

Estudo de técnicas de detecção de anomalias e suas aplicações

Helena Almeida Victoretti

Luciana de Melo e Abud

TRABALHO DE CONCLUSÃO DE CURSO
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
DA
UNIVERSIDADE DE SÃO PAULO

Orientador: Prof. Dr. João Eduardo Ferreira

Coorientador: Dr. Pedro Losco Takecian

São Paulo, 2015

Resumo

Anomalias são dados que não se comportam de forma esperada, podendo indicar eventos que merecem maior atenção do estudioso do domínio. Sendo assim, detectá-las de forma correta é importante.

Este projeto tem por objetivo estudar as principais técnicas de detecção de anomalias e implementar uma técnica para ser utilizada em um problema real.

A escolha da técnica de detecção de anomalias a ser utilizada depende das características dos dados sendo analisados, que interferem na qualidade da técnica. Por essa razão a escolha da técnica utilizada neste trabalho foi baseada no estudo e comparada com uma técnica previamente implementada para o mesmo problema.

Esta monografia foi desenvolvida para a disciplina MAC 0499 - Trabalho de Formatura Supervisionada e está dividida duas partes:

- Parte Objetiva: Contém os fundamentos e a parte técnica do trabalho.
- Parte Subjetiva: Contém a parte subjetiva do trabalho como dificuldades encontradas e a relação do projeto com o curso de graduação.

Sumário

I	Parte Objetiva	3
1	Introdução	4
1.1	Motivação	5
1.2	Objetivo	5
2	Conceitos e levantamento das principais técnicas	7
2.1	Detecção de anomalias	7
2.2	Principais técnicas de detecção de anomalias	7
2.2.1	Baseadas em classificadores	8
2.2.2	Baseadas em distância	12
2.2.3	Baseadas em densidade	13
2.2.4	Baseadas em <i>clustering</i>	14
2.2.5	Baseadas em modelos estatísticos	15
2.3	Validador estatístico de lotes de dados	17
3	Detecção de anomalias baseada em distância	21
3.1	Método do k-ésimo vizinho mais próximo (kNN)	22
3.1.1	Algoritmo <i>simple nested-loop</i>	22
3.1.2	Algoritmo <i>index-based</i>	23
3.1.3	Algoritmo <i>partition-based</i>	24
3.1.4	Algoritmo <i>nested-loop ANNS</i>	25
3.2	Métricas de distância	26
4	Implementação do módulo de detecção de anomalias	28
4.1	Detector de anomalias	28
4.2	Algoritmo kNN	29
5	Estudo de caso	32
5.1	Atributo comportamental tipo sanguíneo	32
5.2	Atributo comportamental sexo	33
6	Conclusão	37
II	Parte Subjetiva	38
7	Luciana de Melo e Abud	39
7.1	Desafios e frustrações	39
7.2	Relacionamento do trabalho com o curso do BCC	39
8	Helena Almeida Victoretti	41
8.1	Desafios e frustrações	41
8.2	Relacionamento do trabalho com o curso do BCC	41
	Apêndice	43

Parte I

Parte Objetiva

1 Introdução

Detecção de anomalias refere-se ao problema de encontrar instâncias de um conjunto de dados que apresentam um grande afastamento das demais, ou que são inconsistentes com elas. Tais instâncias são denominadas anomalias. Nas palavras de D. Hawkins [1], anomalia é “uma observação que se desvia tanto das outras observações a ponto de levantar suspeita de que ela foi gerada por um mecanismo diferente”. Por frequentemente indicarem eventos importantes, anomalias requerem uma maior análise para que se possa chegar a uma conclusão sobre suas causas.

A Figura 1 ilustra exemplos de dados normais e de uma anomalia.

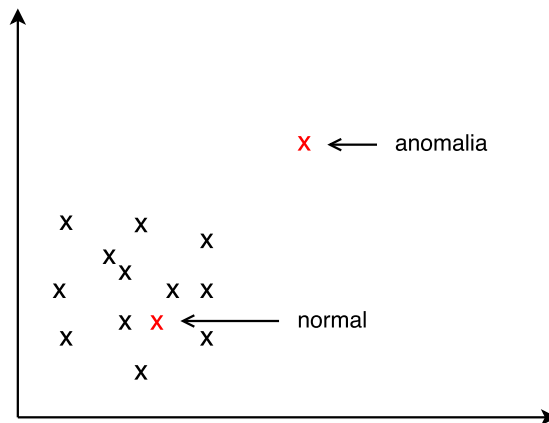


Figura 1: Exemplo de instâncias normais e de uma anomalia

Técnicas de detecção de anomalias são úteis em áreas como mineração de dados, limpeza de dados e *data warehousing* [2]. Elas são utilizadas em sistemas com diferentes finalidades. Alguns exemplos são:

- Detecção de invasão em redes de computadores [3; 4]: os padrões de comportamento da rede são analisados e são gerados mapeamentos de transmissão de seus pacotes. Uma anomalia detectada nesses mapeamentos pode indicar uma invasão na rede.
- Verificação de fraude bancária [5]: a detecção de uma anomalia neste contexto pode significar uma fraude no uso de cartão de crédito. As fraudes são refletidas em registros de transações e correspondem a pagamentos de alto valor, compra de itens que nunca foram adquiridos anteriormente pelo cliente, taxa elevada de compras, entre outros.
- Detecção de doença por análise de imagens [6]: a anomalia é detectada no processamento de imagens de ressonância magnética de mama, e pode indicar presença de tumor.

Existem diferentes técnicas de detecção de anomalias que podem ser aplicadas em um conjunto de dados, e cada uma possui suas características. Elas utilizam métodos de diferentes áreas, como aprendizado de máquina, e podem ser baseadas em classificadores, em métodos estatísticos, em distâncias, em densidade, entre outros [7].

A escolha da técnica a ser utilizada depende basicamente do domínio do problema e dos dados de entrada fornecidos [8].

Além de saber aplicar cada técnica, é importante entender as características e o funcionamento de cada uma.

1.1 Motivação

Um *data warehouse* é uma das principais arquiteturas utilizadas para a integração de dados provenientes de diversos bancos de dados distintos, porém um dos principais problemas enfrentados na construção de um *data warehouse* é a consistência dos dados recebidos. Para tentar resolver tal problema foi idealizado um sistema de validação estatística de lote de dados na tese de doutorado de Pedro Takecian [8] e desenvolvido por Eduardo Dias Filho [9].

Esse sistema de validação foi desenvolvido para o projeto REDS-II (*Retrovirus Epidemiology Donor Study - II*) [10], uma iniciativa cujo intuito é desenvolver projetos de pesquisa com enfoque em segurança transfusional de sangue, e cujo banco de dados armazena dados provenientes de diversos anos de doações de seus hemocentros participantes.

O sistema tem por objetivo validar lotes de dados recebidos pelas diferentes fontes de dados. Um lote de dado, no contexto do validador, é um conjunto de arquivos de dados referentes a um período de funcionamento do sistema transacional.

Esses lotes são recebidos de forma passiva, ou seja, as organizações responsáveis pelos sistemas transacionais ficam com a responsabilidade de fornecer os dados necessários de maneira correta e precisa, respeitando um formato de envio dos dados previamente estabelecido. Dessa forma não há uma garantia sobre a consistência dessas informações recebidas, uma vez que perde-se o controle do mecanismo de geração dos dados. Assim, é necessária uma validação dos lotes recebidos antes que estes sejam inseridos no *data warehouse*, de modo a não tornar o banco inconsistente.

Para manter a corretude dos dados, é preciso verificar se os atributos dos lotes respeitam um determinado domínio e se os dados seguem uma lógica previamente determinada. Essas verificações são as validações sintáticas e semânticas, respectivamente. Entretanto, essas verificações não são suficientes. O sistema de validação descrito acima realiza uma validação estatística dos lotes recebidos, baseada na proporção dos atributos de cada lote recebido.

A validação, neste caso, consiste em verificar se um lote recebido é válido ou inválido, baseando-se em lotes previamente classificados. Como em geral existem poucos lotes inválidos, para a classificação do lote utiliza-se detecção de anomalias. Assim, a escolha da técnica de detecção a ser utilizada é um importante fator para a eficiência do sistema de validação.

O sistema desenvolvido recebe módulos com implementações de diferentes métodos de detecção de anomalias. Esses métodos são implementados seguindo uma certa interface e obedecendo a certas regras, e depositados no núcleo do sistema como módulos.

Atualmente, o sistema realiza a detecção de anomalias utilizando um módulo baseado em um modelo estatístico baseado em distribuições gaussianas. Embora esse modelo seja apropriado para muitas situações, existem casos para os quais ele não é adequado, sendo necessário algum outro método de detecção que se adeque melhor ao problema.

1.2 Objetivo

Este projeto tem como objetivo estabelecer um estudo de caracterização e delimitação de algumas das diferentes técnicas de detecção de anomalias. O estudo possibilita escolher

uma técnica de detecção de anomalias que seja adequada ao domínio de um problema real para ser estudado. Este problema consiste em implementar um módulo de detecção de anomalias para o núcleo do sistema de segurança transfusional de sangue. A escolha dessa técnica será um desafio nesse trabalho, uma vez que precisamos encontrar uma técnica que se adeque às características dos dados do problema de validação de lotes de dados. Após implementada, a técnica será aplicada em um conjunto de dados, a ser escolhido, de modo que seja possível testar o desempenho e a qualidade da técnica escolhida.

2 Conceitos e levantamento das principais técnicas

2.1 Detecção de anomalias

Anomalias são instâncias de um conjunto de dados que não seguem um comportamento padrão esperado. Essas instâncias não possuem uma característica específica e são de grande interesse para os especialistas do domínio em questão, pois indicam eventos importantes e requerem maior atenção. Elas podem ser univariadas (possuem apenas uma dimensão) ou multivariadas (possuem mais de uma dimensão).

As anomalias podem ser classificadas em três grupos [7; 11]:

1. Anomalias pontuais: uma instância do conjunto de dados é considerada uma anomalia quando comparada com o conjunto todo de dados. São as anomalias mais simples de serem detectadas e possuem o maior número de técnicas de detecção de anomalias.
2. Anomalias coletivas: um subconjunto do conjunto de dados é considerado anômalo, mas as instâncias individuais desse subconjunto não necessariamente são consideradas anômalas.
3. Anomalias contextuais: uma instância do conjunto de dados é considerada uma anomalia em relação a um determinado contexto, mas fora deste não necessariamente é considerada anomalia. Nesse caso cada instância de dado é caracterizada por atributos contextuais (que determinam o contexto) e atributos de comportamento (características não contextuais da instância). Uma anomalia contextual também pode ser classificada como pontual ou coletiva.

Detecção de anomalias refere-se ao problema de encontrar instâncias em um conjunto de dados que não possuem um comportamento esperado - as anomalias.

As técnicas de detecção, em geral, consistem em determinar quais instâncias de um dado conjunto são anomalias, atribuindo a essas instâncias um rótulo (anomalia ou normal) ou uma pontuação de anomalia [7; 11]. As técnicas delimitadas nesta seção seguem esse comportamento. Entretanto, uma outra forma de detecção de anomalias se resume em determinar se uma certa instância é uma anomalia quando analisada em um conjunto de dados. A técnica implementada e delimitada na Seção 4 foi adaptada para seguir esse comportamento.

2.2 Principais técnicas de detecção de anomalias

Existem muitas técnicas de detecção de anomalias e o uso de cada uma depende principalmente do conjunto de dados a ser analisado.

Há duas principais fases envolvidas nas técnicas de detecção de anomalias: a fase de treino e a fase de teste. A fase de treino consiste em uma fase inicial do detector de anomalias, em que se define parâmetros utilizados pelo detector a partir de dados do conjunto de treino. A fase de teste consiste em determinar se um dado do conjunto de teste é ou não uma anomalia. O conjunto de teste contém as instâncias a serem classificadas como anomalias ou normais e é normalmente distinto do conjunto de treino [12].

Uma primeira forma de classificação das técnicas leva em conta a rotulação dos dados de treinamento.

Deste modo, as técnicas podem ser divididas em três classes [11]:

- Supervisionadas: assume-se que o conjunto de dados de treino é composto por instâncias rotuladas como normais ou anômalas.
- Semi-supervisionadas: assume-se que o conjunto de dados de treino é composto por instâncias rotuladas para apenas uma das classes (anômala ou normal). Normalmente as instâncias normais são as rotuladas, pois acontecem com mais frequência e possuem geralmente um padrão conhecido.
- Não supervisionadas: assume-se que o conjunto de treino não é composto por instâncias rotuladas.

Uma técnica de detecção de anomalias pode rotular as instâncias de teste ou identificar uma pontuação de anomalia para cada instância.

Algumas das principais técnicas de detecção de anomalias podem ser classificadas em cinco categorias [7; 11]:

1. Baseadas em classificadores.
2. Baseadas em distância.
3. Baseadas em densidade.
4. Baseadas em *clustering*¹.
5. Baseadas em métodos estatísticos.

2.2.1 Baseadas em classificadores

Um classificador consegue classificar os dados recebidos entre anômalos ou normais. Essas técnicas possuem dois estágios: um de treino e um de teste.

Existem quatro principais subcategorias:

1. Baseadas em Redes Neurais: As técnicas dessa subcategoria podem ser utilizadas para classificação dos dados em uma classe ou em mais de uma classe [11]. Uma rede neural é um dispositivo de processamento que se baseia na estrutura dos neurônios dos seres vivos. Ele é tipicamente organizado em camadas, como ilustrado na Figura 2 [13].

A primeira camada corresponde às variáveis de entrada, e a última, às classes de saída (ou à classe de saída). Pode haver uma ou mais camadas (chamadas de “ocultas”) entre elas. Cada conexão entre dois nós é associada a um peso, e os valores de saída dos nós são determinados por uma função aplicada a eles. Os valores dos nós da camada de entrada são determinados pelos dados de entrada, enquanto que os valores dos demais nós são calculados a partir dos valores de uma função de ativação aplicada aos nós das camadas anteriores e dos pesos das arestas de conexão [14].

Na fase de treino, a rede neural aprende valores para os pesos das arestas de forma que, para qualquer instância de entrada rotulada, a saída seja uma boa aproximação do rótulo [15]. Para isso, os pesos são modificados camada por camada

¹Neste trabalho, o termo *clustering* não será traduzido para “agrupamento” por ser um termo reconhecido na área na sua forma em inglês.

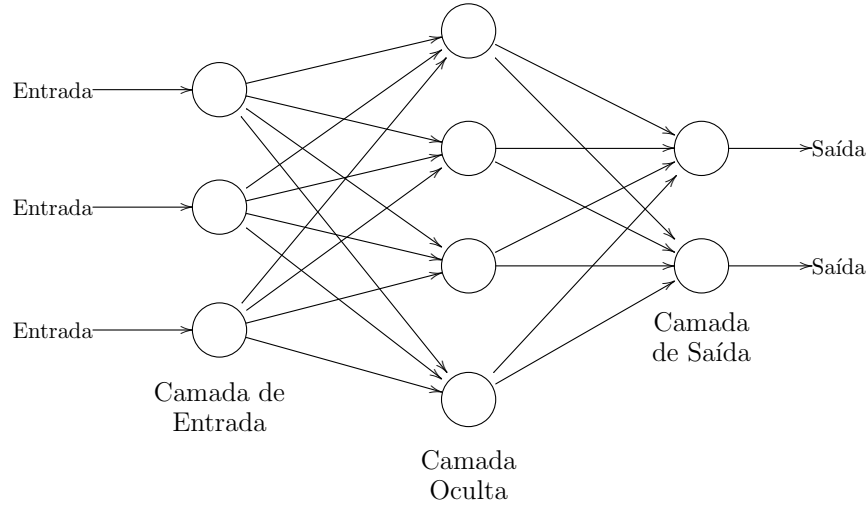


Figura 2: Exemplo da estrutura de uma rede neural

de modo a minimizar o erro do resultado. Já no estágio de teste a rede recebe uma instância, e se ela obtiver o rótulo de uma classe normal é classificada como normal, senão como anômala [11].

Um exemplo de aplicação dessa técnica é a detecção de fraude de cartão de crédito. John Akhilomen [16] desenvolveu um sistema de detecção para *cyber credit card fraud*. *Cyber credit card fraud* pode ser definido como a utilização de informações roubadas de cartões de crédito para compras e/ou pagamentos na internet. O sistema utiliza detecção de anomalias supervisionada baseada em Redes Neurais, e por meio de dados referentes ao comportamento padrão do usuário verifica se a transação que está sendo executada é ou não fraudulenta. Na fase de treino, a Rede Neural utiliza como dados de entrada as informações sobre a localização geográfica, o *e-mail* e telefone cadastrado, os tipos de produtos e serviços comprados e os sites normalmente utilizados pelo usuário nos últimos um ou dois anos. A Figura 3 ilustra um diagrama de representação da Rede Neural utilizada.

Um outro exemplo de uso dessa técnica é o algoritmo proposto por Ghosh et al. [17], que utiliza redes neurais para detectar ataques a redes de computadores, classificando os dados recebidos entre normais e anômalos.

2. Baseadas em Redes Bayesianas: Utilizada em classificação com uma ou mais de uma classe e com instâncias rotuladas na fase de treino. Uma rede Bayesiana é uma estrutura gráfica probabilística, representada por um digrafo acíclico, em que cada nó do grafo representa uma variável aleatória X_i ($i \in \{1, 2, \dots, d\}$, $d \in \mathbb{N}$), e as arestas representam dependências probabilísticas entre as variáveis [18]. Essa rede define uma distribuição de probabilidade conjunta, dada por:

$$P(X_1, X_2, \dots, X_d) = \prod_{i=1}^d P(X_i | \Pi_{X_i}),$$

em que Π_{X_i} é o conjunto de todos os pais de X_i (nós que são origem das arestas que têm destino em X_i) [19].

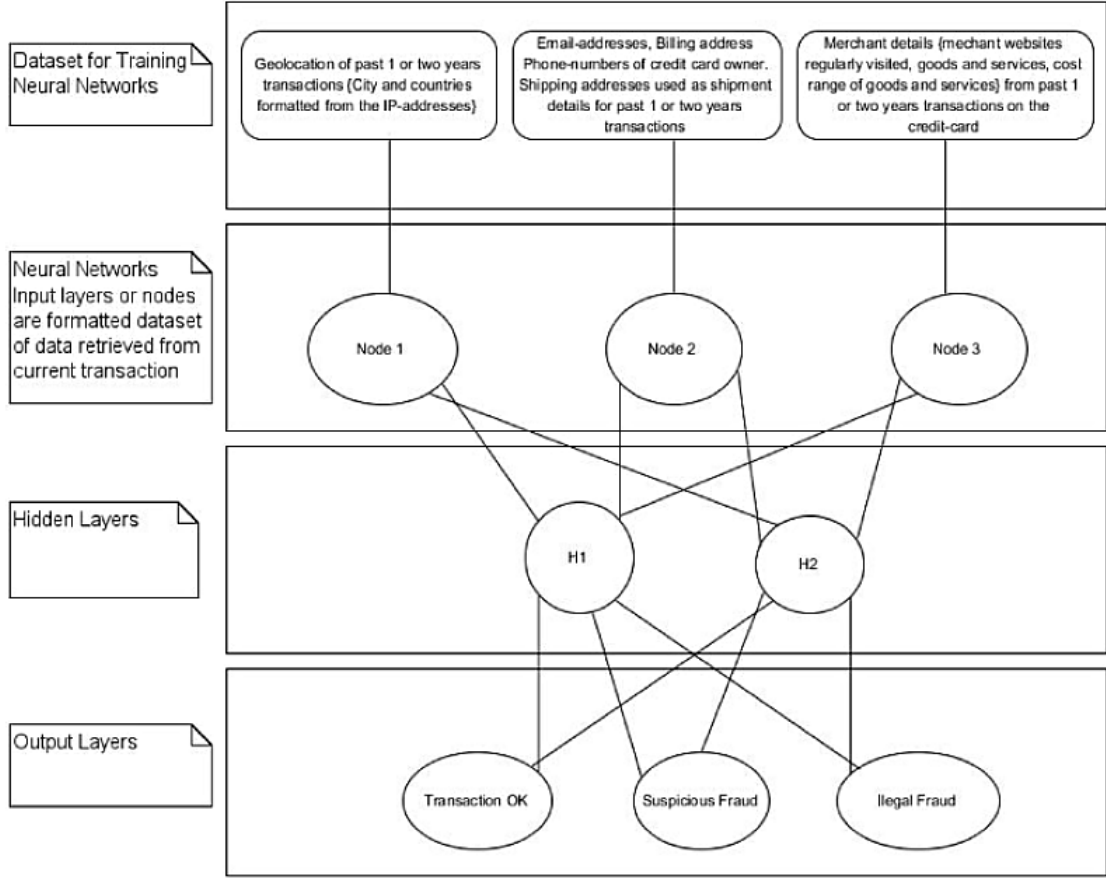


Figura 3: Rede Neural do sistema de detecção para cyber credit card fraud. Extraída de [16]

A estrutura de um classificador simples baseado em rede Bayesiana é dada como na Figura 4. Nesse caso, o nó raiz é o nó que representa a variável de classe, e cada nó que representa as variáveis de dimensão tem como único pai o nó de classe. Ou seja, $\Pi_C = \emptyset$ e $\Pi_{X_i} = \{C\}$, para todo $1 \leq i \leq d$.

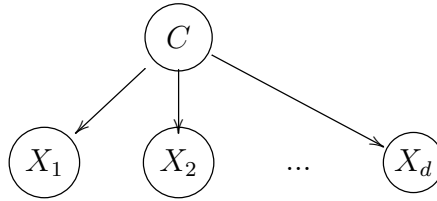


Figura 4: Exemplo da estrutura de uma rede Bayesiana simples.

Assim, na fase de treino, calcula-se a probabilidade $P(C = c_k)$ ($k \in \{1, \dots, K\}$), que é a probabilidade de uma instância pertencer à classe c_k , em que K é o número de classes do problema. Na fase de teste, ao receber uma instância (x_1, x_2, \dots, x_d) de dimensão d , calcula-se a probabilidade condicional de cada classe, dada por:

$$P(C = c_k | X_1 = x_1, \dots, X_d = x_d) = P(C = c_k) \cdot \frac{\prod_{i=1}^d P(X_i = x_i | C = c_k)}{P(X_1 = x_1, \dots, X_d = x_d)}.$$

A instância pode ser então atribuída à classe com a maior probabilidade. Se a probabilidade de cada classe for baixa, então a instância é classificada como anômala [20]. Uma outra abordagem considera a existência de apenas duas classes: a de instâncias normais e a de anomalias, e atribui a instância a ser analisada à classe de maior probabilidade [21].

Um exemplo de aplicação dessa técnica é um sistema de detecção de intrusão em redes desenvolvido por Sebyala et al. [21]. Nessa abordagem, o comportamento do sistema é analisado (considerando uso de CPU e memória, por exemplo) de forma a detectar possíveis invasões ou ataques. A variável de classe C pode assumir dois valores: verdadeiro (indicando um possível ataque ao sistema) e falso (indicando um comportamento normal). São utilizadas ainda variáveis binárias para representar atributos observados do sistema, como por exemplo, X_1 e X_2 indicando uso de CPU acima de 90% e uso de memória acima de 70%, respectivamente. Nesse modelo assume-se que os atributos observados (X_1 e X_2) são condicionalmente independentes dado o valor da classe C .

Outras aplicações podem ser destacadas em áreas como detecção de fraudes em redes de comunicação (como redes de telefonia celular) [22], detecção de atividades suspeitas através de imagens de sistemas de vigilância [23] e detecção de intrusão a redes de computadores [24].

3. Baseadas em Máquinas de Vetores Suporte (*Support Vector Machine - SVM*): Utilizada em classificação com uma classe normal. A ideia principal dessa técnica é encontrar uma fronteira de separação entre duas classes - a de instâncias normais e a de anomalias [25]. Um exemplo de uma fronteira de separação é ilustrado na Figura 5.

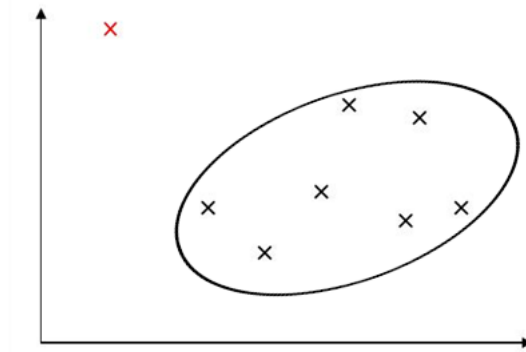


Figura 5: Exemplo de uma fronteira de separação entre instâncias normais (em preto) e uma instância anômala (em vermelho).

No estágio de treino, aprende-se a fronteira da região dos dados normais. No estágio de teste, para cada instância recebida, verifica se ela está dentro ou fora da fronteira. Se estiver dentro é classificada como normal; se estiver fora, como anômala [11].

Um exemplo do uso dessa técnica foi dado na realização de testes de motores de jatos. Por meio da observação das vibrações, Hayton et al. [26] utiliza SVM para detectar vibrações anormais. Os dados utilizados para isso são médias ponderadas das amplitudes das vibrações dos motores para dez diferentes intervalos de velocidade, obtendo assim instâncias de dimensão 10.

4. **Baseadas em Regras:** Utilizada em classificação em uma mais classes normais. A classificação das instâncias é baseada em um conjunto de regras da forma “se, então”, que definem um conjunto de condições que devem ocorrer para que uma determinada decisão seja tomada [27]. Na fase de treino, aprendem-se as regras sobre as quais os dados normais se comportam, utilizando algoritmos de aprendizagem de regras. Na fase de teste, se nenhuma regra captura a instância de teste, ela é considerada anômala. Caso contrário, é considerada normal [11].

Um exemplo do uso dessa técnica é na detecção de anomalias em tráfego de dados em redes de computadores. Mahoney et al. [28] desenvolveu um sistema que aprende regras para detectar tais anomalias. Um outro exemplo é o sistema de detecção precoce de surtos de doenças, desenvolvido por Wong et al. [29], que busca por padrões anômalos em dados provenientes de departamentos de emergência, aprendendo regras que caracterizam tais padrões.

As vantagens das técnicas baseadas em classificadores são a existência de muitos algoritmos poderosos de classificação e a rapidez do processo da fase de teste. As desvantagens são a necessidade de os dados de entrada serem rotulados e a falta de uma pontuação de anomalia associada às instâncias (apenas são classificadas como normais ou anômalas).

2.2.2 Baseadas em distância

As técnicas dessa categoria levam em consideração a distância entre uma instância do conjunto de dados a ser analisada e seus vizinhos. Dependendo do tipo dos dados de entrada, utilizam-se diferentes métodos para determinar a distância [30]. Para dados multivariados, normalmente, calcula-se a distância para cada uma das dimensões separadamente e, depois, os resultados são unidos.

Há dois principais métodos:

1. **Método da vizinhança local :** A ideia desse método é encontrar os vizinhos locais de uma determinada instância O . Se uma fração menor que p (definida previamente) de instâncias está a uma distância menor ou igual a D (definida previamente) de O , então O é considerada uma anomalia. Uma anomalia nesse contexto pode ser chamada de *DB(p , D)-Outlier* [7]. Para obter os vizinhos de uma dada instância pode-se utilizar um dos seguintes algoritmos: *nested-loop algorithm*, *index-based algorithm* ou *cell boxed algorithm* [31]. Esse método não atribui uma pontuação para as instâncias classificadas como anomalias, o que, para algumas aplicações, pode ser uma desvantagem.
2. **Método do k -ésimo vizinho mais próximo ($kNN - k$ nearest neighbors):** Nesse método, cada instância é associada a uma pontuação de anomalia, que é calculada a partir da sua distância ao seu k -ésimo vizinho mais próximo (o valor de k é definido previamente) [7; 11]. Quanto maior for a distância, maior sua pontuação de anomalia. A determinação das instâncias anômalas se obtém selecionando as n (definido previamente) instâncias que possuem maior pontuação de anomalia, ou seja, maior distância ao seu k -ésimo vizinho mais próximo. Como no método anterior, pode-se utilizar um dos seguintes algoritmos para encontrar tal distância: *nested-loop algorithm*, *index-based algorithm* ou *partition based algorithm* [32]. A ideia deste método pode ser estendida para a k -ésima região densa mais próxima.

Um exemplo de utilização desse método é detectar por meio de imagens de vídeo comportamentos humanos anômalos. Os dados utilizados por Leeuwen et al. [33] na detecção de comportamentos anômalos não possuem uma distribuição homogênea, e por isso utilizam esse método.

A vantagem desse tipo de técnica é que não é levada em consideração nenhuma característica especial dos dados de entrada. Eles não precisam possuir uma distribuição estatística (como as técnicas baseadas em modelos estatísticos descritas na Seção 2.2.5) [7; 30]. As desvantagens são o fato dessas técnicas suporem a existência de métricas de distância adequadas para estabelecer a semelhança entre dois pontos [30] e a ineficiência da maior parte dessas técnicas quando as instâncias são multivariadas e possuem uma alta dimensionalidade [7]. Porém, existem estudos que tentam desenvolver algoritmos eficientes para essas técnicas [31; 32; 34; 35].

2.2.3 Baseadas em densidade

As técnicas desta categoria estimam a densidade da vizinhança de cada instância. Uma instância que pertence a uma vizinhança pouco densa é considerada anomalia. Essas técnicas são, geralmente, mais complexas que as baseadas em distância, portanto são mais caras computacionalmente. A diferença entre as técnicas que pertencem a esta categoria são os métodos utilizados para atribuir a pontuação de anomalia às instâncias.

Os principais métodos são:

1. *LOF (Local Outlier Factor)*: Para qualquer instância de dados, a pontuação *LOF* é igual à razão entre a densidade local média dos k vizinhos mais próximos da ocorrência e a densidade local da instância em si [11; 36].
2. *COF (Connectivity-based Outlier Factor)*: A principal diferença entre *LOF* e *COF* é a forma como são calculadas as k vizinhanças mais próximas de uma dada instância. Em *COF* as vizinhanças são calculadas de modo incremental. Depois de calculada a vizinhança, a pontuação *COF* é calculada do mesmo modo que a pontuação *LOF*. A pontuação *COF* é melhor que a *LOF*, pois consegue capturar regiões que são linhas retas. Porém, é computacionalmente mais cara [11; 37].
3. *ODIN (Outlier Detection using In-degree Number)*: Para uma dada instância p , o valor *ODIN* é igual ao número dos k vizinhos mais próximos da instância que possuem p como um dos k vizinhos mais próximos. A pontuação de anomalia é o inverso do valor *ODIN* [11].
4. *INFLO (Influenced Outlierness)*: Há situações em que o *LOF* não consegue caracterizar as anomalias corretamente, como, por exemplo, quando há duas regiões com densidades distintas próximas uma da outra. Para isso o *INFLO* usa, além dos vizinhos mais próximos, os valores de vizinho mais próximo reverso, que são os pontos que possuem a instância como um dos vizinhos mais próximos [7].
5. *MDEF (Multi-granularity Deviation Factor)*: O *MDEF* de uma instância p é igual ao desvio padrão das densidades locais dos vizinhos mais próximos da instância p (inclusive a própria p). O inverso de *MDEF* é a pontuação de anomalia da instância [11].

Um módulo de detecção de intrusões a redes de computadores, denominado MINDS (*Minnesota Intrusion Detection System*), foi implementado por Etzos et al. [38] utilizando o método *LOF* para atribuir pontuações de anomalia para os dados.

As vantagens das técnicas dessa categoria são a eficácia (geralmente se mostram mais eficazes que as técnicas baseadas em distância) e a possibilidade de serem utilizadas em dados não rotulados. A desvantagem é o custo computacional mais elevado quando comparado às técnicas baseadas em distância.

2.2.4 Baseadas em *clustering*

Clustering é um método utilizado para agrupar dados similares. As técnicas que utilizam esse método podem ser divididas em três subcategorias que levam em consideração diferentes regras para determinar uma anomalia [11]:

1. Regra 1: Assume-se que instâncias normais pertencem a algum *cluster*² e instâncias anômalas a nenhum.
2. Regra 2: Assume-se que uma instância normal está próxima do centro do *cluster* mais próximo e uma instância anômala está distante do centro do *cluster* mais próximo.
3. Regra 3: Assume-se que instâncias normais pertencem a *clusters* grandes e densos, enquanto instâncias anômalas pertencem a *clusters* pequenos e esparsos.

Utilizando as regras 1 e 2, se as anomalias formarem um *cluster*, elas não serão detectadas.

Um algoritmo de *cluster* basicamente agrupa as instâncias em K grupos disjuntos, em que K é um inteiro positivo. O algoritmo em geral segue quatro passos:

1. Define o número K de *clusters*
2. Inicializa os centros dos K *clusters*. Isso pode ser feito dividindo arbitrariamente todas as instâncias de treino entre os K *clusters*, calculando seus centros e verificando que são todos distintos entre si. Uma outra alternativa seria escolher arbitrariamente K instâncias de treino distintas e inicializá-las como os centros dos *clusters*.
3. Para cada instância de treino, calcula sua distância ao centro de cada *cluster*, e atribui a instância ao *cluster* cujo centro é o mais próximo.
4. Recalcula os centros de cada *cluster* modificado.
5. Repete o passo 3 até que os centros não mudem mais.

Para o cálculo da distância entre as instâncias e os centros pode-se utilizar diferentes métodos, como por exemplo a função de distância Euclidiana, dada pela fórmula:

$$F(x, y) = \sqrt{\sum_{i=1}^d (x_i - y_i)^2},$$

em que d é a dimensão de cada instância e $x = (x_1, \dots, x_d)$ e $y = (y_1, \dots, y_d)$ são dois vetores de dimensão d .

²Neste trabalho, o termo *cluster* não será traduzido para “grupo” por ser um termo reconhecido na área na sua forma em inglês.

A grande diferença entre as técnicas baseadas em *clusters* e as baseadas em distância (vizinhos mais próximos) é que nas baseadas em *clusters* a instância é analisada em relação à sua distância aos *clusters*, enquanto as baseadas em distância são analisadas em relação à sua vizinhança local.

Um exemplo de aplicação das técnicas baseadas em *clustering* é na detecção de fraude de cartão de crédito. Bolton et al. [39] idealizou um mecanismo de detecção de fraude em cartão de crédito que utiliza detecção de anomalias baseada em *clusters*. A diferença principal entre essa técnica e a técnica utilizada no sistema baseado em redes neurais de John Akhilomen [16] descrita na Seção 2.2.1 é a sua propriedade de ser não supervisionada.

Esse mecanismo consiste em criar *clusters* de transações similares, em que o perfil de uma conta seria determinado pelos *clusters* que possuem transações dessa conta. Uma transação anômala de uma conta seria uma transação que não pertenceria ao perfil da conta, ou seja, não pertenceria a nenhum dos *clusters* que possuem transações da conta. O grande problema dessa técnica seria a escolha das variáveis e métricas de distância utilizadas.

Um outro exemplo do uso de técnicas baseadas em *clustering* é na detecção de invasões em redes. Leung et al. [40] utiliza detecção de anomalias baseadas em *clustering* para detectar invasões em redes por meio de dados não rotulados.

As vantagens dessas técnicas são a possibilidade de serem utilizadas com dados não rotulados, a existência de muitos algoritmos de *clustering* e a rapidez do processo da fase teste. A desvantagem é a dependência dos algoritmos de *clustering* que não são adaptados para detectar anomalias, mas sim para formar *clusters* [7; 11].

2.2.5 Baseadas em modelos estatísticos

As técnicas dessa categoria se baseiam no fato de que existe uma distribuição estatística que modela o conjunto de dados. As instâncias anômalas são as que não são modeladas por essa distribuição.

Na fase de treino um modelo estatístico é generalizado para o conjunto de dados e na fase de teste, se uma instância não corresponde ao modelo, ela é considerada como anômala.

Existem duas subcategorias de métodos [7; 11]:

1. Métodos paramétricos: Assumem explicitamente um modelo estatístico para o conjunto de dados.
 - a. Baseados no Modelo Gaussiano: Assume-se que os dados são gerados por meio de uma distribuição gaussiana. Seja x uma instância desse conjunto então:

$$x \sim \mathcal{N}(\mu, \sigma^2)$$

E a função de probabilidade de x é dada por:

$$p(x, \mu, \sigma^2) = \frac{1}{\sigma(\sqrt{2\pi})} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

em que μ é a média e σ^2 é a variância [12].

A fase de treino normalmente utiliza *MLE* (*Maximum Likelihood Estimates*) para determinar a média e o desvio padrão da distribuição gaussiana. Na fase de teste utiliza-se uma métrica para determinar as instâncias anômalas. Dentre elas, podemos destacar:

- i. F1-Score: Sejam os seguintes conceitos:

- Verdadeiros positivos (VP): número de instâncias originalmente anômalas classificadas corretamente pelo algoritmo como anomalias.
- Falsos positivos (FP): número de instâncias originalmente normais classificadas incorretamente pelo algoritmo como anomalias.
- Falsos Negativos (FN): número de instâncias originalmente anômalas classificadas incorretamente pelo algoritmo como normais.
- Verdadeiros negativos (VN): número de instâncias originalmente normais classificadas corretamente pelo algoritmo como normais.

A métrica F1-Score é definida como:

$$F1-Score = \frac{2 * P * C}{P + C},$$

em que

$$P = \frac{VP}{VP + FP} \text{ e } C = \frac{VP}{VP + FN}.$$

Uma instância x é definida como normal se $p(x) \geq \varepsilon$ e anômala caso contrário. Utiliza-se a métrica F1-Score para se estimar o melhor valor de ε [12].

- Teste da média-variância: Se uma instância está a uma distância maior que três vezes o desvio padrão, então é considerada anômala.
- Teste do *box-plot*: Através do *box-plot*, calcula-se o *IQR* (*Inter quartile range*), o quartil 1 (Q1) e o quartil 3 (Q3). Se uma instância estiver a uma distância maior que 1.5*IQR de Q1 ou a uma distância maior que 1.5*IQR de Q3, é considerada anômala [7; 30].
- Teste Grubb: Utilizado para detectar anomalias em conjuntos de dados univariados, é obtido pela fórmula:

$$G = \frac{|x - \mu|}{\sigma^2},$$

em que x é uma instância do conjunto, μ é a média, σ^2 é o desvio padrão [11]. Uma instância é dita anômala ou normal, de acordo com o valor obtido G .

- Qui-quadrado: Para instâncias multivariadas pode-se utilizar a média de qui-quadrado dada pela fórmula:

$$\chi^2 = \sum_{i=1}^N \frac{(x_i - \mu_i)^2}{\mu_i}, i \in \{1, 2, \dots, N\},$$

em que μ_i é o valor esperado da i -ésima característica da instância x e x_i é a i -ésima característica da instância x [11].

- Modelo de Regressão [11]:

- Encontrar um modelo de regressão que se encaixe no conjunto de dados.
- Para cada instância de teste, o residual (parte da instância que não é explicada pelo modelo de regressão) é usado para determinar a pontuação de anomalia da instância.

- Mistura de distribuições paramétricas estatísticas [11]:

- i. Utiliza uma distribuição para modelar as instâncias normais e outra para modelar as instâncias anômalas.
 - ii. Utiliza uma mistura de distribuições paramétricas para modelar as instâncias normais. Se nenhuma distribuição modelar uma dada instância, então esta será considerada anômala.
2. Métodos não-paramétricos: Usam um modelo estatístico não paramétrico, tal que a estrutura da distribuição dos dados não é definida previamente, mas obtida a partir do conjunto de dados.
 - a. Baseadas em Histogramas [7; 11]:
 - i. Instâncias univariadas: Obtém-se o histograma a partir do conjunto de dados de treino. Na fase de teste é verificado se a instância pertence a alguma das colunas do histograma. Se pertencer é considerada normal, senão anômala. Uma outra alternativa na fase de teste seria atribuir como pontuação de anomalia a altura da coluna a qual a instância pertence.
 - ii. Instâncias multivariadas: Constrói-se um histograma para cada característica das instâncias.
 - b. Baseadas em função kernel: Utiliza-se uma função kernel para obter uma aproximação da função de densidade da distribuição dos dados. Uma instância de teste é declarada como anômala se obtém um valor baixo para a função de densidade da distribuição. Pode ser usada para instâncias univariadas ou multivariadas [7; 11].

Um exemplo do uso dessa técnica é na detecção de ataques na *web*. Kruegel et al. [41] utiliza diferentes técnicas baseadas em modelos estatísticos para detecção de anomalias que poderiam ser determinadas como ataques na *web*.

As vantagens dessas técnicas são a existência de uma justificativa matemática para a classificação da instância entre anomalia ou normal, a possibilidade de serem usadas em dados não rotulados e a eficiência quando se obtém uma boa função de densidade da distribuição dos dados a serem analisados. As desvantagens são a necessidade dos dados seguirem uma distribuição estatística, o aumento da dificuldade de encontrar uma distribuição que modele o conjunto de dados quanto maior for a dimensionalidade, além de possuírem uma performance limitada quando há poucas instâncias de treino [7; 11; 30].

2.3 Validador estatístico de lotes de dados

Um dos objetivos deste trabalho consiste em aplicar uma das técnicas de detecção de anomalias em um problema real. Para isso vamos utilizar um sistema de validação estatístico de lotes de dados.

A arquitetura de um *data warehouse* pode ser utilizada para integrar dados. Isto pode ser feito através de um sistema integrador. Esse sistema pode extrair os dados dos bancos de dados envolvidos de forma ativa ou passiva. A forma mais utilizada é a extração passiva, pelo fato de que na forma ativa o sistema integrador precisa ter acesso a vários bancos de dados de instituições distintas, o que nem sempre é possível, enquanto a extração passiva é feita através do recebimento de dos dados enviados pelos bancos de dados envolvidos no sistema. A extração de dados passiva exige que seja estabelecido um formato de como os dados devem ser enviados para o sistema integrador. Nesse sistema, uma estrutura de lote de dados é utilizada para o recebimento dos dados, que consiste

em um conjunto de arquivos em que cada linha do arquivo corresponde a um registro da tabela de um banco de dados relacional. Tal conjunto de arquivos corresponde a dados extraídos de um sistema transacional referentes a um período de funcionamento desse sistema.

Com a extração passiva dos dados, perde-se a confiabilidade na correteza dos dados recebidos. Portanto o sistema integrador precisa realizar uma validação dos lotes recebidos. O processo de validação é constituído de regras sintáticas e semânticas que têm por objetivo verificar a correteza dos dados recebidos. Porém, essas regras não conseguem verificar alguns erros, como por exemplo: Seja l um lote que está com o formato correto, mas uma análise global dos dados mostra que as proporções dos atributos nos lotes não estão corretas. Essa verificação não consegue ser feita pelas regras sintáticas e semânticas.

Para resolver tal problema, foi desenvolvido um sistema validador estatístico dos lotes de dados [8]. Esse validador utiliza medidas estatísticas dos lotes de dados para verificar a validade destes.

O método de validação estatística dos lotes de dados consiste em decidir se um determinado lote é válido ou inválido, baseando-se em lotes já classificados. O problema de classificação possui três características importantes: a quantidade de lotes inválidos é pequena, os lotes inválidos são importantes para a detecção de onde os erros estão sendo produzidos e os lotes inválidos não possuem uma característica comum. Por esses três fatores o método de validação estatística caracteriza-se como um problema de detecção de anomalias.

As medidas estatísticas extraídas dos lotes utilizadas pelo validador são as proporções dos valores dos atributos de interesse do lote. Para obter tais proporções os atributos precisam possuir categorias, portanto é preciso dividir os atributos numéricos em intervalos de forma que cada intervalo consista em uma categoria. Assim para cada atributo de interesse, ou seja, que se quer validar, obtemos as proporções de cada categoria. As Tabelas 1 e 2 exemplificam as proporções de um atributo categórico e de um atributo numérico em um lote de dados, respectivamente.

Fator Rh sanguíneo	Proporção no lote
Positivo	0.46
Negativo	0.45
Indeterminado	0.09

Tabela 1: *Proporções em um lote exemplo das categorias do atributo fator Rh sanguíneo. Extraída de [8]*

Faixa etária	Proporção no lote
idade < 18	0.00
$18 \leq \text{idade} < 28$	0.25
$28 \leq \text{idade} < 38$	0.22
$38 \leq \text{idade} < 48$	0.23
$48 \leq \text{idade} < 58$	0.19
$58 \leq \text{idade} < 68$	0.10
$68 \leq \text{idade}$	0.01

Tabela 2: *Proporções em um lote exemplo do atributo numérico faixa etária já categorizada. Extraída de [8]*

Assim podemos modelar a validação dos lotes como detecção de anomalias, em que cada lote consiste em vetores de características cujas dimensões são as proporções encontradas nos lotes, em que cada vetor representa um atributo de interesse. Por exemplo, considere um lote que contém os seguintes atributos e categorias:

- Tipo sanguíneo: A, B, AB, O, desconhecido
- Gênero: M (masculino), F (feminino), outros
- Estado civil: casado, solteiro, divorciado, viúvo, outros

Então, os vetores x, y, z de características que representam esse lote podem ser descritos como:

$$\begin{aligned} x &= (x_1, x_2, x_3, x_4, x_5) \\ y &= (y_1, y_2, y_3) \\ z &= (z_1, z_2, z_3, z_4, z_5) \end{aligned}$$

Neste caso, o vetor x contém proporções das categorias do atributo tipo sanguíneo, y contém proporções das categorias do atributo gênero e z contém as proporções das categorias do atributo estado civil.

Cada um desses vetores é uma instância em um detector de anomalias, ou seja, para validar esse lote seriam necessárias três instâncias de detector de anomalias. Essa divisão é feita porque a dimensão do vetor seria muito grande se fosse utilizado um único vetor de características para representar um lote, podendo causar problemas na detecção de anomalias, uma vez que quanto maior a dimensão, mais lotes de treinamento seriam necessários. Além disso, não seria possível descobrir qual ou quais atributos são responsáveis pelo comportamento anômalo.

Nesse contexto, chamamos de *validador global* o validador que valida o lote como um todo e de *validador individual* o validador que valida cada atributo do lote.

Os validadores individuais não precisam receber o lote inteiro. Eles podem receber apenas dados referentes ao atributo que irão validar, o que é chamado de lote reduzido.

Assim, podemos definir o processo de validação V de um lote l com n atributos de interesse do seguinte modo:

Sejam v_1, v_2, \dots, v_n os validadores individuais e sejam l_1, l_2, \dots, l_n os respectivos lotes reduzidos. Então:

$$V(l) = v_1(l_1) \wedge v_2(l_2) \wedge \dots \wedge v_n(l_n).$$

Ou seja, V será válido para o lote l se todos os lotes reduzidos de l forem válidos. Caso contrário, V será inválido para l .

Como as dimensões de cada instância representam proporções das categorias de atributos de um lote, percebe-se que uma das dimensões não é necessária, uma vez que a soma das proporções de uma instância resulta em 1. Dessa forma, uma dimensão pode ser escrita em função das demais, e assim o validador que recebe uma instância d -dimensional para validar pode escolher $d-1$ dimensões para classificá-la.

Um validador pode também levar em consideração dependências simples entre atributos, em que as proporções das categorias de um atributo dependem de um contexto que é determinado pelo valor de outros atributos. Por exemplo, lotes provenientes de diferentes hemocentros podem possuir uma variação do perfil dos doadores em relação ao sexo, de forma que em São Paulo exista um equilíbrio maior na quantidade de homens e mulheres entre os doadores, enquanto que em Pernambuco há uma predominância de homens. Desse modo, percebe-se uma certa dependência entre os atributos *sexo* e *hemocentro*. Chamamos de atributo comportamental aquele que desejamos validar, e de atributos contextuais aqueles que fornecem um contexto para a validação. Os atributos contextuais devem ser categorizados, assim como os comportamentais.

Esse sistema de validação foi idealizado por Pedro Takecian em sua tese de doutorado [8].

Por lidar com técnicas de detecção de anomalias, o sistema de validação estatístico será utilizado neste trabalho para que possamos aplicar uma das técnicas estudadas e analisar seu funcionamento no conjunto de dados do sistema transacional.

3 Detecção de anomalias baseada em distância

Nesta seção explicaremos a escolha da técnica de detecção de anomalias baseada em distância a ser implementada como um módulo do sistema validador descrito na Seção 2.3. Faremos também um estudo mais detalhado da técnica escolhida, apresentando seus principais algoritmos e um estudo sobre a escolha da função de distância a ser utilizada. Os principais algoritmos das técnicas baseadas em distâncias, que serão apresentados nesta seção, são utilizados para encontrar em um conjunto de dados as instâncias que são anomalias. Como o problema a ser revolido neste trabalho é verificar se um lote é normal ou anômalo quando analisado em um conjunto de dados, na Seção 4 serão discutidas a implementação do módulo e a adaptação dos algoritmos para a resolução desse problema.

O módulo desenvolvido para o detector de anomalias do sistema é baseado no modelo de detecção gaussiana (uma técnica de detecção de anomalia baseada em modelos gaussianos que utiliza como métrica, F1-score, essa técnica foi discutida na seção 2.2.5) que modela adequadamente muitas situações, mas em alguns casos algum outro método é necessário para representar melhor o problema. A Figura 6 exemplifica um caso em que o modelo gaussiano não representa as instâncias apropriadamente, em que se desconhece a razão das instâncias se agruparem em dois conjuntos distintos.

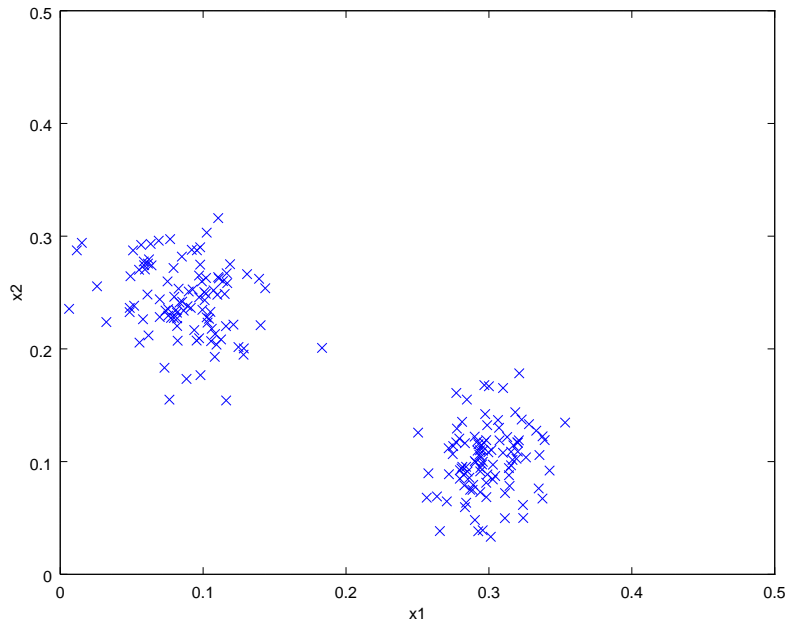


Figura 6: Exemplo de instâncias que não são adequadamente representadas pelo modelo gaussiano. Extraída de [8]

Uma técnica baseada em distância se adequaria a essa situação. No contexto das técnicas baseadas em distância, não é feita nenhuma hipótese sobre os dados, e elas generalizam muitos conceitos de métodos que assumem que os dados seguem uma certa distribuição (como métodos estatísticos) [7]. Além disso, essas técnicas possibilitam tratar instâncias cujas dimensões são valores contínuos e com dependência entre atributos.

Existem dois principais tipos de técnicas baseadas em distância: o método da vizinhança local e o método do k -ésimo vizinho mais próximo (apresentado normalmente como kNN). A principal diferença entre esses dois métodos está na definição de anomalia.

Uma anomalia no contexto da vizinhança local pode ser chamada de *DB-outlier* (*distance based outlier*) [31], e é definida como:

“Uma instância O em um conjunto T de dados é um $DB(p, D)$ -outlier se pelo menos uma fração p ($0 < p < 1$) de instâncias em T se encontra a uma distância maior que D de O .”

Uma anomalia no contexto do k -ésimo vizinho mais próximo pode ser chamada de D_n^k -outliers [32], e é definida como:

“Dado um conjunto de entrada com N instâncias e parâmetros n e k , uma instância p é um D_n^k -outlier se não mais que $n - 1$ outras instâncias p' possuem $D^k(p') > D^k(p)$.”

Nessa definição, n representa a quantidade de anomalias a serem encontradas e $D^k(p)$ a distância entre p e seu k -ésimo vizinho mais próximo. Além disso, $D^k(p)$ representa uma pontuação de anomalia, ou seja, quanto maior o $D^k(p)$, maior o grau de anomalia de p .

A principal diferença entre as definições de anomalia dos dois métodos é a necessidade de definir o parâmetro adicional D para detectar os DB -outliers, que é muitas vezes difícil de ser encontrado [32; 35]. Porém para obter os D_n^k -outliers é preciso definir previamente a quantidade de anomalias que se deseja encontrar. Além disso, pela definição de DB -outlier não se atribui às anomalias uma pontuação.

Como no sistema de validação deve-se decidir se um determinado lote recebido é válido ou inválido, o parâmetro n do método do k -ésimo vizinho mais próximo não será utilizado, uma vez que não precisaremos encontrar n anomalias no conjunto de dados, e sim decidir se uma dada instância é anômala ou não levando em consideração os lotes de treinamento.

Sendo assim, optamos pelo método do k -ésimo vizinho mais próximo para estudo e implementação do módulo do sistema de validação.

3.1 Método do k -ésimo vizinho mais próximo (kNN)

Considere a definição de D_n^k -outliers descrita anteriormente. Descreveremos alguns dos principais algoritmos utilizados para encontrá-los.

Os algoritmos descritos nas seções 3.1.1, 3.1.2 e 3.1.3 foram desenvolvidos por Ramaswamy et al. [32], e, para sua delimitação, algumas definições são necessárias.

Seja $MINDIST(p, R)$ a menor distância entre uma instância p e um retângulo R , de forma que cada ponto em R esteja a uma distância de pelo menos $MINDIST(p, R)$ de p , e seja $MAXDIST(p, R)$ a maior distância entre uma instância p e um retângulo R , de forma que qualquer ponto em R esteja a uma distância de no máximo $MAXDIST(p, R)$ de p . De forma similar definimos $MINDIST(R, S)$ e $MAXDIST(R, S)$ como a menor e a maior distância entre dois retângulos de limite mínimos R e S , respectivamente.

Considere também a aproximação de um conjunto de instâncias utilizando seu retângulo de limite mínimo (*minimum bounding rectangle*), definido por um retângulo R d -dimensional com vértices $r = \{r_1, r_2, \dots, r_d\}$ e $r' = \{r'_1, r'_2, \dots, r'_d\}$ formando sua diagonal principal, em que d é a dimensão das instâncias do conjunto, e tais que $r_i \leq r'_i$ para todo $1 \leq i \leq d$.

3.1.1 Algoritmo *simple nested-loop*

O algoritmo consiste em calcular $D^k(p)$ para cada instância p do conjunto de entrada e retorna as n instâncias com maior valor de D^k . Para isso mantém, para cada instância p

do conjunto de entrada, uma lista com suas distâncias aos seus k vizinhos mais próximos. Assim, para cada instância $q \neq p$ do conjunto de entrada, se a distância entre p e q for menor que alguma distância na lista, então ela é adicionada na lista. Se a lista está completa (se já contém k elementos) então a maior distância da lista é removida e a nova distância é inserida. Ao final, a maior distância na lista de vizinhos de p é o valor de $D^k(p)$.

Esse algoritmo possui complexidade computacional $O(N^2)$, onde N é o tamanho do conjunto de entrada [32; 35].

3.1.2 Algoritmo *index-based*

Uma forma de diminuir a complexidade do algoritmo *simple nested-loop* é utilizar uma estrutura de índice multidimensional, como por exemplo uma árvore R^* , que é uma variação de uma árvore R [42].

Uma árvore R é uma árvore balanceada por altura similar a uma árvore B , e é estruturada de forma a agrupar objetos próximos, representando-os por retângulos de limite mínimos, como ilustrado na Figura 7. Cada nó que não é folha da árvore armazena dois tipos de informação: uma maneira de identificar o nó filho e o retângulo de limite de todas as entradas desse nó filho. Cada nó folha, por sua vez, armazena um conjunto de instâncias e seu retângulo de limite mínimo [43].

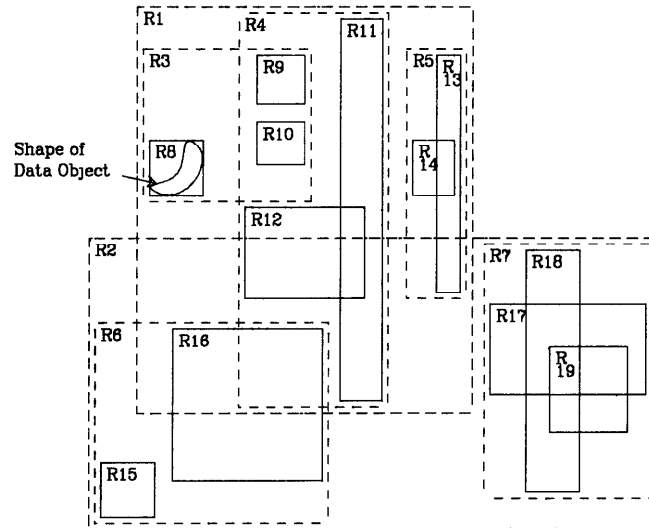
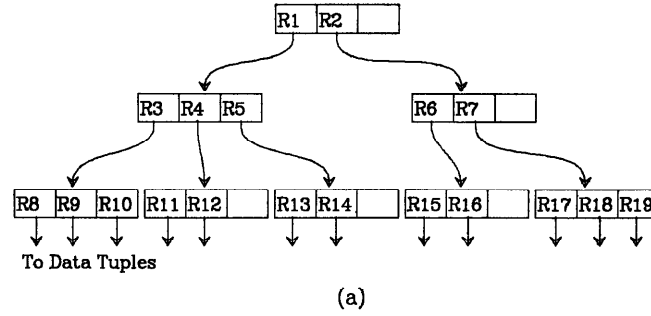


Figura 7: Exemplo simples de uma árvore R bidimensional. Extraída de [43].

A principal diferença entre as árvores R^* e as árvores R é que as árvores R^* otimizam a cobertura e a sobreposição de cada retângulo de limite mínimo [42].

Utilizando essa estrutura de dados, uma otimização pode ser feita de forma a diminuir o número de cálculos de distância. Suponha que calculamos $D^k(p)$ para uma instância p considerando apenas um subconjunto dos dados de entrada. Esse valor será um limitante superior para o valor real de $D^k(p)$, uma vez que se a menor distância entre p e o retângulo de limite mínimo de um nó na árvore R^* for maior que o valor de $D^k(p)$ encontrado até o momento, então nenhuma das instâncias na sub-árvore definida por esse nó estarão entre os k vizinhos mais próximos de p . Assim, há uma redução da quantidade de computações das distâncias.

Uma outra otimização pode ser feita considerando que procuramos pelos n melhores candidatos a anomalias. Suponha que os n melhores candidatos sejam armazenados durante cada passo do algoritmo. Seja D_{nmin} o mínimo dentre os valores D^k dessas n instâncias. Se o valor $D^k(p)$ de algum p calculado for menor que D_{nmin} , então p não pode ser uma anomalia, uma vez que D_{nmin} é o menor valor de $D^k(o)$, com o sendo uma anomalia.

O cálculo de $D^k(p)$ de uma instância p pode ser feito utilizando uma lista ligada de nós, que é inicializada com a raiz da árvore R^* , e um *heap* que armazena os k vizinhos mais próximos de p encontrados, inicializado como vazio.

Ao longo do algoritmo, a lista é ordenada de forma crescente em relação a $MINDIST(p, Nó)$, em que $Nó$ é um elemento da lista. Essa distância é calculada utilizando o retângulo de limite mínimo de cada elemento $Nó$ da lista. Já o *heap* armazena os vizinhos de p em ordem decrescente em relação às suas distâncias a p .

Assim, $D^k(p)$ é inicializado com infinito e cada nó da lista é examinado. Se o nó não corresponde a uma folha na árvore R^* , ele é removido da lista, seus filhos são inseridos nela e a lista é reordenada. Além disso, é verificado se $D^k(p)$ é menor ou igual a $MINDIST(p, nó)$ para cada nó da lista. Se for, então nenhuma das instâncias do nó fará parte dos k vizinhos mais próximos de p . Logo o nó é deletado da lista.

Se o nó corresponde a uma folha, então para cada instância q armazenada por esse nó, se a distância entre q e p for menor que $D^k(p)$, q é inserido no *heap* de vizinhos de p . Se o número de elementos no *heap* for maior que k , então o elemento do topo do *heap* é deletado (o elemento cuja distância a p é a maior dentre os elementos do *heap*). Se o número de elementos for igual a k , então $D^k(p)$ recebe o valor da distância entre p e t , onde t é o elemento do topo do *heap*. Além disso, se a qualquer momento $D^k(p)$ for menor que D_{nmin} , então p não pode ser uma anomalia, como discutido anteriormente, e assim o método termina.

Esse algoritmo é mais eficiente do que o algoritmo *simple nested-loop*, mas no pior caso ainda é de ordem quadrática em relação ao tamanho do conjunto de entrada.

3.1.3 Algoritmo *partition-based*

O problema do algoritmo *index-based* é o seu alto custo computacional. Como tipicamente o valor de n é pequeno, muitos cálculos de distâncias não são necessários, levando a um desperdício de computação.

O algoritmo *partition-based* visa eliminar instâncias cujas distâncias aos seus k -ésimos vizinhos mais próximos são muito pequenas, e que portanto são fracas candidatas a anomalias. A principal ideia do algoritmo é particionar o conjunto de entrada dos dados, e dessa forma eliminar cada partição assim que se descobre que ela não contém anomalias.

Para isso utiliza-se um algoritmo de *clustering* para agrupar os dados, de forma que cada *cluster* formado seja uma partição. As partições podem ser armazenadas em uma

estrutura de índice, sendo representadas pelos seus retângulos de limite mínimos. A partir disso, para cada partição P , o limitante superior $P.sup$ e o inferior $P.inf$ dos valores de D^k das suas instâncias são calculados, de modo que para cada $p \in P$, $D^k(p) \geq P.inf$ e $D^k(p) \leq P.sup$.

Com esses valores, podemos identificar partições que contém instâncias que são candidatas a anomalias. Suponhamos que temos o valor D_{nmin} como descrito na seção 3.1.2, que é o menor valor de D^k dos n melhores candidatos a anomalias. Então se $P.sup < D_{nmin}$ para alguma partição P , então nenhuma de suas instâncias pode ser anômala. Apenas partições P com $P.sup \geq D_{nmin}$ são partições com instâncias candidatas a anomalias. O cálculo de D_{nmin} pode ser feito mantendo as partições em ordem decrescente de seus valores de limitante inferior. Sejam $P_1, P_2, \dots, P_l, l \in \mathbb{N}$, partições com os maiores valores de limitante inferior tais que o número de instâncias em cada partição seja no mínimo n . Então $D_{nmin} = \min\{P_i.inf : 1 \leq i \leq l\}$.

As instâncias que devem ser analisadas para o cálculo de $D^k(p)$ para cada $p \in P$, sendo P uma partição, são aquelas que estão em partições cuja distância a P é de no máximo $P.sup$. Sendo assim, seja $P.vizinhos$ o conjunto de tais partições. Desse modo, as partições P com $P.sup$ maior ou igual a D_{nmin} e as partições em $P.vizinhos$ podem ser armazenadas em uma estrutura de índice multidimensional, como uma árvore R^* , e assim, o algoritmo *index-based* pode ser utilizado para computar as n melhores anomalias dentre as instâncias nessas partições. Como em geral o número de instâncias candidatas a anomalias é pequeno, o algoritmo terá uma performance melhor do que quando executado sobre todas as instâncias do conjunto de entrada.

O cálculo de $P.sup$ e $P.inf$ de uma partição P pode ser feito utilizando dois *heaps* que armazenam em ordem decrescente cada partição Q em relação a $MINDIST(P, Q)$ e a $MAXDIST(P, Q)$ e uma lista ligada de nós análoga àquela utilizada no algoritmo descrito na seção 3.1.2. Desse modo, a lista é inicializada com a raiz da árvore R^* , os valores de $P.sup$ e $P.inf$ são inicializados com infinito, e os *heaps* são inicializados como vazios.

Assim, para cada nó da lista, verifica-se se corresponde a um nó folha na árvore R^* . Se sim, então para cada partição Q presente no nó, se $MINDIST(P, Q)$ for menor que $P.inf$, então Q é inserido em um dos *heaps* e, dessa forma, $P.inf$ é ajustado com o valor de $MINDIST(P, R)$, em que R é a partição no topo do *heap*. Analogamente, se $MAXDIST(P, Q)$ for menor do que $P.sup$, então Q é inserido no outro *heap* e, dessa forma, $P.sup$ é ajustado com o valor de $MAXDIST(P, R)$, em que R é a partição no topo do *heap*. Se o nó não corresponde a uma folha na árvore R^* , ele é removido da lista, seus filhos são inseridos nela e a lista é reordenada. Além disso, a cada nó examinado, é verificado se, para cada elemento $Nó$ na lista, $P.sup \leq MAXDIST(P, Nó)$ e $P.inf \leq MINDIST(P, Nó)$. Se sim, então o nó é deletado da lista.

3.1.4 Algoritmo *nested-loop ANNS*

O algoritmo mais simples para encontrar os D_n^k -outliers é o algoritmo *simple nested-loop* [35]. O algoritmo *index-based* e o algoritmo *partition-based* são algoritmos mais eficientes que o *simple nested-loop*, porém sua boa performance acontece apenas em conjuntos de dados com poucas dimensões.

Um algoritmo simples baseado no *simple nested-loop* foi proposto por Bay et al. [34] que utiliza uma regra simples de *pruning*³ e a aleatoriedade dos dados do conjunto de

³O termo *pruning* não será traduzido para “poda” por ser um termo mais utilizado na área na sua

entrada. Este algoritmo para a maioria dos testes realizados por Bay et al. obteve complexidade linear no tamanho do conjunto de entrada. No pior caso, o algoritmo continua sendo de ordem quadrática.

A regra de *pruning* utilizada é chamada de *Approximate Nearest Neighbour Search – ANNS*. A principal ideia por trás dessa regra é encontrar as n melhores anomalias. Para isso, salva-se um valor c que representa a menor distância entre uma instância e seu k -ésimo vizinho mais próximo encontrado até um certo momento pelo algoritmo. Durante o processo de encontrar $D^k(p)$ para alguma instância p , como $D^k(p)$ tende a diminuir quanto mais vizinhos são processados, se obtivermos um valor de distância entre p e algum vizinho menor que o valor de c , então podemos descartar essa instância p , pois ela não será uma anomalia. Como o valor de c ao longo da execução aumenta, a eficiência do *pruning* aumenta também [34; 35].

O algoritmo pode ser descrito da seguinte forma (Algoritmo 1) [34; 35]:

Algoritmo 1 Algoritmo *nested-loop ANNS*

entrada: k (número de vizinhos mais próximos a ser considerado), n (número de anomalias a serem encontradas), conjunto D de dados de entrada

saída: O (conjunto de anomalias encontradas)

```

1: Inicialize  $c = 0$ 
2: Inicialize  $O$  como vazio
3: para cada  $b$  em  $D$  faça
4:   Inicialize  $Vizinhos(b)$  como vazio
5:   para cada  $d$  em  $D$ ,  $d \neq b$  faça
6:     se a quantidade de elementos em  $Vizinhos(b)$  for menor que  $k$  ou a distância
       entre  $b$  e  $d$  for menor que a distância entre  $b$  e os elementos de  $Vizinhos(b)$ 
       então
7:       Atualize os valores de  $Vizinhos(b)$  com as  $k$  instâncias do conjunto  $Vizinhos(b)$ 
          $\cup d$  que estão mais próximas de  $b$ 
8:       se  $D^k(b) < c$  então
9:         break
10:      fim se
11:    fim se
12:  fim para
13:  Atualize o conjunto  $O$  obtendo as  $n$  melhores anomalias entre as que estão no
    conjunto  $O \cup b$ 
14:  Atualize  $c$  com o valor da menor distância para os  $k$ -ésimos vizinhos mais próximos
    dos elementos que estão em  $O$ 
15: fim para

```

3.2 Métricas de distância

Um importante fator a ser considerado na utilização de uma técnica de detecção de anomalias baseada em distância é a escolha da função de cálculo da distância utilizada.

As métricas de distância mais utilizadas são as L_k norm⁴. Elas são definidas por:

forma em inglês.

⁴Não foi traduzido por não encontrar uma referência para esse conjunto de métricas em português

$$x, y \text{ pertencem a um conjunto de dimensão } d \text{ e } L_k(x, y) = \left(\sum_{i=1}^d |x_i - y_i|^k \right)^{1/k}$$

A métrica L_1 é a distância de Manhattan e a métrica L_2 é a distância Euclidiana.

Muitas técnicas baseadas em distância utilizam como métrica a distância Euclidiana [32; 33; 34]. A distância Euclidiana funciona corretamente e descreve de forma intuitiva a distância para espaços de 2 ou 3 dimensões. Porém em espaços com alta dimensão pode ser difícil a escolha de um métrica de distância.

Segundo os estudos do artigo “On the Surprising Behavior of Distance Metrics in High Dimensional Space” [44] a eficácia da L_k norm em espaços com dimensão alta diminui conforme k aumenta. Portanto para espaços com alta dimensão a distância de Manhattan é melhor que a distância Euclidiana. Isso acontece pois:

Seja $Dmax_d^k$ a distância máxima entre um ponto específico (por exemplo a origem) e os N pontos de um conjunto de dimensão d . Seja $Dmin_d^k$ a distância mínima entre um ponto específico (por exemplo a origem) e os N pontos de um conjunto de dimensão d . Temos:

- $Dmax_d^1 - Dmin_d^1$ diverge para o infinito quando d aumenta.
- $Dmax_d^2 - Dmin_d^2$ converge para um valor positivo maior que zero quando d aumenta.
- $Dmax_d^k - Dmin_d^k$ ($k > 2$) converge para zero quando d aumenta.

Ou seja, quanto maior o k , menor é a diferença entre a maior distância a um certo ponto e a menor distância para esse mesmo ponto, perdendo assim a ideia de “perto” e “longe”.

Uma outra métrica que pode ser utilizada é a distância de Mahalanobis, que leva em consideração a correlação entre os dados no cálculo da distância e é invariante à escala das variáveis. Sua função é definida por:

$$F(x, y) = \sqrt{(x - y)' \Sigma^{-1} (x - y)},$$

em que $x = (x_1, x_2, \dots, x_d)$ e $y = (y_1, y_2, \dots, y_d)$ são dois vetores de dimensão d e Σ é a matriz de covariância dos dados. Se essa matriz for a identidade, então a distância de Mahalanobis é equivalente à distância Euclidiana [45].

Essa métrica apresenta a desvantagem do alto custo de se calcular a matriz de covariância [46].

4 Implementação do módulo de detecção de anomalias

Nesta seção detalharemos a implementação do método do k -ésimo vizinho mais próximo (kNN) como um módulo de detecção de anomalias para o sistema de validação descrito na Seção 2.3. Utilizamos a distância de Manhattan como métrica para calcular a distância entre duas instâncias e o algoritmo *nested-loop ANNS* delimitado na Seção 3.1.4 para obter os valores dos parâmetros necessários para o funcionamento do método.

4.1 Detector de anomalias

O núcleo do sistema de validação estatístico é responsável por conter os algoritmos de detecção de anomalias que são utilizados para a validação dos dados, de forma que seja possível adicionar novos detectores em forma de módulos, desde que suas estruturas respeitem uma certa interface e certas regras.

Para que um módulo possa ser integrado ao sistema, ele deve implementar os seguintes métodos:

- **módulo.criaDetector:** Esse método do módulo visa gerar novos detectores de anomalias (objetos da classe detector). Deve receber de entrada os parâmetros *número_dimensões*, *valor_treinamento*, *absoluto* e *parâmetros_adicionais*. O primeiro é um inteiro que representa o número de dimensões do atributo em questão a ser validado. Já *valor_treinamento* e *absoluto* são um inteiro e um booleano, respectivamente, que juntos indicam a periodicidade com que o treinamento será executado em um detector. O valor de *valor_treinamento* indicará a quantidade de novas operações (adição, alteração ou remoção) sobre instâncias no detector que devem ser realizadas para que o detector possa atualizar seu treinamento. Se o valor de *absoluto* for *Verdadeiro*, então *valor_treinamento* representa o número de operações que devem ser realizadas sobre as instâncias para um novo treinamento, caso contrário (*absoluto* valer *Falso*), representa um percentual em relação à quantidade de instâncias que o detector já possui. Além disso, *parâmetros_adicionais* é um dicionário que permite a entrada de parâmetros, na forma de pares (*chave*, *valor*), que podem ser necessários para o funcionamento de algumas técnicas de detecção de anomalias. Dessa forma, o método retorna um objeto detector criado a partir dessas configurações.
- **detector.adiciona:** Esse método da classe detector permite a inclusão de uma instância de treinamento com sua respectiva marca. Uma marca representa os seguintes valores: *Válido (rotulado)*, que indica que a instância (lote reduzido) foi confirmada como válida por um especialista do domínio; *Supostamente válido (não rotulado)*, que representa que a instância passou pelo processo de validação e foi considerada válida pelo algoritmo; *Inválido (rotulado)*, que indica que o lote foi confirmado como inválido por um especialista do domínio; *Supostamente inválido (não rotulado)*, que representa que a instância passou pelo processo de validação e foi considerada inválida pelo algoritmo e *Desconhecido (não rotulado)*, que indica que não se sabe nada a respeito da validade da instância. O método recebe os parâmetros *instância* e *marca*, em que o primeiro consiste em um objeto que contém o identificador da instância (lote reduzido) e um vetor com as proporções de suas

categorias, e o segundo é a marca associada à instância. A instância é armazenada em uma lista de espera para que possa fazer parte do treinamento quando ele for realizado. Caso a instância já esteja no treinamento, a marca associada a ela é substituída pela nova marca recebida.

- **detector.remove:** Esse método do detector permite que uma instância previamente inserida (ou agendada para inserção) e sua respectiva marca sejam removidas do detector. Ele recebe o parâmetro *id_instância*, que representa o id da instância que se deseja remover. A remoção é colocada em uma lista de espera, para que a instância seja removida na próxima vez que o treinamento for realizado.
- **detector.classifica:** Permite a classificação de instâncias pelo detector, utilizando o algoritmo de detecção de anomalias. Esse método recebe o parâmetro *instância*, que é um objeto que contém a identificação da instância a ser classificada e um vetor com as proporções de suas categorias, e retorna *Verdadeiro* ou *1* se a instância for classificada como normal e *Falso* ou *0* se for classificada como anomalia.
- **detector.estaAtivo:** Esse método indica se o detector está ativo, ou seja, se está apto a receber pedidos de validação das instâncias. Ele retorna um booleano *Verdadeiro* se estiver pronto e *Falso* caso contrário. O detector só poderá ser usado se esse método retornar *Verdadeiro*.

Para implementar as funções acima, é sugerido que se tenha duas estruturas de dados gerais: uma lista das instâncias já treinadas e suas respectivas marcas e uma fila das alterações a serem feitas no conjunto de dados.

4.2 Algoritmo kNN

As técnicas de detecção de anomalias em sua maioria resolvem o problema de encontrar as instâncias anômalas em um conjunto de dados. Entretanto, no contexto do sistema validador estatístico de dados, o problema se resume em definir se uma determinada instância é uma anomalia quando analisada em um conjunto de dados. Dessa forma, adaptamos o método do k-ésimo vizinho mais próximo delimitado na Seção 3 de forma a realizar a classificação de uma instância entre anomalia ou normal.

A ideia por trás da fase de teste desse método se baseia em classificar uma instância como normal se a sua distância ao seu k-ésimo vizinho mais próximo for relativamente pequena, e como anômala caso contrário. O Algoritmo 2 descreve um algoritmo que classifica a instância entre normal ou anômala.

Esse algoritmo utiliza um conjunto *kVizinhos* para armazenar a lista dos *k* vizinhos mais próximos da instância a ser classificada. Dessa forma, sempre que se encontra um vizinho mais próximo da instância do que os demais que estão na lista, *kVizinhos* é atualizado. Além disso, dois métodos são utilizados: *dist*, que calcula a distância entre duas instâncias (por exemplo, distância Euclidiana ou de Manhattan) e *distKNN*, que retorna a distância entre a instância e seu k-ésimo vizinho mais próximo encontrado até um certo momento.

Assim, ao final do algoritmo, se a distância entre a instância e seu k-ésimo vizinho mais próximo for menor que um dado valor *c*, essa distância é considerada relativamente pequena e portanto a instância é classificada como normal. Caso contrário, como anomalia.

Algoritmo 2 Algoritmo de classificação

entrada: x (elemento a ser classificado), k , c (*threshold*), D (conjunto de dados de entrada)

saída: Anomalia ou normal

```
1: Inicialize  $kVizinhos$  como vazio
2: para cada  $b$  em  $D$  faça
3:    $d = dist(b, x)$ 
4:   se a quantidade de elementos em  $kVizinhos$  for menor que  $k$  ou  $d < distKNN(x, kVizinhos)$  então
5:     Atualize  $kVizinhos$  com as  $k$  instâncias do conjunto  $kVizinhos \cup d$  que estão mais próximas de  $x$ 
6:   fim se
7: fim para
8: se a distância de  $x$  ao seu  $k$ -ésimo vizinho for menor ou igual a  $c$  então
9:   devolva normal
10: senão
11:   devolva anomalia
12: fim se
```

Entretanto o funcionamento desse algoritmo depende dos parâmetros c e k . Para determiná-los, utilizaremos na fase de treino o algoritmo descrito na Seção 3.1.4, uma vez que é um algoritmo que utiliza uma estrutura de dados simples e se mostra eficiente na prática [34]. Os parâmetros serão então treinados a fim de maximizar a eficiência do algoritmo.

Utilizaremos o índice *F1-Score* [47] como métrica para quantificar a eficiência do algoritmo quando executado com um determinado valor de k . Esse índice é calculado a partir de uma comparação entre os rótulos originais das instâncias e os rótulos obtidos pela execução do algoritmo, considerando os seguintes conceitos:

- Verdadeiros positivos (VP): número de instâncias originalmente anômalas classificadas corretamente pelo algoritmo como anomalias.
- Falsos positivos (FP): número de instâncias originalmente normais classificadas incorretamente pelo algoritmo como anomalias.
- Falsos Negativos (FN): número de instâncias originalmente anômalas classificadas incorretamente pelo algoritmo como normais.
- Verdadeiros negativos (VN): número de instâncias originalmente normais classificadas corretamente pelo algoritmo como normais.

Dessa forma, a métrica *F1-Score* é calculada a partir de dois conceitos: precisão (P) e cobertura (C). A precisão é obtida pela razão entre o número de instâncias corretamente classificadas como anomalias e o número total de instâncias classificadas pelo algoritmo como anomalias. Assim, temos:

$$P = \frac{VP}{VP + FP}$$

Já a cobertura é calculada a partir da razão entre o número de instâncias corretamente classificadas como anomalias e o número de instâncias originalmente anômalas. Temos:

$$C = \frac{VP}{VP + FN}$$

Assim, obtemos o cálculo do índice *F1-Score*:

$$F1-Score = \frac{2 * P * C}{P + C}$$

O cálculo do *F1-Score* deve ser feito para diversos valores de k , e aquele que maximizar o valor do *F1-Score*, será utilizado no algoritmo de classificação ([Algoritmo 2](#)). Esse cálculo deve ser repetido a cada novo treinamento.

O parâmetro c será obtido a partir do último valor assumido pela variável c no algoritmo descrito na Seção 3.1.4 quando ele é executado com o melhor valor de k (o que maximiza o valor do *F1-Score*). Ou seja, c representará o menor valor de $D^k(p)$ encontrado para determinado k em que p é uma anomalia.

O algoritmo de treinamento pode ser descrito da seguinte maneira ([Algoritmo 3](#)):

Algoritmo 3 Algoritmo de treinamento

entrada: D (conjunto de treino), R (conjunto de rótulos das instâncias de treino), n (quantidade de anomalias no conjunto de treino), N (quantidade de instâncias de treino)

saída: $melhor_k$

```

1: Inicialize  $f1 = 0$ 
2: Inicialize  $melhor\_k = 0$ 
3: para  $k = 1$  até  $\sqrt{N}$  faça
4:    $rotulos = algoritmoSE(D, n, k)$ 
5:    $f1\_aux = F1score(rotulos, R)$ 
6:   se  $f1\_aux > f1$  então
7:      $f1 = f1\_aux$ 
8:      $melhor\_k = k$ 
9:   fim se
10: fim para
11: devolva  $melhor\_k$ 
```

A escolha do intervalo em que k varia foi feita baseando-se em um critério frequentemente utilizado de atribuir $k = \sqrt{N}$, em que N é o número de instâncias do conjunto de entrada, o que torna o processo menos ineficiente do que seria caso k variasse em um intervalo de 1 a N . [48; 49].

Nesta implementação estamos utilizando instâncias de $d-1$ dimensões (em que d é a quantidade de categorias do atributo que está sendo analisado) em vez de d dimensões, uma vez que como descrito na seção 2.3, uma dimensão pode ser descrita com base nas outras dimensões, por se tratar de proporções. A escolha da dimensão a ser eliminada é feita sobre a que possui maior variância, ou seja, a que apresenta maior dispersão dos valores, com a intenção de reduzir a linearidade dos dados.

Para o funcionamento correto do algoritmo, o conjunto das instâncias de treino deve estar rotulado corretamente, uma vez que o método implementado nesse trabalho é supervisionado.

A implementação em *Python* [50] dos algoritmos descritos nessa seção está detalhada no apêndice desse trabalho.

5 Estudo de caso

Para testar a implementação do módulo descrita na Seção 4 realizamos um estudo de caso, que será descrito nesta seção.

Aplicamos o módulo baseado no modelo gaussiano implementado por Eduardo Dias Filho [9] e o módulo desenvolvido neste trabalho em diferentes conjuntos de dados (simulados e reais) para realizar uma análise comparativa entre os dois métodos.

Os lotes que utilizamos são provenientes ou baseados nos dados do sistema *Bloddiss* (*Blood Donation Data Integration System*)⁵, que foi desenvolvido para o projeto REDS-II pelo grupo DATA como um sistema para integração de dados que recebe lotes de doação de sangue de diversos hemocentros. Como os atributos de cada lote são inúmeros, escolhemos apenas alguns deles para serem utilizados no processo de validação, fazendo com que a visualização dos dados e de suas classificações fique mais simplificada. Criamos, então, validadores para classificar lotes considerando atributos como tipo sanguíneo e sexo.

5.1 Atributo comportamental tipo sanguíneo

Para realizar a validação de lotes considerando o atributo tipo sanguíneo como comportamental, utilizamos dados reais do sistema *Bloddiss* e dados simulados que seguem uma distribuição gaussiana. Para gerar esses dados, nos baseamos nos valores de média e desvio padrão dos dados reais. Dessa forma foram gerados lotes com dados aleatórios seguindo uma distribuição normal com os parâmetros de média e desvio padrão dos dados reais. As instâncias reais e as geradas podem ser visualizadas na Figura 8. Nesse caso consideramos o atributo tipo sanguíneo como atributo comportamental, cujas categorias são O, A, B e AB. Como os valores de cada categoria correspondem a proporções da categoria no lote, uma das dimensões pode ser escrita em função das demais. Optamos então por eliminar a categoria AB por motivos ilustrativos, e dessa forma a figura mostra as proporções das categorias O, A e B de cada lote representado por um “x”. A categoria que representa tipo sanguíneo desconhecido foi desconsiderada neste trabalho com o intuito de possibilitar a visualização dos dados.

Para realizar o treinamento dos detectores no sistema, são necessárias também instâncias inválidas. Uma forma simples de gerar anomalias considerando apenas o atributo tipo sanguíneo é inverter as proporções dos tipos O e AB, pois a proporção de amostras de sangue do tipo O são muito mais comuns na população do que as do tipo B.

Nessa configuração, os dois métodos - gaussiano e k-ésimo vizinho mais próximo - apresentaram bons resultados. Ambos classificaram corretamente instâncias normais e anômalas, como as ilustradas nas Figuras 9 e 10, respectivamente. O parâmetro obtido no treinamento do módulo gaussiano nessa configuração foi $\epsilon = 0.0000227601832971$, enquanto que os parâmetros obtidos no treinamento do módulo do k-ésimo vizinho mais próximo foram $c = 0.30376012593272605$ e $k = 13$.

⁵Disponível em <https://data.ime.usp.br/bloddiss>

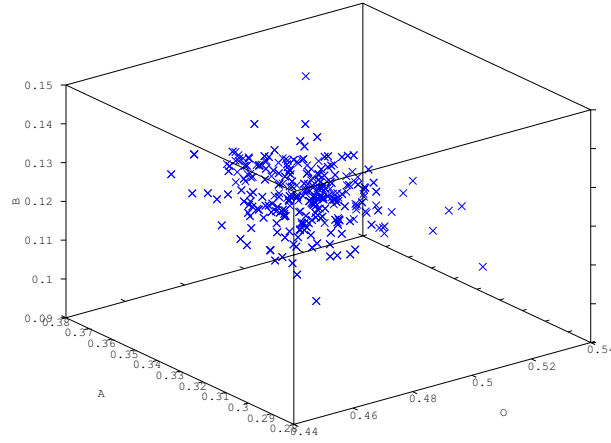


Figura 8: *Instâncias reais e instâncias geradas seguindo uma distribuição gaussiana. Cada eixo corresponde às proporções de cada categoria (O, A e B) em cada lote representado por um “x”.*

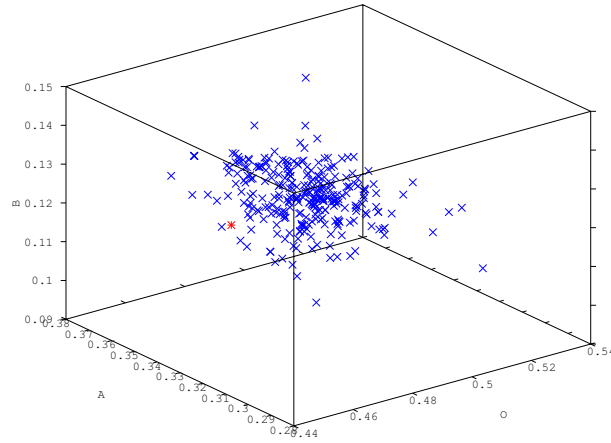


Figura 9: *Instâncias válidas reais, instâncias geradas seguindo uma distribuição gaussiana (em azul) e instância válida a ser classificada (em vermelho). Cada eixo corresponde às proporções de cada categoria (O, A e B) em cada lote representado por um “x”.*

5.2 Atributo comportamental sexo

O método de detecção de anomalias gaussiano não modela adequadamente situações em que os dados se encontram agrupados, como visto na Seção 3. Desse modo, para testar o funcionamento dos dois algoritmos utilizamos sexo como atributo comportamental. Como os valores possíveis para esse atributo são Feminino e Masculino, uma das dimensões foi eliminada e assim obtivemos instâncias univariadas para o validador. Como podemos ver na Figura 11, as instâncias seguem uma distribuição em *cluster*. Por se tratarem de instâncias univariadas, acrescentamos o atributo *id* ao gráfico para possibilitar uma melhor visualização dos dados. Foram utilizadas instâncias reais do sistema *Bloddis* e instâncias geradas a partir de parâmetros obtidos pelos dados reais.

As instâncias inválidas para o treinamento foram geradas a partir dos pontos médios das instâncias válidas, como ilustrado na Figura 12. Nessa configuração, o método da gaussiana não obteve sucesso na detecção da anomalia, enquanto que o método do k-ésimo

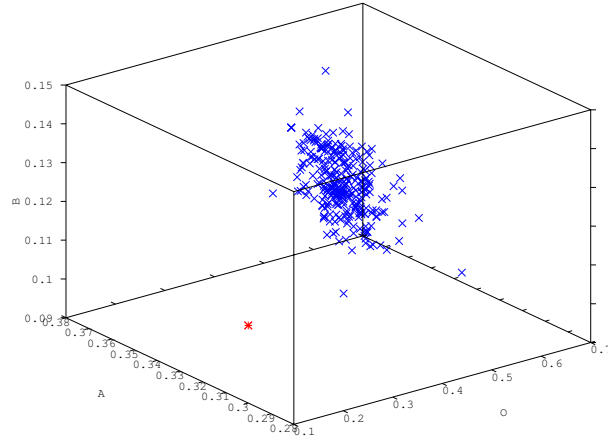


Figura 10: *Instâncias válidas reais, instâncias geradas seguindo uma distribuição gaussiana (em azul) e uma instância inválida a ser classificada (em vermelho). Cada eixo corresponde às proporções de cada categoria (O, A e B) em cada lote representado por um “x”.*

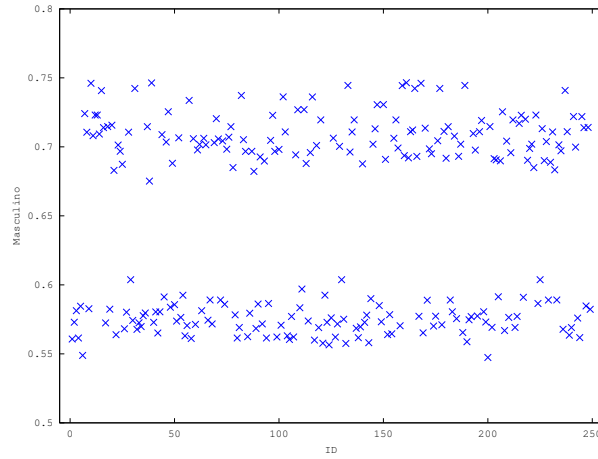


Figura 11: *Instâncias reais e instâncias geradas seguindo uma distribuição em agrupamento. O eixo horizontal corresponde ao id da instância, enquanto que o eixo vertical corresponde à categoria Masculino. Cada lote está representado por um “x”.*

vizinho mais próximo classificou corretamente a instância como anômala. O parâmetro obtido no treinamento do módulo gaussiano foi $\epsilon = 0.00603469157095$, e no caso do módulo do k-ésimo vizinho mais próximo foram $c = 0.02300002$ e $k = 2$.

No caso de instâncias válidas, ambos os métodos apresentaram a mesma performance, classificando corretamente as instâncias em configurações similares à instância na Figura 13.

A visualização da distribuição dos dados sem eliminarmos uma dimensão (ou seja, um lote sendo uma instância bidimensional) pode ser obtida pela Figura 14.

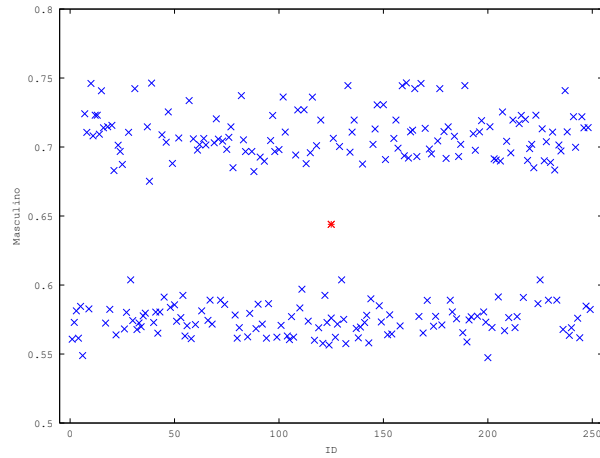


Figura 12: *Instâncias reais e instâncias geradas seguindo uma distribuição em agrupamento (em azul) e uma instância inválida a ser classificada (em vermelho). O eixo horizontal corresponde ao id da instância, enquanto que o eixo vertical corresponde à categoria Masculino. Cada lote está representado por um “x”.*

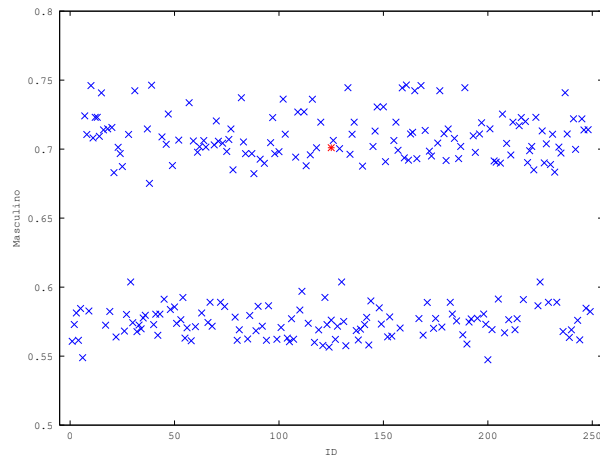


Figura 13: *Instâncias reais e instâncias geradas seguindo uma distribuição em agrupamento (em azul) e uma instância válida a ser classificada (em vermelho). O eixo horizontal corresponde ao id da instância, enquanto que o eixo vertical corresponde à categoria Masculino. Cada lote está representado por um “x”.*

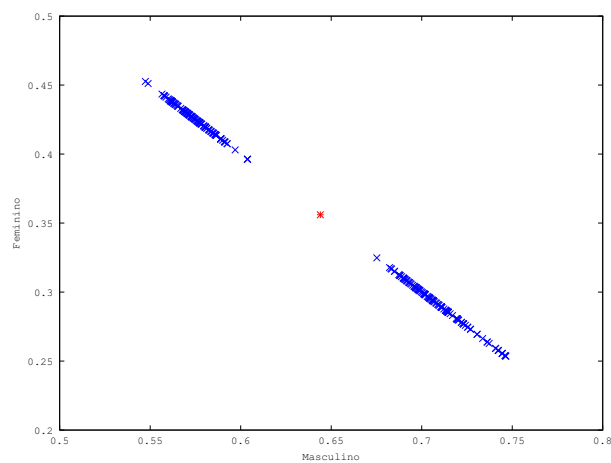


Figura 14: *Instâncias reais e instâncias geradas seguindo uma distribuição em agrupamento (em azul) e uma instância válida a ser classificada (em vermelho). O eixo horizontal corresponde à categoria Masculino, enquanto que o eixo vertical corresponde à categoria Feminino. Cada lote está representado por um “x”.*

6 Conclusão

Detecção de anomalias é uma importante ferramenta para várias áreas de estudos, pois anomalias são importantes fontes de informação. Por serem dados não esperados, raros e por não possuírem um padrão, elas podem significar eventos importantes e requerem maior análise do especialista em questão.

Existem diversas aplicações para detecção de anomalias, e diferentes técnicas. As técnicas se diferem por sua forma de encontrar e determinar as anomalias. Elas possuem vantagens e desvantagens, pois cada técnica depende das características dos dados estudados. Não existe uma técnica que seja melhor que a outra para todos os casos, a escolha da melhor técnica depende do que se quer obter no estudo e dos dados estudados.

No estudo de caso, percebemos que a distribuição dos dados, ou seja, a forma como estão agrupados interfere na qualidade da técnica de detecção de anomalias. A técnica de detecção gaussiana funciona muito bem para muitos conjuntos de dados, porém quando os dados formam *clusters*, a técnica de detecção gaussiana não consegue encontrar as anomalias, enquanto uma técnica baseada em distância consegue. Porém as técnicas baseadas em distância possuem uma maior complexidade computacional que a técnica de detecção gaussiana.

Uma limitação apresentada pelo método implementado neste trabalho foi a incapacidade de lidar com dados que não estejam corretamente classificados, ou seja, anomalias com rótulos de instâncias normais e instâncias normais com rótulos de anomalias, mesmo que apareçam em pequenas proporções. Nesse caso, o funcionamento do algoritmo é instável, podendo ser comprometido ou não por essas instâncias.

Para corrigir esse problema, algumas alterações são necessárias, como eliminar a regra de *prunning* do algoritmo (que funciona sob a hipótese dos dados estarem corretamente rotulados) e também a dependência do parâmetro n . A implementação dessa solução faz parte de trabalhos futuros, bem como melhorar a eficiência da técnica utilizando estruturas de dados apropriadas. Outro ponto a ser realizado em trabalhos futuros é calcular para cada técnica sua taxa de acerto, para assim possibilitar uma comparação das técnicas mais precisa com a utilização de alguma metodologia estatística.

Parte II

Parte Subjetiva

7 Luciana de Melo e Abud

7.1 Desafios e frustrações

Um dos maiores desafios e também uma grande frustração deste trabalho foi lidar com o sistema de validação estatístico, uma vez que sua implementação está incompleta e apresenta inconsistências e *bugs* frequentes. O treinamento do sistema teve que ser feito à mão, ou seja, alterando o código do sistema cada vez que se desejava uma configuração diferente dos dados.

Como o núcleo do sistema não está pronto, o método não pôde ser acoplado da maneira em que o sistema foi idealizado. Sendo assim, foi necessário alterar uma linha do código do sistema para chamar o módulo implementado.

Identificar as partes que precisariam ser alteradas foi bastante difícil, ainda mais porque nós não sabíamos *python*.

Uma grande frustração também foi perceber a inabilidade do método implementado funcionar com instâncias com rótulos incorretos. Percebemos isso perto do final do desenvolvimento do projeto, e então não conseguimos corrigir a tempo. Essa correção faz parte de trabalhos futuros.

Outra frustração que tive no começo do desenvolvimento deste trabalho foi a dificuldade de escrever de forma “científica”. Dada a minha inexperiência, o tempo que levava para escrever era absurdamente grande. Felizmente esse tempo foi reduzindo ao longo do trabalho.

Como desafio, também posso citar a escolha da técnica a ser implementada. Foi uma tarefa difícil porque precisávamos de uma técnica que funcionasse bem em diferentes configurações dos dados, e que não fosse tão ineficiente.

7.2 Relacionamento do trabalho com o curso do BCC

Abaixo, seguem as matérias que foram mais relevantes para o desenvolvimento deste trabalho:

- **MAC0122 - Princípios de Desenvolvimento de Algoritmos:** Essa matéria foi bastante útil por ter dado uma boa intuição de desenvolvimento de algoritmos. Foi fundamental para o meu aprendizado de programação, e me deu uma boa base para a implementação do módulo deste trabalho.
- **MAC0323 - Estruturas de Dados:** Também foi bastante útil por esclarecer qual estrutura de dados utilizar para resolver cada problema, seja listas, dicionários, *heaps*, etc. Apesar de não termos usado nenhuma estrutura de dados mais complexa (usamos apenas listas e dicionários), pode ser interessante para trabalhos futuros fazer uso dessas EDs para melhorar a performance do módulo.
- **MAC0426 - Sistemas de Banco de Dados:** Essa matéria foi útil por ser praticamente a primeira a falar sobre armazenamento de dados, e dessa forma foi possível aprender a como lidar com os dados e assim como analisá-los. Como o sistema de validação estatístico utiliza um banco de dados, essa matéria se mostrou bastante necessária no desenvolvimento deste projeto.
- **MAC0439 - Laboratório de Bancos de Dados:** Foi uma matéria bastante interessante e também útil para esse trabalho por se tratar de uma matéria prática

de banco de dados, dando uma intuição maior sobre como funcionam sistemas que utilizam bancos de dados na prática.

- **MAE0121 - Introdução a Probabilidade e a Estatística I:** Essa matéria foi útil principalmente para o entendimento dos métodos estatísticos estudados nesse trabalho.
- **MAE0212 - Introdução a Probabilidade e a Estatística II:** Também foi útil para o entendimento do método gaussiano implementado para o sistema, além dos métodos estatísticos estudados.

8 Helena Almeida Victoretti

8.1 Desafios e frustrações

A escolha da técnica a ser desenvolvida para o sistema de validação foi um dos grandes desafios do trabalho, pois precisávamos de uma técnica que se adequasse a vários tipos de dados mantendo a sua performance. Outro desavio foi o estudo das técnicas, pois a maioria dos artigos e estudos de detecção de anomalias são para encontrar anomalias em conjuntos de dados e não dada uma instância identificá-la como anômala ou normal, que era o nosso foco de interesse.

A escrita “científica” foi um desafio, também. Por possuir nenhuma prática em escrever textos científicos, o começo do desenvolvimento da monografia se mostrou difícil.

Um desafio que esperava encontrar era a linguagem *Python* que não havia trabalhado antes e por sua vez foi utilizada para implementar a técnica de detecção de anomalias, porém essa se mostrou uma linguagem de programação rápida de ser aprendida. O desafio quanto a linguagem ocorreu no processo estudar o código do sistema e como alterá-lo.

O sistema utilizado no estudo de caso foi idealizado para se poder acoplar módulos com diferentes técnicas de anomalias sem o módulo precisar ter conhecimento de como o sistema havia sido implementado. Portanto, imaginamos que nosso módulo seria acoplado ao sistema e usaríamos o sistema para testar a técnica de detecção de anomalias escolhida e a técnica já existente. Porém o sistema não estava completamente implementado, gerando um desafio e frustração para o nosso projeto. Pois foi preciso inserir nosso módulo de detecção de anomalias direto no código do sistema, interferindo na realização do estudo de caso.

Além disso, como precisamos utilizar bastante o sistema para a realização do estudo de caso, o fato do sistema não ser muito simples de utilizar (não ser “*user friendly*”), causou outra frustração em relação ao projeto.

Outro ponto frustrante, foi quando percebemos que a técnica de detecção de anomalia escolhida perde sua qualidade quando utilizamos instâncias com rótulos incorretos. A correção desse problema faz parte de trabalhos futuros.

8.2 Relacionamento do trabalho com o curso do BCC

As matérias estudadas que se mostram mais relevantes para o desenvolvimento do projeto foram:

- **MAC110 - Introdução à Computação:** Quando entrei na faculdade não sabia programar e foi com MAC110 que tive minha primeira experiência com a computação, por isso é uma das matérias mais importantes para o desenvolvimento desse projeto.
- **MAC122 - Princípios de Desenvolvimento de Algoritmos:** Assim como MAC110 foi meu primeiro contato com a programação, MAC122 foi a matéria em que aprendi a desenvolver um algoritmo, como pensar no problema e criar uma solução através da programação. Portanto, foi uma matéria importante que possibilitou o desenvolvimento do nosso módulo de detecção de anomalia.
- **MAC323 - Estruturas de Dados:** As técnicas de detecção de anomalias trabalham com muitos dados e a forma como se estrutura e trabalha com esses dados interfere na qualidade e performance das técnicas.

- **MAC426 - Sistemas de Bancos de Dados:** Foi a partir dessa matéria que tivemos contato com armazenamento de dados e análise de dados. Como utilizamos dados reais de um banco de dados, essa matéria se mostrou útil.
- **MAC0439 - Laboratório de Bancos de Dados:** Essa matéria nos proporcionou um estudo de banco de dados na prática, mostrando como funciona a estrutura de um banco de dados, de forma que nos foi útil para entender a razão de se desenvolver um *data warehouse* e o sistema de validação.
- **MAE121 - Introdução à Probabilidade e Estatística I e MAE212 - Introdução à Probabilidade e Estatística II:** Essas matérias nos proporcionaram uma noção básica de estatística que nos permitiu entender e estudar os métodos de detecção de anomalias baseados em métodos estatísticos.

Apêndice

Implementação dos algoritmos

Os algoritmos foram escritos em *Python* [50].

```
def distanceFunction(x, y):
    #distância de manhattan (implementada em scipy.spatial)
    return distance.cityblock(x,y)

# retorna a maior distância entre b e os elementos em N
def maxDist(b, N):
    if(len(N) == 0):
        return 0
    return distanceFunction(b, N[len(N)-1])

# retorna os k elementos em (N U {d}) que estão mais próximos de b
def kNearestNeighbors(k, b, N, d):
    if(len(N) == 0):
        N.append(d)
    else:
        newIndex = -1
        dist = distanceFunction(b, d)
        for i in range(0, len(N)):
            if(dist < distanceFunction(b, N[i])):
                newIndex = i
                break
        # se a distância entre b e d é menor do que alguma
        # distância entre b e um elemento em N,
        # então insere d na posição desse elemento
        if(newIndex != -1):
            N.insert(newIndex, d)
        # caso contrário, insere no fim da lista
        else:
            N.append(d)
        # se a lista ficou com mais que k elementos, retira o último
        # (retira o vizinho mais distante)
        if(len(N) > k):
            N.pop()
    return N

# encontra as n anomalias em D baseando-se no método kNN
def algoritmoANNS(D, k, n):
    c = 0.0
    O = []
    Neighbors = []
    for i in range(0, len(D)):
        b = D[i]
        Neighbors.append([])
        for j in range(0, len(D)):
```

```

        if(i == j): continue
        d = D[j]
        if(len(Neighbors[i]) < k or
            distanceFunction(b,d) < maxDist(b, Neighbors[i])):
            Neighbors[i] = kNearestNeighbors(k, b, Neighbors[i], d)
            if(maxDist(b, Neighbors[i]) < c):
                break
    O = bestOutliers(n, O, b, i, Neighbors)
    c = maxDist(O[0][0], Neighbors[O[0][1]])

result = ['valid']*len(D)
for i in range(0, len(O)):
    result.pop(O[i][1])
    result.insert(O[i][1], 'invalid')
return [result, c]

# retorna as n melhores anomalias em (O U {b}), em relação às distâncias
# aos k-ésimos vizinhos mais próximos de cada instância.
# as instâncias são armazenadas em O em ordem crescente das distâncias
# aos k-ésimos vizinhos mais próximos.
def bestOutliers(n, O, b, i, N):
    if(len(O) == 0):
        O.append([b, i])
    else:
        newIndex = -1
        distKNN = maxDist(b, N[i])
        for k in range(0, len(O)):
            if(distKNN < maxDist(O[k][0], N[O[k][1]])):
                newIndex = k
                break
        if(newIndex != -1):
            O.insert(newIndex, [b, i])
        else:
            O.append([b, i])
        if(len(O) > n):
            O.pop(0)
    return O

# encontra os melhores valores de c e k.
def treinamento(D, R, n):
    f = 0.0
    faux = 0.0
    bestK = 0
    bestC = 0.0
    N = math.ceil(len(D)**(0.5))
    for k in range(1, N):
        [labels, c] = algoritmoANNS(D, k, n)
        faux = F1Score(labels, R)
        if(faux > f):
            f = faux

```

```

        bestK = k
        bestC = c
    self.k = bestK
    self.c = bestC

# classifica x como normal (1) ou anômala (0).
def classify(x, k, c, D) :
    kNeighbors = []
    for i in range(0, len(D)):
        d = D[i]
        if(len(kNeighbors) < k or
            distanceFunction(x, d) < maxDist(x, kNeighbors)):
            kNeighbors = kNearestNeighbors(k, x, kNeighbors, d)
    if(maxDist(x, kNeighbors) <= c):
        return 1
    return 0

# VP: Verdadeiro Positivo, FP: Falso Positivo
# VN: Verdadeiro Negativo, FN: Falso Negativo
def F1Score(predicted, original):
    VP, FP, VN, FN = 0, 0, 0, 0
    for i in range(0, len(predicted)):
        if(predicted[i] == 'valid'):
            if(original[i] == 'valid'):
                VN = VN + 1
            else:
                FN = FN + 1
        else:
            if(original[i] == 'valid'):
                FP = FP + 1
            else:
                VP = VP + 1

    if ((VP+FP) == 0 or (VP+FN) == 0):
        return 0

    P = VP/(VP+FP)
    C = VP/(VP+FN)

    if (P+C) == 0:
        return 0

    return 2*P*C/(P+C)

```

Referências

- [1] D.M. Hawkins. *Identification of Outliers*. Monographs on applied probability and statistics. Chapman and Hall, 1980. 4
- [2] H. Jair Escalante. A comparison of outlier detection algorithms for machine learning, 2005. 4
- [3] Paulo M. Mafra, Joni da Silva Fraga, Vinicius Moll, and Altair Olivo Santin. POLVO-IIDS: Um sistema de detecção de intrusão inteligente baseado em anomalias, 2008. 4
- [4] Gerhard Münz, Sa Li, and Georg Carle. Traffic anomaly detection using kmeans clustering, 2007. 4
- [5] Prakash N. Kalavadekar Amruta D. Pawar and Swapnali N. Tambe. A survey on outlier detection techniques for credit card fraud detection. *IOSR Journal of Computer Engineering*, 16(2):44–48, 2014. 4
- [6] Clay Spence, Lucas Parra, and Paul Sajda. Detection, synthesis and compression in mammographic image analysis with a hierarchical image probability model, 2001. 4
- [7] Ji Zhang. Advancements of outlier detection: A survey. *EAI Endorsed Trans. Scalable Information Systems*, 1:e2, 2013. 4, 7, 8, 12, 13, 15, 16, 17, 21
- [8] Pedro L. Takecian. Diretrizes metodológicas e validação estatística de dados para a construção de *data warehouses*, 2014. 4, 5, 18, 19, 20, 21
- [9] Eduardo Dias Filho. Implementação de um sistema de validação estatística configurável de dados, 2014. 5, 32
- [10] Anna Bárbara Carneiro-Proietti, Ester C. Sabino, Divaldo Sampaio, Fernando A. Proietti, Thelma T. Gonzalez, Cláudia D.L. Oliveira, João E. Ferreira, Jing Liu, Brian Custer, George B. Schreiber, Edward L. Murphy, and Michael P. Busch. Demographic profile of blood donors at three major brazilian blood centers: results from the international reds-ii study, 2007 to 2008. *Transfusion*, 50(4):918–925, 2010. 5
- [11] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1 – 15:58, 2009. 7, 8, 9, 11, 12, 13, 14, 15, 16, 17
- [12] Andrew Ng. Machine learning: Lecture xv. anomaly detection. <https://class.coursera.org/ml-005/lecture/preview>. Universidade de Stanford, Plataforma Coursera. 7, 15, 16
- [13] Josef Burger. A basic introduction to neural networks. <http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html>. Acessado em 18/05/2015. 8
- [14] David Reby, Sovan Lek, Ioannis Dimopoulos, Jean Joachim, Jacques Lauga, and Stéphane Aulagnier. Artificial neural networks as a classification method in the behavioural sciences. *Behavioural Processes*, 40(1):35 – 43, 1997. 8

- [15] Gábor Horvath. Neural networks in system identification. *NATO SCIENCE SERIES SUB SERIES III COMPUTER AND SYSTEMS SCIENCES*, 185:43–78, 2003. [8](#)
- [16] John Akhilomen. Data mining application for cyber credit-card fraud detection system. In *Proceedings of the 13th International Conference on Advances in Data Mining: Applications and Theoretical Aspects*, ICDM’13, pages 218–228, Berlin, Heidelberg, 2013. Springer-Verlag. [9](#), [10](#), [15](#)
- [17] Anup K. Ghosh, Aaron Schwartzbard, and Michael Schatz. Learning program behavior profiles for intrusion detection. In *Proceedings of the 1st Conference on Workshop on Intrusion Detection and Network Monitoring - Volume 1*, ID’99, pages 6–6, Berkeley, CA, USA, 1999. USENIX Association. [9](#)
- [18] Irad Ben-Gal. *Bayesian Networks*. 2008. [9](#)
- [19] Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian network classifiers. *Mach. Learn.*, 29(2-3):131–163, November 1997. [9](#)
- [20] Sakshi Babbar and Sanjay Chawla. On bayesian network and outlier detection. 2010. [11](#)
- [21] Abdallah Abbey Sebyala, Temitope Olukemi, Lionel Sacks, and Dr. Lionel Sacks. Active platform security through intrusion detection using naive bayesian network for anomaly detection. In *In: Proceedings of London communications symposium*, 2002. [11](#)
- [22] M. Taniguchi, M. Haft, J. Hollmen, and V. Tresp. Fraud detection in communication networks using neural and probabilistic methods. In *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, volume 2, pages 1241–1244 vol.2, May 1998. [11](#)
- [23] H. Buxton. Advanced visual surveillance using bayesian networks. In *Image Processing for Security Applications (Digest No.: 1997/074)*, *IEE Colloquium on*, pages 9/1–9/5, Mar 1997. [11](#)
- [24] Neethu B. Adaptive intrusion detection using machine learning, 2013. [11](#)
- [25] Wenjie Hu, Yihua Liao, and V. Rao Vemuri. Robust anomaly detection using support vector machines. [11](#)
- [26] Paul Hayton, Bernhard Schölkopf, Lionel Tarassenko, and Paul Anuzis. Support vector novelty detection applied to jet engine vibration spectra. *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS 13*, pages 946–952, 2000. [11](#)
- [27] Luiz Gustavo Moro Senko. Um método baseado em lógica paraconsistente para detecção de inconsistências em classificadores à base de regras, 2006. [12](#)
- [28] Philip K. Chan Matthew V. Mahoney. Learning rules for anomaly detection of hostile network traffic, 2003. [12](#)

- [29] Gregory Cooper Michael Wagner Weng-Keen Wong, Andrew Moore. Rule-based anomaly pattern detection for detecting disease outbreaks. In *Proceedings of the 18th National Conference on Artificial Intelligence*. MIT Press, 2002. Also available online from <http://www.cs.cmu.edu/simawm/antiterror>. 12
- [30] Marco A.F. Pimentel, David A. Clifton, Lei Clifton, and Lionel Tarassenko. A review of novelty detection. *Signal Processing*, 99:215 – 249, 2014. 12, 13, 16, 17
- [31] Edwin M. Knorr, Raymond T. Ng, and Vladimir Tucakov. Distance-based outliers: Algorithms and applications, 2000. 12, 13, 21
- [32] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. Efficient algorithms for mining outliers from large data sets. *SIGMOD Rec.*, 29(2):427–438, May 2000. 12, 13, 22, 23, 27
- [33] Coen van Leeuwen, Arvid Halma, and Klammer Schutte. Anomalous human behavior detection: an adaptive approach, 2013. 13, 27
- [34] Stephen D. Bay and Mark Schwabacher. Mining distance-based outliers in near linear time with randomization and a simple pruning rule, 2003. 13, 25, 26, 27, 30
- [35] Gustavo H. Orair, Carlos H. C. Teixeira, Wagner Meira, Jr., Ye Wang, and Srinivasan Parthasarathy. Distance-based outlier detection: Consolidation and renewed bearing, September 2010. 13, 22, 23, 25, 26
- [36] Markus Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. Lof: Identifying density-based local outliers. *PROCEEDINGS OF THE 2000 ACM SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA*, pages 93–104, 2000. 13
- [37] Jian Tang, Zhixiang Chen, Ada Wai-Chee Fu, and David Wai-Lok Cheung. Enhancing effectiveness of outlier detections for low density patterns. In *Proceedings of the 6th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining, PAKDD '02*, pages 535–548, London, UK, UK, 2002. Springer-Verlag. 13
- [38] Levent Ertoz, Eric Eilertson, Aleksandar Lazarevic, Pang-Ning Tan, Vipin Kumar, Jaideep Srivastava, and Paul Dokas. Minds - minnesota intrusion detection system. *Next Generation Data Mining*, pages 199–218, 2004. 13
- [39] Richard J. Bolton, David J. Hand, and David J. H. Unsupervised profiling methods for fraud detection. In *Proc. Credit Scoring and Credit Control VII*, pages 5–7, 2001. 15
- [40] Kingsly Leung and Christopher Leckie. Unsupervised anomaly detection in network intrusion detection using clusters. In *Proceedings of the Twenty-eighth Australasian Conference on Computer Science - Volume 38, ACSC '05*, pages 333–342, Darlinghurst, Australia, Australia, 2005. Australian Computer Society, Inc. 15
- [41] Christopher Kruegel and Giovanni Vigna. Anomaly detection of web-based attacks. In *Proceedings of the 10th ACM Conference on Computer and Communications Security, CCS '03*, pages 251–261, New York, NY, USA, 2003. ACM. 17

- [42] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. The r^* -tree: An efficient and robust access method for points and rectangles, May 1990. 23
- [43] Antonin Guttman. R-trees: A dynamic index structure for spatial searching, 1984. 23
- [44] Charu C. Aggarwal, Alexander Hinneburg, and Daniel A. Keim. On the surprising behavior of distance metrics in high dimensional spaces, 2001. 27
- [45] Bahadır Durak. A classification algorithm using mahalanobis distance clustering of data with applications on biomedical data sets, 2011. 27
- [46] Victoria Hodge and Jim Austin. A survey of outlier detection methodologies, October 2004. 27
- [47] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008. 30
- [48] Ahmad Basheer Hassanat, Mohammad Ali Abbadi, Ghada Awad Altarawneh, and Ahmad Ali Alhasanat. Solving the problem of the K parameter in the KNN classifier using an ensemble learning approach. *CoRR*, abs/1409.0919, 2014. 31
- [49] Marcel Jirina and Marcel Jirina Jr. Classifiers based on inverted distances, 2011. 31
- [50] Python. <http://www.python.org/>. Acessado em 13/10/2014. 31, 43