar

e ranquear os atributos. Esta avaliação pode ser feita com diferentes técnicas que normalmente se dividem em técnicas de avaliação individual ou coletiva. Técnicas de avaliação individual analisam cada atributo pertencente a instância do problema de maneira isolada, gerando então um ranque de atributos organizados pela sua relevância. Exemplos de métricas de avaliação para ranqueamento de atributos são Ganho de Informação (InfoGain) e Relief [Robnik-Sikonja & Kononenko, 1997].

Outras técnicas utilizadas são aquelas de avaliação coletiva. Este tipo de método realiza a avaliação dos atributos pertencentes ao subconjunto de maneira coletiva, preocupando-se não só com a relevância isolada de cada elemento, mas também com sua interação com os demais atributos pertencentes ao subconjunto. Esta característica é muito importante, pois em situações reais quase sempre não é o bastante selecionar os n atributos mais relevantes, com análise isolada de cada um, e declarar o subconjunto resultante como de alta qualidade. Problemas com redundância entre os atributos de um subconjunto são frequentes e é melhor trata-los.Um exemplo de Filtro com avaliação coletiva é a técnicas baseadas em Correlação.

O simulador NICeSim fornece ao usuário um ranking de estimativa de mérito dos atributos obtido através da execução do algoritmo InfoGain. Este algoritmo foi escolhido devido ao seu rápido desempenho computacional e por ter apresentado bons resultados com o conjunto de testes utilizado neste trabalho. Durante o processo de preparação da base de dados, antes da execução da mesma no simulador NICeSim, a biblioteca Weka foi utilizada para execução de uma série de testes wrapper que serão detalhados em seções futuras.

2.3 A biblioteca Weka

A suíte Weka (do inglês: *Waikato Environment for Knowledge Analysis*) é uma ferramenta que existe desde 1992 e que almeja ser um repositório unificado e de fácil acesso a técnicas de aprendizagem de máquina. Por muitos anos no meio acadêmico a Weka tem sido um repositório de algoritmos e métodos com livre acesso para cientistas. Outra importante característica da Weka é sua estrutura de arcabouço que permite a pesquisadores interessados implementar novos algoritmos e integrá-los à biblioteca [Hall et al., 2009].

Nos dias atuais a Weka é reconhecida como um sistema chave na área de pesquisa de mineração de dados e aprendizagem de máquinas. Ela adquiriu ampla aceitação pela academia e por grandes empresas [Piatetsky-Shapiro, 2014].

Este trabalho fez uso tanto da aplicação quanto da biblioteca Java disponibilizada

como parte da suíte Weka. Todos os algoritmos de aprendizagem e seleção de atributos utilizados neste trabalho foram providos pela Weka. As funcionalidades do simulador NICeSim podem ser vistas como uma extensão desta suíte, permitindo que usuários realizem simulações de maneira interativa, algo que antes não era possível somente com o ambiente Weka.

Capítulo 3

Materiais e Métodos

3.1 Conjunto de Dados

Origem

O simulador NICeSim foi desenvolvido tendo em mente ajudar na compreensão de problemas reais através do uso de técnicas de aprendizagem de máquina e mineração de dados de maneira interativa, fornecendo ao usuário estimativas sobre a importância das variáveis que fazem parte do problema e permitindo o acompanhamento detalhado das mudanças na variável de saída ao se modificar as entradas.

Para testar a eficácia da aplicação NICeSim escolhemos utilizar o problema do tratamento de recém nascidos em UTI neonatais, uma situação que tem sido foco de estudos do Departamento de Medicina e Enfermagem da Universidade Federal de Viçosa por alguns anos. Os dados aqui apresentados foram coletados no Hospital de Ensino São Sebastião na cidade de Viçosa, Minas Gerais. Estes dados foram coletados inicialmente para a realização de pesquisas sobre a sepse neonatal [Freitas et al., 2012], uma doença que afeta recém nascidos nos primeiros dias de vida. O hospital onde os dados foram coletados se tornou uma unidade referência no serviço de saúde para gravidezes de alto risco em 2009. Sua UTI neonatal foi inaugurada em março de 2004 e atendeu um total de 1059 casos até dezembro de 2010, onde 70% destes foram tratamentos de prematuros [Freitas, 2011].

Preparaçao dos Dados

A primeira preocupação durante o design e desenvolvimento do NICeSim foi a de preparar os dados disponíveis para que estes estivessem alinhados com nossos objetivos. O conjunto de dados inicial possuía 114 atributos e 293 instâncias. Este conjunto de dados não estava balanceado, 39 das instâncias eram casos positivos (onde houve a morte do recém-nascido) e o restante das instâncias eram de casos negativos. A fim de remediar esta situação foi decidido o uso de um algoritmo sensível ao custo (*cost sensitive*), onde uma matriz de custo é utilizada para ditar penalidades ao algoritmo de aprendizagem quando este erra uma classificação fornecendo um resultado falso negativo. Em outras palavras a base de dados foi artificialmente balanceada aumentando-se o custo de erro de má classificação para a classe menos presente. Este recurso de sensibilidade ao custo foi desenvolvido na aplicação de forma que a mesma será capaz de analisar qualquer conjunto válido de entradas fornecido e então sugerir uma matriz de custo que balanceie este conjunto para que o algoritmo escolhido (ANN ou SVM) seja capaz de aprender com maior eficácia.

O próximo passo foi identificar quais atributos eram relevantes dentro dos 114 iniciais. Na realização desta tarefa o conjunto de dados foi submetido a um processo de seleção de atributos. Seleção de atributos (no ingles: *Feature Selection*) é uma das etapas do processo de extração do conhecimento. Essa etapa tem como objetivo diminuir a complexidade do conjunto de dado através da redução da dimensionalidade do mesmo [Guyon & Elisseeff, 2003]. A primeira parte do processo de seleção foi a utilização do conhecimento prévio do assunto [Freitas, 2011] para remover atributos que não seriam relevantes para a simulação a ser feita. Após a remoção destes atributos, restaram aqueles presentes na tabela 3.1. Após esta primeira etapa de análise humana dos atributos passamos a trabalhar na seleção computacional dos atributos.

Como a primeira etapa da seleção reduziu consideravelmente a dimensionalidade da base de dados, foi possível que a segunda etapa da seleção fosse realizada através do uso de uma técnica wrapper. Todos os possíveis subconjuntos de atributos foram utilizados para a execução do algoritmo SVM de aprendizagem, este teste foi realizado com SVM e não ANN pois o algoritmo SVM permitiu a execução dos testes em menor tempo. O algoritmo de aprendizagem foi executado em conjunto com técnicas de validação cruzada [Hall et al., 2009] a fim de obter resultados mais confiáveis. Realizando este processo foi possível atribuir uma pontuação para todos os possíveis subconjuntos de atributos presentes no conjunto obtido na etapa anterior.

Usando como base o ranking obtido foram escolhidos os seguintes atributos durante esta etapa de seleção: apgarMenor7, IUGR, IG, DMH, PCA, IdadeNutricaoPa-

Tabela 3.1. Candidate Attributes

Atributo	Descrição
HIS	Hemorragia intracranial severa
SatAdmo2	Saturaçao de oxigenio do recém nascido no momento da admissao na UTI neonatal
IdadePesoRecuperado	Idade (em dias) de quando o recem nascido recuperou seu peso normal de nascimento
IdadeNutricaoParental	Idade em dias de vida do início da nutrição parenteral (nutrição administrada pela veia)
CorticoideNeonatal	Corticoide feito na mãe antes do parto, quando realizado, tem impacto na redução do óbito de bebês prematuros
SPreanimacao	Se o recém nascido foi reanimado em sala de parto, ou seja, ao nascimento. Quando é necessária a reanimação, significa que o recém nascido nasceu em estado grave
IUGR	Se o recém nascido prematuro nasceu pequeno para a sua idade gestacional, ou seja, se este atributo é verdadeiro ele "cresceu inadequadamente"intraútero. Isso se relaciona com óbito.
DMH	Doença de membrana hialina, imaturidade pulmonar associada à prematuridade
IdadeDietaPlena	Idade em dias de vida que o recem nascido atingiu por via enteral (por sonda) as necessidades nutricionais diárias
IG	Idade gestacional em semanas e dias(ex: 29,6 = 29 semanas e seis dias)
SepsesTardiaEFungica	Apresentaçao de sepse (normal ou fúngica) dentro dos primeiros 7 dias de vida do recem nascido.
pH1a12h	Medida do pH no sangue do recem nascido colhido nas primeiras 12 horas de vida - quanto mais baixo, mais grave
PCA	Persistência do canal arterial - doença cardíaca que aparece em prematuros
apgarMenor7	Reflete as condições do RN aos 5 minutos de vida (é um escore, cujo ponto de corte é considerado 7)
IdadeNutricaoEnteral	Idade em dias em que a alimentação enteral (por sonda) foi iniciada para o RN

rental e IdadeDietaPlena (detalhes sobre os mesmos podems er vistos na Tabela 3.1). A escore dos demais grupos de atributos obtiveram uma pontuação não satisfatória quanto à sensibilidade e a sensitividade. Trabalhos futuros com conjuntos de dados mais amplos podem mudar esse quadro.

Esta estratégia de wrapping foi viável nesta etapa devido ao reduzido número de atributos presentes na base de dados. Como o NICeSim foi desenvolvido visando um uso genérico ele faz uso da técnica de filtro InfoGain para fornecer informações sobre mérito de atributos ao usuário. O algoritmo InfoGain usa como base para seu cálculo

de mérito o ganho de informação fornecido por cada atributo.

O ganho de informação de um atributo reflete quanta informação, relativa a classe de saída do problema, que aquele atributo pode fornecer. O ganho de informação de um atributo a_i relativo a uma classe L_j é definido pela função: $GI(a_i) = H(L_j) - H(L_j|a_i)$. Nesta função $H(L_j)$ representa a entropia da classe L_j, $H(a_i)$ a entropia do atributo a_i e $H(L_j|a_i)$ a entropia condicional.

Os sete atributos escolhidos passaram por mais uma fase de testes e análises, onde foram considerados principalmente como cada um daqueles atributos se relacionava, com o objetivo de auxiliar instituições e profissionais a no entendimento e tratamento de recém nascidos internados em UTIs. Esta etapa de análise serviu também como teste dos primeiros protótipos do simulador NICeSim. Os médicos envolvidos nesta pesquisa utilizaram os resultados obtidos em simulações realizadas pelo NICeSim para decidir se todos os atributos deveriam estar no conjunto final. Esta análise demonstrou que três dos atributos presentes neste conjunto obtido anteriormente foram coletados de forma que não faria sentido serem utilizados em nosso simulador para a finalidade estipulada nos objetivos iniciais. Os atributos removidos foram: PDA, ageParentalNutrition e agePlainDiet.

3.2 Criando o simulador

Após termos visto o processo que levou à versão final do conjunto de treinamento, balanceado e com os atributos propriamente selecionados, veremos agora o processo de desenvolvimento da aplicação NICeSim.

3.2.1 O algoritmo de Aprendizagem

NICeSim provê ao usuário duas opções de algoritmo de aprendizagem. Decidimos fornecer múltiplas opções pois algoritmos diferentes podem ter melhores resultados dependendo do conjunto de treinamento utilizado na aplicação. Ambos os algoritmos de aprendizagem fazem parte da ferramenta Weka [Hall et al., 2009], uma biblioteca amplamente conhecida e utilizada no meio científico para mineração de dados. A primeira opção de algoritmo explicada serão as Redes Neurais Artificiais (ANN, do inglês: Artificial Neural Network) e em seguida veremos mais sobre as Máquinas de Vetores de Suporte.

Redes Neuras Artificiais

ANNs são módulos computacionais que foram criados tendo como inspiração o sistema nervoso central de animais. Estes modelos evoluíram com o tempo para incorporar diversos conceitos da estatística e processamento de sinais. Uma ANN possui nós (ou neurônios) que são interligados por conexões ponderadas que são responsáveis por transmitir informações entre os mesmos. Neste trabalho foi utilizado uma implementação de ANN conhecida como Perceptron-Multicamadas (MP, do inglês: Multilayer-Perceptron). MP faz parte de um grupo de ANN que utiliza alimentação direta. Isso quer dizer que em um MP os neurônios só enviam informações para camadas que estão à frente de si, não sendo permitidos ciclos dentro da rede. A aprendizagem dentro da rede contida no MP acontece por um processo conhecido como retro-propagação de erro, um processo onde os pesos das conexões entre os nós da rede são ajustados após a ocorrência do processamento de uma instância cujo valor de saída é conhecido. Este ajuste é baseado na quantidade de erro produzida comparando-se o valor obtido pelo MP e o valor esperado da instância que foi processada. A Figura 3.1 demonstra de

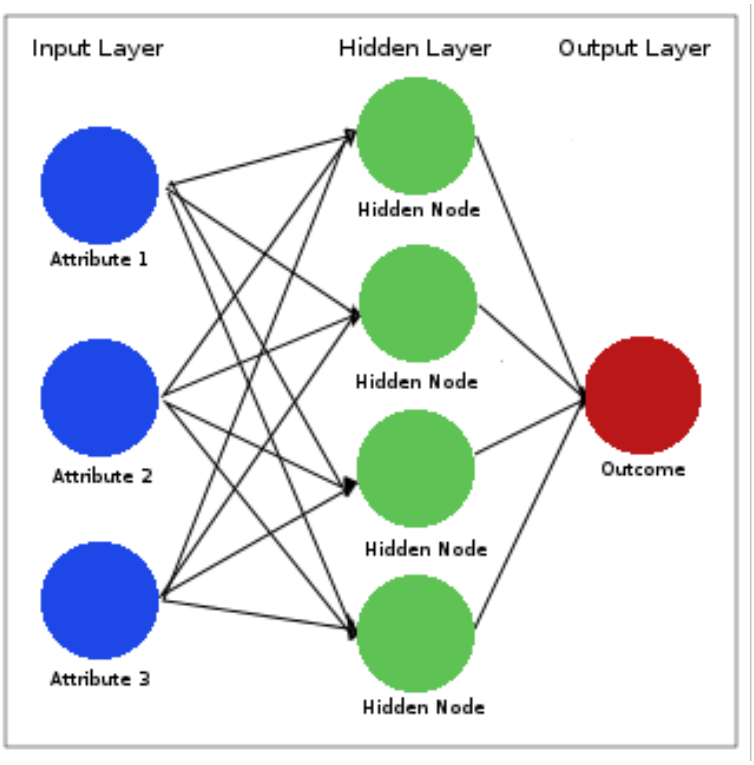

Figura 3.1. Representação simplificada do funcionamento do algoritmo MP

forma simplificada o funcionamento do algoritmo MP. Os círculos em azul na figura representam a camada de entrada, input layer, do MP. Esta camada corresponde aos

atributos utilizados no conjunto de dados. Após a camada de entrada o algoritmo pode possuir uma ou mais camadas ocultas, ou hidden layers (representadas por círculos verdes), que são responsáveis por processar as informações através dos pesos de suas conexões. Por último temos a camada de saída, (output layer), representada pelo círculo vermelho na figura. A camada de saída apresenta o valor final obtido pelo MP e que representa a predição do algoritmo para a saída referente aos atributos inseridos originalmente na rede. Caso o valor real seja conhecido a retro-propagação poderá ser realizada baseada no erro entre o valor predito e o valor real.

Se definirmos a saída real de um neurônio k como sendo $y_k(n)$ e a saída desejada como $d_k(n)$. Temos o erro desta saída na função:

$$e_k(n) = d_k(n) - y_k(n)$$

Se w_{kj} denota o peso da conexão entre dois nós k e j e $x_j(n)$ a saída do nó j durante a iteração n do algoritmo. Temos o ajuste necessário ao peso w_{kj} na função:

$$\Delta w_{kj}(n) = \eta e_k(n) x_j(n)$$

onde η é uma constante, conhecida como taxa de aprendizagem, que dita a velocidade com a qual os pesos são ajustados [Haykin, 1998].

Máquina de Vetores de Suporte

O algoritmo de Máquina de Vetores de Suporte (SVM, do inglês: Support Vector Machine) está presente no aplicativo como segunda opção para a aprendizagem do simulador. SVMs são conhecidas por ter uma alta capacidade de aprendizagem mesmo quando o conjunto de treinamento apresenta poucos parâmetros. Elas também são robustas e têm um bom desempenho mesmo que haja presença de ruído no modelo. Outra importante vantagem do algoritmo SVM é sua eficiência computacional quando comparado a outros métodos [Steinwart & Christmann, 2008]. Similar ao algoritmo MP, a SVM é um modelo supervisionado, ou seja, seu treinamento é realizado através de análise de uma série de exemplos de treinamento onde o valor de saída já é conhecido. SVM utiliza-se de técnicas de otimização para reconhecer padrões em um determinado conjunto de dados. A ideia formal por traz do algoritmo SVM é de construir um hiperplano que seja capaz de separar os dados de entrada em duas classes, quando se trata de classificação binária, como é o caso deste trabalho. Em uma situação ótima este hiperplano deve delimitar o maior espaço de margem possível entre os pontos de classes diferentes. Uma margem maior é desejável pois leva a uma redução nos erros durante a generalização, tornando a SVM mais capaz de predizer a classe de uma entrada

de dados nova. A Figura 3.2 exemplifica a ideia do algoritmo SVM. Nesta figura os

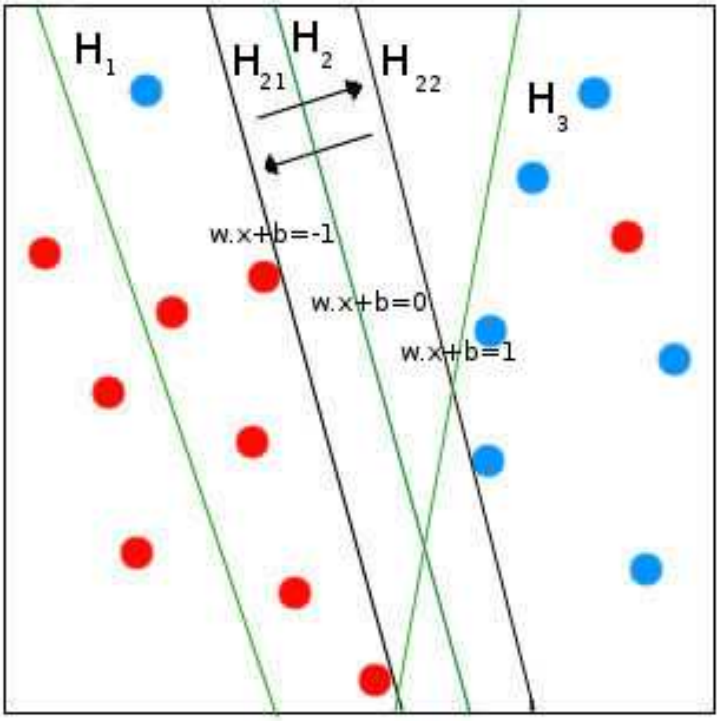

Figura 3.2. Representação simplificada do funcionamento do algoritmo SVM

círculos azuis e vermelhos representem objetos de 2 classes diferentes presentes em um conjunto de dados bidimensional. As 3 retas H_1, H_3 e H_3 são 3 possíveis delimitadores que tentam separar os objetos destas classes. A reta A possui um poder de separação baixo, resultando em muitos objetos de classes diferentes dividindo o mesmo lado do plano. As retas H_2 e H_3 têm igual poder de divisão para os objetos presentes no plano, porém a reta H_2 é superior à reta H_3 pois podemos traçar uma margem maior entre os 2 objetos mais próximos de ambos os lados da reta. A existência de uma margem maior para a reta H_2 quer dizer que objetos futuros, ainda não presentes no plano, tem uma maior chance de serem corretamente classificados pela reta H_2. Isto significa uma menor chance de overfitting, ou seja, minimiza uma superadequação do modelo aos dados de treinamento em questão. Esta superadequação poderia levar a um maior erro de generalização, ainda que com baixo erro de treinamento.

O algoritmo SVM pode ser formulado como um problema de otimização quadrática. A formulação descrita a seguir corresponde a classificação binária, ou seja, classificação envolvendo duas classes discretas. As N instâncias de um problema de classificação binária arbitrário podem ser representadas pela tupla (x_i, y_i), onde $x_i = (x_{i1}, x_{i2}, ..., x_{id})$ é um vetor contendo os valores dos atributos da i-ésima instância

e y_i indica a classe a qual x_i pertence. Para conveniência, suponhamos que $y_i \in \{-1, 1\}$. Qualquer hiperplano pode ser descrito como uma série de pontos x que satisfaçam:

$w \cdot x + b = 0$,

onde w denota um vetor normal no hiperplano e $\frac{b}{||w||}$ denota o deslocamento do hiperplano a partir da origem através de w. Então se encontrarmos w e b que descrevem H_2 na Figura 3.2 e usarmos -1 e $+1$ ao invés de vermelho e azul, respectivamente, para representar as classes deste hiperplano poderemos predizer a classe de uma dada instância x_i através da seguinte função:

$$y = \begin{cases} -1 & \text{se } w \cdot x + b < 0 \\ +1 & \text{se } w \cdot x + b > 0 \end{cases}$$

Além disto, podemos alterar as escalas de w e b de forma que H_{21} e H_{22} possam ser representados como:

$$\begin{aligned} H_{21}: &\quad \text{se } w \cdot x + b = -1 \\ H_{22}: &\quad \text{se } w \cdot x + b = 1 \end{aligned}$$

Fazendo uso da geometria, temos que a distância entre H_{21} e H_{22} (margens da linha de decisão) é $\frac{2}{||w||}$. Consequentemente queremos encontrar w e b que minimizem $f(w) = ||w||$ que safisfaça:

$$\begin{aligned} &\text{se } w \cdot x + b \leq -1 \quad \text{se } y = -1 \\ &\text{se } w \cdot x + b \geq -1 \quad \text{se } y = 1 \end{aligned}$$

Para que possamos então transformar esta função em um problema de otimização quadrática, podemos substituir $||w||$ por $\frac{1}{2}||w||^2$, mantendo a mesma solução, tendo em vista que os valores de w e b permanecem inalterados. Temos então que o procedimento de aprendizagem em uma SVM linear pode ser definido pelo seguinte problema de otimização restrito:

$$\begin{gathered} min \frac{1}{2}||w||^2 \\ \text{sujeito a } y_i(w \cdot x + b) \geq 1,\ i = 1, 2, 3...N. \end{gathered}$$

Este problema de otimização convexa com restrições de desigualdade pode ser resolvido pelo método multiplicador de Lagrange juntamente com as condições de Karush-Kuhn-Tucker [Cristianini & Shawe-Taylor, 2000] [Avriel, 2003]. Estas técnicas em conjunto com funções kernel dados não separáveis e não lineares, levam a formulação mais comum da SVM contida na literatura [Cristianini & Shawe-Taylor, 2000] [Noble, 2006] [Tan et al., 2006].

A execução da SVM neste trabalho se deu com o parâmetro de complexidade $C = 1$ e utilizando de construção de modelos logísticos, a fim de produzir as probabilidades necessárias. A inclusão das funções de kernel não agregaram benefício algum para o simulador, sendo assim procedemos com a SVM linear que tem um tempo de treinamento mais curto.

3.2.2 Estrutura da Aplicação

Esta seção é dedicada a explicar a estrutura da aplicação e o funcionamento de seu código através de diagramas de classe e de sequência.

Durante o desenvolvimento os seguintes requisitos foram considerados:

- Permitir ao usuário simular problemas de um domínio modelado;
- Fornecer estimativas de acurácia estatística do simulador;
- Suportar os mesmos arquivos e formatos que a ferramenta Weka para os modelos de treinamento;
- Suportar problemas de diferentes domínios;
- Interface de uso simples e funcional;

Assim sendo as próximas seções detalharão a estrutura e desenho da aplicação.

Diagramas de Classe

O Diagrama ilustrado na figura 3.3 apresenta a representação UML dos 3 principais componentes da aplicação NICeSim. Todos os componentes ilustrados no diagrama fazem parte do pacote de controle da aplicação, onde a maior parte da lógica da mesma se concentra.

Um dos componentes ilustrados no diagrama representa a classe AppController. Esta classe é o ponto de entrada da aplicação e é responsável por parametrizar as entradas do usuário e instanciar apropriadamente os demais componentes da aplicação. Os principais métodos deste componente e suas respectivas funções são:

openFromFile: lida com a inicialização dos dados de treinamento a partir de um arquivo especificado pelo usuário. É responsável pela instanciação do componente ClassificationController;

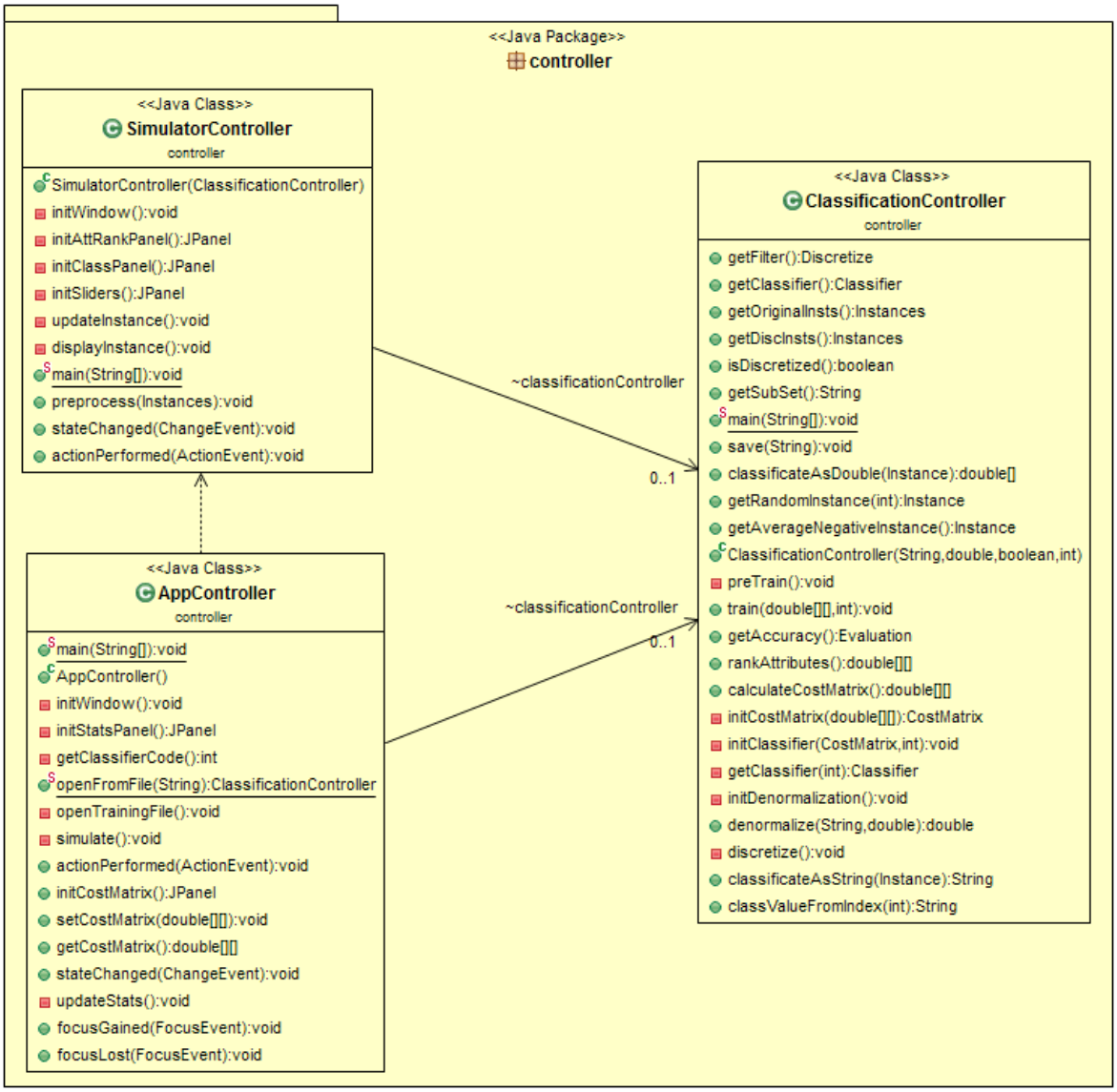

Figura 3.3. Diagrama de Classe dos componentes presentes no pacote controller da aplicação

initCostMatrix, setCostMatrix e getCostMatrix: métodos responsáveis pela interação do usuário com a matriz de custos. Manipulam as informações exibidas e inseridas no painel de custos a fim de permitir o controle do balanceamento artificial ao usuário;

initStatsPanel e updateStats: Métodos responsáveis pela inicialização e atualização (respectivamente) do painel de estatísticas. A atualização das estatísticas de performance do algoritmo de aprendizagem ocorrem caso algum parâmetro tenha sido modificado pleo usuário;

simulate: Inicia a simulação, instanciando um objeto da classe SimulationController com todos os parâmetros relevantes.

O próximo componente da Figura 3.3 que analisaremos sera a classe ClassificationController. Esta classe é a principal interface de comunicação entre o simulador(SimulationController) e a API Weka. Seus principais métodos e responsabilidades são:

preTrain e train: executa o pre-treino e treino do algoritmo de aprendizagem. O pre-treino consiste na etapa de processamento e filtragem do conjunto de dados;

rankAttributes: calcula e retorna o ranking de estimativa de importância dos atributos presentes no conjunto utilizado;

getAccuracy: retorna as estatísticas de performance do algoritmo de aprendizagem, para que estas possam ser exibidas para o usuário pelo componente AppController.

classificateAsDouble: método essencial do componente de classificação, é responsável por retornar a percentagem de chance de que a instância recebida como parâmetro seja pertencente a cada uma das classes existentes no atributo de saída do conjunto;

Por último na Figura 3.3 temos a representação da classe SimulationController, encarregada de controlar o fluxo de simulação na aplicação. Durante o processo de definição da parâmetros o controle da aplicação é mantido principalmente pela classe AppController. Após o início da simulação a classe SimulationController toma controle da aplicação. Seus principais métodos são:

preProcess: prepara os componentes de interface com o usuário do simulador, usando como base as informações contidas no conjunto de treinamento;

displayInstance e updateInstance: displayInstance apresenta ao usuário as informações de uma instância do conjunto de dados e updateInstance modifica essas informações caso o usuário solicite a visualização de uma nova instância.

stateChanged: monitora modificações que o usuário possa fazer na instancia atual e atualiza as informações necessárias no componente SimulationController caso haja alguma mudança.

Em parágrafos futuros desta seção a interação entre estes componentes será melhor detalhada através do uso de diagramas de sequência.

Em adição aos componentes apresentados anteriormente a aplicação possui um modulo utilitário empregado amplamente como interface de comunicação com a biblioteca Weka e para a execução de algumas tarefas auxiliares.

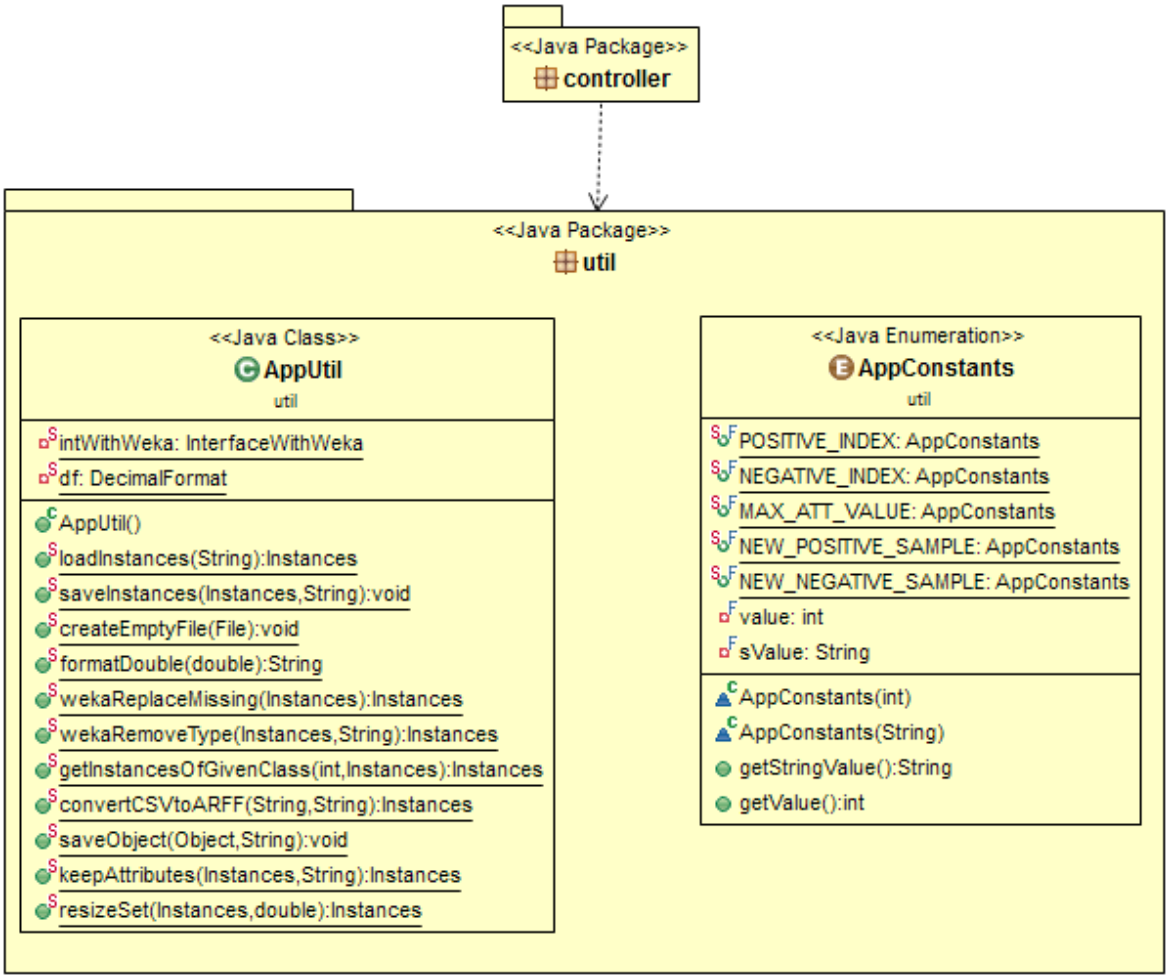

Figura 3.4. Diagrama de Classe representando as classes utilitárias do NICeSim

A Figura 3.4 ilustra em um Diagrama de Classes os componentes do pacote utilitário da aplicação NICeSim. Esse pacote é composto pela classe AppUtil e pela enumeração AppConstants. A classe AppUtil possui uma série de métodos estáticos auxiliares que executam funções como: manipular arquivos de disco contendo informações sobre conjuntos de dados, realizar conversões entre formatos numéricos, redimensionar conjuntos de dados para o treinamento e separar as instâncias utilizadas pela aplicação tendo como base suas classes. Essas funções de ajuda são utilizadas por todas as classes do pacote controller.

O outro componente do pacote utilitário é a enumeração AppConstants, esta enumeração é utilizada por todas as classes da aplicação e fornece basicamente uma maneira fácil de se identificar valores de saída nas instâncias processadas.

Vistos os componentes que formam a aplicação, é importante também analisarmos em mais detalhes a interação entre os mesmos. A Figura 3.5 apresenta uma versão simplificada do diagrama de sequência do início da aplicação ate o início da simulação. A execução real da aplicação envolve mais passos do que os apresentados no diagrama, mas os principais métodos do fluxo de inicialização são aqui apresentados. Neste diagrama temos a classe AppController como ponto de entrada na aplicação. A

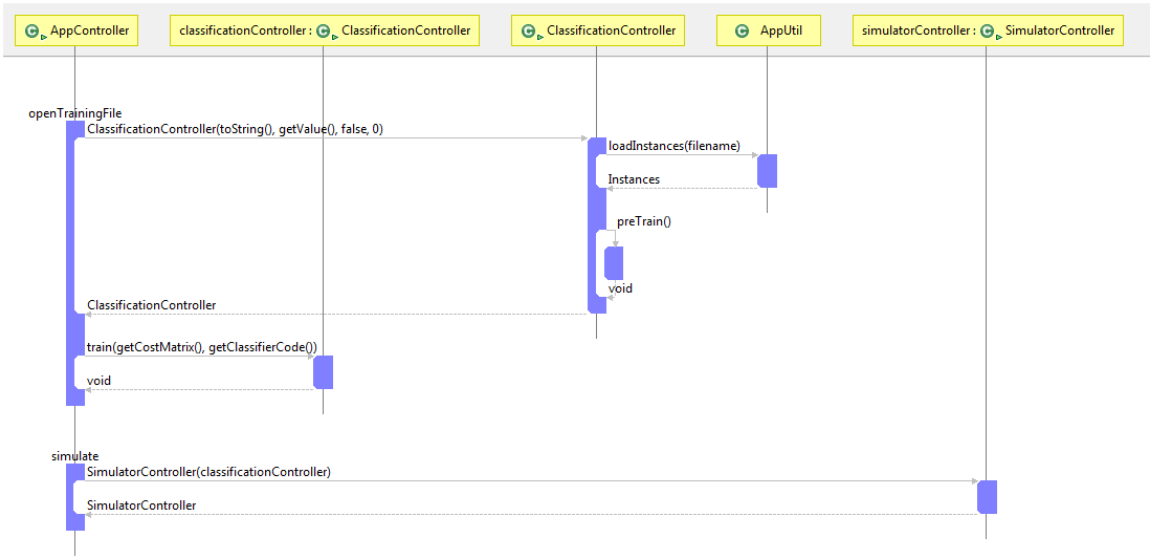

Figura 3.5. Diagrama de Sequência demonstrando o fluxo de inicialização da aplicação

primeira ação executada pelo usuário é a abertura do arquivo de entrada contendo as informações a serem utilizadas para treinamento. Com estas informação o componente ClassificationController é criado e em sua construção os parâmetros padrões (como a matriz de custos) são calculados.

Com os parâmetros padrões definidos o algoritmo de aprendizagem é treinado e as estatísticas de performance do mesmo são calculadas e apresentadas ao usuário. Neste ponto a aplicação aguarda que o usuário modifique algum parâmetro (o que desencadeia um novo treinamento) ou inicie a simulação. O início da simulação marca a instanciação do componente SimulationController pelo componente AppController.

Após iniciada a simulação o componente SimulationController toma controle da aplicação, e inicia um fluxo que pode ser visto de maneira simplificada na Figura 3.6.

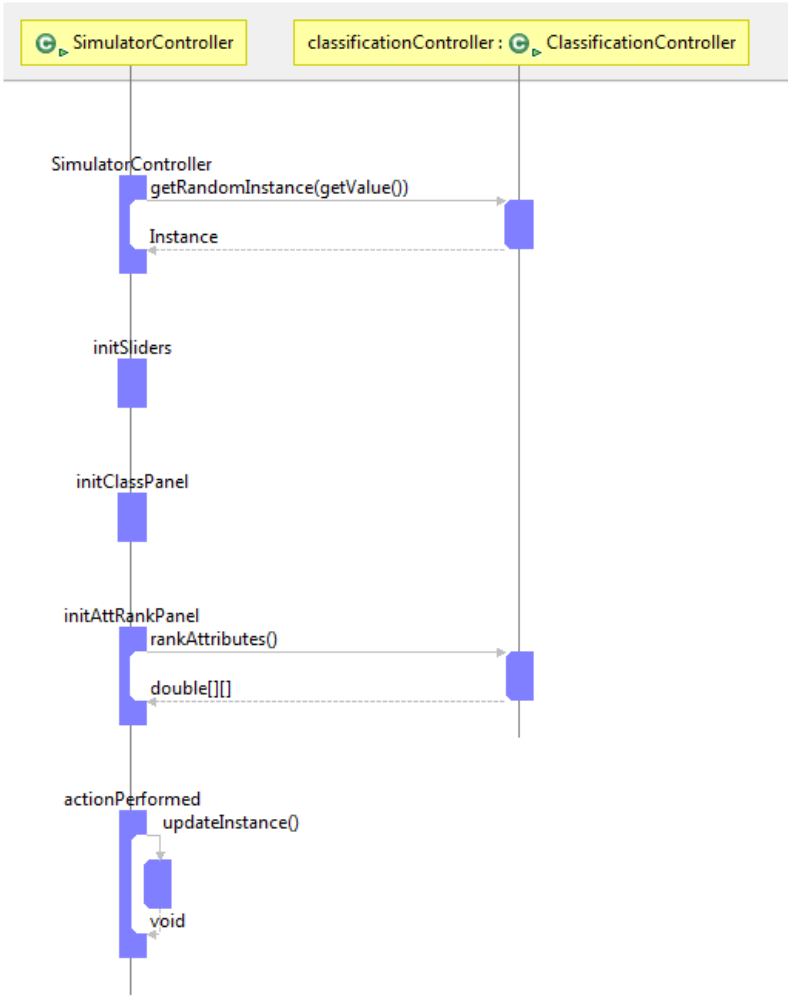

Figura 3.6. Diagrama de Sequência com o fluxo interno da classe Simulation-Controller

O SimulatorController utiliza o componente ClassificationController para execução da lógica de predição e o primeiro passo de sua inicialização é a obtenção de uma instância aleatória pertencente ao conjunto de treinamento. Esta instância é então utilizada para definir valores iniciais na tela de simulação. Estes valores são inseridos nos componentes denominados sliders, e estes são utilizados pelo usuário para manipular as instâncias após terem sido propriamente inicializados e conectados com os atributos presentes no conjunto de dados.

Em seguida o fluxo do controle de simulação inicia a geração do ranking de atributos, essa informação é calculada pelo componente ClassificationController e então retornada para o SimulatorController para que este possa exibir a informação para o usuário. Internamente a classe ClassificationController utiliza o ranqueamento através

do algoritmo InfoGain existente na biblioteca Weka.

A última parte do diagrama de sequência apresentado na Figura 3.6 denominada como actionPerformed representa a adaptação do simulador a modificações realizadas pelo usuário. O componente SimulatorController monitora as modificações realizadas nos sliders e garante a sincronia entre a instância que se está simulando e as informações apresentadas na janela da aplicação.

3.2.3 Apresentação do Simulador

A Figura 3.7 apresenta a tela inicial do NICeSim. Em destaque vermelho pode ser visto o botão que deve ser utilizado para fornecer os dados de entrada para o aplicativo. Estes dados devem estar contidos em um arquivo que siga apropriadamente o padrão Attribute-Relation File Format(ARFF) [Hall et al., 2009] que é utilizado pela biblioteca Weka. Este arquivo deve conter os dados a serem utilizados pelo algoritmo de aprendizagem. Em destaque verde na imagem estão as opções disponíveis de algoritmos de aprendizagem. Na versão aqui apresentada o usuário pode escolher entre os algoritmos SVM e MP. No destaque azul da imagem pode-se ver a opção de utilizar somente uma percentagem do conjunto fornecido para o treinamento, o que permite um tempo de processamento menor em conjuntos de dados muito grandes. Outro uso importante para esta função é a de utilizar parcialmente os dados em caso de um conjunto de dados muito grande, tendo em vista que isso poderia prejudicar o desempenho da aplicação.

Na Figura 3.8 vemos a janela da aplicação após a escolha de um arquivo de entrada. Em destaque azul nesta imagem podemos ver a matriz de custos sugerida pela aplicação para os dados fornecidos. Na imagem exemplo os dados de entrada utilizados tratam-se do conjunto final de quatro atributos descritos em seções anteriores. Podemos ver na imagem os valores utilizados para atingir o balanceamento artificial nos dados de exemplo. No destaque vermelho da Figura 3.8 temos um painel com informações estatísticas sobre a acurácia do simulador após o treinamento do algoritmo de aprendizagem selecionado. Este painel possui informações importantes para a medição da eficácia da simulação a ser executada. Todos os valores presentes neste painel foram obtidos através de testes de validação cruzada utilizando os dados de treinamento fornecidos. As informações apresentadas no painel são:

Correctly Classified Instances: Número e percentagem de instâncias corretamente classificadas pelo simulador;

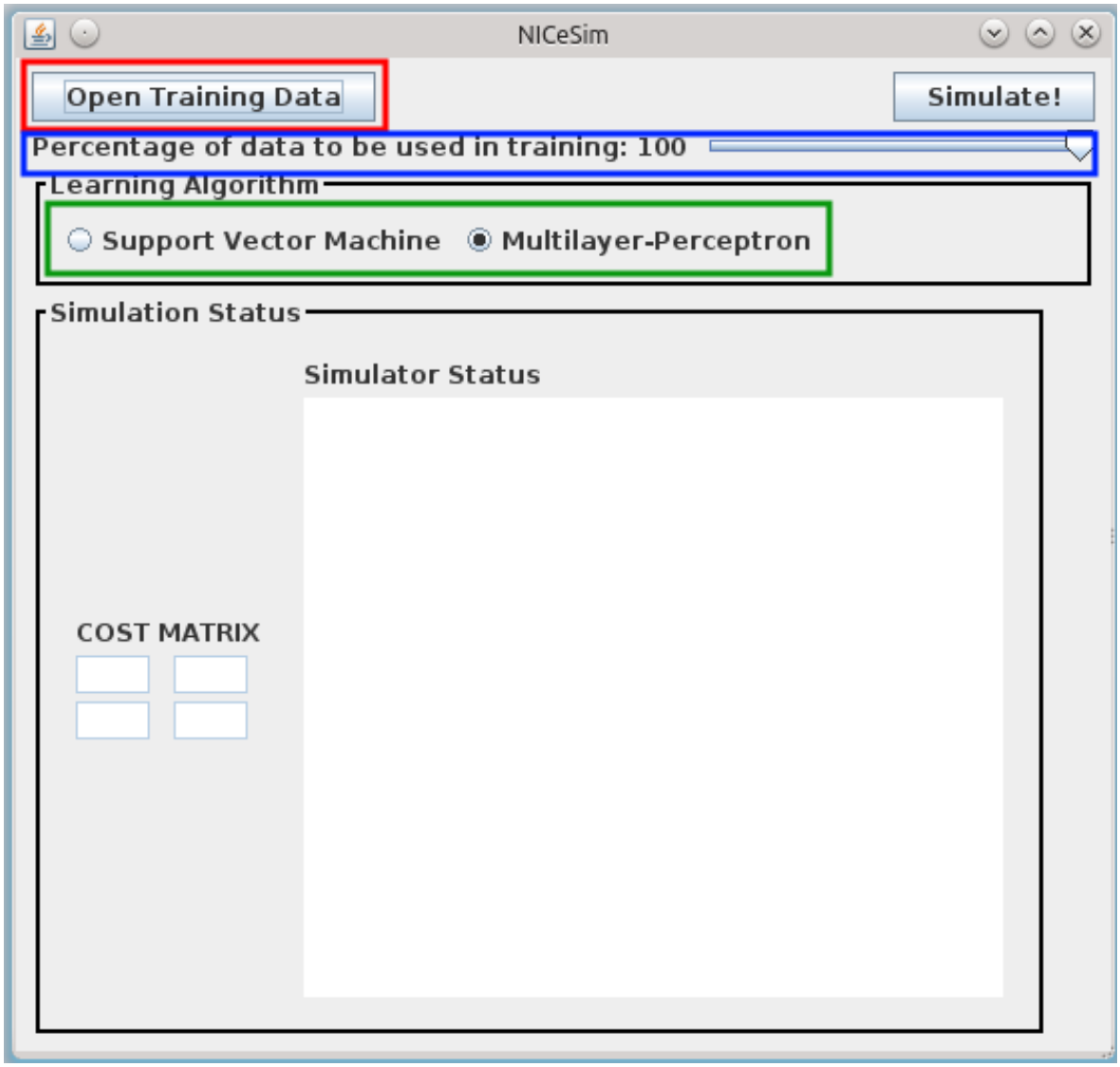

Figura 3.7. Imagem inicial da aplicação NICeSim

Incorrectly Classified Instances: Número e percentagem de instâncias incorretamente classificadas pelo simulador;

Kappa Statistic: Medida do coeficiente de Cohen Kappa, uma medida estatística conservadora que leva em consideração acertos que podem ter ocorrido por acaso [Carletta, 1996];

Mean Absolute Error: Erro médio absoluto cometido pelo simulador;

Root Mean Squared Error: Raiz do erro quadrático médio. Representa basicamente o desvio padrão dentro da amostra entre os valores previstos e os valores observados;

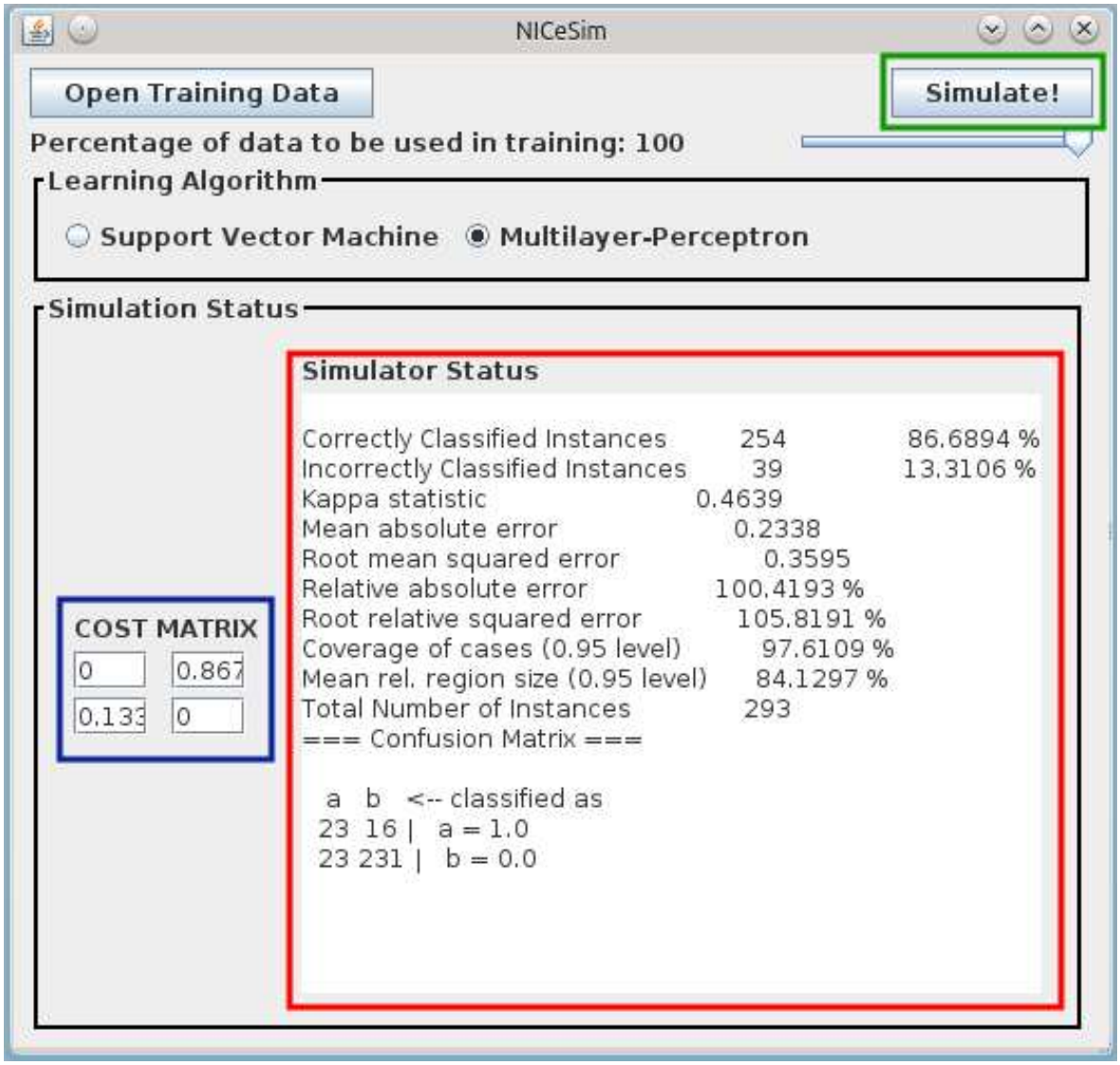

Figura 3.8. Tela da aplicação após a escolha dos dados de entrada

Relative Absolute Error: Erro absoluto relativo. Este valor representa a divisão do erro absoluto do simulador pelo erro absoluto obtido por um simulador fictício que sempre preveja a média dos valores do conjunto de treinamento;

Root Relative Squared Error: Similar ao erro absoluto relativo, mas calculado para o erro quadrático médio;

Consufion Matrix: Matriz de confusão. Demonstra a relação entre as instâncias classificadas como verdadeiro positivo, falso positivo, verdadeiro negativo e falso negativo.

Ainda na Figura 3.8 temos a área em destaque verde, onde está presente o botão

de simulação. Após definir todos os parâmetros necessários (ou utilizar os valores sugeridos) o usuário pode dar início à simulação clicando neste botão.

Alguns instantes após dar início à simulação o usuário deverá ter acesso à janela de simulação, vista na Figura 3.9. Nesta janela podemos notar em destaque verde que o aplicativo fornece como guia para o usuário um ranking dos atributos presentes no conjunto. Este ranking é obtido através da execução do algoritmo Info Gain nos dados de treinamento. O Algoritmo info gain fornece estimativas da importância dos atributos através de uma medição da diferença entre a entropia do conjunto de treinamento completo e a entropia do mesmo conjunto sub-dividido a partir dos diferentes valor do atributos que se está analisando. Na Figura 3.9 as áreas destacadas em azul e

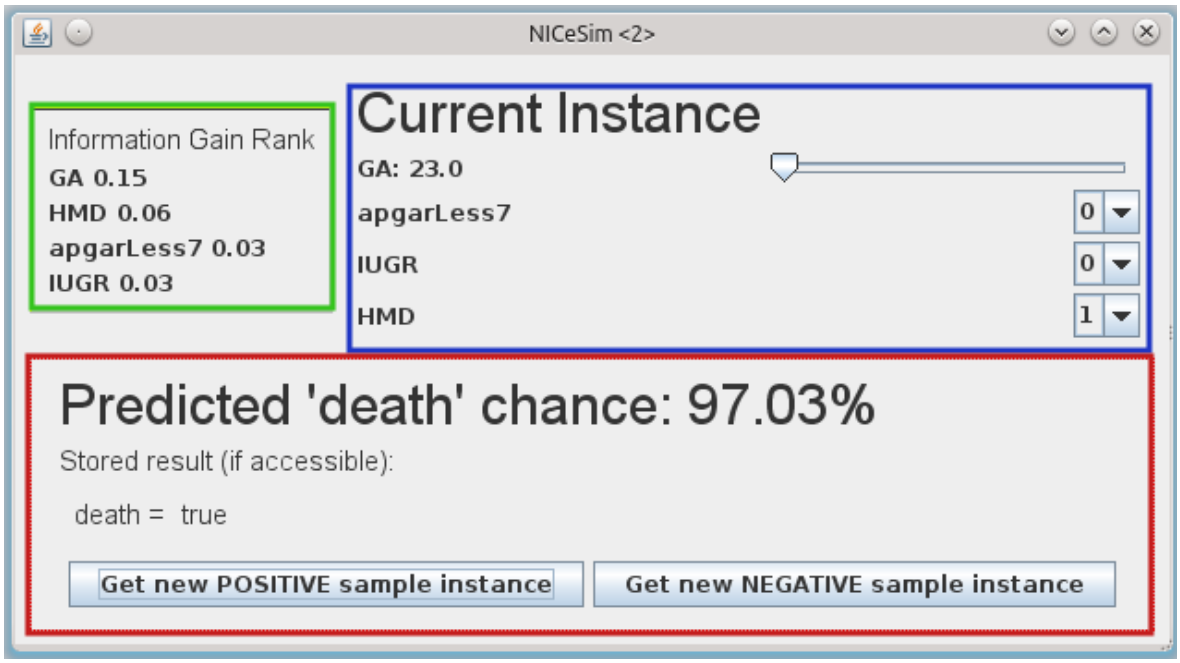

Figura 3.9. Tela de simulação da aplicação NICeSim

vermelho trabalham em conjunto. A área azul apresenta os controles responsáveis pela manipulação da instância que está atualmente sendo simulada. O usuário pode manipular estes controles a fim de modificar os valores dos atributos da instância. Qualquer modificação na instância simulada é refletida em tempo real no painel destacado em vermelho da figura. Este painel apresenta os resultados da simulação. No exemplo apresentado na Figura 3.9 o resultado da simulação da instância atual é apresentado, demonstrando 97,03% de chance de resultado positivo (morte = verdadeiro) para esta instância. Como esta é uma instância presente no conjunto de treinamento podemos ver também na imagem que o valor armazenado (Stored Result) para esta instância é verdadeiro (true). Este resultado armazenado é apresentado para consulta do usuário e

não influencia na execução do simulador. Para algumas instâncias podemos ter o caso da predição apresentar um resultado diferente do resultado armazenado no conjunto de treinamento. O que representaria uma predição incorreta realizada pelo simulador. Ainda no painel destacado em vermelho temos dois botões que o usuário pode utilizar para acessar instâncias do conjunto de treinamento. O botão da esquerda modifica os parâmetros da simulação para aqueles correspondentes a alguma instância aleatória que contenha desfecho positivo e o botão da direita faz o mesmo, mas para uma instância que possua desfecho negativo.

Capítulo 4

Resultados e Discussões

4.1 Performance Estatística

O software produzido durante este trabalho se mostrou um simulador robusto, pois faz uso de algoritmos de aprendizagem com capazes de obter uma baixa taxa de erro mesmo na presença de ruídos. NICeSim pode ser utilizado por instituições, profissionais e pelo poder público na tarefa de melhor entender os diversos fatores que afetam o tratamento de recém-nascidos em unidades neonatais de terapia intensiva. Nosso trabalho começou com uma base de dados contendo 114 atributos, destes pudemos apontar 4 como sendo cruciais na simulação a ser executada.

Utilizando a versão final do conjunto de dados e do aplicativo desenvolvido foi possível obter 73,04% e 86,68% de acurácia com os algoritmos de SVM e MP respectivamente. É importante notar que apesar de terem sido obtidos melhores resultados em alguns dos testes realizados com o conjunto de dados intermediário estes não são resultados de melhor qualidade pois nem todos os atributos presentes neste conjunto estão alinhados com os objetivos deste trabalho.

Na tabela 4.1 é possível ver os resultados obtidos nos testes estatísticos realizados. Nesta tabela Conjunto Intermediário se refere ao conjunto de 15 atributos obtido em etapas iniciais da seleção de atributos. Conjunto Final é o conjunto de 4 atributos que obtivemos como resultado final da seleção de atributos. Na coluna tipo de teste o valor *Split* represente testes realizados utilizando 66% da base de dados como conjunto de treinamento e os 34% restantes como conjunto de testes. Um valor Validação Cruzada nesta coluna representa testes que utilizaram técnica de estimação por validação cruzada. Em todos os experimentos de validação cruzada o número de partes (*folds*) utilizadas foi de 10.

Conjunto	Tipo de Teste	Parâmetro	SVM	MP
Intermediário	Split	Acurácia	86%	93%
Intermediário	Split	Sensibilidade	91,67%	75%
Intermediário	Split	Especificidade	85,22%	95,5%
Intermediário	Validação Cruzada	Acurácia	88,05%	92,15%
Intermediário	Validação Cruzada	Sensibilidade	79,49%	71,8%
Intermediário	Validação Cruzada	Especificidade	89,37%	95,3%
Final	Split	Acurácia	77%	84%
Final	Split	Sensibilidade	100%	83,3%
Final	Split	Especificidade	73,9%	84,1%
Final	Validação Cruzada	Acurácia	73,04%	86,68%
Final	Validação Cruzada	Sensibilidade	82,10%	69,2%
Final	Validação Cruzada	Especificidade	71,70%	89,37%

Tabela 4.1. Estatísticas de Performance do Simulador

Deve-se salientar que o desempenho dos algoritmos do NICeSim está diretamente atrelado ao desempenho da suíte Weka. Como esta biblioteca é utilizada como fonte dos algoritmos de mineração, melhoras em seus algoritmos e métodos tem impacto positivo no simulador.

Ademais, as funcionalidade e uso do NICeSim não estão restritas ao conjunto de dados aqui apresentados. Qualquer conjunto de dados que siga as especificações ARFF e que mantenha um atributo de saída contendo duas classes pode ser utilizado neste simulador.

4.2 Uso prático

Para a testagem preliminar do software, foram realizadas algumas simulações dirigidas à estimativa do risco de óbito de crianças hipotéticas, na dependência das variáveis clínicas de maior interesse, com destaque para a doença da membrana hialina (HMD), idade gestacional (apresentada como GA durante a seção de preparação de dados) e Apgar.

A idade gestacional é usualmente definida como a melhor estimativa entre o exame de ultrassonografia gestacional precoce (<20 semanas), data da última menstruação, anotação obstétrica e exame clínico pelo escore New Ballard [Cohen-Wolkowiez et al., 2009]. Foram categorizados como prematuros extremos aqueles cuja IG<28 semanas; muito prematuros aqueles com IG entre 28 e 31 semanas; e prematuros moderados os com IG entre 32 e 36 semanas [Behrman & Butler, 2007].

A Escala de Apgar diz respeito à avaliação clínica do recém-nascido - tradicionalmente no 1º e no 5º minutos de vida -, através da apreciação de cinco sinais objetivos: frequência cardíaca, respiração, tônus muscular, irritabilidade reflexa e cor da pele [Freitas et al., 2012]. A cada um destes sinais atribui-se uma pontuação de 0 a 2, de modo que o Apgar pode variar de zero a dez em seu escore final. Assim, o recém-nascido será classificado como sem asfixia (Apgar 8 a 10), com asfixia leve (Apgar 5 a 7), com asfixia moderada (Apgar 3 a 4) e com asfixia grave (Apgar 0 a 2).

A doença da membrana hialina é um quadro relacionado à imaturidade pulmonar do recém-nascido, associada à deficiência de surfactante - substância importante para a manutenção da estabilidade da parede dos pulmões - e caracterizada pela ocorrência de significativa dificuldade respiratória do bebê. As estimativas de possibilidade de evolução para o óbito com base nesses parâmetros são apresentadas na Tabela 4.2.

Tabela 4.2. Risco de evolução para a morte de crianças com e sem HMD, de acordo com o Apgar e a Idade Gestacional, em UTI neonatal.

Apgar	HMD	Idade Gestacional (GA)	Risco de Óbito
<=7	Presente	24 semanas	97,05%
		28 semanas	97,04%
		32 semanas	41,57%
>7	Presente	24 semanas	96,64%
		28 semanas	40,28%
		32 semanas	10,46%
<=7	Ausente	24 semanas	97,05%
		28 semanas	96,65%
		32 semanas	0,19%
>7	Ausente	24 semanas	6,77%
		28 semanas	0,12%
		32 semanas	0,17%

Observa-se, com base nos dados obtidos, que as estimativas produzidas pelo software são consistentes com as informações clínicas disponíveis na literatura, ou seja, a existência de síndrome da membrana hialina, o nascimento antes da plena maturidade do bebê (infantes com menores idades gestacionais) e a presença de um Apgar menor do que 7 são fatores que se correlacionam, claramente, a piores prognósticos dos recém-nascidos (ou seja, maior risco de evolução para a morte) [Freitas et al., 2012] [Rego et al., 2010] [Fanaroff et al., 2003].

4.2.1 Trabalhos Futuros

Existem algumas melhorias futuras que gostaríamos de desenvolver. Uma destas evoluções seria permitir ao usuário escolher mais opções de algoritmos de aprendizagem. Isto teria uma grande importância na ampliação das possibilidades de uso do simulador tendo em vista que situações de domínios diferentes podem apresentar melhores resultados com outros algoritmos.

Outro trabalho futuro interessante seria a utilização do simulador com bases de dados maiores, tendo em vista que todos os testes realizados e aqui descritos foram feitos utilizando uma base de dados relativamente pequena (293 instâncias).

Outra possibilidade interessante seria a formação da base de dados com vistas à experimentação com o simulador, o que não foi o caso da base aqui utilizada. Isto faria com que a coleta das informações fosse mais adequada a algum propósito específico, maximizando as chances de obter conhecimento útil que demandariam mais esforço e custo via métodos não computacionais para serem elucidados.

Capítulo 5

Conclusões

NICeSim é uma ferramenta de simulação que faz uso da API Weka para prover um aplicativo de fácil uso. Utilizando este aplicativo pode-se realizar simulações em interativas de qualquer problema que esteja propriamente descrito no padrão ARFF e que possua uma variável de saída binária.

Neste trabalho demonstramos o uso do NICeSim no problema de tratamento e diagnóstico de recém nascidos em UTIs neonatais. Pudemos verificar que o simulador alcança resultados estatísticos satisfatórios e de acordo com conhecimentos prévios vindo da literatura.

Por utilizar a biblioteca Weka como provedora de algoritmos de aprendizagem e mineração de dados o NICeSim garante sua compatibilidade com um formato de dados bastante utilizado no meio acadêmico para análise de conhecimentos.

As simulações realizadas pelo aplicativo são interativas e o simulador é capaz de auxiliar no entendimento das características do domínio de um problema. Este objetivo é alcançado pois o NICeSim permite que o usuário analise atributos e crie instâncias fictícias, tendo como retorno informações sobre como suas modificações influenciam a variável de saída do problema.

Tais experimentações podem servir para validar uma série de hipóteses sobre o domínio do problema de maneira rápida e praticamente sem custo, levando à geração de conhecimento de modo facilitado.

Referências Bibliográficas

Avriel, M. (2003). *Nonlinear programming: Analysis and methods*. Dover Publications, New york, USA.

Behrman, R. E. & Butler, A. S. (2007). Preterm birth: causes, consequences, and prevention. *Washington, DC: The National Academies Press*.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Carletta, J. (1996). Assessing agreement on classification tasks: The kappa statistic. *Comput. Linguist.*, 22(2):249--254.

Christine L Tsien, Isaac S Kohane, N. M. (2000). Multiple signal integration by decision tree induction to detect artifacts in the neonatal intensive care unit. *Artificial intelligence in medicine*, 19(3):189--202.

Cofiño, A. S.; Gutiérrez, J. M.; Jakubiak, B. & Melonek, M. (2003). Implementation of data mining techniques for metereological applications. *Realizing Teracomputing*, pp. 215–240.

Cohen-Wolkowiez, M.; Moran, C.; Benjamin, D. K.; Cotten, C. M.; Clark, R. & DK, B. (2009). Early and late onset sepsis in late preterm infants. *Pediatr Infect Dis J*, 28(12):1052–6.

Cristianini, N. & Shawe-Taylor, J. (2000). *An Introduction to support vector machines and other kernel-based learning methods*. Cambridge University Press, Cambridge, UK.

Davenport, T. H. & Harris, J. G. (2007). Competing on analytics: The new science of winning. *Harvard Business School Press*.

Fanaroff, A. A.; Hack, M. & Walsh, M. C. (2003). The nichd neonatal research network: changes in practice and outcomes during the first 15 years. *Semin Perinatol*, 27(4):281–7.

Fayyad, U.; Piatetsky-shapiro, G. & Smyth, P. (1996). From data mining to knowledge discovery in databases. *AI Magazine*, 17:37--54.

Freitas, B. A. C. (2011). Parâmetros clínicos, epidemiológicos e nutricionais de recém-nascidos prematuros atendidos em uma unidade de terapia intensiva neonatal no municÃpio de viçosa-mg. Master's thesis, Federal University of ViÃ§osa, Department of Nutrition and Health.

Freitas, B. A. C.; Peloso, M.; Manella, L. D.; Franceschini, S. C. C.; Longo, G. Z.; Gomes, A. P. & Siqueira-Batista, R. (2012). Sepse tardia em prÃ©-termos de uma unidade de terapia intensiva neonatal: anÃ¡lise de trÃªs anos. *Revista Brasileira de Terapia Intensiva*, 24:79–85.

Guyon, I. & Elisseeff, A. (2003). An introduction to variable and feature selection. *The Journal of Machine Learning Research archive*, 3:1157–1182.

Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P. & Witten, I. H. (2009). The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18.

Han, J. (2005). *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Haykin, S. (1998). *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edição.

Hollander, Z.; Chen, V.; Sidhu, K.; Lin, D.; Ng, R. T.; Balshaw, R.; Cohen-Freue, G. V.; Ignaszewski, A.; Imai, C.; Kaan, A.; Tebbutt, S. J.; Wilson-McManus, J. E.; McMaster, R. W.; Keown, P. A. & McManus, B. M. (2012). Predicting acute cardiac rejection from donor heart and pre-transplant recipient blood gene expression. *The Journal of Heart and Lung Transplant*.

Kang, U. J. & Auinger, P. (2012). Activity enhances dopaminergic long-duration response in parkinson disease. *Neurology*.

Monique Frize, Colleen M Ennett, M. S. H. C. T. (2001). Clinical decision support systems for intensive care units: using artificial neural networks. *Medical engineering and physic*, 23(3):217--225.

Monique Frize, R. W. (2000). Clinical decision-support systems for intensive care units using case-based reasoning. *Medical engineering and physics*, 22(9):671--677.

Nakayama, N.; Oketani, M.; Kawamura, Y.; Inao, M.; Nagoshi, S.; Fujiwara, K.; Tsubouchi, H. & S, M. (2012). Algorithm to determine the outcome of patients with acute liver failure: a data mining analysis using decision trees. *Journal of Gastroenterol*.

Noble, W. S. (2006). What is a support vector machine? *Nat. Biotechnol.*, 24:1565 –1567.

Piatetsky-Shapiro, G. (2014). Kdnuggets news on sigkdd service award.

Reddy, L. C. (2011). A review on data mining from past to the future. *International Journal of Computer Applications*, 15(7).

Rego, M. A.; Franca, E. B.; Travassos, A. P. & Barros, F. C. (2010). Assessment of the profile of births and deaths in a referral hospital. *J Pediatr (Rio J)*, 86(4):295–302.

Robnik-Sikonja, M. & Kononenko, I. (1997). An adaptation of relief for attribute estimation in regression.

Steinwart, I. & Christmann, A. (2008). *Support vector machines*. Springer.

Tan, P. N.; Steinbach, M. & Kumar, V. (2006). *Introduction to data mining*. Addison-Wesley, Boston.

Weber, B. G. & Mateas, M. (2009). A data mining approach to strategy prediction computational intelligence and games. *IEEE Symposium on Date*.

Witten, I. H. & Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Anexo A

Artigo Submetido ao Journal Artificial Intelligence in Medicine

NICeSim: A simulator based on machine learning techniques to support clinical decision making in neonatal intensive care units

Tiago Geraldo Ferreira[a,1], Fabio R. Cerqueira[a,1,*], Brunnella Alcantara Chagas de Freitas[b], Andreia Patricia Gomes[b], Alcione de Paiva Oliveira[a], Sylvia do Carmo Castro Franceschini[c], Rodrigo Siqueira-Batista[b]

[a] *Informatics Department, Universidade Federal de Viçosa, Minas Gerais State, Brazil*
[b] *Medicine and Nursing Department, Universidade Federal de Viçosa*
[c] *Health and Nutrition Department, Universidade Federal de Viçosa*

Abstract

This paper describes NICeSim, an open-source simulator that uses machine learning (ML) techniques to aid health professionals to better understand the diagnosis, treatment and prognostic of premature newborns in neonatal intensive care units (ICU). The application was developed and tested using a database of data collected in a brazilian public hospital. The available data was used to feed a ML pipeline that was designed to create a simulator capable of predicting the outcome (death probability) for newborns in neonatal ICU. Here, we present the techniques that are the basis of the simulator as well as some results obtained with the above mentioned dataset. Our statistical experiments showed that the resulting model for death prediction could achieve an accuracy of 86.7% and an area under the ROC curve of 0.84. This significant accuracy demonstrates that NICeSim might be used for hypothesis testing so to minimize in vivo experiments. Indeed, in an experiment performed by physicians of the research team, three key attributes were varied to understand how they affect the risk of death. The results showed that the model provides predictions that

*Corresponding author
Email addresses: tiago.g.f@gmail.com (Tiago Geraldo Ferreira), fabio.cerqueira@ufv.br (Fabio R. Cerqueira), frcerqueira@gmail.com (Fabio R. Cerqueira)

[1] Ferreira and Cerqueira contributed equally to this work.

are in very good agreement with the literature, demonstrating that NICeSim might be an important tool for supporting decision making in clinical practice. More than that, the method might be used as a template for the creation of similar computational solutions for any scenario of interest as long as appropriate data is provided.

Keywords: Clinical decision making, machine learning, bioinformatics, computational simulation

---

1. Introduction

Machine learning (ML) algorithms have been the basis of data mining frameworks for detecting patterns in large datasets. These frameworks compose computational solutions for many complex scenarios such as decision-making in business [1], and weather forecast [2], just to mention a few. In many entertainment and educational games, ML has also a crucial role in improving the artificial intelligence of computational agents [3]. In the medical field, ML approaches had, likewise, a great impact, being applied in a myriad of complex problems such as to reach a more precise diagnosis and prognosis of patients with severe diseases, to simulate drug reactions, and to predict the probability of rejection in transplants [4, 5, 6].

The ML framework described here is proposed as a general simulation method for understanding critical variables in complex problems. As long as appropriate data is provided, a learning model can be build to predict the probability of some outcome of interest given the values of key attributes of some instance being analyzed. The user has the chance of changing the attribute values to notice how they affect the outcome, so that hypotheses might be raised and easily tested in a quick and low-cost fashion.

Particularly in this work, we applied the proposed method for the case of premature newborns admitted to intensive care unities (ICU). The implementation of the method for this case gave rise to NICeSim, a simulator that predicts the risk of death of premature newborns. The resulting tool has the potential

of being very useful to aid health professionals to understand the weight that each newborn feature has in the outcome. For the dataset used here, the key attributes: Apgar Scale, Hyaline Membrane Disease, and Gestational Age are shown to have a great impact in the newborn risk of death.

Although there were some similar works in the past [7, 8], they were more concerned for providing real-time decision support in neonatal ICUs. NICeSim focuses the analysis of punctual data collected at the time of death or discharge of patients to build a ML model. The system provides the outcome prediction for instances whose attributes of interest can be easily altered. As a result, hypotheses for new treatments to minimize premature births and death of premature newborns might be build and tested in a very practical manner. Therefore, this kind of simulation might be of great utility for the government, hospitals, and related institutions that need to better understand the circumstances that lead to death at neonatal ICUs. Once the causes and effects are understood, large-scale national campaigns might be launched to decrease newborn deaths.

In NICeSim, the user can choose one of two ML algorithms to build a learning model: Support Vector Machine (SVM) or Neural network (ANN). We provide two options since different datasets might have different characteristics. Consequently, SVM might present a better performance in one case, while ANN might perform better in others. In the data used here, ANN showed a better result. The statistical experiments using 10-fold cross validation showed an accuracy of 86.7% and an area under the ROC (receiver operating characteristic) curve of 0.84. Using the resulting model, physicians simulated different scenarios for three key attributes of premature newborns: Apgar Scale, Hyaline Membrane Disease, and Gestational Age. The predicted outcomes in this experiments have shown a very good agreement of NICeSim to expected results already described in the literature, demonstrating that this system might provide a great support for research of new treatments to minimize severe problems regarding premature newborns.

2. Material and methods

2.1. Dataset and its preparation

We built a training set (TS) for the learning algorithms from a database with data collected in the Neonatal Intensive Care Unit of the General Hospital São Sebastião, at Viçosa, Minas Gerais State, Brazil, during the years 2008-2010 [9]. In 2009, this hospital became a health service reference unit for high risk pregnancies. Its neonatal ICU was inaugurated in March of 2004 and totaled 1,059 calls in December of 2010, out of which 70% were cases of premature newborns [10].

The first concern during the design and implementation of NICeSim was to preprocess the available data so to achieve our goals. The initial dataset had 114 attributes and 293 instances. This dataset was not balanced: 39 instances were positive cases (regarding newborn deaths in the neonatal ICU), while the remaining consisted of negative instances. To counteract the imbalance problem, we use a Cost Sensitive algorithm [11], where a cost matrix is applied to dictate that the penalty for predicting a false negative is higher than the penalty for a false positive.

The next step was to find the relevant attributes among the 114 initial ones. For this task, we submitted the dataset to an attribute selection process. Attribute selection, also known as feature selection, is a common step in the knowledge discovery process [12]. It aims at diminishing the complexity of a dataset by reducing its dimensionality. Many attributes in the initial set were identified as out of the scope of our simulations. Hence, based on a pre-analysis by our partner physicians as well as a deep study on the previous work that originated our dataset [9], we reduced the initial set of attributes to only the 15 presented in Table 1.

After this initial selection, the dimensionality of our dataset was greatly reduced. Then, a next attribute selection stage was performed, but, this time, using ML algorithms. Notice that the drastic reduction of attributes allowed us to run our learning algorithms for all possible subsets of the remaining attributes.

Table 1: Remaining attributes after attribute selection by medical knowledge

Attribute	Description
SIH	Occurrence of Severe intracranial hemorrhage
o2SatAdmission	Occurrence of Oxygen saturation just after ICU admission
ageWeigthRecovered	Age (days) when the newborn recovered its birth weight
ageParentalNutrition	Age (days) at the beginning of parenteral nutrition
antenatalCorticos	Whether corticosteroids where given to the pregnant before birth
reanimationDR	Whether the newborn was resuscitated at birth
IUGR	Occurrence of Intrauterine Growth Restriction
HMD	Occurrence of Hyaline Membrane Disease
agePlainDiet	Age (days) when reached nutritional requirements by enteral route
GA	Gestational age (weeks)
lateOnsetSepsis	Occurrence of Late Onset Sepsis
pH1to12h	Blood pH measure during the first 12 hours of life
PDA	Occurrence of Patent Ductus Arteriosus
apgarLess7	Occurrence of low Apgar Score, cutoff point is 7
ageEnteralNutrition	Age (days) when enteral feeding was initiated

By doing so, we could assign a score to each subset based on the sensitivity and specificity provided by the model resultant of the given attribute subset. Attribute selection methods that train a new model for each candidate subset and analyze the resulting model performance are also known as Wrapper methods [5]. We used 10-fold cross validation for each subset evaluation to minimize the possibility of overfitting. The subset that presented the better score value is composed by the following attributes: apgarLess7, IUGR, GA, HMD, PDA, ageParentalNutrition, agePlainDiet. Therefore, the other 8 attributes showed no evidence, for the given dataset, that they could be important to predict the outcome variable representing the occurrence of death.

The 7 selected features went through additional tests and analysis. At this stage, we analyzed mainly how each attribute would relate to our goals. Considering that the available database was originally built for different purposes [9, 10], our partner physicians concluded that three variables, in the way they where collected, would not make sense for the objectives of this work: PDA, ageParentalNutrition and agePlainDiet. As a consequence, the final dataset is composed by the four remaining attributes: apgarLess7, IUGR, GA, HMD.

Thus, our learning models are focused on detecting how these attributes affect the probability of death in premature newborns.

2.2. Bioethical aspects

This study involves humans as research participants (data collected in neonatal ICU). The project was submitted for analysis and approval by UFV Ethical Committee for research with human subjects (CEP), in accordance with Resolution 196/1996 and Resolution 466/2012 of the Brazilian National Health Council (Conselho Nacional de Saúde do Brasil).

2.3. The learning algorithms

NICeSim provides the user with two options of learning algorithms. We decided to do so because ML algorithms may show different performance for different training sets that present distinct characteristics. Both learning algorithms used here are part of the Weka library, a well known and extensible tool widely adopted for machine learning tasks [13].

2.3.1. Artificial neural networks

Artificial neural networks are an approach inspired by biological neural systems aiming at creating a powerful learning technique [14, 15, 13, 16]. Alike the human brain, ANNs are composed of a set of nodes connected by directed links. The so-called perceptron was the first proposed model [17]. In this simplified architecture, just two types of nodes (neurons) are present: Input nodes and a single output node. The former type of nodes represent attributes, whereas the latter node type represents the ANN output. Every input node is connected to the output node by a weighted link. The weights mean the strength of synaptic connections between neurons. In a perceptron, the output node calculates the weighted sum of the input values, subtracts the this sum by a bias value, and applies the result to a function termed activation function to generate the final output. The perceptron uses the signum function, i.e., if the sum value is positive, then it outputs $+1$, if it is negative, then the output is -1 [16]. Thus,

the perceptron training process is the adaptation of the weights until obtaining a satisfactory relation between input and output regarding what is observed in the training dataset.

The perceptron concepts have quickly evolved to a more complete approach called multilayer neural network. In this case, the network might contain several intermediary layers defined as hidden layers (Figure 1). Furthermore, more sophisticated activation functions are used, such as sigmoid (logistic) and hyperbolic tangent functions. These improvements, including an output layer with one or more nodes, provide more complex and useful decision boundaries. Additionally, the learning process generally applies a method called backpropagation, where the deviation between observed and expected outputs is used in a weight update formula in reverse direction, i.e., weights at level $d+1$ are updated before weights at level d [16].

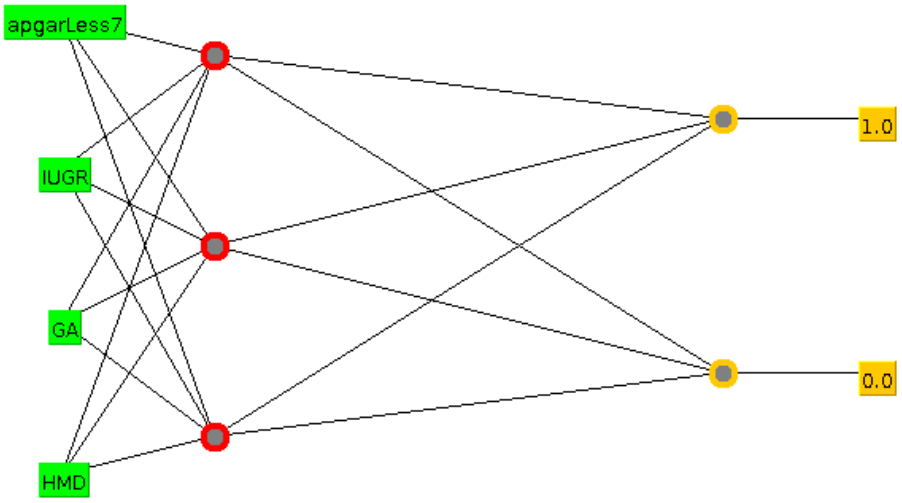

Figure 1: ANN architecture used in our approach. In this case, the hidden layer contains three nodes: $\lceil \frac{5}{2} \rceil$, where 5 is the number of attributes, including the outcome variable.

In NICeSim, we use the multilayer with backpropagation approach. The NICeSim ANN architecture is illustrated in Figure 1. As can be noticed, the input layer nodes correspond to the already-mentioned four features, there is one hidden layer with three nodes, and the output layer has two nodes, since we wish

to perform binary classification (death or not, where value 1 indicates death). We used sigmoid as activation function (so probabilities can be generated), epochs = 500, learning rate = 0.3, and momentum = 0.2.

2.3.2. Support vector machines

The second option of ML algorithm available to the user is the Support Vector Machine (SVM) method [18, 19, 16]. SVM constitutes a very powerful classification method. Figure 2 helps to understand the basic idea behind the construction of a decision boundary in a linear SVM. This example shows a two-dimensional dataset containing instances of two different classes (labeled by colors). This data is linearly separable, meaning that it is possible to find a hyperplane (a straight line in dimension 2) to distinguish elements of different classes as such, i.e., elements having different labels lie in opposite sides of the hyperplane. The Figure shows H_1 representing one such decision boundary. Notice that H_2 is likewise a valid hyperplane. As a matter of fact, there is an infinity of possible hyperplanes that perfectly separate the data. In Figure 2, H_1 has two associated hyperplanes H_{11} and H_{12}. Hyperplane H_{11} is obtained by moving a parallel copy of H_1 toward the elements of class blue until reaching the closest blue instance(s). Hyperplane H_{12} is generated moving a parallel copy of H_1 to the opposite direction until touching the closest red instance(s). H_2 has also its associated hyperplanes H_{21} and H_{22} obtained in the same fashion. Given a hyperplane representing a decision boundary, the distance between the two associated parallel hyperplanes is defined as the margin of the classifier. The fundamental principle in linear SVM is delineated from the fact that decision boundaries with large margins tend to minimize overfitting, whereas those with small margins, even having low training errors, generalize poorly on unknown data [16]. As a consequence, hyperplane H_1 in Figure 2 lead to a better model when compared to H_2.

The SVM method is formulated as a quadratic optimization problem for which well-established algorithms can be employed to find global optima of the objective function. The N instances in a TS with d attributes can be represented

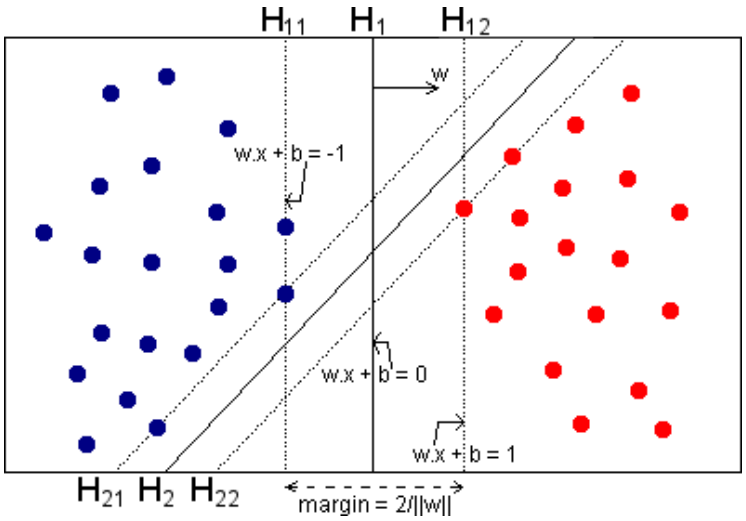

Figure 2: Decision boundary of a linear SVM classifier.

by tuples of the form $(\mathbf{x}_i, y_i)$, where $\mathbf{x}_i = (x_{i1}, x_{i2}, ..., _{id})^T$ is a vector containing the attribute values of the i^{th} instance and y_i indicates the class to which $\mathbf{x}_i$ belongs. For convenience, let $y_i \in \{-1, 1\}$. Any hyperplane can be described by the set of points $\mathbf{x}$ satisfying:

$$\mathbf{w} \cdot \mathbf{x} + b = 0, \tag{1}$$

where $\mathbf{w}$ is a normal vector to the hyperplane and $\frac{b}{\|\mathbf{w}\|}$ denotes the hyperplane offset from the origin along $\mathbf{w}$. Therefore, considering that we found $\mathbf{w}$ and b that describe H_1 in Figure 2 and using -1 and $+1$ as class labels in place of blue and red, respectively, the class of a given instance $\mathbf{x}_i$ can be predict as follows:

$$\text{y} = \begin{cases} -1, & \text{if } \mathbf{w} \cdot \mathbf{x}_i + b < 0; \\ +1, & \text{if } \mathbf{w} \cdot \mathbf{x}_i + b > 0. \end{cases} \tag{2}$$

Furthermore, it is possible to rescale $\mathbf{w}$ and b such that H_{11} and H_{12} can be represented as follows:

$$\begin{aligned} H_{11}: &\quad \mathbf{w} \cdot \mathbf{x} + b = -1, \\ H_{12}: &\quad \mathbf{w} \cdot \mathbf{x} + b = 1. \end{aligned} \tag{3}$$

By using geometry, the distance between H_{11} and H_{12}, i.e., the margin of the decision boundary, turns out to be $\frac{2}{\|\mathbf{w}\|}$. Consequently, we want to find $\mathbf{w}$ and

b that minimizes $f(\mathbf{w}) = \|\mathbf{w}\|$. In order to switch this task to a quadratic optimization problem, we can replace $\|\mathbf{w}\|$ with $\frac{1}{2}\|\mathbf{w}\|^2$, as the solution does not change (the same $\mathbf{w}$ and b are found in both cases). As a result, the learning procedure in linear SVM can be seen as the following constrained optimization problem:

$$\begin{aligned} & \min \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{subject to} \quad & y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \quad i = 1, 2, ..., N. \end{aligned} \tag{4}$$

This convex optimization problem with inequality constraints can be solved by the Lagrange multiplier method along with the Karush-Kuhn-Tucker (KKT) conditions [18, 20]. These techniques along with the use of kernel functions for nonlinearly-separable data, and some other convenient transformations, lead to the more familiar formulation of SVM contained in the main literature about this subject [18, 19, 16].

The execution of an SVM procedure in this work uses the complexity parameter $C = 1$ and buildLogisticModels = *true* to produce probabilities. The inclusion of kernel functions did not present any improvement to the simulator. Then, we proceeded with the linear SVM, since the training time is shorter.

2.4. The simulator

The simulator is composed by two screens. The initial one is shown in Figure 3. It is where the user provides all necessary information before starting the simulation: The training set, percentage of instances for training, the entries of the cost matrix, and the ML algorithm.

The TS is a text file in Attribute-Relation File Format (ARFF) [13]. After providing the TS, the model is built according to the given cost matrix and the chosen ML algorithm. When the model is ready, some statistical measures related to a 10-fold cross validation is presented, summing up the learning performance, as can be seen in Figure 3. Notice that the entries of the cost matrix are automatically filled with values that compensate the imbalance problem. The default values are as follows. In entries $[1, 1]$ and $[2, 2]$ that represent the costs of getting a true positive and a true negative, respectively, the values are

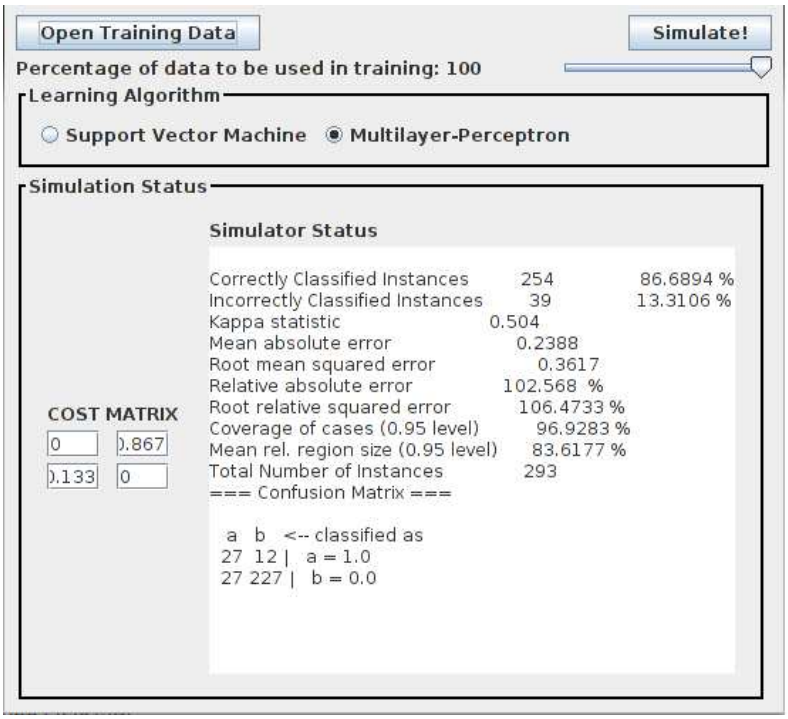

Figure 3: Input Screen. In this screen the user can set all the input parameters for the simulator.

zero, since we do not want to penalize correct classifications. In entry $[1, 2]$ that represents the cost of getting a false negative, the value is set to the percentage of negative instances. Finally, the entry $[2, 1]$ used for penalizing a false positive is filled with the percentage of positive values. In our dataset, there are 39 positive (death) instances and 254 negative instances. As a consequence, the penalization for a false negative is more than six times higher than in the case of a false positive so to overcome the imbalance issue. However, the user has the chance of changing the cost matrix values if it is convenient.

Once the model is built, the simulations may start off. This procedure is executed in the second screen of NICeSim, as depicted in Figure 4. When getting to this screen, a random instance of the TS is selected and has its attribute values (upper-right corner) and outcome (bottom) presented. Notice that the outcome is shown as a binary value (true or false - death or not) as well as a

percentage representing the chance of death. It is also possible to randomly get a new positive or negative instance from the TS, pressing the buttons available on the bottom.

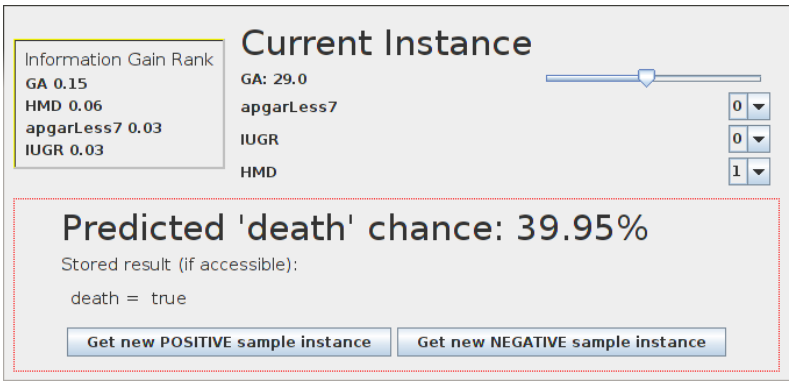

Figure 4: Simulation Screen. The user can change any value for the attributes and see the immediate effect on the outcome variable.

For performing simulations of interest, the instance's attribute values may be altered accordingly. For each alteration, the outcome is automatically predicted by the model and displayed on the screen, allowing a quick analysis of the impact that each changing might cause.

One important thing to notice is the attribute merit rank that is provided in the upper-left corner. Each attribute has a quality, represented by a score, assigned by the Information Gain (IG) method [16]. The information gain of an attribute expresses how much information this attribute yields on the class of the instances. The IG of an attribute a regarding to a class variable c is given by:

$$IG(c, a) = ent(c) - ent(c|a), \tag{5}$$

where $ent(c)$ is the entropy of the variable class c, and $ent(c|a)$ is the conditional entropy. The entropy of a class variable c is defined as:

$$ent(c) = -\sum_{i=1}^{k} p(i) \log_2 p(i), \tag{6}$$

where k is the number of class labels and p_i is the proportion of instances belonging to class i.

The IG scores make up a very useful information since they highlight the attributes that may yield the most significant effects in the outcome. Therefore, a specialist can combine knowledge with IG scores to perform simulations in an optimized manner.

2.5. Machine learning framework

Even though this work focuses in a specific case for newborns in ICUs, it is important to emphasize the general aspects presented here, i.e., as long as appropriate data is available, the machine learning steps presented in this work might be applied to any domain of interest, serving as a powerful tool for hypothesis testing and, consequently, decision making. Everything we have presented so far might be summarized by the following machine learning framework:

1. Conversion of raw data into an appropriate format: Here we chose the ARFF format;
2. Data preprocessing: Dimensionality reduction by attribute selection. It is important the participation of specialists so that a mix of knowledge and mathematical methods can be used to extract the most important variables. If convenient, sampling, discretization and binarization may also be useful;
3. Experiments with many ML algorithms: After the two steps above, the final TS is ready, allowing ML experiments to begin. The statistical performance of the obtained models has to be analyzed to select the ML approaches that perform better on the available data. It is important to remember the use of a cost matrix to counteract the imbalance issue;
4. Coding of a GUI (graphical user interface): Program a GUI embedding the learning algorithms chosen in the previous step, so that it makes easier the attribute value manipulation and its consequent effects in the outcome. Here, an attribute ranking method, such as Information Gain, might be of great interest to provide some optimized directions to the user.

3. Results and Discussion

3.1. Performance evaluation of SVM and ANN

To evaluate the SVM and the ANN models on the ICU dataset, we used two different statistical approaches largely adopted in the absence of testing data for measuring how general the models are: Data split and cross-validation [16].

Table 2: Statistical evaluation of SVM and ANN models in the simulator for the ICU data. Split refers to using 66% of the dataset for training, and the remaining for testing. The performance was also measured through 10-fold cross validations.

Test type	Measure	SVM	ANN
Split	Accuracy	77%	84%
Split	Sensitivity	Statistical p 100%	83.3%
Split	Specificity	73.9%	84.1%
10-fold CV	Accuracy	73.04%	86.68%
10-fold CV	Sensitivity	82.10%	69.2%
10-fold CV	Specificity	71.70%	89.4%

In the data split procedure, the TS is divided into two parts, one for training and another for testing. The most popular proportion is 66% for training and the remaining for testing.

In cross validation, the dataset is split into k disjunct parts of roughly the same size. One part is taken as test data, while the remaining $k-1$ parts are used to build the model. This process is repeated k times, each time using a different partition for testing. At the end, the validation results are averaged over the k executions. The most popular value for k is 10, which is the value we used in our experiments.

Table 2 summarizes the performance of each learning algorithm on ICU dataset in terms of sensitivity, specificity, and accuracy, measured by both validation methods described above. It can be noticed that ANN could achieve better accuracies and specificities, whereas SVM presented better sensitivities.

It demonstrates, thus, that it is important to provide both options in the simulator so that the user can choose the most suitable one. Furthermore, the simulator accepts other datasets, as long as the same structure is maintained. As a result, a different dataset might present a different characteristic, making the learning algorithms to show distinct performances.

The areas under the ROC curve were 0.83 and 0.84 for SVM and ANN, respectively. Figure 5 shows the ROC curve of the ANN model. As can be noticed in the figure and in Table 2, the prediction power demonstrated by the learning procedures is very good. As a consequence, we can assign a higher confidence to the simulation process. This is demonstrated in the next section.

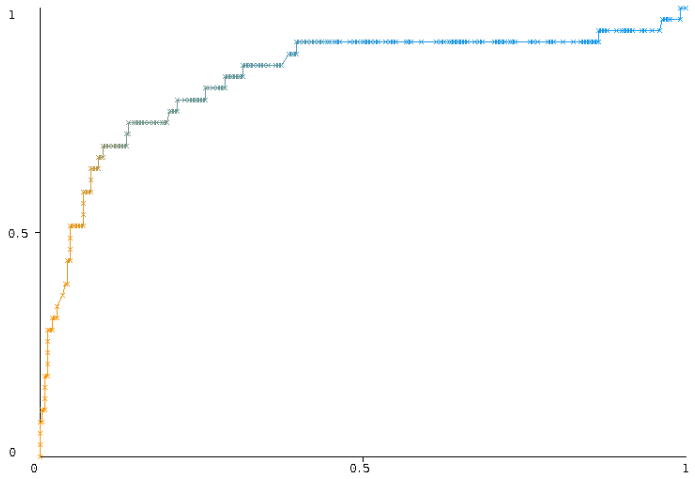

Figure 5: ROC curve of the ANN model. The area under the curve is 0.84.

3.2. Application for clinical purposes

This section describes some of the practical experiments we performed with NICeSim. After using the ANN approach to build a model on the ICU dataset, we executed a simulation targeting the risk of death of newborns, using 3 out of the 4 attributes present in the data: Apgar (less than 7), HMD, and GA. As a result, we could evaluate the importance of these attributes and the coherence of the model in order to check out the simulator validity.

The Apgar scale represents the clinical evaluation of newborns measured in the first 5 minutes of life. This scale measures 5 main subjects: Appearance (complexion), pulse rate, reflex irritability, activity, and respiratory effort. Each one of these measures can have a score ranging from 0 to 2, resulting in a final score ranging from 0 to 10. Scores above 7 are generally regarded as normal, while scores of 7 or less possibly mean that the newborn needs medical attention.

Hyaline Membrane Disease, or HMD, is related to the pulmonary maturity of newborns. This is associated with insufficient surfactant production. The main characteristic of HMD is the increased effort needed for the newborn to breathe.

GA (Gestational Age) is usually defined by an estimate that combines the obstetric ultrasound, the first day of the woman's last normal menstrual period, and the New Ballard score [21]. Here, we separated the newborns as follows [22]:

- Extreme preterm (EP): GA less than 28 weeks,
- high preterm (HP): GA ranging from 28 to 31 weeks, and
- moderate preterm (MP): GA ranging from 32 to 36 weeks.

The predictions of death chance based on the above-mentioned attributes are presented on Table 3. We can see that the outcomes provided by NICeSim are consistent with clinical information available in the literature. For instance, the scenario where the Apgar score is low, HMD is positive, and the gestational age is low demonstrates that all these factors clearly correlate to worse prognostics of newborns, which is previously shown by many authors [23, 24, 10]. Each of the attributes Apgar score and HMD, individually, demonstrates to be strongly correlated to the outcome as well. Either the Apgar score below 7 or the occurrence of HMD leads to a higher chance of death, mainly in EPs and HPs. Furthermore, when the Apgar score is 7 or more, and HMD is negative, the chances of death are very low, even for extreme preterms.

Table 3: Death risk of newborns according to: Apgar score, occurrence of HMD, and gestational age

Apgar	HMD	Gestational Age	Death Risk (%)
≤ 7	+	24	97.05
		28	97.04
		32	41.57
> 7	+	24	96.64
		28	40.28
		32	10.46
≤ 7	-	24	97.05
		28	96.65
		32	0.19
> 7	-	24	6.77
		28	0.12
		32	0.17

4. Conclusion

In this paper, we presented a machine learning framework to build a simulator for any domain of interest provided that the appropriate data is available. The idea is to allow an easy manipulation of attributes related to the problem in order to see the effects that the alterations cause to the outcome attribute. As a consequence, specialists might quickly raise and test new hypotheses with virtually no cost.

To validate the proposed framework, we first built a dataset that was extracted from a database with data collected in an intensive care unit for newborns in a public hospital of Brazil. After this, we implemented a simulator that used this data to build a machine learning model to predict the chance of death of premature newborns. The simulator allows the changing of the main attributes related to newborn death, providing the immediate prediction of death probability for each alteration. The results demonstrate that the model is very accurate and, as a consequence, the simulator produces answers that are in very good agreement with current clinical information. This can be very use-

ful for researches in the health field, since new hypotheses can be easily tested, narrowing the number of possibilities to validate *in vivo*.

In a future work, we intend to work closely with health professionals to build a more comprehensive dataset. Notice that the database that originated the data used here was not intended for the purposes presented in this work. We strongly believe that constructing a database with the original objective of running the described computational simulations would maximize the results that such in silico experiments could produce.

Acknowledgements

We would like to thank the funding agencies Capes, FAPEMIG, and CNPq for the financial support for this project.

URL for software download

The software is open-source and is available under the URL:

`http://sourceforge.net/projects/nicesim`

References

[1] T. Davenport, J. Harris, Competing on analytics: The new science of winning, Harvard Business School Press, 2007.

[2] A. S. Cofio, J. M. Gutirrez, B. Jakubiak, M. Melonek, Implementation of data mining techniques for metereological applications, Realizing Teracomputing (2003) 215–240.

[3] B. G. Weber, M. Mateas, A data mining approach to strategy prediction, in: IEEE symposium on computational intelligence and games, 2009, pp. 140–147.

[4] N. Nakayama, M. Oketani, Y. Kawamura, M. Inao, S. Nagoshi, K. Fujiwara, H. Tsubouchi, S. Mochida, Algorithm to determine the outcome of patients

with acute liver failure: A data-mining analysis using decision trees, Journal of Gastroenterology 47 (6) (2012) 664–677.

[5] U. J. Kang, P. Auinger, Activity enhances dopaminergic long-duration response in parkinson disease, Neurology 78 (15) (2012) 1146–9.

[6] Z. Hollander, V. Chen, K. Sidhu, D. Lin, R. T. Ng, R. Balshaw, G. V. Cohen-Freue, A. Ignaszewski, C. Imai, A. Kaan, S. J. Tebbutt, J. E. Wilson-McManus, R. W. McMaster, P. A. Keown, B. M. McManus, Predicting acute cardiac rejection from donor heart and pre-transplant recipient blood gene expression, The Journal of Heart and Lung Transplantation 32 (2) (2013) 259 – 265.

[7] M. Frize, R. Walker, Clinical decision-support systems for intensive care units using case-based reasoning, Medical Engineering and Physics 22 (9) (2000) 671 – 677.

[8] M. Frize, C. M. Ennett, M. Stevenson, H. C. Trigg, Clinical decision support systems for intensive care units: Using artificial neural networks, Medical Engineering and Physics 23 (3) (2001) 217 – 225.

[9] B. A. C. Freitas, Clinical, epidemiological and nutritional parameters of premature newborns treated in a neonatal intensive care unit in the municipality of viçosa-MG (Portuguese), Master's thesis, Universidade Federal de Viçosa, Department of Nutrition and Health, Brazil (2011).

[10] B. A. C. Freitas, M. Peloso, L. D. Manella, S. C. C. Franceschini, G. Z. Longo, A. P. Gomes, R. Siqueira-Batista, Late-onset sepsis in premature newborns treated in a neonatal intensive care unit: A three-years analysis (Portuguese), RBTI-Revista Brasileira de Terapia Intensiva 24 (2012) 79–85.

[11] C. Elkan, The foundations of cost-sensitive learning, in: M. K. P. Inc (Ed.), Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, Stoneham: Butterworth-Heinemann, 2001, pp. 16–27.

[12] I. Guyon, A. Elisseeff, An introduction to variable and feature selection, The Journal of Machine Learning Research archive 3 (2003) 1157–1182.

[13] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I. H. Witten, The weka data mining software: An update, ACM SIGKDD Explorations Newsletter 11 (1) (2009) 10–18.

[14] T. M. Mitchell, Machine Learning, McGraw-Hill, Singapore, 1997.

[15] P. Baldi, S. Brunak, Bioinformatics: The machine learning approach, 2nd Edition, The MIT Press, Massachusetts, 2001.

[16] P. N. Tan, M. Steinbach, V. Kumar, Introduction to data mining, Addison-Wesley, Boston, 2006.

[17] F. Rosenblatt, The perceptron, a perceiving and recognizing automaton Project Para, Cornell Aeronautical Laboratory report, Cornell Aeronautical Laboratory, 1957.

[18] N. Cristianini, J. Shawe-Taylor, An Introduction to support vector machines and other kernel-based learning methods, Cambridge University Press, Cambridge, UK, 2000.

[19] W. S. Noble, What is a support vector machine?, Nat. Biotechnol. 24 (2006) 1565 –1567.

[20] M. Avriel, Nonlinear programming: Analysis and methods, Dover Publications, New york, USA, 2003.

[21] M. Cohen-Wolkowiez, C. Moran, D. K. Benjamin, C. M. Cotten, R. Clark, B. DK, Early and late onset sepsis in late preterm infants, Pediatr Infect Dis J 28 (12) (2009) 1052–6.

[22] R. E. Behrman, A. S. Butler, Preterm birth: Causes, consequences, and prevention, Washington, DC: The National Academies Press.

[23] A. A. Fanaroff, M. Hack, M. C. Walsh, The NICHD neonatal research network: Changes in practice and outcomes during the first 15 years, Semin Perinatol 27 (4) (2003) 281–7.

[24] M. A. Rego, E. B. Franca, A. P. Travassos, F. C. Barros, Assessment of the profile of births and deaths in a referral hospital, J Pediatr (Rio J) 86 (4) (2010) 295–302.