

UNIVERSIDADE FEDERAL DO RIO GRANDE

Vinicius Lucena dos Santos

**Proposta de um modelo preditivo de tentativa
de suicídio para usuários de drogas utilizando
técnicas de Mineração de Dados**

Brasil

2018

UNIVERSIDADE FEDERAL DO RIO GRANDE

Vinicius Lucena dos Santos

**Proposta de um modelo preditivo de tentativa de suicídio
para usuários de drogas utilizando técnicas de Mineração
de Dados**

Trabalho acadêmico apresentado ao Curso de Engenharia de Computação da Universidade Federal do Rio Grande como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Computação.

Orientador: Eduardo Nunes Borges

Universidade Federal do Rio Grande – FURG

Centro de Ciências Computacionais

Curso de Engenharia de Computação

Brasil

2018

Resumo

Segundo um estudo realizado pelo Ministério da Saúde, o número de suicídios no Brasil aumentou durante 16 anos consecutivos, entre os anos de 2000 a 2016. Os principais fatores de risco para o suicídio incluem doença mental ou física, abuso de álcool e drogas dentre outros. O principal objetivo do trabalho apresentado nesta monografia é contribuir com a construção de modelos preditivos de tentativa de suicídio. Tais modelos utilizam algoritmos de Aprendizado de Máquina (AM) para tarefas de classificação. Como objetivo específico, pretende-se descobrir quais os atributos que mais influenciam na tentativa do suicídio.

Palavras-chave: Classificação. Knowledge Discovery in Databases. Tentativa de Suicídio. Machine Learning.

Abstract

According to a study carried out by the Ministry of Health of Brazil, the number of suicides in Brazil increased for 16 consecutive years between the years 2000 to 2016. The main risk factors for suicide include mental or physical illness, alcohol and drug abuse among others. The main objective of the work presented in this monograph is to contribute with the construction of predictive models of suicide attempt. These models use machine learning algorithms (ML) for classification tasks. The specific objective is to find out which attributes have the most influence in attempting a suicide.

Keywords: Classification. Knowledge Discovery in Databases. Suicide Attempt. Machine Learning.

Lista de ilustrações

Figura 1 – Processo de KDD	18
Figura 2 – Exemplo de um Ensemble Learning	19
Figura 3 – Exemplo de uma divisão randômica do <i>dataset</i> original.	21
Figura 4 – Ideia do algoritmo AdaBoost	23
Figura 5 – Um exemplo de um conjunto de dados linearmente separável em duas dimensões.	24
Figura 6 – Adicionando <i>features</i> para tornar o <i>dataset</i> linearmente separável . . .	25
Figura 7 – Sensibilidade com a escala dos dados	26
Figura 8 – Matriz de Confusão	26
Figura 9 – Exemplo de uma Curva ROC	27
Figura 10 – Cronograma	33

Lista de abreviaturas e siglas

KDD	Knowledge Discovery in Database
SVM	Support Vector Machines
HCPA	Hospital das Clínicas de Porto Alegre
PPGPSQ	Programa de Pós Graduação em Psiquiatria e Ciências do Comportamento
CPAD	Centro de Pesquisa em Álcool e Drogas

Sumário

1	INTRODUÇÃO	15
1.1	Objetivos	16
1.2	Estrutura da monografia	16
2	FUNDAMENTAÇÃO TEÓRICA	17
2.1	Descoberta de Conhecimento	17
2.1.1	Fases do KDD	17
2.2	Ensemble Learning	19
2.2.1	Bagging e Pasting	20
2.3	Florestas Aleatórias	20
2.3.1	Vantagens e Desvantagens	20
2.4	AdaBoost	21
2.4.1	Funcionamento	22
2.4.2	Vantagens e Desvantagens	23
2.5	SVM	24
2.5.1	Classificação Não-linear	25
2.5.2	Vantagens e Desvantagens	25
2.6	Métricas de Avaliação	26
3	METODOLOGIA	29
3.1	Os dados	29
3.2	Parametrização dos algoritmos	29
3.3	Avaliação dos Resultados	29
4	RESULTADOS ESPERADOS	31
5	CRONOGRAMA	33
	REFERÊNCIAS	35

1 Introdução

De acordo com um estudo realizado pela Organização Mundial da Saúde em 2014, mais de 1 milhão de pessoas tiram a própria vida por ano, é a segunda maior causa de mortes entre jovens de 15 a 29 anos de idade. No Brasil, em média, o número de suicídio é de 11 mil por ano, sendo a quarta maior causa de mortes entre os jovens do sexo masculino. Considerando apenas o grupo etário mais economicamente produtivo (15-44 anos), o suicídio é uma das três principais causas de morte em todo mundo, dessa forma, nota-se que também há impactos econômicos no ato de tirar a própria vida.

Segundo um estudo realizado pelo Ministério da Saúde, o número de suicídios no Brasil aumentou durante 16 anos consecutivos, entre os anos de 2000 a 2016. Os principais fatores de risco para o suicídio incluem doença mental ou física, abuso de álcool e drogas, mudança súbita na vida, como a perda de emprego, término de um casamento ou a combinação desses com outros acontecimentos. Para os amigos e familiares do suicida, o impacto é significativo, tanto imediatamente quanto a longo prazo.

O Hospital das Clínicas de Porto Alegre (HCPA) é uma instituição pública e universitária, ligada ao Ministério da Educação e à Universidade Federal do Rio Grande do Sul (UFRGS). O HCPA é responsável por prestar assistência a sociedade, bem como capacitar pessoas e realizar pesquisas de cunho científico. A pesquisa produzida no HCPA, introduz novos conhecimentos, técnicas e tecnologias que beneficiam toda a sociedade, além de formar novas gerações de pesquisadores. Um dos centros de pesquisa dentro do HCPA é o Centro de Pesquisa em Álcool e Drogas (CPAD) que trata de usuários de substâncias psicoativas.

O Departamento de Psiquiatria e Medicina Legal, da Universidade Federal do Rio Grande do Sul possui cursos de extensão, especialização, mestrado, doutorado e pós-doutorado. Um dos programas de pós-graduação é o Programa de Pós Graduação em Psiquiatria e Ciências do Comportamento (PPGPSQ) que foi desenvolvido para a produção e difusão de conhecimento científico e tecnológico em Psiquiatria.

Esta monografia faz parte do projeto de pós-doutorado do Prof. Dr. Eduardo Nunes Borges, intitulado *Análises avançadas de dados brasileiros sobre drogas com inteligência artificial e translação para a clínica* o qual pretende-se utilizar algoritmos e técnicas de *Machine Learning* para estimar o tempo de internação de pacientes usuários de substâncias psicoativas. Também pretende-se calcular a probabilidade de abandono de um usuário de drogas internado no HCPA. O projeto de pós-doutorado faz parte do PPGPSQ/UFRGS e utiliza dados no CPAD/HCPA.

1.1 Objetivos

O principal objetivo do trabalho apresentado nesta monografia é contribuir com a construção de modelos preditivos de tentativa de suicídio. Tais modelos utilizam algoritmos de Aprendizado de Máquina (AM) para tarefas de classificação.

Como objetivo específico pretende-se descobrir os atributos que mais influenciam na tentativa de um suicídio, isto é, pretende-se interpretar os modelos aprendidos. O tipo de interpretação dependerá do tipo de algoritmo utilizado.

1.2 Estrutura da monografia

O restante desta monografia está dividida da seguinte forma. O Capítulo 2 apresenta a Fundamentação Teórica das técnicas de *Machine Learning* utilizadas para construir os modelos de classificação propostos. O Capítulo 3 trata da metodologia utilizada, bem como a parametrização dos algoritmos e as métricas de avaliação empregadas. O Capítulo 4 apresenta os resultados esperados dos modelos. Por fim, no Capítulo 5 é apresentado o cronograma de atividades que serão desempenhadas para a conclusão deste Projeto de Graduação.

2 Fundamentação Teórica

Este capítulo aborda a fundamentação teórica necessária para o entendimento do presente trabalho, bem como dos modelos de classificação propostos. Primeiramente é introduzido o conceito de Descoberta de Conhecimento em Banco de Dados. Após, é detalhado o conceito e funcionamento de três algoritmos usados para classificação, são eles: Florestas Aleatórias, Support Vector Machine e Adaboost. Por fim, o capítulo encerra tratando das principais métricas de avaliação para modelos de classificação.

2.1 Descoberta de Conhecimento

A primeira vez que apareceu o termo *Descoberta de Conhecimento em Banco de Dados*, do inglês *Knowledge Discovery in Databases* (KDD), foi em 1989, por Piatetsky-Shapiro, com o objetivo de enfatizar que o produto final do processo é o conhecimento.

Para [Fayyad, Piatetsky-Shapiro e Smyth \(1996\)](#), o KDD refere-se ao processo geral de descobrir conhecimento útil a partir dos dados, essa extração de conhecimento é um processo não-trivial. A Figura 1 mostra o processo de KDD proposto por pelos autores [Fayyad, Piatetsky-Shapiro e Smyth \(1996\)](#). O termo processo implica que o KDD é composto por várias etapas, as quase são iterativas e interativas. Iterativa reflete o ciclo entre as etapas e, interativa reflete o caráter interdisciplinar do processo, uma vez que o KDD se aplica em inúmeras áreas e para resolver o problema muitas vezes é necessário a ajuda de um *expert* naquela área.

A seguir, será dada uma visão geral de cada uma das fases.

2.1.1 Fases do KDD

1) Seleção: A primeira etapa no processo é chamada de seleção, que é responsável por selecionar um *dataset* a partir de uma base de dados maior. A escolha desse *dataset* dependerá do domínio do problema que está tentando resolver. Em geral, a escolha é feita por um especialista ou por algum algoritmo de extração de características. O conjunto de dados selecionado é chamado de *Target Data*.

2) Pré-processamento: Após ter o conjunto de dados a ser utilizado, é comum que esses dados não estejam em boa qualidade, dessa forma, é necessário tratá-los e limpá-los. Operações básicas realizadas nesse nível devem lidar com:

- **Valores faltantes:** Valores nulos. Ocorre quando alguém deixa um campo de um formulário em branco, por exemplo.

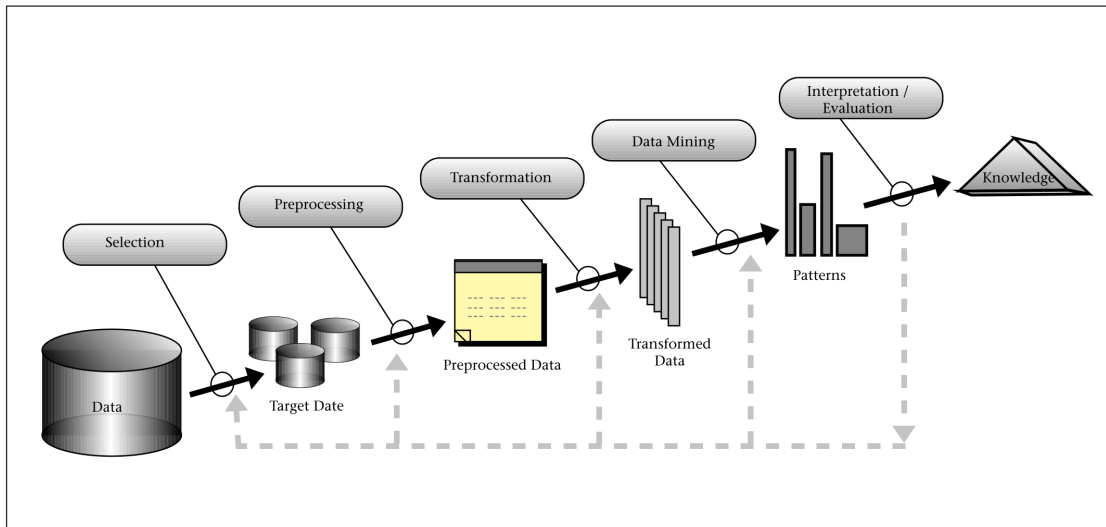


Figura 1 – Processo de KDD

(FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996)

- **Outliers:** Quando um valor foge muito do padrão dos dados. Deve ser identificado e eliminado do conjunto de dados.
- **Dados derivados:** Ocorre quando um dado pode ser obtido a partir de outro. Por exemplo, a idade de um indivíduo pode ser obtida a partir do seu ano de nascimento, dessa forma, não faz muito sentido ter esses dois atributos no conjunto de dados.

3) Transformação: Uma vez que os dados estão limpos e tratados, é necessário armazená-los e formatá-los adequadamente para que os algoritmos de aprendizado possam ser utilizados. A título de exemplo, se for usar um algoritmo de agrupamento baseado na distância euclidiana, é necessário converter todos os dados para numérico.

4) Mineração de Dados: Embora todas as etapas sejam importantes, a etapa de Mineração de Dados é a que recebe maior atenção. Conforme [Fayyad, Piatetsky-Shapiro e Smyth \(1996\)](#), mineração de dados é a aplicação de algoritmos específicos para extrair padrões a partir dos dados. De acordo com os mesmos autores, os algoritmos de mineração de dados podem ser divididos em cinco categorias: Associação, Classificação, Regressão, Segmentação e Sumarização. A tarefa na qual este projeto se encontra é a de Classificação.

A tarefa de classificação é responsável por identificar a qual classe um determinado registro pertence. Inicialmente, o modelo é “alimentado” com conjunto de registros já rotulados, isto é, é fornecida a qual classe os registros pertencem. Com esses dados, o algoritmo treina e aprende padrões que fazem com que um registro pertença a uma classe ou a outra (esse tipo de aprendizado é conhecido como aprendizado supervisionado).

5) Interpretação/Validação: Uma vez que o modelo já foi treinado, é medido a sua capacidade de generalização. Nesta etapa é fundamental a presença do especialista,

visto que, a melhor métrica de avaliação dependerá do negócio. Caso o desempenho não seja satisfatório, é possível retornar para qualquer etapa no processo de KDD, e o ciclo recomeça até chegar a um modelo que satisfaça o negócio.

2.2 Ensemble Learning

Um *ensemble* é composto por um grupo de preditores, que nada mais são do que algoritmos de aprendizado de máquina supervisionados. A técnica de combinar vários preditores é conhecida como *Ensemble Learning* (GERON, 2017). Existem vários métodos baseados nessa estratégia, os mais conhecidos na literatura são: *bagging*, *boosting* e *stacking*.

A figura 2 ilustra um *ensemble* composto por quatro preditores, dos quais três predizeram que uma nova instância é da classe 1 e um preditor disse que a nova instância é da classe 2. Como neste caso, o *ensemble* escolhe a classe de acordo com a votação majoritária de seus preditores, é atribuída a classe 1 à nova instância.

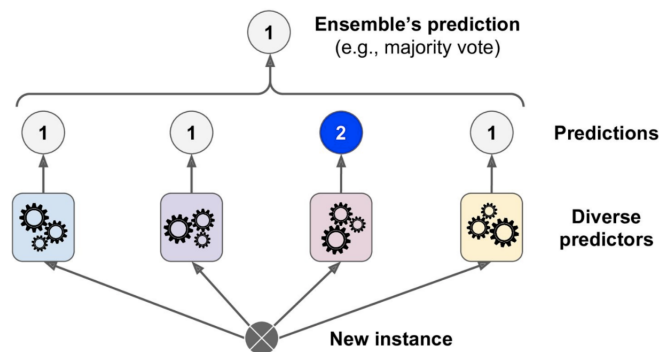


Figura 2 – Exemplo de um Ensemble Learning
(GERON, 2017)

Embora seja mais comum utilizar *Ensemble Learning* para classificação, também é possível utilizar essa técnica para problemas de regressão.

Uma maneira muito simples de agregar os resultados de cada preditor, a fim de chegar em um classificador ainda mais robusto, é escolher a classe com a maior quantidade de votos. Essa estratégia é conhecida como *Hard Voting Classifier* (GERON, 2017).

Contudo, é possível que haja classificadores bons e ruins, caso o número de preditores ruins supere o número de bons, é possível que o método *ensemble* seja prejudicado e tenha um resultado inferior ao melhor preditor. Para contornar esse tipo de problema, uma estratégia possível é atribuir pesos diferentes para os preditores, essa estratégia é conhecida como *Soft Voting Classifier* (GERON, 2017).

2.2.1 Bagging e Pasting

Quando for construir um modelo *ensemble*, uma abordagem possível é utilizar o mesmo algoritmo para treinar os preditores. Contudo, caso os preditores sejam treinados com o mesmo *dataset*, todos os preditores terão o mesmo desempenho, dessa forma, é necessário que os preditores sejam treinados com uma amostra aleatória do conjunto de dados. Caso a amostragem seja feita *com reposição*, esse método é chamado de *bagging* (BREIMAN, 1996). Caso a amostragem seja feita *sem reposição*, o método é chamado de *pasting* (BREIMAN, 1999).

2.3 Florestas Aleatórias

Florestas Aleatórias (do inglês *Random Forest*) é um algoritmo de aprendizado do tipo *ensemble* cujo os algoritmos são Árvores de Decisão (QUINLAN, 1986) treinadas com o método *bagging* (ver Seção 2.2.1). De acordo com Breiman (2001) a definição formal do algoritmo Floresta Aleatória é dada por:

Definição 1. *Uma floresta aleatória é um classificador que consiste em uma coleção de classificadores de árvores de decisão $h(x, \theta_k)$, onde θ_k são as features para o classificador k e, x é o vetor de entrada.*

O algoritmo de Floresta Aleatória introduz um certo nível de aleatoriedade ao treinar os classificadores; em vez de receberem o mesmo conjunto de dados, as árvores de decisão recebem um subconjunto diferente. Para se escolher o *dataset* de treinamento de cada árvore de decisão, é feita uma amostragem aleatória com reposição, dessa forma, é possível que uma instância seja escolhida mais de uma vez para compor o conjunto de treinamento de uma árvore de decisão.

Por exemplo, na figura 3 o *dataset* original foi dividido em 2 subconjuntos de dados, os quais serão utilizados para treinar os algoritmos de árvore de decisão. Nota-se que os subconjuntos θ_1 e θ_2 possuem tanto *features* quanto instâncias diferentes. Esse processo é importante, pois aumenta a capacidade de generalização do modelo de classificação (BREIMAN, 2001).

2.3.1 Vantagens e Desvantagens

Um ponto positivo das Florestas Aleatórias é o fato desse tipo de algoritmo poder ser usado tanto para problemas de classificação quanto para problemas de regressão.

Uma grande qualidade das florestas aleatórias é o fato de ser possível identificar as melhores *features*, em outras palavras, é possível saber quais são os atributos mais importantes para prever se um determinado registro irá pertencer a uma classe ou a

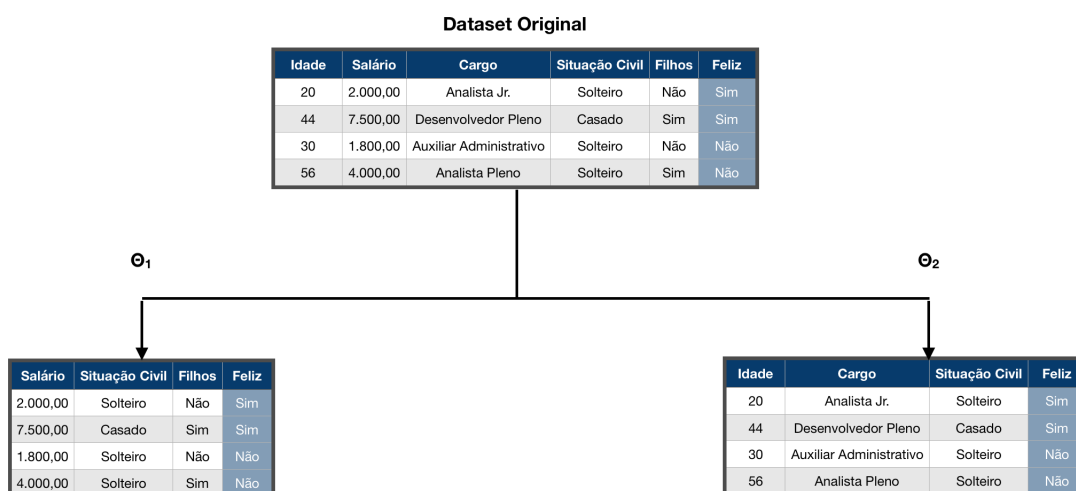


Figura 3 – Exemplo de uma divisão randômica do *dataset* original.

outra. Por exemplo, se o modelo está sendo treinado para identificar uma doença de acordo com os sintomas, é possível descobrir quais são os sintomas que mais influenciam em um paciente possuir uma determinada doença. Isso é uma característica importante, pois é possível compreender melhor o comportamento dos dados e interpretar o modelo aprendido.

Um grande problema de alguns algoritmos de aprendizado supervisionado é o *overfitting*, isto ocorre quando um modelo aprende muito bem o conjunto de treino e sua capacidade de generalização fica comprometida por conta disso. Na maioria das vezes, esse problema não irá acontecer com o modelo de Floresta Aleatória.

Quanto mais árvores na floresta melhor tende a ser a acurácia do modelo, ou seja, melhor o modelo se ajusta aos dados. Por outro lado, se há muitos preditores, o tempo de gasto para treinar o modelo acaba se tornando demasiadamente elevado, além de aumentar as chances do modelo sofrer *overfitting*.

2.4 AdaBoost

O modelo *boosting* refere-se a um método *ensemble* (sessão 2.2) para resolver problemas de classificação e regressão. A ideia geral do *boosting* é combinar vários algoritmos fracos em um modelo robusto. Esses algoritmos são treinados em sequência, onde cada um tenta corrigir seu antecessor (KEARNS, 1988).

O algoritmo do tipo *boosting* mais conhecido na literatura é o *Adaptive Boosting* ou simplesmente *AdaBoost*, que foi proposto por Freund e Schapire (1997). O algoritmo pode ser usado para melhorar o desempenho de qualquer algoritmo de *machine learning*.

Contudo, AdaBoost funciona melhor com preditores fracos ¹.

O preditor mais comum para ser usado com o AdaBoost são as árvores de decisão com um nível. Pelo fato dessas árvores possuírem apenas um nível, são conhecidas na literatura como *Decision Stump* (algo como Toco de Decisão, em vez de Árvore de Decisão).

2.4.1 Funcionamento

O trabalho de [Freund, Schapire e Abe \(1999\)](#) foi utilizado como referência para a explicação do funcionamento do algoritmo que será dada a seguir.

Inicialmente, é fornecido como entrada para o algoritmo, o conjunto de treino $(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)$, onde cada x_i pertence ao conjunto de instâncias X , e cada y_i pertence ao conjunto de rótulos Y . Para problemas envolvendo a classificação binária, considera-se $Y = \{-1, +1\}$.

O algoritmo Adaboost utiliza T preditores (os autores chamam os preditores de *hipóteses*) denotando as hipóteses como sendo $h_1(x), h_2(x), \dots, h_T(x)$, onde x representa uma instância a ser classificada pela hipótese $h_t(x)$.

Uma das principais ideias do algoritmo é manter um conjunto de pesos sobre o conjunto de treinamento. Isto é, para cada instância no conjunto de treinamento há um peso associado e, a medida que o algoritmo evolui, quanto mais difícil for para classificar uma instância de treinamento, maior será o peso atribuído a ela. De modo que, para os preditores subsequentes, a chance de classificar corretamente esse dado é maior. O peso dessa distribuição para um dado de treinamento i na rodada t é denotado por $D_t(i)$.

Inicialmente, todas as instâncias no conjunto de treinamento possuem exatamente o mesmo peso $w_i = \frac{1}{N}$, onde N é o tamanho do conjunto de treinamento. Mas, a cada rodada, o peso das amostras classificadas erroneamente são aumentados, dessa forma, os preditores são forçados a focar nas amostras mais difíceis de serem classificadas. A cada rodada t o peso de cada amostra é normalizado de acordo com o somatório de pesos de todas as amostras $Z_t = \sum_{i=1}^N D_t(i)$

A figura 4 ilustra a ideia do algoritmo AdaBoost, cada classificador fraco realiza uma classificação bem simples, focando em acertar corretamente as amostras com maior peso. No final, cada classificador fica “especializado” em classificar uma parte do conjunto e, a agregação de todos os classificadores fracos se transforma em um classificador forte, ou hipótese final, como os autores chamam.

¹ Preditores fracos são aqueles que alcançam uma acurácia pouco acima de um preditor aleatório.

Algoritmo 1: ADABOOST

Entrada: $(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)$, onde $x_i \in X$ e $y_i \in Y = \{-1, 1\}$

Saída: Classificador $H(x)$

1 Inicialize $D_1(i) = \frac{1}{N}$.

2 **para** cada $t = 1, \dots, T$ **faça**

3 Treine o preditor usando a distribuição D_t

4 Calcule a taxa de erro ϵ_t

5 Calcule o peso do preditor $\alpha_t = \frac{1}{2} \times \frac{1-\epsilon_t}{\epsilon_t}$

6 Atualize os pesos dos dados $D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t}, & \text{se } h_t(x_i) = y_i \\ e^{\alpha_t}, & \text{se } h_t(x_i) \neq y_i \end{cases}$

7 **fim**

8 **retorna** $\text{sign}\left(\sum_{t=1}^T \alpha_t h_t(x)\right)$

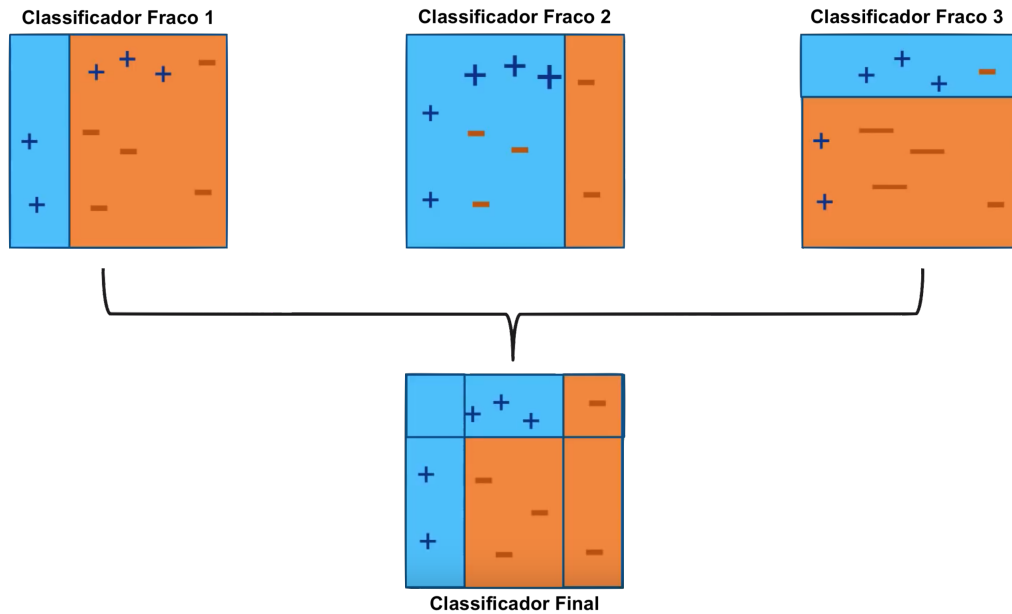


Figura 4 – Ideia do algoritmo AdaBoost

2.4.2 Vantagens e Desvantagens

De acordo com os autores do algoritmo, [Freund, Schapire e Abe \(1999\)](#), o AdaBoost possui várias vantagens. O algoritmo é rápido, simples e fácil para programar. Originalmente possui apenas um parâmetro, que é o número de iterações T . Além disso, é possível utilizar qualquer algoritmo de predição como classificador fraco. Finalmente, de acordo com os autores, a principal contribuição do AdaBoost é que vez de tentar construir um algoritmo de aprendizagem que classifique de forma correta todos os dados do conjunto de dados, é necessário encontrar apenas classificadores que sejam um pouco melhor que a classificação aleatória. Por fim, uma boa propriedade do AdaBoost é a sua capacidade de detectar *outliers*, como o algoritmo atribui pesos maiores a amostras mais difíceis de

serem classificadas, geralmente essas amostras são ruídos.

Por outro lado, vale a pena fazer algumas ressalvas. A performance de qualquer algoritmo *boosting* é claramente dependente dos dados e dos classificadores fracos. Além disso, o algoritmo pode falhar caso não haja dados de treinamentos suficientes.

2.5 SVM

Máquinas de Vetores de Suporte, do inglês *Support Vector Machines*, é um tipo de algoritmo de Aprendizado de Máquina supervisionado utilizado para problemas de classificação envolvendo duas ou mais classes. O algoritmo foi proposto por (CORTES; VAPNIK, 1995).

A ideia do SVM é mapear vetores de entrada em alguma dimensão maior, através de algum mapeamento não-linear, escolhido a priori. Neste novo espaço um hiperplano é construído para separar os dados (CORTES; VAPNIK, 1995).

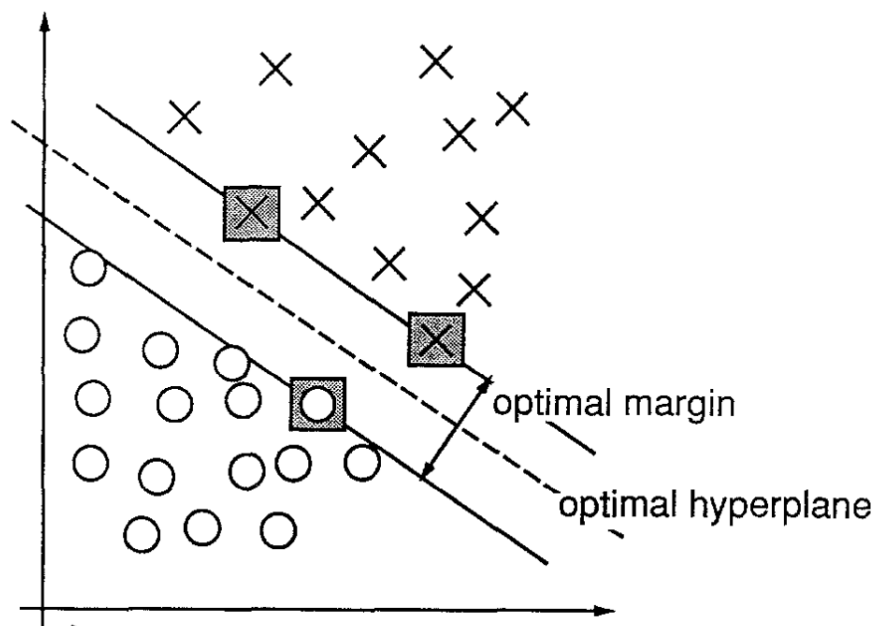


Figura 5 – Um exemplo de um conjunto de dados linearmente separável em duas dimensões.
(CORTES; VAPNIK, 1995)

A figura 5 mostra um conjunto de dados que pode ser separado de maneira linear. O objetivo do SVM é encontrar o melhor hiperplano, de maneira que a margem² seja a maior possível.

² Margem é o dobro da distância entre o hiperplano e o ponto mais perto

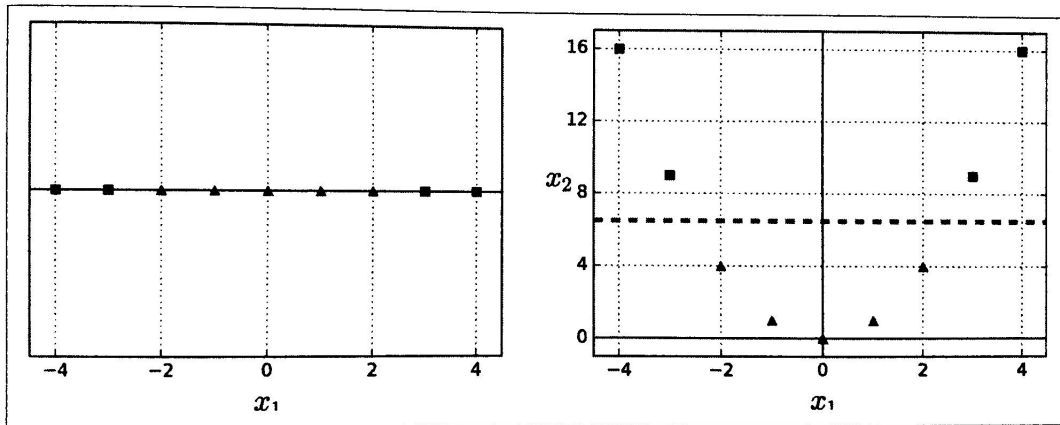


Figura 6 – Adicionando *features* para tornar o *dataset* linearmente separável
(GERON, 2017)

2.5.1 Classificação Não-linear

Embora o SVM funcione muito bem para classificação quando os dados são linearmente separáveis, a grande maioria dos *datasets* não são, nem de perto, linearmente separáveis. Para lidar com esse tipo de problema, a solução encontrada é utilizar mais *features* de acordo com alguma função de mapeamento, conhecido como *kernel* (GERON, 2017). Em alguns casos, essa transformação resulta em um novo *dataset* linearmente separável. O desenho à esquerda da figura 6 representa uma distribuição de dados não-linear. Após adicionar uma segunda *feature* $x_2 = (x_1)^2$, a distribuição resultante passou a ser linearmente separável, como mostra o desenho à direita. Além de *kernel* polinomial de 2º grau, existem outros tipos de função de mapeamento. As mais conhecidas são: Gaussiânica, Sigmoidal e Polinomial.

2.5.2 Vantagens e Desvantagens

Um *Support Vector Machine* é um modelo de classificação poderoso e versátil, capaz de realizar classificação linear e não linear, regressão e até detecção de *outliers*. SVMs são particularmente bem adequadas para problemas de classificação complexos e com *datasets* não muito grandes (GERON, 2017).

Um lado negativo do SVM é sua sensibilidade à escala dos dados de entrada. A figura 7 ilustra bem esse problema: no desenho da esquerda, onde x_0 e x_1 estão em escalas diferentes, a margem é bem pequena. Quando os dados são normalizados, como mostra no desenho da direita, a margem fica bem maior, possibilitando melhor generalização na hora de classificar novas instâncias.

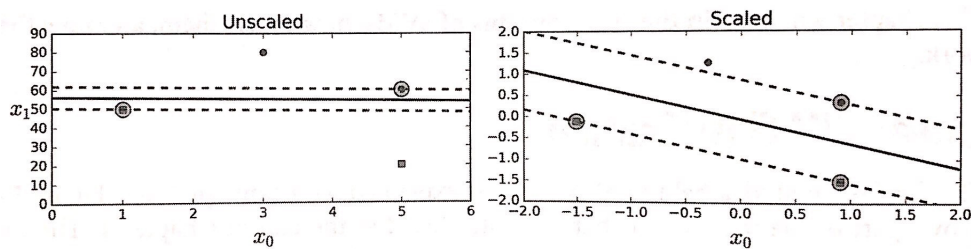


Figura 7 – Sensibilidade com a escala dos dados
(GERON, 2017)

2.6 Métricas de Avaliação

Uma etapa importante em qualquer tarefa de mineração de dados é avaliar os modelos aprendidos. As métricas de avaliação servem para que o pesquisador tenha subsídios de escolher qual o melhor modelo para o problema em questão. Antes de explicar as métricas de fato, será discutido a matriz de confusão.

A matriz de confusão é uma maneira de organizar os acertos e erros de algum modelo de classificação. Existem quatro situações possíveis: VP (Verdadeiro Positivo) ocorre quando uma amostra é da classe positiva e o modelo prediz corretamente. VN (Verdadeiro Negativo) ocorre quando a amostra é da classe negativa e o modelo acerta. FP (Falso Positivo) ocorre quando um amostra é da classe negativa mas o modelo classifica como sendo da classe positiva. E por fim, FN (Falso Negativo) ocorre quando uma amostra é da classe positiva mas o modelo classifica com sendo da casse negativa.

		Valor Predito	
		Positivo	Negativo
Valor Verdadeiro	Positivo	VP Verdadeiro Positivo	FN Falso Negativo
	Negativo	FP Falso Positivo	VN Verdadeiro Negativo

Figura 8 – Matriz de Confusão

De acordo com os valores de VP, VN, FP e FN, pode-se calcular várias medidas, as mais comuns, e as que foram utilizadas neste trabalho, são:

- Acurácia (ACC): A acurácia é a métrica de avaliação mais comum. Todavia, ela não é uma boa forma de avaliar um modelo quando há grande desbalanceamento de

classes.

$$Acurácia = \frac{VP + VN}{VP + VN + FP + FN} \quad (2.1)$$

- Precisão (PRE): A Precisão (Precision), avalia de quanto o modelo previu como sendo da classe positiva, quanto que ele acertou. A fórmula para o cálculo da acurácia é dado por:

$$Precisão = \frac{VP}{VP + FP} \quad (2.2)$$

- Revocação (REV): A medida Revocação, do inglês Recall, avalia o quanto o modelo acertou da classe negativa dividido por quanto era possível ter acertado. A Revocação também é conhecida como Taxa de Verdadeiro Negativo.

$$Revocação = \frac{TP}{TP + FP} \quad (2.3)$$

- F1-Score (F1): A medida F1-Score é uma média geométrica das duas medidas anteriores.

$$F1 = 2 \times \frac{Precisão \times Revocação}{Precisão + Revocação} \quad (2.4)$$

- Área abaixo da Curva ROC (AUC): A curva ROC (*Receiver Operating Characteristic*) é obtida traçando a Taxa de Verdadeiro Positivo pela Taxa de Falso Positivo. A medida utilizada é a área abaixo dessa curva resultante. Um exemplo dessa curva é mostrada na figura 9.

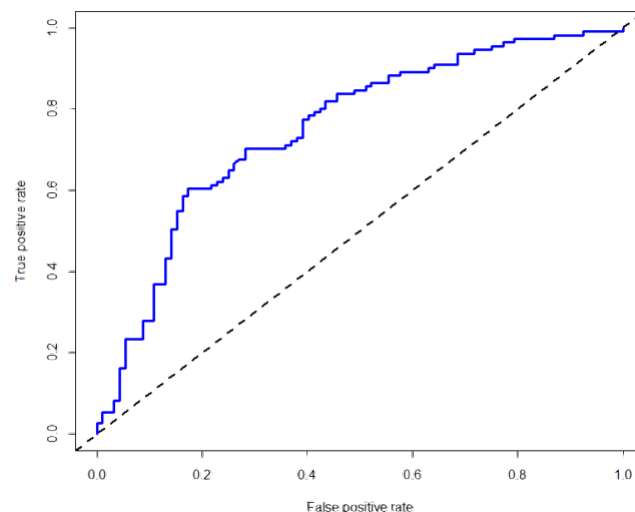


Figura 9 – Exemplo de uma Curva ROC

3 Metodologia

Este capítulo descreve a metodologia utilizada no desenvolvimento desta monografia. O capítulo inicia-se com a descrição dos dados (Seção 3.1). Após, é relatado a parametrização dos algoritmos empregados (Seção 3.2). Por fim, as métricas de avaliação dos algoritmos são apresentadas (Seção 2.6).

3.1 Os dados

Esta monografia foi concebida em parceria com o Hospital das Clínicas de Porto Alegre (HCPA), da Universidade Federal do Rio Grande do Sul (UFRGS). Os dados utilizados são de dependentes químicos internados de forma voluntária na Unidade de Internação para Tratamento de Alcoolismo e Dependência Química do HCPA.

3.2 Parametrização dos algoritmos

Quando se trata de uma pesquisa científica, é importante garantir a repetibilidade e reprodutibilidade dos experimentos. A tabela 1 mostra as configurações dos algoritmos utilizados. Tais algoritmos já foram discutidos em detalhes no capítulo 2.

Algoritmo	Parâmetros
Random Forest	numero de árvores = 10000
	semente = 123
	critério de divisão = entropy
	máxima profundidade = None
AdaBoost	preditor = None
	número de preditores = 50
	taxa de aprendizado = 1
	semente = 123
SVM	C = 1
	kernel = linear
	semente = 123

Tabela 1 – Configuração dos algoritmos de classificação.

3.3 Avaliação dos Resultados

Para o atual trabalho, as métricas de avaliação utilizadas foram:

- Acurácia (ACC)

- Precisão (PRE)
- Revocação (REV)
- F1-Score (F1)
- Área abaixo da Curva ROC (AUC)

Essas métricas foram escolhidas por serem as mais comuns quando se trata de problemas de classificação. No entanto, a melhor métrica dependerá do domínio do problema. Tais métricas foram discutidas na seção [2.6](#).

4 Resultados Esperados

Neste capítulo são discutidos os resultados que são esperados até a conclusão desta monografia.

Como dito no capítulo 1, o principal objetivo do atual projeto é utilizar técnicas de mineração de dados para propor modelos preditivos a fim de predizer se um determinado paciente tentará cometer suicídio ou não. Mais especificamente, o objetivo geral deste trabalho é propor (ao menos um) modelo de classificação com medida de Acurácia acima de 0.80 com a Revocação acima de 0.90.

5 Cronograma

A figura 10 descreve o cronograma de andamento do Projeto de Graduação.

Atividade		Dez	Jan	Fev	Mar	Abr	Mai	Jun	Jul	Ago
Levantamento Bibliográfico		X								
Fundamentação Teórica		X	X							
Pré-processamento dos dados		X	X							
Criar os modelos	Random Forest			X						
	SVM				X					
	Adaboost					X				
Análise e Interpretação							X			
Redação da Monografia		X	X	X	X	X	X	X		
Revisão									X	
Entrega do PG										X

Figura 10 – Cronograma

Referências

BREIMAN, L. Bagging predictors. *Machine learning*, Springer, v. 24, n. 2, p. 123–140, 1996. Citado na página 20.

BREIMAN, L. Pasting small votes for classification in large databases and on-line. *Machine learning*, Springer, v. 36, n. 1-2, p. 85–103, 1999. Citado na página 20.

BREIMAN, L. Random forests. *Machine learning*, Springer, v. 45, n. 1, p. 5–32, 2001. Citado na página 20.

CORTES, C.; VAPNIK, V. Support-vector networks. *Machine learning*, Springer, v. 20, n. 3, p. 273–297, 1995. Citado na página 24.

FAYYAD, U. M.; PIATETSKY-SHAPIRO, G.; SMYTH, P. From data mining to knowledge discovery in databases. *AI Magazine*, v. 17, p. 37–54, 03 1996. Citado 2 vezes nas páginas 17 e 18.

FREUND, Y.; SCHAPIRE, R.; ABE, N. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, JAPANESE SOC ARTIFICIAL INTELL, v. 14, n. 771-780, p. 1612, 1999. Citado 2 vezes nas páginas 22 e 23.

FREUND, Y.; SCHAPIRE, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, Elsevier, v. 55, n. 1, p. 119–139, 1997. Citado na página 21.

GERON, A. *Hands-On Machine Learning with Scikit-learn & TensorFlow*: Concepts, tools, and techniques to build intelligent systems. [S.l.]: O'Reilly, 2017. Citado 3 vezes nas páginas 19, 25 e 26.

KEARNS, M. Learning boolean formulae or finite automata is as hard as factoring. *Technical Report TR-14-88 Harvard University Aikem Computation Laboratory*, 1988. Citado na página 21.

QUINLAN, J. R. Induction of decision trees. *Mach. Learn.*, Kluwer Academic Publishers, Hingham, MA, USA, v. 1, n. 1, p. 81–106, mar. 1986. ISSN 0885-6125. Disponível em: <<http://dx.doi.org/10.1023/A:1022643204877>>. Citado na página 20.