Pontifícia Universidade Católica
do Rio de Janeiro

Igor Oliveira Vasconcelos

Detecção móvel e online de anomalia em múltiplos fluxos de dados: Uma abordagem baseada em processamento de eventos complexos para detecção de comportamento de condução

Tese de Doutorado

Tese apresentada ao Programa de Pós-graduação em Informática da PUC-Rio como requisito parcial para obtenção do grau de Doutor em Informática.

Orientador: Prof. Markus Endler

Rio de Janeiro
Março de 2017

Igor Oliveira Vasconcelos

Detecção móvel e online de anomalia em múltiplos fluxos de dados: Uma abordagem baseada em processamento de eventos complexos para detecção de comportamento de condução

Tese apresentada como requisito parcial para obtenção do grau de Doutor pelo Programa de Pós-graduação em Informática da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Markus Endler
Orientador
Departamento de Informática – PUC-Rio

Profª. Noemi de La Rocque Rodriguez
Departamento de Informática – PUC-Rio

Prof. Hélio Côrtes Vieira Lopes
Departamento de Informática – PUC-Rio

Prof. Paulo de Figueiredo Pires
Departamento de Informática - UFRJ

Prof. Methanias Colaço Rodrigues Junior
Departamento de Informática - UFS

Prof. Marcio da Silveira Carvalho
Coordenador Setorial do Centro
Técnico Científico – PUC-Rio

Rio de Janeiro, 31 de março de 2017

Igor Oliveira Vasconcelos

Bacharel em Ciência da Computação pela Universidade Tiradentes (UNIT) e Mestre em Ciência da Computação pela Universidade Federal de Pernambuco (UFPE).

Ficha Catalográfica

Vasconcelos, Igor Oliveira

Detecção móvel e online de anomalia em múltiplos fluxos de dados: uma abordagem baseada em processamento de eventos complexos para detecção de comportamento de condução / Igor Oliveira Vasconcelos; orientador: Markus Endler. – 2017.

102 f. : il. color. ; 30 cm

Tese (doutorado)–Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2017.

Inclui bibliografia

1. Informática – Teses. 2. Detecção online de anomalia. 3. Processamento de eventos complexos. 4. Sensoriamento à bordo do veículo. 5. Detecção online do comportamento de condução. 6. Dispositivos móveis. I. Endler, Markus. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Esta tese é dedicada aos meus pais, minha filha e à minha esposa. Dedico especialmente a Giovanna que ao longo de seis anos e meio, entre mestrado e doutorado, viu seu pai dedicar-se aos estudos, e por isso muitas vezes ausente, sem nunca tecer uma única queixa.

Ao meu pai (*in memoriam*), meu amigo, parceiro, irmão, confidente, meu herói e exemplo de perseverança. O senhor lutou bravamente durante anos e mesmo quando já estava um pouco mais debilitado me incentivou ir ao Rio em busca de um sonho. Assim como o senhor, que ao graduar-se após os 40 anos disse: *vim, vi* e *venci* (*Veni, vidi, vici*). Hoje eu escrevo: vim, vi e vencemos, pois estivestes comigo durante as dificuldades do doutorado e durante a escrita de cada oração desta tese. Certo de que um dia poderei te dizer estas palavras e te dar um abraço.

Agradecimentos

Ao meu irmão Rafael por ter insistido que ingressasse no doutorado.

Ao meu grande amigo Luciano que embarcou comigo rumo ao Rio.

Ao CNPq e à PUC-Rio, pelos auxílios concedidos, sem os quais este trabalho não poderia ter sido realizado.

Aos meus amigos do LAC e da PUC, principalmente, Bruno e Roriz. Brunão, obrigado pelo acolhimento e atenção.

Ao meu orientador Markus Endler pelas orientações, revisões, críticas e conselhos, ao professor Methanias Colaço pelas orientações, sugestões e contribuições.

Aos meus pais pelo investimento em minha educação. Contudo, faço um agradecimento especial à minha grande mãe, uma leoa, que mesmo nos momentos mais difíceis, jamais deixou de investir no bem mais valioso que ela dizia que poderia me dar, minha formação.

A Camila, minha amada esposa, sem você talvez eu não terminasse essa jornada. Suportou a distância e a ausência física durante os dois primeiros anos. Em meio ao caos, você foi capaz de serenamente me ouvir, aconselhar, motivar e tem conseguido me aturar por quatro anos e meio. Te amo.

Ao meu sogro, Carlos Alberto, pelas conversas, conselhos, amparo e cuidado, principalmente nos momentos difíceis. Muito obrigado ao senhor e à toda família.

Ao grande arquiteto do universo por permitir viver cada uma das dificuldades e aprender com os erros. Graças a sua infinita sabedoria chego ao fim dessa etapa da vida uma pessoa melhor.

Resumo

Vasconcelos, Igor Oliveira; Endler, Markus. **Detecção móvel e online de anomalia em múltiplos fluxos de dados: Uma abordagem baseada em processamento de eventos complexos para detecção de comportamento de condução.** Rio de Janeiro, 2017. 102p. Tese de Doutorado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Dirigir é uma tarefa diária que permite uma locomoção mais rápida e mais confortável, no entanto, mais da metade dos acidentes fatais estão relacionados à imprudência. Manobras imprudentes podem ser detectadas com boa precisão, analisando dados relativos à interação motorista-veículo, por exemplo, curvas, aceleração e desaceleração abruptas. Embora existam algoritmos para detecção *online* de anomalias, estes normalmente são projetados para serem executados em computadores com grande poder computacional. Além disso, geralmente visam escala através da computação paralela, computação em grid ou computação em nuvem. Esta tese apresenta uma abordagem baseada em complex event processing para a detecção *online* de anomalias e classificação do comportamento de condução. Além disso, objetivamos identificar se dispositivos móveis com poder computacional limitado, como os *smartphones*, podem ser usados para uma detecção *online* do comportamento de condução. Para isso, modelamos e avaliamos três algoritmos de detecção *online* de anomalia no paradigma de processamento de fluxos de dados, que recebem os dados dos sensores do *smartphone* e dos sensores à bordo do veículo como entrada. As vantagens que o processamento de fluxos de dados proporciona reside no fato de que este reduz a quantidade de dados transmitidos do dispositivo móvel para servidores/nuvem, bem como se reduz o consumo de energia/bateria devido à transmissão de dados dos sensores e possibilidade de operação mesmo se o dispositivo móvel estiver desconectado. Para classificar os motoristas, um mecanismo estatístico utilizado na mineração de documentos que avalia a importância de uma palavra em uma coleção de documentos, denominada frequência de documento inversa, foi adaptado para identificar a importância de uma anomalia em um fluxo de dados, e avaliar quantitativamente o grau de prudência ou imprudência das manobras dos motoristas. Finalmente, uma avaliação da abordagem (usando o algoritmo que

obteve melhor resultado na primeira etapa) foi realizada através de um estudo de caso do comportamento de condução de 25 motoristas em cenário real. Os resultados mostram uma acurácia de classificação de 84% e um tempo médio de processamento de 100 milissegundos.

Palavras-chave

Detecção *online* de anomalia; processamento de eventos complexos; sensoriamento à bordo do veículo; detecção *online* do comportamento de condução; dispositivos móveis.

Abstract

Vasconcelos, Igor Oliveira; Endler, Markus (Advisor). **A mobile and online outlier detection over multiple data streams: A complex event processing approach for driving behavior detection.** Rio de Janeiro, 2017. 102p. Tese de Doutorado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Driving is a daily task that allows individuals to travel faster and more comfortably, however, more than half of fatal crashes are related to recklessness driving behaviors. Reckless maneuvers can be detected with accuracy by analyzing data related to driver-vehicle interactions, abrupt turns, acceleration, and deceleration, for instance. Although there are algorithms for online anomaly detection, they are usually designed to run on computers with high computational power. In addition, they typically target scale through parallel computing, grid computing, or cloud computing. This thesis presents an online anomaly detection approach based on complex event processing to enable driving behavior classification. In addition, we investigate if mobile devices with limited computational power, such as smartphones, can be used for online detection of driving behavior. To do so, we first model and evaluate three online anomaly detection algorithms in the data stream processing paradigm, which receive data from the smartphone and the in-vehicle embedded sensors as input. The advantages that stream processing provides lies in the fact that reduce the amount of data transmitted from the mobile device to servers/the cloud, as well as reduce the energy/battery usage due to transmission of sensor data and possibility to operate even if the mobile device is disconnected. To classify the drivers, a statistical mechanism used in document mining that evaluates the importance of a word in a collection of documents, called inverse document frequency, has been adapted to identify the importance of an anomaly in a data stream, and then quantitatively evaluate how cautious or reckless drivers' maneuvers are. Finally, an evaluation of the approach (using the algorithm that achieved better result in the first step) was carried out through a case study of the 25 drivers' driving

behavior. The results show an accuracy of 84% and an average processing time of 100 milliseconds.

Keywords

Online anomaly detection; complex event processing; in-vehicle sensing; online driving behavior detection; mobile devices.

Sumário

Lista de figuras

Lista de tabelas

1
Introdução

Este capítulo apresenta a motivação para a tese, os objetivos do trabalho e como a tese está organizada.

1.1
Motivação

Dirigir é uma tarefa cotidiana que se tornou uma necessidade da sociedade moderna, principalmente nas grandes cidades e, segundo Owsley [1], está associada à qualidade de vida das pessoas em alguns casos. Contudo, devido ao comportamento imprudente – definido por Tasca [2] como um comportamento que deliberadamente aumenta o risco de colisão e é motivado por impaciência, aborrecimento, hostilidade ou uma tentativa de poupar tempo –, há um número crescente de acidentes no trânsito. Nesta linha, de acordo com o relatório global de segurança no trânsito da Organização Mundial da Saúde [3], 1,24 milhão de pessoas morrem anualmente no trânsito e estima-se que de 20 a 50 milhões de pessoas envolvem-se em acidentes não fatais. Além disso, estima-se que os recursos financeiros aportados globalmente para lidar com as consequências dos acidentes girem em torno de $ 518 bilhões de dólares [4]. Contudo, estudos indicam que os motoristas tendem a ser relativamente mais prudentes quando são monitorados ou quando são fornecidos comentários (*feedback*) a respeito das manobras [5], [6].

No entanto, os Sistemas de Transportes Inteligentes (*Intelligent Transportation Systems* – ITS) atuais ainda contam com uma infraestrutura composta por sensores e câmeras estacionários instalados em estradas, impossibilitando a coleta, agregação e análise dos dados, especialmente em tempo real [7]–[9], durante todo um percurso. Além disso, devido ao alto custo de instalação e manutenção, são geralmente restritos à certas vias ou seções de uma via [9]–[11]. Em contraste, a Internet das Coisas (*Internet of Things* – IoT) objetiva conectar de forma pervasiva bilhões [12] de coisas ou objetos

inteligentes, tais como veículos, sensores, atuadores e *smartphones*. Tendo em vista os dados de sensores que podem ser coletados a bordo do veículo (e.g., velocidade, aceleração, desaceleração, rotação do motor e uso do acelerador), o objetivo desta tese é detectar anomalia no fluxo de dados gerados por estes sensores e realizar uma classificação *online* do estilo de condução do motorista.

No entanto, alguns dispositivos móveis utilizados na IoT possuem capacidade de armazenamento (memória disponível) e processamento limitados [13] e, geralmente, os algoritmos de detecção de anomalias são projetados para serem executados em computadores com grande poder computacional a partir de grandes *datasets*, constitui-se um desafio adaptá-los às restrições dos dispositivos [14], [15]. Este desafio torna-se ainda mais complicado em um cenário cujo fluxo de dados de múltiplos sensores (i.e., múltiplos fluxos de dados) precisa ser processado em paralelo, em um dispositivo com recursos limitados. Assim, várias soluções de IoT, incluindo as que se interessam por condução veicular segura, são projetadas e desenvolvidas para realizar o processamento de dados em um ambiente de computação na nuvem, devido à capacidade "ilimitada" destes ambientes em termos de armazenamento e processamento. Por exemplo, os autores em [16] propuseram uma solução para classificar o comportamento de condução coletando dados do veículo e encaminhando-os para processamento em um servidor remoto na nuvem. Já os autores em [17], [18], propuseram um *middleware* de computação na nuvem para a Internet dos Veículos (*Internet of Vehicle*). No entanto, devido ao grande volume do fluxo de dados gerado por alguns objetos inteligentes móveis (i.e., sensores ou dispositivos a bordo do veículo) torna-se oneroso transmiti-los para a nuvem. Além disso, a latência da rede e desconexões impossibilitam a realização da análise em tempo real do fluxo de dados.

Fluxo de dados é definido como uma sequência contínua de itens, que aparentemente não têm fim (i.e., *unbounded*), no qual não é possível controlar a ordem dos dados [19]. Uma característica do fluxo de dados é sua natureza dinâmica [20], esta característica pode ser avaliada a partir de dois pontos de vista distintos: no primeiro, como a distribuição dos dados pode mudar a medida que o fluxo de dados evolui (janelas temporais), torna-se um desafio identificar anomalias à medida que os dados chegam [20], [21]; no segundo, mudanças de contexto em um cenário móvel podem ocasionar mudanças nos dados disponíveis,

e portanto, é necessário que o algoritmo de descoberta de padrões [14], [22], [23] adapte-se a tais mudanças [24]. Por exemplo, em um cenário real, as leituras de sensores veiculares e dados disponíveis dependem do fabricante e do modelo do veículo [25]. Além disso, os dispositivos móveis não possuem o mesmo conjunto de sensores embarcados e sensores a bordo do veículo podem ser adicionados ou removidos a qualquer momento.

Neste contexto, a mineração do fluxo de dados móveis aparece como uma técnica chave [15] para análise em tempo real dos fluxos de dados gerados por sensores do próprio dispositivo móvel e por sensores que estão nas proximidades. No cenário abordado nesta tese, o dispositivo móvel é usado para receber e analisar os fluxos de dados dos sensores embarcados no veículo, tais como velocidade e rotação do motor, em vez de enviá-los diretamente para análise por um serviço na nuvem. Argumentamos que a análise do comportamento de condução, utilizando tais fluxos de dados, pode ser mapeada para o problema de *detecção de anomalia* em fluxos de dados. Este problema refere-se à encontrar padrões nos dados que não estão em conformidade com (ou desviam suficientemente de) comportamentos esperados [14], [26], por exemplo, mudanças repentinas de faixas e frenagens bruscas. Embora a detecção de anomalia tenha sido amplamente estudada, não têm havido muitos trabalhos de pesquisa na detecção de anomalia em fluxos de dados [15], [20], [21]. Até o momento, baseado em uma revisão quase-sistemática da literatura, não foram encontradas soluções para o problema da detecção *online* de anomalias em fluxos de dados de dispositivos móveis com limitação de recursos (em contraste com recursos virtualmente ilimitados do ambiente em nuvem) através do processamento de eventos complexos.

1.2 Definição do Problema

Segundo Hawkins [27], uma anomalia é uma observação que se desvia tanto das outras observações que desperta suspeitas de que tenha sido gerada por um mecanismo diferente. Uma anomalia geralmente contém informação útil sobre características anormais dos sistemas ou de uma entidade [28]. A detecção de anomalia é um campo de estudo multidisciplinar para a extração de padrões a

partir de grandes *datasets*[1], englobando um amplo espectro de técnicas tais como inferência estatística, aprendizado de máquina e mineração de dados [14], [23], [29]. Além disso, tem sido aplicada extensivamente em uma grande variedade de aplicações, como detecção de fraudes em operações com cartão de crédito, intrusão de redes, falhas em sistemas críticos, reconhecimento de fala e monitoramento de tráfego [14], [29]. Em todas estas aplicações, a maioria dos dados possuem um modelo "normal", e as anomalias são consideradas como desvios deste modelo [22].

Do grande número de pesquisas sobre detecção de anomalia, a maioria dos métodos existentes opera em todo o *dataset* para detectá-las [14], [30], foi projetado para lidar com a análise *offline* [26] de um grande volume de dados ou são ineficientes quando aplicados em fluxo de dados [21]. É desafiador adaptar as técnicas/algoritmos existentes de detecção de anomalias para realizarem o processamento dos múltiplos fluxos de dados nos próprios dispositivos móveis, uma vez que originalmente foram projetados e desenvolvidos para ambientes cujos recursos de memória e processamento são abundantes [19]. Além disso, tais técnicas/algoritmos são tratadas como uma "caixa-preta" [31], isto é, mudanças dificilmente podem ser feitas no processamento interno dos algoritmos. Recentemente, técnicas de descoberta de anomalias em fluxo de dados móveis tem sido adotada em várias aplicações emergentes, tais como *mobile crowd sensing*, *mobile activity recognition*, *intelligent transportation systems* e *mobile healthcare* [15], pela necessidade de processamento em tempo real do fluxo de dados gerados pelos sensores de dispositivos móveis e pelos sensores que estão nas proximidades destes. Neste contexto, a adaptação dos algoritmos clássicos de detecção de anomalias para operar sobre fluxo de dados de dispositivos móveis é um desafio, visto que: (i) a detecção em fluxo de dados está restrita a uma janela de tempo; (ii) acessos aleatórios aos dados dos sensores não são possíveis; (iii) o algoritmo precisa adaptar-se aos dados disponíveis e (iv) padrões devem ser descobertos preferencialmente com uma única análise sobre o fluxo dados [15], [19], [32]. Além disto, os autores em [14] destacam outros fatores que tornam o problema de detecção de anomalia um problema ainda mais complexo:

1 Esta tese considera um *dataset* um conjunto de dados armazenado na memória principal ou secundária que permita acesso randômico e cuja atualização dos dados ocorra fortuitamente – o que não ocorre em fluxo de dados.

- Geralmente, fluxos de dados gerados por sensores contêm ruído que, por sua vez, tende a ser semelhante às anomalias, tornando difícil distingui-lo ou removê-lo;
- É muito difícil definir uma distribuição dos dados que representa comportamentos normais. Além disso, o limite entre o comportamento normal e anormal, muitas vezes não é preciso. Assim, uma observação anômala que fica próxima da fronteira pode realmente ser normal, e vice-versa;
- A disponibilidade de *datasets* para treinamento e validação é normalmente um grande problema;
- A noção exata de anomalia é diferente dependendo do domínio de aplicação e a formulação de detecção de anomalia é induzida tanto pela natureza dos dados como pela disponibilidade de dados.

As técnicas atualmente utilizadas para avaliação do estilo de condução, em sua grande maioria, utilizam modelos/técnicas (e.g., modelos baseados em redes neurais, teoria fuzzy e modelos ocultos de Markov) com uma boa acurácia [33]. Contudo, não foram projetados para o processamento de fluxo de dados [34], e de acordo com [33], [35], [36]: redes neurais possuem baixo desempenho de processamento em tempo real, impossibilita entender o comportamento físico por se tratar de uma "caixa-preta" e como possui método empírico de ajuste dos parâmetros (e.g., número de camadas), pode ter uma fase de treinamento longa. Modelos de Markov requerem hipóteses artificiais e teoria fuzzy requer um conhecimento prévio para geração de regras. Além disso, estes modelos/técnicas de identificação do comportamento do motorista são estáticos [33], ou seja, seus parâmetros são pré-fixados. Por exemplo, redes neurais requerem um número fixo de parâmetros de entrada [33], [36]. Contudo, as situações de condução (que são influenciadas pelo estado psicológico do motorista, por fatores do trânsito e clima) são dinâmicas e nem sempre todas as informações que uma técnica ou algoritmo necessitam como entrada estarão disponíveis.

1.3 Justificativa

Em diversas aplicações, tal como salientado na Seção 1.2, o conceito de fluxo de dados é mais apropriado que o conceito de *dataset.* Como salientado por Kontaki *et al*. [37], existem poucos trabalhos de pesquisa que estudam o problema da detecção de anomalia em fluxo de dados, tais como [19], [32], [37]–[39]. No entanto, nestes trabalhos, os algoritmos ou técnicas não foram projetados para serem executados em dispositivos cujo poder de processamento e memória são limitados.

Dado o exposto, esta tese apresenta uma abordagem leve (*lightweight*), para detecção de anomalias em fluxo de dados gerado pelos sensores de dispositivos móveis e de qualquer fonte de dado que esteja nas proximidades, por meio do processamento de eventos complexos (*Complex Event Processing* – CEP). CEP é um conjunto de técnicas e ferramentas que fornece um modelo de processamento assíncrono para a detecção *online* de situações de interesse [40], por exemplo, uma manobra agressiva de um motorista. Estas técnicas fornecem um rico conjunto de conceitos e operadores para o processamento de eventos, que inclui uma linguagem de consulta denominada *Continuous Query Language* (CQL) [41], [42], regras e primitivas de transformação de eventos tais como agregação, filtragem, detecção de padrões e produção de eventos derivados. No CEP, os eventos são continuamente processados, analisados, manipulados e então, eventos derivados são enviados para os consumidores de eventos. Por meio destes conceitos, é possível desenvolver módulos de processamento independentes responsáveis por parte da lógica de processamento de um algoritmo.

1.4 Questões de Pesquisa

Como discutido na Seção 1.2, o problema de pesquisa desta tese é adaptar *algoritmos clássicos de detecção de anomalias*, originalmente concebidos para processamento *offline* e em computadores com boa capacidade de processamento e memória, para que possam processar *online* e em dispositivos móveis. Apesar dos avanços tecnológicos, tais dispositivos ainda possuem poder de processamento e/ou armazenamento restritos. Sendo assim, as seguintes questões

de pesquisa são levantadas sobre a detecção de anomalias no paradigma de *Complex Event Processing*:

- Como adaptar os algoritmos de detecção de anomalias para o paradigma CEP, provendo uma acurácia aceitável?
- Como os algoritmos devem adaptar-se às mudanças no fluxo de dados?

A partir destas indagações, um conjunto de questões subjacentes é colocado para discussão:

- Qual o melhor algoritmo em termos de acurácia?
- Qual o melhor algoritmo em termos de precisão?
- Qual o melhor algoritmo em termos de cobertura?
- Qual o melhor algoritmo em termos de medida F?
- Qual o melhor algoritmo em termos de tempo médio de execução?
- Qual o melhor algoritmo em termos de consumo de memória?
- Qual o melhor algoritmo em termos de consumo de processador?

A partir do problema definido na Seção 1.2 e das questões de pesquisa elencadas, faz-se necessária a elaboração de uma suposição passível de investigação dentro do escopo desta tese. A suposição em questão é: Algoritmos procedurais de detecção de anomalia podem ser adaptados para o paradigma de *Complex Event Processing*, permitindo análise em tempo real do fluxo de dados de dispositivos móveis com pouca capacidade de processamento e memória.

Para investigar esta suposição, três algoritmos de detecção de anomalia clássicos foram adaptados, testados e avaliados. Destes, dois são baseados em inferência estatística (Z-Score e Box Plot) e foram escolhidos por serem evidentemente eficientes [29] e exigirem um processamento relativamente simples. O terceiro algoritmo é baseado em clusterização (K-Means) e foi escolhido por ser uma técnica extremamente eficiente para agrupar dados através da identificação de padrões ocultos [43].

1.5 Objetivos e Contribuições

De acordo com os pressupostos, suposição e questões de pesquisa apresentados anteriormente, o objetivo desta tese é: propor uma abordagem

lightweight baseada em CEP, para detecção *online* de anomalias em múltiplos fluxos de dados dinâmicos (i.e., não há garantia de que todos os dados sempre estarão disponíveis) de dispositivos móveis, com capacidades limitadas de processamento e armazenamento – contrastando com ambientes de nuvem.

Dado que o processamento (detecção *online* de anomalias) é realizado em um dispositivo móvel, esta abordagem proporciona os seguintes benefícios: (i) processamento dos dados dos sensores com pouca centralização se comparado com o ambiente de nuvem, (ii) economia de largura de banda, (iii) capacidade de encontrar anomalias mesmo com o dispositivo móvel desconectado, (iv) diminuição da necessidade de armazenamento de dados e (v) mecanismo para identificação de anomalias em dispositivo com poucos recursos de processamento e memória.

Mais especificamente, as contribuições da tese são:

- Implementação e avaliação da abordagem proposta, isto é, desenvolvimento de um protótipo, na plataforma *Android*, que realiza a identificação *online* de anomalias em um ambiente real a partir de dados provenientes dos sensores do *smartphone* do motorista e de um dispositivo OBD, e classifica os motoristas em prudentes e imprudentes;
- Criação e disponibilização de um *dataset* com dados significativos da condução dos motoristas.

1.6 Cenário Motivador

Como salientado da Seção 1, os sistemas de transportes inteligentes ainda contam com uma infraestrutura composta por sensores estacionários instalados em determinados segmentos de estradas e avenidas. Esta característica impossibilita a coleta, agregação e análise dos dados em tempo real [7]–[9] durante todo um percurso. Na tentativa de superar tais limitações, os fabricantes de automóveis desenvolveram produtos que auxiliam o motorista no processo de condução, os chamados *Advanced Driver Assistance Systems* – ADAS. Alguns ADAS, como o *Automatic Crash Notification System* da BMW e o *OnStar* da GM, obtêm dados do veículo a partir de sensores ou dispositivos embarcados (e.g., câmeras e sensores do controle de estabilidade) para a prevenção e detecção de colisões

(e.g., sensores de impacto para ativação dos *airbags*), direção assistida [44], [45] e geração *offline* de relatórios de condução, como é o caso do Chevrolet Malibu 2016 [46]. Contudo, os ADAS normalmente são disponibilizados apenas em veículos de alto padrão e mais novos, ficando assim, inacessíveis para a maioria dos motoristas [47]–[49], mesmo em países desenvolvidos. Além disso, a instalação em veículos usados é inviável ou extremamente cara. E quando os ADAS se tornam obsoletos, a atualização ou troca por sistemas mais novos e eficientes é uma tarefa difícil [49], como foi o caso do *OnStar* que perdeu 500 mil assinantes devido ao equipamento de comunicação analógico não ser compatível com a nova infraestrutura de comunicação digital [49].

Em contrapartida, há trabalhos que propõe o uso de *smartphone* para avaliar o estilo de condução de motoristas [5], [25], [45], [50]–[54]. O uso de *smartphones* para avaliar o estilo de condução deve-se a sua ampla disseminação, boa capacidade de armazenamento e processamento, bem como variedade de sensores embarcados. Além disso, *smartphones* possuem uma variedade de interfaces sem fio (e.g., Bluetooth LE, Wif-Fi Direct, NFC e ANT+), o que possibilita a comunicação direta com outros dispositivos a bordo do veículo (e.g., dispositivos de diagnóstico OBD-II, câmeras e dispositivos vestíveis) [49]. Assim, o *smartphone* pode atuar como um centro (*hub*) de processamento dos dados dos diferentes sensores/dispositivos a bordo do veículo para classificar o estilo de condução. Por exemplo, usando o Bluetooth LE, um *smartphone* é capaz de se conectar e receber dados de sensores (e.g., velocidade, rpm e posição do acelerador) embarcados no veículo usando o padrão OBD-II e combiná-los, através de regras CEP, com os dados de sensores do próprio *smartphone* (tal como direção e aceleração). Por fim, os *smartphones* permitem desenvolver sistemas independentes do veículo, ubíquos, e provêm um conjunto diversificado de dados para a análise do estilo de condução.

Pelo exposto, acreditamos que a avaliação do estilo de condução do motorista pode se beneficiar de uma abordagem detecção *online* de anomalias em múltiplos fluxos de dados cuja unidade de processamento é um dispositivo móvel com poder de processamento e/ou armazenamento limitado.

1.7 Delimitação de Escopo e Pressupostos da Tese

A noção de anomalia depende do domínio da aplicação, e esta tese dedica-se a identificar anomalias nos fluxos de dados relacionados à detecção do estilo de condução de motoristas. Além disso, a tese considera os seguintes pressupostos:

- A maior parte das instâncias pertencentes ao fluxo de dados são normais. Apenas uma pequena parcela dos dados são anômalos [55];
- "Dados anômalos" são estatisticamente diferentes dos "dados normais" [2];
- O consumo de energia do dispositivo móvel não é um requisito crítico, visto que em um veículo o *smartphone* pode ser facilmente carregado quando necessário.

Contudo, vale ressaltar que os dois primeiros pressupostos se completam. Considerando apenas primeiro pressuposto, temos que "dados anômalos" podem representar comportamentos semelhantes daqueles relativos aos "dados normais". Já o segundo pressuposto afirma que dados anômalos representam um comportamento completamente diferente daquele descrito pelos "dados normais".

1.8 Organização da Tese

O restante do documento é organizado da seguinte forma:

- O Capítulo 2 apresenta a fundamentação teórica que fornece uma visão geral sobre os conceitos sobre detecção de anomalias, processamento de eventos complexos e sobre a avaliação do estilo de condução;
- O Capítulo 3 apresenta os principais trabalhos relacionados a esta teste;
- O Capítulo 4 descreve os detalhes da proposta para detecção *online* de anomalia baseada em CEP;
- O Capítulo 5 expõe o planejamento do estudo de caso realizado;
- O Capítulo 6 explica a preparação e execução do estudo de caso, expondo as avalições e comparações com abordagens relacionadas;

- O Capítulo 7, por fim, revisa e discute as ideias centrais apresentadas na tese e propõe linhas de trabalhos futuros, além de listar os resultados obtidos.

2
Fundamentação Teórica

Este capítulo apresenta os principais conceitos sobre processamento de eventos complexos, detecção de anomalia e avaliação do estilo de condução para uma melhor compreensão do trabalho descrito nesta tese.

2.1
Detecção de Anomalia

Detecção de anomalia normalmente assume que anomalias nos dados são raras quando comparadas com instâncias normais e quando anomalias ocorrem, desviam significativamente do restante da amostra [29]. No entanto, "significativamente" constitui um julgamento subjetivo que pode ou não classificar uma instância como anômala. Por exemplo, analisando o padrão de agrupamento principal na Figura 1 (a) e (b) nota-se que são bastante parecidos, embora existam diferenças significativas fora destes grupos principais. Na Figura 1 (a) o ponto *A* claramente desvia significativamente do restante dos pontos e, portanto, é um ponto anômalo. No entanto, na Figura 1 (b) é bem mais subjetivo, visto que o ponto *A* está em uma região esparsa de dados. Assim, torna-se mais difícil afirmar com confiança que este dado difere significativamente dos demais pontos. É bastante provável que este ponto de dado represente ruído distribuído aleatoriamente. Isso porque o ponto *A* parece ajustar-se a um padrão representado por outros pontos distribuídos aleatoriamente.

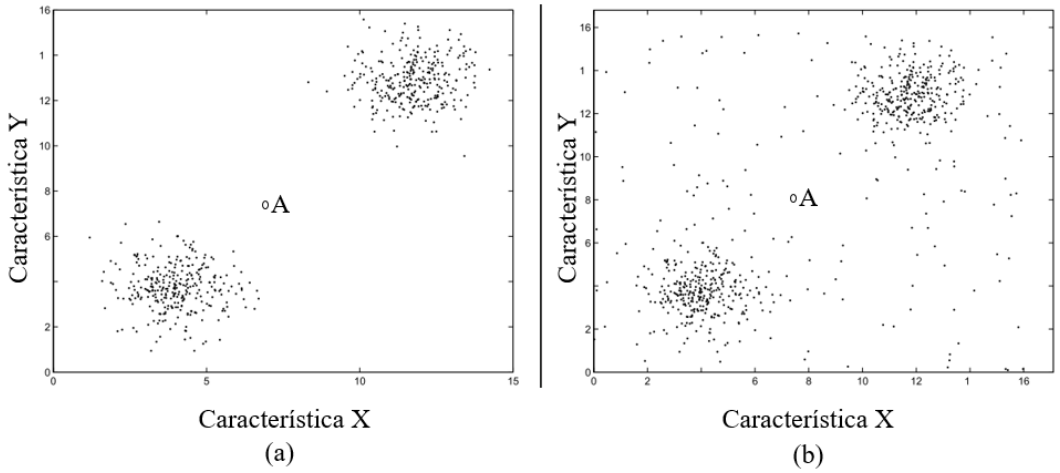

Figura 1. Diferença entre ruído e anomalia. Adaptado de [22].

Portanto, embora a detecção/remoção do ruído seja importante, nem sempre é possível classificar uma instância (dado) como *normal*, *ruído* ou *anomalia* precisamente como exemplificado na Figura 2, e escolhas dependem de critérios específicos de cada aplicação. Dessa forma, o ruído pode ser interpretado como o limite semântico entre as instâncias normais e anômalas [22]. Assim, alguns autores [56], [57] utilizam o termo *anomalia fraca* (ruído) e *anomalia forte* para caracterizar essa diferença. Devido à dificuldade de classificação supracitada, nesta tese o termo anomalia refere-se a uma instância que pode ser considerada uma anormalidade ou ruído.

A maioria dos algoritmos de detecção de anomalias usam uma medida (pontuação) para mensurar o nível da anomalia (i.e., *outlierness*), tal como vizinho mais próximo, *clustering* ou inferência estatística [14], [22], [23], [29]. E nesses algoritmos, anomalias normalmente recebem uma pontuação muito maior do que os ruídos [22]. Os autores em [14] salientam que embora as técnicas de *clustering* e vizinhos mais próximos sejam abordagens semelhantes, por computarem a distância entre todos os pares de instâncias de dados, há uma diferença fundamental – as técnicas baseadas em *clustering* avaliam cada instância em relação ao cluster ao qual pertencem, enquanto que as técnicas baseadas em vizinhos mais próximos, analisam cada instância em relação à sua vizinhança.

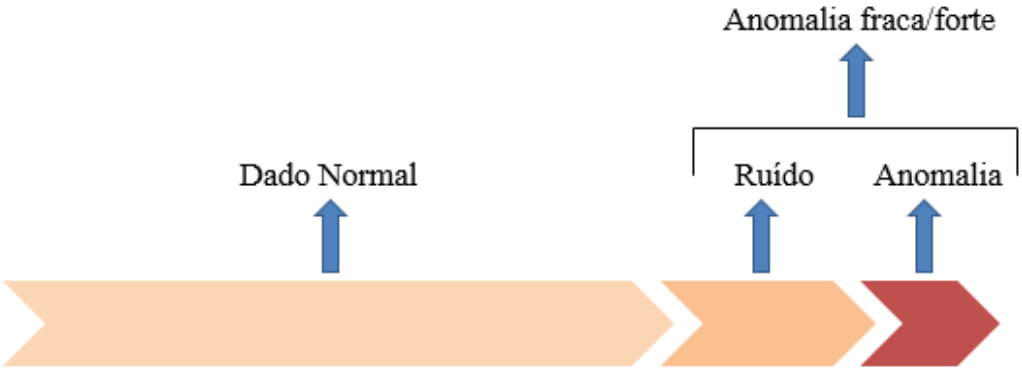

Figura 2. Classificação ideal dos dados. Adaptado de [22].

Abordagens *estatísticas* para a detecção de anomalia assumem que instâncias normais ocorrem em regiões de alta probabilidade de um modelo estocástico, enquanto anomalias ocorrem em regiões de baixa probabilidade [14], [58]. Como as abordagens foram inicialmente projetadas para detectar anomalia em *datasets*, estas regiões são estáticas. Contudo, em fluxo de dados estas regiões são dinâmicas, ou seja, a cada janela de tempo tem-se novas regiões. O algoritmo *Standard Score* (ou Z-Score) é uma técnica estatística que possibilita identificar anomalias com uma única análise sobre o fluxo dados. O Z-Score torna diferentes tipos de dados comparáveis e de fácil interpretação [59]. Contudo, foi projetado para detectar anomalia em *datasets* unidimensionais. O Z-Score descreve a localização da pontuação (*raw score's location*) de uma instância de dado (bruto) em termos de quão longe acima ou abaixo da média está, medido em unidades do desvio padrão [59]. Um Z-Score igual a zero significa que a instância de dado bruto é igual a média. O Z-Score é calculado como mostrado na equação (1), onde Z é o Z-Score de uma instância, *X* representa o valor da instância, μ representa a média da amostra e σx representa o desvio padrão da média. Este cálculo estabelece uma pontuação sem unidade de medida já que não está relacionada a unidade original do dado (e.g., km/h ou m/s²). O Z-Score mede a disparidade em número de unidades de desvio padrão e, por conseguinte, pode ser mais facilmente utilizado para comparações [60] dos dados.

$$Z = \frac{X - \mu}{\sigma x} \tag{1}$$

De acordo com os autores em [59], Z-Score possui basicamente dois componentes: (i) um sinal, positivo ou negativo, indicando se a pontuação bruta

está acima ou abaixo da média, e (ii) o valor absoluto Z-Score indicando a distância em relação à média. Segundo os autores em [14], [22], [29], uma boa regra geral considera todas as instâncias de dado cujos módulos do Z-Score é maior que 3 como anomalias. Isso porque em uma distribuição gaussiana a região compreendida entre [μ e ±3σx] contém 99.7% das instâncias [14], [29]. Após computar o Z-Score para cada instância, o algoritmo verifica a distribuição dos Z-Scores (*Z-Distribution*), isto é, a frequência relativa dos scores brutos de uma população ou amostra. A Figura 3 mostra a *Z-Distribution* normal perfeita (a.k.a curva normal padrão). É possível notar que 50% dos *scores* estão abaixo da média, 50% estão acima da média, aproximadamente 68% da distribuição está entre ± 1 (σx) a partir da média e menos de 1% da distribuição dos *scores* são maiores que $|\pm 3|$. Se estes Z-Scores fossem obtidos a partir de dados de condução, isto significaria que na maior parte do tempo o motorista manteve um comportamento sem mudanças abruptas de velocidade ou direção. Nos casos em que detectou-se anomalias, o motorista pode ter realizado manobras evasivas para evitar acidentes ou de fato comportou-se de forma imprudente, mas o número de instâncias anômalas é insuficiente para considerá-lo um motorista imprudente. O ponto forte do Z-Score reside no fato de que não requer parâmetros de configuração e anomalias são descobertas com uma passagem sobre o fluxo de dados. No entanto, o Z-Score pode ter sua capacidade de identificar anomalia com precisão comprometida caso haja poucas instâncias disponíveis [29].

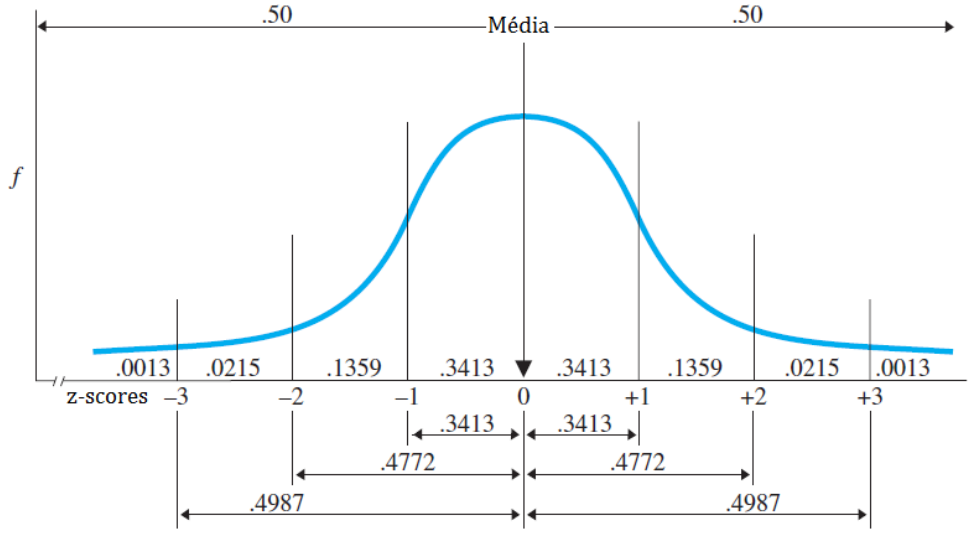

Figura 3. Frequência dos Z-Scores. Adaptado de [59].

Box Plot é a abordagem que produz uma representação gráfica para identificar anomalias, é provavelmente a técnica estatística mais simples para detectar anomalia em *datasets* univariados e multivariados que não faz suposições sobre o modelo de distribuição de dados [14], [29]. É usado para avaliar e comparar (principalmente pontos extremos ou anômalos), por exemplo, o perfil de uso do acelerador de motoristas. O Box Plot tornou-se a técnica padrão para apresentar graficamente os dados através de um sumário de 5 números (*5-number summary*), que consiste na menor observação não anômala (*min*), quartil inferior (*Q1*), mediana (*Q2*), quartil superior (*Q3*) e maior observação não anômala (*max*). Por fim, o intervalo interquartil (*Interquartile Range – IQR*) – a diferença entre Q3 e Q1 – é calculado. Isto significa que 25% das observações são menores do que o primeiro quartil, 50% são menores do que o segundo quartil e 75% são menores do que o terceiro quartil. Anomalias são os pontos além da menor e maior observação não anômala [29]. Laurikkala, Juhola e Kentala [61] sugerem uma heurística de (1.5 x IQR) para determinar o ponto a partir do qual se consideram anomalias. Entretanto, de acordo com autores em [29], tal heurística precisaria ser melhor avaliada de acordo com os diferentes *datasets*. O Box Plot típico pode ser visto na Figura 4.

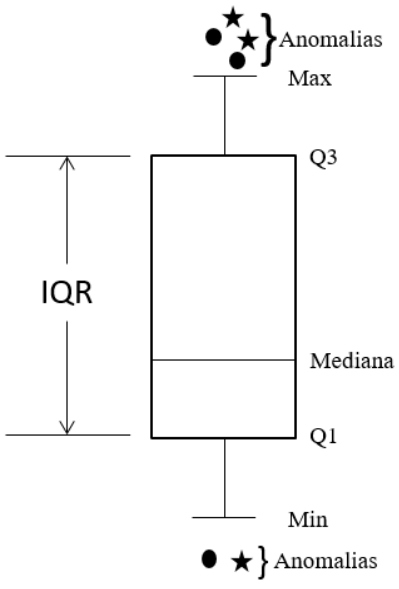

Figura 4. Gráfico Box Plot.

A abordagem para detecção de anomalias baseada em *Clustering* [62] é uma técnica exploratória de análise de dados em que um conjunto de dados de entrada, normalmente multidimensionais, são classificados em grupos, isto é, *clusters* de

dados similares. Além disso, é essencialmente uma técnica não supervisionada precedida por uma fase de treinamento e teste curta e semi-supervisionada [28], [63] usada para identificar anomalias. Técnicas que operam em um modo semi-supervisionado presumem que apenas as instâncias de dados rotuladas como normal estão disponíveis para treinamento [14]. As abordagens baseadas em *Clustering* utilizam uma medida de distância, tal como distância Euclidiana, para mensurar a distância entre uma instância e a centroide do *cluster*. Tais abordagens podem ser divididas em três grupos segundo [14]. A primeira categoria baseia-se no seguinte pressuposto: *instâncias normais pertencem a um cluster, enquanto anomalias não pertencem a cluster algum*. Estas técnicas aplicam um algoritmo de *clustering* conhecido sobre um *dataset* e qualquer instância que não pertença a algum *cluster* é considerada anômala. Exemplos desses algoritmos são o DBSCAN [64], ROCK [65], SNN [66]. Contudo, o objetivo central destas técnicas é encontrar *clusters*, e portanto, não são otimizadas para achar anomalias. A segunda categoria baseia-se no seguinte pressuposto: *instâncias normais encontram-se próximas do centroide do cluster mais próximo, enquanto anomalias encontram-se distantes do centroide do cluster mais próximo*. Vários algoritmos foram propostos como *Self-Organizing Maps* (SOM) [67], *K-Means* [68] e *Expectation Maximization* (EM) [69]. Estes algoritmos podem operar em um modo semi-supervisionado, em que os dados de treinamento são agrupados e as instâncias pertencentes aos dados de teste são comparadas com os *clusters* para obter uma pontuação de anomalia para cada instância de dado de teste [70].

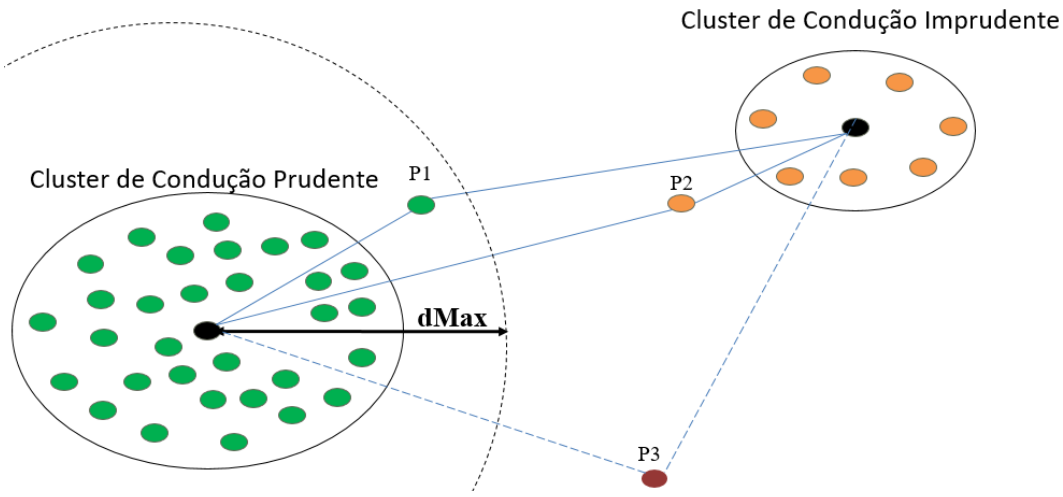

Figura 5. Abordagem de clusterização baseada em distância.

Seguindo os pressupostos dos algoritmos das categorias supracitadas, considerando dois *clusters*, como mostrado na Figura 5, os pontos P1, P2 e P3 são considerados anomalias, uma vez que não pertencem a nenhum *cluster* (categoria 1). Já baseado no pressuposto da segunda categoria, como P1 está próximo do centroide do *cluster* de condução prudente, é considerado um dado normal. Já P2 está próximo do centroide do *cluster* de condução imprudente, portanto, é considerado uma anomalia. Contudo, como a distância de P3 para ambos centroides é maior que *dMax*, P3 é considerado uma instância anômala. Esta abordagem possui dois problemas. O primeiro é determinar o valor para *dMax* – o analista precisa analisar o parâmetro de forma experimental e analisar os resultados [71]. O segundo problema reside fato que pode ser extremamente oneroso coletar e rotular dados anormais [29]. Por exemplo, a coleta de dados que representem um comportamento de condução imprudente pode ser perigosa, pois pode causar de fato, acidentes de trânsito. Assim, um algoritmo de *clustering* deve ser capaz de identificar anomalias mesmo com poucas instâncias de dados que representam comportamento de condução imprudente.

É possível observar que na Figura 5 anomalias formaram um pequeno grupo (*clusters*). As técnicas acima discutidas não serão capazes de detectar tais anomalias caso não se tenha um *dataset* rotulado para treinamento e testes. Para abordar esta questão, há uma terceira categoria de técnicas baseadas no seguinte pressuposto: *instâncias normais pertencem a clusters grandes e densos, enquanto anomalias pertencem a clusters pequenos e esparsos.* Algoritmos que baseiam-se nesse pressuposto consideram instâncias pertencentes a *clusters* cujo tamanho ou densidade estejam abaixo de um limiar como anômala. Breunig [72] introduziu o conceito de "anomalia local" *(local outliers*), que são instâncias periféricas em relação às suas vizinhanças locais, particularmente com respeito às densidades da vizinhança. A pontuação de uma instância anômala é conhecida como *Local Outlier Factor* (LOF). Dada uma instância, a pontuação LOF é igual à razão entre a média da densidade local dos k-vizinhos mais próximos e a densidade local da própria instância. Embora o conceito de anomalias locais seja útil, a computação do LOF para cada instância requer um grande número de buscas dos k-vizinhos mais próximos e pode ser computacionalmente cara [19]. Aggarwal e Yu [57] discutiram uma técnica para detecção de anomalia que encontra instâncias

anômalas observando a distribuição da densidade dos dados. Isto é, a definição de anomalia considera uma instância anômala se a densidade local dos dados é excepcionalmente inferior à densidade média. Embora o algoritmo possua uma medida de distância simples, possui uma computação onerosa [19], assemelhando-se a força bruta [57].

Nesta tese três algoritmos clássicos de detecção de anomalia foram adaptados para o monitoramento contínuo do fluxo de dados, os quais são discutidos nas secções 4.1, 4.2 e 4.3. Escolhemos o Z-Score por ser uma técnica simples que requer uma única análise ao longo do fluxo de dados. O Box Plot foi escolhido também pela simplicidade na identificação de anomalias e por não pressupor uma distribuição dos dados, como é o caso do Z-Score que pressupõe uma distribuição Gaussiana [14]. O último é o K-Means – um algoritmo de *clustering*, baseado na computação de distância, popular e simples que minimiza o erro de agrupamento [73].

2.2 Processamento de Eventos Complexos

Processamento de eventos complexos (*Complex Event Processing* – CEP) é um conjunto de técnicas e ferramentas que proveem um modelo de processamento em memória, sobre fluxo de dados assíncrono, em tempo real (i.e., atraso mínimo) para detecção de situações de interesse [40], [74], [75]. De acordo com Luckham [40], CEP oferece: (i) *ciência de situação* através da utilização de consultas contínuas que correlacionam dados/eventos de diferentes fluxos, (ii) *ciência de contexto* pela incorporação de dados de contexto de um domínio específico (e.g., velocidade máxima de uma via específica) e pela segmentação de fluxos de dados em diferentes visões, tais como janelas temporais ou partições e (iii) *flexibilidade* uma vez que é possível especificar regras de processamento de eventos a qualquer momento, isto é, regras podem ser adicionadas ou removidas enquanto o sistema está em execução (*on-the-fly*).

O conceito central do CEP é a sua linguagem declarativa de processamento de eventos para expressar regras de processamento de eventos (consultas contínuas e especificação de padrões de eventos). Estas regras são baseadas na tríade evento-condição-ação, e usam operadores (e.g., lógicos, numéricos, de

contagem, temporais e causais) sobre os eventos de entrada, permitindo filtrar, classificar, identificar correlações entre eventos e ocorrência de padrões causais ou temporais específicos etc. CEP provê também mecanismos para *Event Pattern Matching*, isto é, a partir de centenas ou mesmo milhares de eventos recebidos por segundo, identificar padrões de eventos significativos para o domínio da aplicação [76].

A maioria dos sistemas CEP possuem o conceito de Agentes de Processamento de Eventos (*Event Processing Agents* – EPAs). EPAs são módulos de *software* que implementam parte da lógica de processamento, encapsulando alguns operadores e regras CEP. Basicamente, um EPA filtra, separa, agrega, transforma e sintetiza novos eventos complexos a partir de eventos simples. Uma manobra imprudente (e.g., uma curva a alta velocidade) é um exemplo de um evento complexo, pois é baseado na composição de vários eventos primitivos, tais como variação significativa em um dos eixos (e_x, e_y, e_z) do acelerômetro, alta velocidade e ângulo de esterçamento do volante.

Uma rede de processamento de eventos (*Event Processing Network* – EPN) é uma rede de EPAs interconectados que implementam a lógica de processamento global de eventos [41]. Em uma EPN, os EPAs estão conceitualmente ligados uns aos outros (ideia de subscrição), ou seja, os eventos de saída de um EPA são transmitidos e tratados posteriormente por outros EPAs, independentemente do tipo específico de mecanismo de comunicação subjacente para a disseminação dos eventos [41]. Isso sugere a adoção de uma arquitetura distribuída para o CEP como um conjunto de *brokers* de eventos (ou EPAs) conectados em uma rede *overlay* (EPN), que implementa funções de roteamento e encaminhamento especializadas com escalabilidade como sua principal preocupação [77], [78]. A Figura 6 mostra a caracterização de uma EPN.

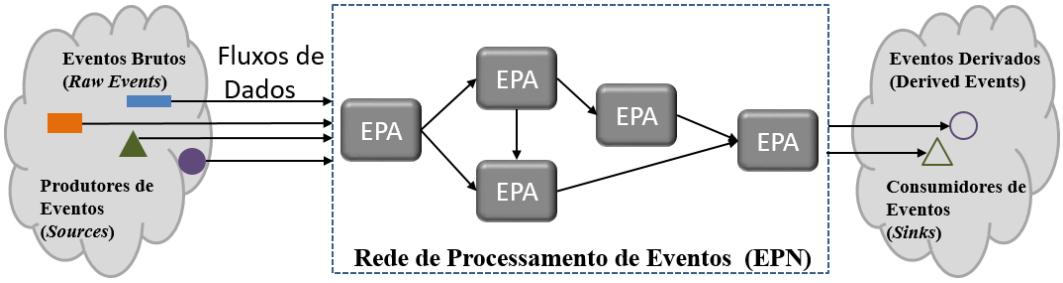

Figura 6. Caracterização típica de uma EPN.

O CEP trabalha com o conceito de janela de tempo (*Time Window*) – uma janela de tempo é um contexto temporal que define quais porções do fluxo de entrada são considerados durante a execução da regra [78], por exemplo, somente os eventos gerados nos últimos 30 segundos [76]. Os modelos de janela de tempo mais comuns são as janelas em lote (*time batch window*) e as janelas deslizantes (*sliding time window*). Com a janela temporal em lote, o CEP suporta o processamento em lote armazenando (*buffering*) todos os eventos produzidos durante um intervalo de tempo Δ e então, aplica as consultas e operadores em todo o conjunto de eventos. A janela deslizante é uma janela de lote em movimento que armazena somente os eventos pertencentes ao último intervalo de tempo Δ, ou seja, a cada novo evento aplica-se as consultas e operadores no conjunto de eventos que chegaram no último intervalo de tempo Δ. A Figura 7 exibe esses comportamentos.

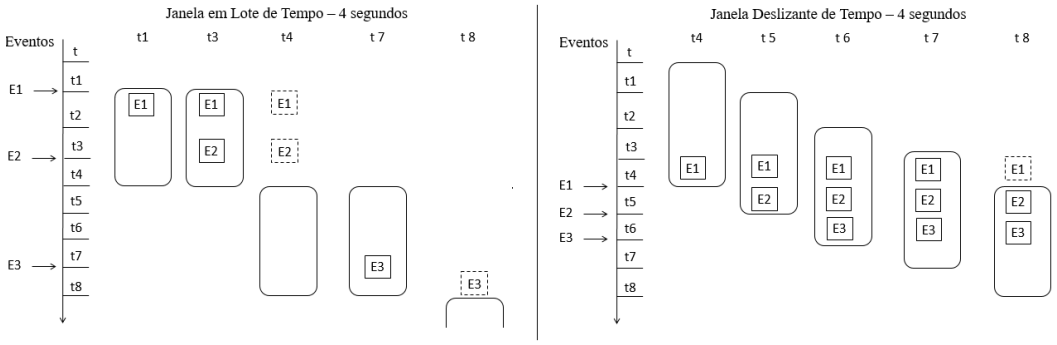

Figura 7. Exemplo do funcionamento das Janelas de Tempo. Adaptado de [79].

Um sistema que analisa o comportamento do motorista pode lidar com informações de vários provedores de dados, isto é, sensores tais como velocidade, rpm, posição do acelerador, acelerômetro e giroscópio. Consequentemente, os fluxos de dados recebidos podem ser muito heterogêneos. Considerando que CEP se concentra em encontrar padrões entre eventos heterogêneos que refletem certas atividades/situações do mundo real e fornece um conjunto de primitivas e conceitos tais como EPA, EPN, mecanismo para detecção de padrão e linguagem de consulta contínua, torna-se possível escrever algoritmos que se adaptam aos dados disponíveis, por meio de regras CEP e pela composição dos EPAs.

2.3
Avaliação do Estilo de Condução

Na maioria dos trabalhos de detecção do estilo de condução, as características de condução do motorista geralmente são baseadas em: (i) movimentos da cabeça, dos olhos e expressões faciais para detectar fadiga, sonolência ou distração [80], [81]; (ii) sinais fisiológicos que também são usados para detectar fadiga ou sonolência a partir da análise dos sinais de atividade do cérebro, coração ou contração muscular [82], [83]; e (iii) interações motorista-veículo, analisando, por exemplo, dados de velocidade, rotação por minuto, posição do acelerador, aceleração, desaceleração e esterço da direção (*steering*) [16], [84], [85].

Há ainda trabalhos baseados em questionários para identificar a personalidade e estilos de condução gerando conhecimento sobre risco iminente de acidentes [86]. Os que visam avaliar o comportamento imprudente são o *Driving Behavior Scale* [87], o *Driving Vengeance Questionnaire* [88], o *Driving Anger Scale* [89], o *Driving Anger Expression Inventory* [90], e, o mais popular deles, o *Manchester Driving Behavior Questionnaire* (DBQ) [91]. Apesar dos questionários serem simples, possuírem baixo custo e rápida aplicação, não são ideais para sistemas de avaliação automático e *online*, visto que podem gerar viés por serem respondidos à *posteriori* e por dependerem da memória dos motoristas [52].

Um mecanismo alternativo para avaliação do estilo de condução é o uso de simuladores. Nesta abordagem, o motorista controla o veículo em um ambiente simulado, que se constitui em uma forma natural e segura para avaliar o estilo de condução. No entanto, embora seja possível criar situações de risco/estresse, simuladores são utilizados com menos frequência para fornecer *feedback* em tempo real durante a condução por serem caros [52]. Alternativamente, em alguns países, a avaliação é realizada através da análise de especialistas. Neste caso, um avaliador certificado avalia e fornece *feedback* em “tempo real” sobre as habilidades e os estilos de condução incorporando dados de contexto, isto é, condição do trânsito e clima. Devido ao alto custo e por requerer entrada de dados do avaliador (o que pode gerar um viés), este tipo de avaliação normalmente só é realizada para verificar se o motorista está apto a dirigir [52].

Portanto, espera-se que a tecnologia de sensoriamento a bordo do veículo aliada à mobilidade e capacidade de coleta, processamento e armazenamento dos dispositivos móveis, ofereça uma opção ao desenvolvimento de técnicas e ferramentas para avaliação do estilo de condução em tempo real, de baixo custo, flexíveis e acessíveis à maioria dos condutores. No entanto, quaisquer sensores, incluindo os dos dispositivos móveis e embarcados no próprio veículo, normalmente enviam uma grande quantidade de eventos (mensagens), gerando um fluxo de dados que deve ser processado de maneira confiável e segura. Arquiteturas orientadas a eventos (*Event-Driven Architectures* – EDAs) tais como o CEP, foram propostas e desenvolvidas para lidar com grandes quantidades de fluxos de dados. Estas arquiteturas aparecem como um novo paradigma [40] para a mineração e processamento de fluxos de dados *online* com o mínimo de atraso. EDAs já são amplamente utilizadas e desempenham um papel importante em aplicações tais como logística, diagnósticos, segurança de rede, detecção de fraude bancária e monitoramento de tráfego [92].

O Capítulo 3 explana sobre os trabalhos relacionados. Inicialmente expomos trabalhos sobre detecção *online* de anomalias em fluxos de dados e em seguida, trabalhos que utilizam dispositivos móveis para classificar o estilo de condução do motorista.

3 Trabalhos Relacionados

Detecção de anomalia abrange aspectos de um amplo espectro de técnicas tais como estatística, aprendizado de máquina e mineração de dados [14], [22], [29]. A maior parte das técnicas de detecção de anomalias existentes resolvem uma formulação específica do problema. Por exemplo, Cui [19] propõe um algoritmo para detecção *online* de anomalia em fluxo de dados objetivando a detecção de intrusão em redes. Para isso, utilizou uma abordagem de *clustering* baseada em grade (*grid-based clustering*) que mantém apenas um resumo dos *clusters*, isto é, o conjunto de células pertencentes ao *cluster*, média dos dados em todas as dimensões e tamanho do *cluster*. Esta proposta considera *clusters* pequenos (esparsos) anômalos e *clusters* grandes (densos) normais. Como medida de anormalidade, Cui utilizou a computação de distância entre *clusters*. Assim, quanto mais distante um *cluster* pequeno for do *cluster* grande mais próximo, maior será o nível de anormalidade de seus dados. No entanto, o algoritmo considera dados anômalos apenas os pertencentes aos *top-k clusters* anômalos, onde *k* é o número máximo de *clusters* considerados anômalos, para garantir que não haja *clusters* anômalos próximos de *clusters* normais. Portanto, quando um novo dado chega, verifica-se a qual *cluster* ele pertence e atualizam-se os sumários dos *clusters* e a lista dos *k* clusters mais anômalos. Se o dado pertencer a um dos *top-k clusters*, então é rotulado como anômalo, caso contrário, normal. Afim de controlar o consumo de memória, o algoritmo utiliza uma estratégia de mescla (*merge*) de todos os *clusters* densos adjacentes e poda todos os clusters anômalos. No caso da mescla dos *clusters*, utiliza-se o algoritmo do vizinho mais próximo (*k-nearest neighbor*). Já a poda exclui todos os *clusters* anômalos próximos aos *clusters* grandes.

Sadik e Gruenwald [39] propuseram uma técnica adaptativa para detecção de anomalias (*Adaptive Outlier Detection for Data Streams* – A-ODDS) baseada em desvios nos dados considerando um fator de contexto global (*Global Deviation Factor* – GDF) e local (*Local Deviation Factor* – LDF), ambos calculados com base na densidade dos vizinhos mais próximos. A densidade da

vizinhança (conjunto dos vizinhos mais próximos) de uma instância no fluxo de dados é calculada *on-the-fly* a partir de uma função de distribuição não paramétrica (*kernel probability density estimator*). O GDF considera todo o histórico dos dados e, na prática, é a distância entre a densidade da vizinhança de uma instância no fluxo de dados e a densidade média da vizinhança de todas as instâncias (dado histórico). O LDF é a distância entre a densidade da vizinhança de uma instância no fluxo e a densidade média da vizinhança das instâncias de janelas de tempo recentes. O desvio padrão do GDF e LDF de cada instância é calculado e se pelo menos um dos desvios for maior que três, a instância é considerada anômala.

O AnyOut [93] é uma técnica para detecção de anomalias em fluxo de dados baseada em *clustering* que representa os dados de forma hierárquica. Logo, o AnyOut utiliza uma estrutura em árvore. A distância entre as instâncias e os centroides dos *clusters* é computada por nível e de cima para baixo. No entanto, sempre que uma nova instância O_{i+1} chega ao fluxo de dados, interrompe-se a descida da instância O_i na árvore. Dessa forma, a cada nível calcula-se a pontuação das instâncias. Ao chegar ao último nível, ou seja, ao chegar ao nível onde todos os nós da árvore são folhas, obtém-se a pontuação média (*mean outlier score*) que é usada para medir o nível de anormalidade da instância. Assim como o trabalho de Sadik e Gruenwald, verifica-se o desvio padrão desta pontuação. O AnyOut utiliza ainda uma segunda forma para mensurar a anormalidade: através do cálculo da densidade a partir do valor da média da instância e do cálculo da matriz de covariância. Assim, a pontuação baseada em distância leva em consideração a distância média para os centroides dos *clusters*, enquanto que a pontuação baseada na densidade leva em consideração a distribuição global dos dados, assumindo uma distribuição gaussiana. Apesar de Aggarwal [22] salientar que a matriz de covariância possa ser facilmente mantida a partir do fluxo de dados, no entanto, não considera correlações ao longo do tempo – constituindo uma desvantagem. Além disso, Aggarwal argumenta que pode ser computacionalmente caro manter a matriz de covariância.

Diferentemente dos trabalhos supracitados, o VEDAS (*vehicle data stream mining*) [94] objetiva a identificação de instâncias anômalas usando um dispositivo com baixo poder computacional, isto é, baixa capacidade de processamento e armazenamento. O VEDAS foi projetado para minerar o fluxo de

dados do veículo. Estes dados são coletados através de um dispositivo OBD-II e são armazenados em um sistema gerenciador de fluxo de dados (*Data Stream Management System – DSMS*) que provê mecanismos para controlá-los e acessá-los através de consultas. O DSMS provê operadores para computar agregação estatística tais como, média, variância e covariância. Estes dados agregados são analisados pelos algoritmos de mineração. Após o pré-processamento dos dados, o VEDAS constrói uma representação dos dados de baixa dimensionalidade através de três técnicas: Análise incremental de Componentes Principais (*incremental Principal Component Analysis* – PCA), transformações de Fourier ou segmentação linear *online*. Apesar, de ser possível escolher dinamicamente qual das técnicas será usada, os autores enfatizam que o PCA não funciona bem para monitoramento *online* com recursos computacionais limitados.

De acordo com os autores, o VEDAS implementa uma coleção de técnicas e algoritmos, inclusive proprietários, para realizar a análise do fluxo. Discutem-se técnicas baseadas em *clustering* e testes estatísticos. Primeiro, os dados do OBD-II são agrupados pelo K-Means para detecção de padrões anormais da "saúde" do veículo (*vehicle health monitoring*). O objetivo da clusterização é identificar representações no espaço que correspondam a um regime de operação seguro do veículo. Já a detecção de padrões de condução incomuns é realizada através da análise da aceleração através do algoritmo de aproximação linear, o *Piecewise Linear Approximation* [95]. Além disso, é realizado um teste estatístico nos dados suavizados pelo algoritmo assumindo uma distribuição gaussiana para identificar padrões incomuns. Os dados utilizados para validação da proposta foram extraídos do *Live For Speed*[2], um simulador de condução. No entanto, nenhuma classificação do motorista é realizada.

O Join Driving [51] é um sistema, baseado apenas nos dados dos sensores do *smartphone*, para detecção do comportamento de condução veicular que ajuda os motoristas a perceberem o nível de imprudência. Contudo, como o *smartphone* pode estar em uma posição arbitrária dentro do veículo, os autores desenvolveram um novo algoritmo para reorientação. Diferentemente das abordagens baseadas na interação motorista-veículo da Seção 2.3, o Join Driving propõe um mecanismo de pontuação para avaliar quantitativamente os eventos de condução (manobras) e

2 https://www.lfs.net/

o nível de conforto dos passageiros com base na ISO 2631-1-1997 [96]. No Join Driving, os dados do acelerômetro são coletados e, em seguida, processados por dois módulos distintos. O primeiro detecta e pontua as manobras, enquanto o segundo analisa o nível de conforto. No entanto, o artigo não deixa claro como a pontuação de cada manobra e o respectivo nível de conforto (ou desconforto) são usados na caracterização do perfil de condução do motorista. Além disso, o processo de pontuação é *offline*, ou seja, a pontuação só é calculada e informada após uma viagem, sem um *feedback* em tempo real.

Com intuito de entender e modelar o estilo de condução imprudente, Hong, Margines e Dey [52] desenvolveram uma plataforma de sensoriamento de baixo custo no veículo. Diferentemente do Join Driving [51], que usa apenas o *smartphone*, nesta plataforma foi adicionado um dispositivo de diagnóstico OBD-II para coletar dados, como velocidade, rotação por minuto e posição do acelerador. Para detectar o movimento de esterçamento do volante foi adicionado um dispositivo denominado *inertial measurement unit* (IMU). Ambos os dispositivos se comunicam com o *smartphone* via Bluetooth. Para caracterizar o estilo de condução, um modelo baseado em aprendizado de máquina analisa os dados da aceleração, do OBD-II e do IMU em seis eventos específicos: (i) *START* – cinco segundos após o veículo entrar em movimento, (ii) *STOP* – cinco segundos antes da parada completa do veículo, (iii) *H-SPEED* – enquanto o veículo move-se acima de 50 km/h, (iv) *TURN* – identificação da realização de uma curva através da análise dos dados do IMU, (v) *B-TURN* – cinco segundos que antecedem a realização da curva e (vi) *A-TURN* – cinco segundos após realização da curva.

Para determinar o comportamento de condução, primeiro é criado o perfil da viagem, em seguida, este perfil é agregado aos perfis obtidos nas últimas três semanas, chamado de perfil do motorista. Por fim, a partir do perfil do motorista e o do aprendizado de máquina, o comportamento de condução do motorista é determinado. Pensando na ausência de informação, os autores construíram três modelos – Modelo 1: considera apenas os sensores do *smartphone*; Modelo 2: considera os dados do *smartphone* e do OBD-II; e o Modelo 3 considera os dados do *smartphone*, do OBD-II e do IMU. Apesar de criar modelos distintos visando diminuir o custo para o motorista, os modelos não se adaptam a ausência de um dado de entrada esperado, por exemplo, o Modelo 2 não é capaz de lidar com a

ausência da velocidade. Além disso, o *feedback* é dado somente com o término da viagem. A proposta desta tese visa superar a limitação da ausência dos dados de entrada através da formação de uma EPN de processamento, ou seja, pela composição de EPAs conceitualmente interconectadas.

O MIROAD [5] é uma plataforma de reconhecimento de estilo de condução que usa apenas o *smartphone* como fonte de dados e unidade de processamento. O MIROAD utiliza o algoritmo *Dynamic Time Warping* (DTW) [97], originalmente desenvolvido para reconhecimento de fala, para classificar as manobras (eventos) realizadas pelo motorista. Para a detecção de manobras imprudentes no MIROAD foram criados cinco *templates*, um para cada tipo de manobra (curva à esquerda e direita, mudanças abruptas de faixa e fortes acelerações e frenagens), totalizando 40 manobras, prudentes e imprudentes. Estes *templates* foram registrados por um único motorista e um único veículo. O DTW cria uma tabela *m x n* em memória, onde *m* é o número de manobras a serem detectadas e *n* é o número de dados de sensor, e computa a distância euclidiana entre os dados dos *templates* armazenados e os dados dos motoristas para determinar se corresponde ou não a uma manobra imprudente.

Vale ressaltar que o MIROAD utiliza apenas dados do acelerômetro e giroscópio para detectar os padrões das manobras. O GPS determina apenas a velocidade e localização dos eventos, e não faz parte da detecção do estilo de condução. No entanto, de acordo com Musculo *et al.* [98], em alguns casos, o DTW pode produzir distorções quando duas sequências variam fortemente ou quando existem singularidades, ou seja, vários pontos de uma série temporal correspondendo a um único ponto.

O trabalho de Aljaafreh, Alshabatat e Najim [99] propõe a utilização de inferência por Lógica Fuzzy para identificação *online* de dados de condução anormais e classificação do comportamento dos motoristas baseado na aceleração e velocidade. A aceleração lateral e longitudinal são categorizadas em intervalos, baixa, média e alta. Já a velocidade é categorizada em cinco intervalos, de muito baixa a muito alta. Os valores destas saídas são usados para classificar os motoristas. A proposta de Quintero, Lopez e Cuervo [16] também utiliza Lógica Fuzzy, no entanto, as variáveis de saídas são inseridas em uma rede neural devidamente treinada para classificação do comportamento dos motoristas. Contudo, a rede neural encontra-se em um servidor remoto, e portanto, todas as

saídas do sistema Fuzzy precisam ser enviadas para este servidor que realiza a análise e classificação *offline* do comportamento dos motoristas. Através desta abordagem é inviável realizar uma análise em tempo real já que os dados precisam ser enviados para um servidor remoto. Além disso, custo com largura de banca é um impeditivo em cenários reais. A Tabela 1 mostra um comparativo das técnicas utilizados nos trabalhos relacionados.

Tabela 1. Comparação dos trabalhos relacionados

Trabalho	Técnica	Algoritmo
Cui [19]	*Clustering*	Distância entre *clusters*
A-ODDS [39]	Densidade	K-NN
AnyOut [93]	*Clustering*	K-Means
VEDAS [94]	*Clustering*/Segmentação Linear	K-Means/Piecewise Linear Approximation
Join Driving [51]	ISO 2631-1	Baseado no conforto dos passageiros
Hong, Margines e Dey [52]	Aprendizado de Máquina	Naive Bayes Classifier
MIROAD [5]	Dynamic Time Warping	Reconhecimento de Gestos
Aljaafreh, Alshabatat e Najim [99]	Fuzzy	Experiência Humana
Quintero, Lopez e Cuervo [16]	Fuzzy / Rede Neural	Experiência Humana/ Back-propagation

4
Proposta

Esta Seção apresenta os algoritmos de detecção de anomalias (Z-Score, Box Plot e K-Means) explicados na Seção 2.1 expressos como um conjunto de regras CEP para operar sobre vários fluxos de dados e permitindo um processamento eficiente em dispositivos móveis.

A Figura 8 mostra uma visão geral do fluxo de processamento dos dados para classificação do comportamento de condução. Os comportamentos de condução dos motoristas podem ser classificados em dois tipos, comportamentos de curta (*short-term behavior*) e longa (*long-term behavior*) duração. O primeiro diz respeito ao comportamento operacional do motorista, por exemplo, a intensidade e frequência que pressiona o pedal do acelerador, do freio ou como manipula volante. O último representa manobras mais complexas/completas, tais como, mudanças de faixas, sequências mais longas de acelerações e desacelerações, ou como curvas são traçadas. A partir dos comportamentos de curta e longa duração é possível detectar padrões de eventos de condução do veículo que permitem a formulação de um perfil que classifica o motoristas em prudente ou imprudente [100].

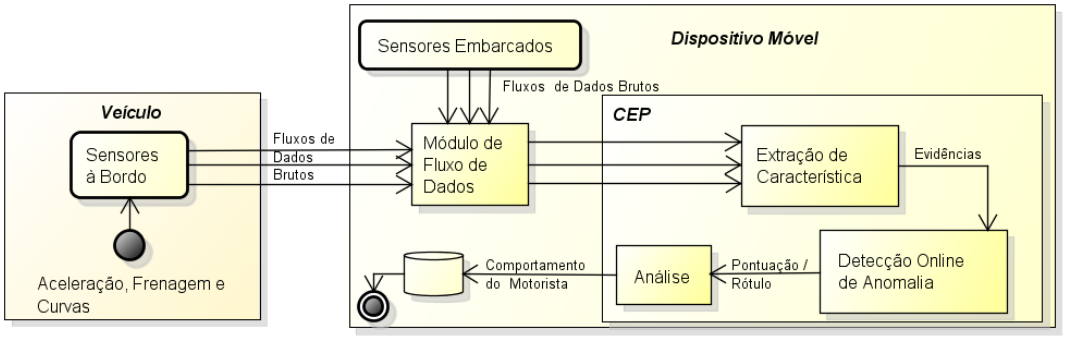

Figura 8. Visão geral do fluxo de processamento.

A fim de detectar dados do comportamento de condução, o *Módulo de Fluxo de Dados* é responsável por descobrir, conectar e ler tanto os dados dos sensores a bordo do veículo quanto os embarcados no próprio dispositivo móvel. Este módulo é capaz de se comunicar com sensores a bordo do veículo através de

tecnologias de comunicação sem fio de curto alcance, tais como Bluetooth e Bluetooth Low Energy. Mais detalhes sobre a descoberta de sensores e processo de comunicação estão disponíveis em [101]. Este módulo atua como um *hub* e encaminha o fluxo de dados para pré-processamento pelo motor CEP. Este pré-processamento de dados brutos (*raw data*) consiste na produção de dados de nível de abstração maior (referidos nesta tese como *Evidência*) que melhor apresentam um aspecto do comportamento do motorista. Esse processo de geração de evidências é conhecido como Extração de Característica (*Feature Extraction*). Uma característica é uma propriedade mensurável que melhor representa um fenômeno e a extração desta característica é o processo de derivar valores de tais características [102]. Como mencionado anteriormente, para avaliar as manobras dos motoristas, algumas características disponíveis precisam ser analisadas e correlacionadas ao longo de um período de tempo. Estas características incluem, por exemplo, velocidade ($V==[v_1, v_2, \ldots, v_n]^T$) e aceleração ($A=[a_1, a_2, \ldots, a_n]^T$). O parâmetro *n* indica o número de sequência da amostra e *T* especifica uma janela de tempo. Nesta avaliação, conforme descrito por Karthik [102], podem ser utilizadas características adicionais, tais como a velocidade média excluindo paradas, aceleração e desaceleração média, mudanças repentinas na (des)aceleração, dentre outras combinações de medições físicas.

A responsabilidade do algoritmo de detecção *online* de anomalia é encontrar padrões nas evidências encontradas que desviam significativamente do comportamento esperado, como salientado na Seção 2.1. Uma característica importante de qualquer algoritmo de detecção de anomalia é a maneira pela qual as anomalias são reportadas [14]. Por um lado, algoritmos de pontuação, como o Z-Score, atribuem uma pontuação a cada evidência estimando o grau de "anormalidade". Por outro lado, os algoritmos de rotulação, como Box Plot, atribuem um rótulo (normal ou anomalia) a cada evidência. Finalmente, estas pontuações (ou rótulos) são analisadas para classificar o comportamento do motorista (i.e., prudente ou imprudente) e atualizar o perfil do condutor.

A abordagem desta tese para detecção de anomalias usando CEP, inspirada em Brenna *et al.* [103] e Kim *et al.* [104], baseia-se em uma máquina finita de estado não determinística (*Nondeterministic Finite State Machine* ou *Nondeterministic Finite Automaton* – NFA) para adaptar os algoritmos propostos através de regras CEP. A NFA é usada para representar a estrutura da sequência

de processamento dos eventos. Como CEP possui uma arquitetura orientada a eventos, nesta tese integramos o processamento de eventos complexos e NFA para derivar automaticamente estados de processamento dos algoritmos clássicos de detecção de anomalia para operar em fluxo de dados. Estes estados de processamento implementam parte da lógica do algoritmo através de regras CEP. Dessa forma, a NFA é orientada a eventos, o que significa que a transição de um estado para outro é disparada por um evento e pode haver vários estados de processamento simultâneos. Na prática, para rastrear todos os estados paralelos, uma pilha grava o conjunto de estados ativos e como este conjunto leva a um novo conjunto de estados ativos [105].

Dessa forma, o motor CEP analisa os fluxos de dados, afim de automaticamente disparar transições de um estado de processamento formalmente especificado para outro(s). A Figura 9 ilustra a integração de CEP e NFA mostrando um NFA simples e uma regra[3] relacionada: o estado de processamento s_0 verifica se no fluxo de dados de entrada houve um evento do tipo *a* seguido[4] por um evento do tipo *b* em uma janela de tempo de tamanho *len*, então gera-se um novo evento e_1. Em seguida, o evento e_1 dispara automaticamente a transição do estado s_0 para o estado s_1 na NFA.

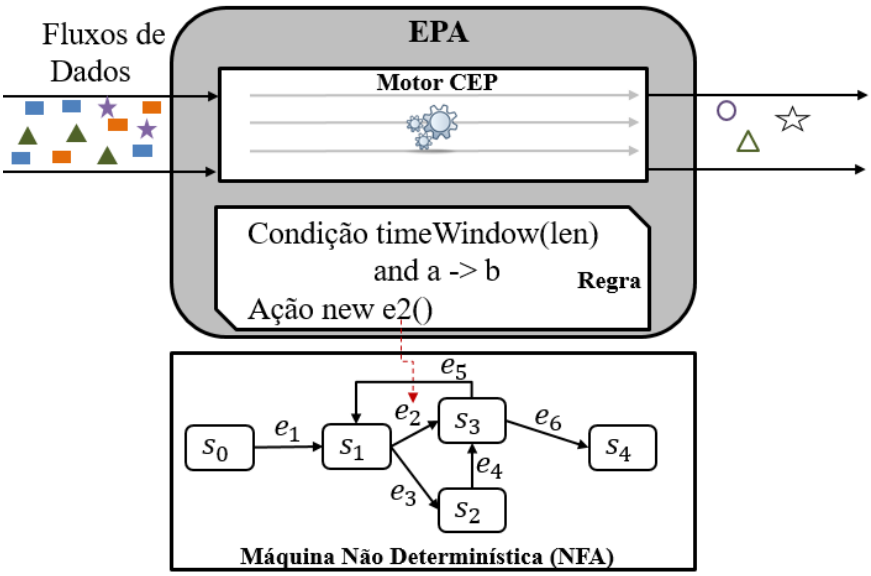

Figura 9. Integração entre CEP e FSM. Adaptado de [74].

Em geral, uma NFA é definida [106] pela quíntupla (S, ∑ E, S_0, δ), onde:

3 Neste exemplo, uma pseudo-linguagem simplificada foi utilizada para expressar regras de processamento de eventos, por ser mais fácil de entender do que uma regra EPL real.

4 O operador de sequência de eventos é expresso pelo operador ->.

- S é o conjunto finito de estados;
- ∑ é o conjunto de dados de entrada (evidências);
- E é o conjunto de eventos;
- S_0 é o estado inicial, $S_0 \in S$;
- δ é a função de transição de estado.

Integrar CEP e NFA gera alguns benefícios. Combina a flexibilidade, ciência de situação e de contexto fornecido pelo CEP com as restrições de transição de estados garantidas pela NFA. Cada estado da NFA realiza parte da lógica de processamento de um determinado algoritmo. A implementação dos algoritmos de detecção de anomalias combinando CEP e NFA é discutida nas seções 4.1, 4.2 e 4.3.

4.1
Algoritmo Z-Score Online baseado em CEP

Como o Z-Score recebe fluxos de dados e não pode esperar até que todas as evidências sejam recebidas do módulo de fluxo de dados, ele segmenta o fluxo em janelas de tempo (*contexto temporal*), cada uma das quais contendo um conjunto de evidências. Em seguida, as evidências são agrupadas por dimensão. Uma transição leva ao estado de processamento responsável por calcular os Z-Scores como mostrado na equação (2). No Z-Score Online, ao contrário do algoritmo Z-Score clássico, a média e desvio padrão são calculados a partir das *evidencias* pertencentes a uma janela deslizante *T* e uma dimensão *D*. As regras de contexto temporal determinam quais instâncias de dados são admitidas em qual janela. Então, o algoritmo calcula o Z-Score das evidências disponíveis em cada janela. No último estado de processamento, a distribuição dos Z-Scores (*Z-Distribution*) é computada conforme mostrado na Figura 3. Por fim, as pontuações (z-scores) de cada evidência, juntamente com a distribuição dos z-scores, são enviadas para o módulo de análise que classificará o motorista em prudente ou imprudente, para cada janela de tempo. A Figura 10 exibe os estados de processamento do Z-Score Online.

$$Z = \left(\frac{X - \mu}{\sigma x} \right)^{T} \quad (2)$$

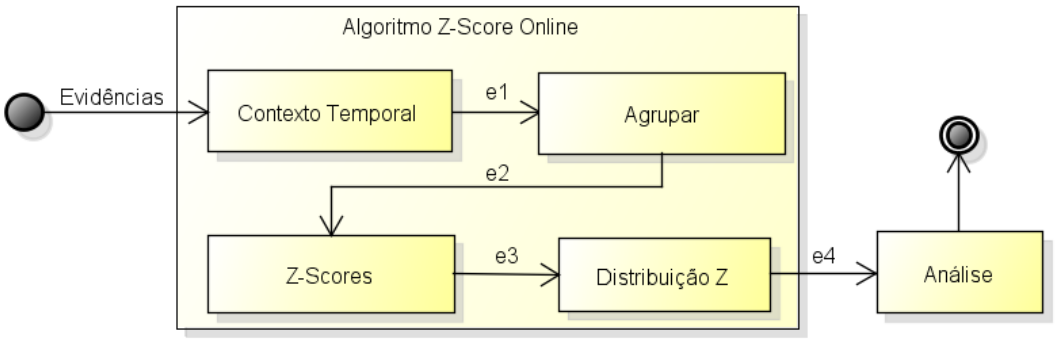

Figura 10. Z-Score online baseado em NFA.

4.1.1 Implementação do Z-Score Online

As regras dos algoritmos foram escritas em *Event Processing Language* – EPL, uma linguagem de consulta contínua, para expressar regras CEP no ESPER [79], um motor de processamento de código aberto. A regra EPL que implementa o Z-Score Online é ilustrada na Figura 11. A cláusula *time* na linha 3 é um operador temporal que segmenta as evidências dos fluxos de dados em janelas deslizantes de tamanho *windowLength*. Esse tipo de janela foi escolhido para evitar o problema exemplificado na Figura 12 – no segmento (1 – 30) o motorista dirigia prudentemente e a partir do instante de tempo 30 iniciou-se uma manobra imprudente. Utilizando a janela de tempo em lote, as evidências iniciais do *Lote 2* não seriam consideradas anômalas por desconsiderar todo o *Lote 1*. A cláusula na linha 4 (Figura 11) realiza o agrupamento das evidências por dimensão. Na projeção da linha 2 é calculado o Z-Score das evidências que pertencem à janela de tempo especificada na linha 3. O cálculo do Z-Score é realizado por dimensão como mostrado na linha 4. Por fim, a cláusula da linha 1 insere os Z-Scores das evidências no evento chamado *Z_SCORE_EVENT*.

```
1. INSERT INTO Z_SCORE_EVENT
2. SELECT (rawValue - avg(rawValue))/stddev(rawValue) AS z_score
3. FROM Evidence.win:time(windowLength sec)
4. GROUP BY dimension
```

Figura 11. Cálculo do Z-Score expresso em EPL

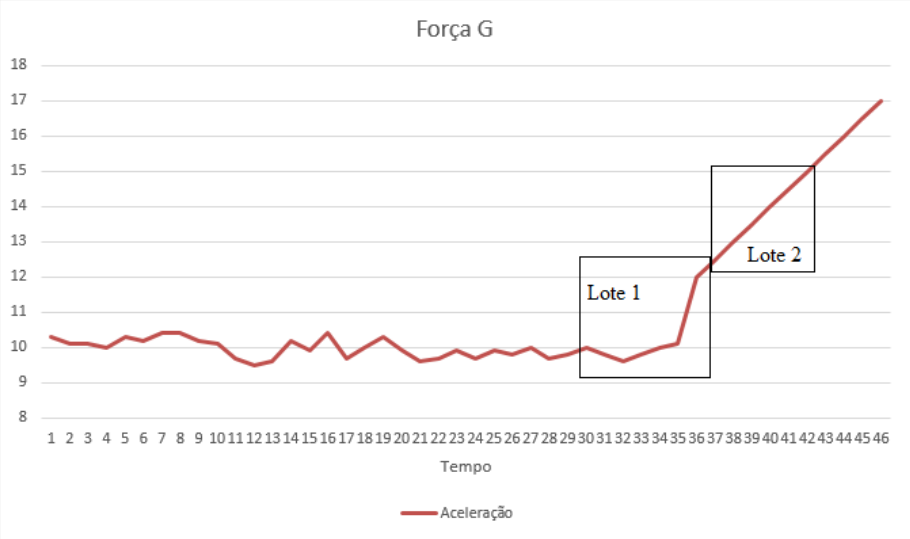

Figura 12. Exemplo de janela em lote

A regra EPL implementada na Figura 13 realiza o cálculo da distribuição dos z-scores. Da linha três a dez é feito o cálculo do percentual de instâncias de acordo com os intervalos da Figura 3. Na linha 11 define-se uma janela em lote para computar a distribuição dos z-scores em um intervalo de tempo fixo.

```
1.  INSERT INTO Z_SCORE_RESULT
2.  SELECT z_score,
3.          (COUNT(*, z_score < -3) * 100)/CAST(COUNT(*), FLOAT) AS l3,
4.         ((COUNT(*, z_score >= -3 AND z_score < -2)) * 100)/CAST(COUNT(*), FLOAT) AS b32,
5.         ((COUNT(*, z_score >= -2 AND z_score < -1)) * 100)/CAST(COUNT(*), FLOAT) AS b21,
6.         ((COUNT(*, z_score >= -1 AND z_score < 0)) * 100)/CAST(COUNT(*), FLOAT) AS b10,
7.         ((COUNT(*, z_score >= 0 AND z_score < 1)) * 100)/CAST(COUNT(*), FLOAT) AS b01,
8.         ((COUNT(*, z_score >= 1 AND z_score < 2)) * 100)/CAST(COUNT(*), FLOAT) AS b12,
9.         ((COUNT(*, z_score >= 2 AND z_score <= 3)) * 100)/CAST(COUNT(*), FLOAT) AS b23,
10.        (COUNT(*, z_score > 3) * 100)/CAST(COUNT(*), FLOAT) AS a3
11. FROM Z_SCORE_EVENT.win:time_batch(windowLenght sec) AS e
12. GROUP BY dimension
```

Figura 13. Regra usada para a distribuição dos z-scores.

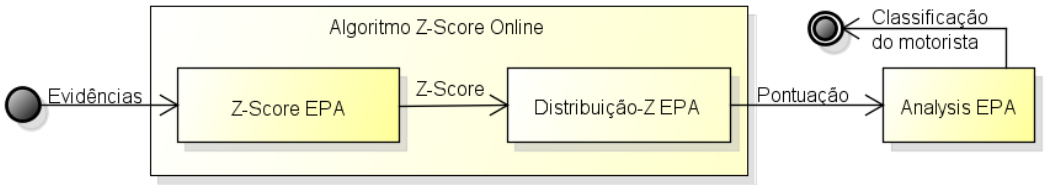

Figura 14. Estados de processamento (EPA) do Z-Score Online.

A estrutura de processamento do Z-Score Online ficou como mostrado na Figura 14. Comparando as Figura 10 e Figura 14 nota-se que dois estados de processamento da NFA (contexto temporal e agrupar) foram omitidos na adequação do Z-Score Online para regras EPL. Ambos estados de processamento

foram encapsulados pelos operadores de contexto (*time*) e de agrupamento (*group by*) da linguagem. A computação dos z-scores foi realizada usando métodos da própria linguagem. A transição para o estado de processamento que computa a distribuição dos z-scores é realizada de forma automática visto que a regra EPL da Figura 13 “consome” os eventos gerados pela regra EPL da Figura 11.

4.2
Algoritmo Box Plot Online baseado em CEP

A estrutura do algoritmo Box Plot Online com seus estados de processamento e suas transições é mostrado na Figura 15. Assim como o Z-Score, o *contexto temporal* é criado e, em seguida, as evidências são agrupadas por dimensão. O próximo estado de processamento calcula a mediana (*Q2*) por dimensão. A transição deste estado leva a dois estados de processamento independentes – cálculo do valor do primeiro (*Q1*) e terceiro (*Q3*) quartil. A transição destes dois estados leva ao estado de processamento para finalizar o *5-number summary* com o cálculo do menor e maior valor não anômalo, *min* e *max* respectivamente. Além disso, computa-se a distância entre os quartis (*IQR*). Por fim, o último estado é responsável por rotular cada evidência em normal ou anômala.

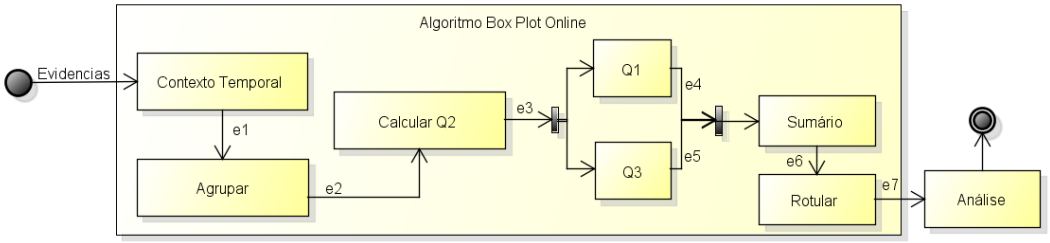

Figura 15. Box Plot online baseado em NFA

4.2.1
Implementação do Box Plot Online

A implementação do Box Plot Online através de regras CEP é ilustrada na Figura 16. Para evitar o cálculo da distância entre todos os pares de evidências, cuja implementação normalmente possui complexidade quadrática [14], optou-se por realizar os cálculos para cada dimensão individualmente (versão unidimensional do algoritmo) e ao final, a EPA de análise correlaciona as evidências anômalas. Na adequação do Box Plot Online para regras EPL nota-se

que, assim como o Z-Score Online, estados de processamento da NFA para criação do contexto temporal e agrupar as evidências foram abstraídas pelos seus respectivos operadores. Então, o primeiro estado de processamento do Box Plot Online é, de fato, o cálculo da mediana (*Q2*). Após o cálculo, dispara-se o evento de transição *Q2* que leva ao estado de processamento que calcula o valor do primeiro e terceiro quatis. Neste estado utilizou-se o conceito de sub-consulta da EPL para calcular o primeiro e terceiro quatis em uma mesma regra, diminuindo assim, o número de estados de processamento. Em seguida, dispara-se um evento (*Q1, Q2 e Q3*) que leva ao estado de processamento que verificará o maior e menor valor não anômalos e o *IQR*. Por fim, com base nas informações obtidas nos estados anteriores, cada evidência é rotulada ("normal" ou anômala).

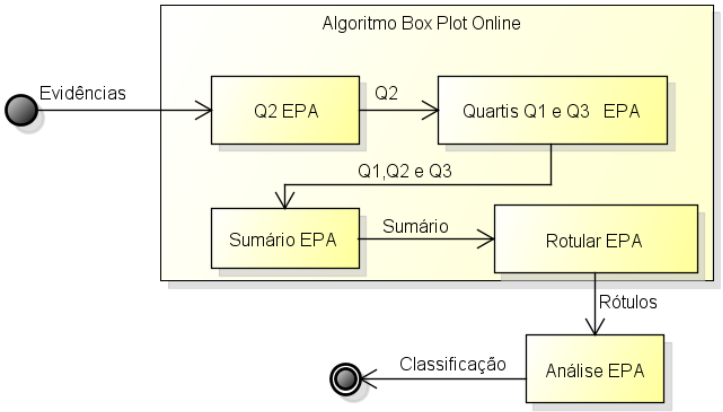

Figura 16. Estados de processamento (EPA) do Box Plot Online

A regra EPL exibida na Figura 17 implementa o cálculo do *Q2* e encapsula os três primeiros estados de processamento da Figura 15. Como saída, esta regra EPL insere a mediana calculada em um fluxo chamado *box_plot_q2_event*. Assim, o primeiro e terceiro quartil (*Q1* e *Q3*) são calculados. Para isso, criou-se uma regra EPL que recebe o fluxo *box_plot_q2_event* e usa a mediana como parâmetro para o cálculo do primeiro (*Q1*) e terceiro (*Q3*) quartis, como mostrado na Figura 18. O resultado é inserido no fluxo *box_plot_q1_q3_event*.

```
1. INSERT INTO BOX_PLOT_Q2_EVENT
2. SELECT median(rawValue) AS q2,
3.        rawValue  AS rawValue
4. FROM  Evidence.win:time(windowLength sec)
5. GROUP BY dimension
```

Figura 17. Cálculo do Q2.

```
INSERT INTO BOX_PLOT_Q1_Q3_EVENT
SELECT rawValue, q2,
 (SELECT median(rawValue) FROM BOX_PLOT_Q2_EVENT WHERE rawValue < q2) AS q1,
 (SELECT median(rawValue) FROM BOX_PLOT_Q2_EVENT WHERE rawValue > q2) AS q3
FROM  BOX_PLOT_Q2_EVENT.win:time(windowLength sec)
GROUP BY dimension
```

Figura 18. Cálculo do Q1 e Q3.

O terceiro estado de processamento calcula *min*, *max* e o *IQR* a partir do fluxo de eventos *box_plot_q1_q3_event*. As expressões *non_outlier_expression* (equação 4) e *outlier_expression* (equação 5) verificam se uma instância é normal ou anômala como proposto por Laurikkala, Juhola e Kentala [61] e explicado na Seção 2.1. As funções *fmin* e *fmax* retornam, respectivamente, a menor e maior evidência não anômala no fluxo de dados *box_plot_q1_q3_event*, respeitando as restrições impostas pela expressão *non_outlier_expression*. Como saída, esta instrução insere o sumário computado de 5 números e o *IQR* no fluxo de dados *box_plot_event*.

$$rawValue \geq (q1 - (1.5 * IQR)) \,\&\, \text{rawValue} \leq (\text{q3} + (1.5 * \text{IQR})) \tag{3}$$

$$rawValue < (q1 - (1.5 * IQR)) \,\|\, rawValue > (q3 + (1.5 * IQR)) \tag{4}$$

```
INSERT INTO BOX_PLOT_EVENT
SELECT rawValue, q1, q2, q3, (q3 - q1) AS IQR,
       fmin(rawValue, non_outlier_expression) AS min_non_outlier,
       fmax(rawValue, non_outlier_expression) AS max_non_outlier
FROM  BOX_PLOT_Q1_Q3_EVENT.win:time(windowLength sec)
GROUP BY dimension
```

Figura 19. Sumário e IQR do Box Plot.

```
INSERT INTO BOX_PLOT_RESULT
SELECT (CASE WHEN non_outlier_expression THEN 'normal'
             WHEN outlier_expression THEN 'outlier'
       ) AS label,
      rawValue, q1, q2, q3,IQR
FROM box_plot_event
```

Figura 20. Regra EPL para rotular as evidências.

O último estado de processamento do Box Plot é rotular as evidências com base no sumário e no IQR. Finalmente, após a rotulação, os dados são enviados para análise e classificação do motorista em prudente ou imprudente. Este processo é feito através da regra EPL da Figura 20. Nota-se que não foi utilizada janela de tempo nesta regra. Isto foi possível porque ao longo das regras anteriores

os dados foram agregados e cada evento possui todas as informações necessárias para rotular a evidência.

4.3
Algoritmo K-Means Online baseado em CEP

A visão geral do K-Means Online baseado em NFA é dada na Figura 21. Diferentemente dos algoritmos anteriores, o K-Means é um algoritmo iterativo. Por este motivo, foi necessário usar uma janela em lote para possibilitar iterações até que o algoritmo convirja sem que novas instâncias sejam adicionadas ao processamento. A escolha da janela em lote ainda evita que o algoritmo entre em um *loop* infinito. Assim, os fluxos de dados das evidências são separados em diferentes *contextos temporais* (janelas de lote). Em seguida, os centroides do(s) *cluster*(s) são escolhidos e cria-se o contexto temporal. A implementação tradicional do algoritmo K-Means escolhe *K* instâncias aleatórias e as define como centroides do *clusters*. A principal desvantagem deste método reside no fato que estas instâncias aleatórias podem não ser bons centroides, levando a um baixo desempenho [73]. Por exemplo, pode ser escolhida uma instância anômala ou um ruído. Nos testes, ainda na fase de desenvolvimento, notamos um desempenho ruim com a escolha aleatória dos valores iniciais dos centroides. Para superar este problema, o algoritmo começa com o conhecimento adquirido previamente na fase de treino. Mais detalhes são dados na Seção 6.1.

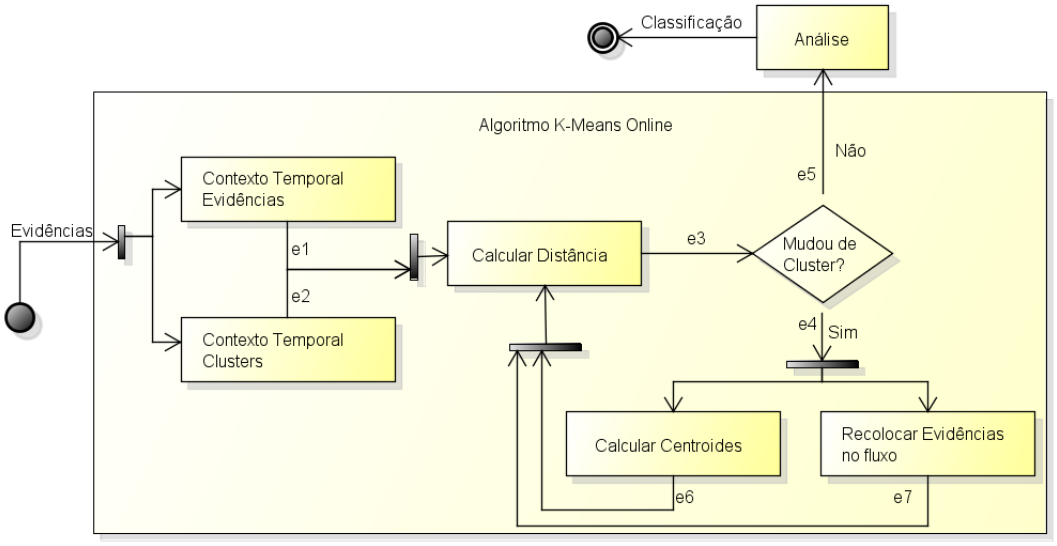

Figura 21. Estados de processamento do K-Means Online.

O terceiro estado de processamento calcula a distância entre as evidências e os centroides dos *clusters*. As evidências são atribuídas ao *cluster* mais próximo. Enquanto houver evidências que mudem de um *cluster* para outro em uma iteração, calcula-se novos centroides para os *clusters* e reinsere-se as evidências no fluxo de processamento para uma nova iteração. Quando calcular as distâncias e o *cluster* mais próximo de cada evidência não mudar, a formação dos *clusters* é enviada para análise.

4.3.1 Implementação do K-Means Online

A implementação do K-Means por meio de regras CEP segue o fluxo de estados de processamento e transições mostrado na Figura 21. Para o K-Means foi necessário criar duas janelas em lote nomeadas (*named window*) como mostrada na Figura 22. Este tipo de janela é globalmente visível e pode ser acessada por várias outras regras EPLs. A função da regra EPL da Figura 22 (a) é *bufferizar* as evidências por um período de tempo (e.g., 20 segundos). Todas as evidências *bufferizadas* são enviadas para a regra EPL de cálculo das distâncias. Enquanto o algoritmo está processando o fluxo de dados, esta janela recomeça a preencher o *buffer*. A segunda janela nomeada, exibida na Figura 22 (b), cria um *buffer* contento as informações dos *clusters*.

```
1. CREATE window EvidenceStreamEvent.win:time_batch(timeWindowLength sec) AS
2.        SELECT * FROM Evidence
                                                                          a
3. ____________________________________________________________________
4. CREATE window ClusterStreamEvent.win:time_batch(timeWindowLength sec) AS
5.        SELECT * FROM Cluster
                                                                          b
```

Figura 22. Criação das janelas em lote nomeadas

A regra EPL da Figura 23, realiza uma junção (*join*) entre os fluxos das evidências (*EvidenceStreamEvent*) e dos *clusters* (*ClusterStreamEvent*). É criado o evento *min_distance_matrix* contendo a distância de uma evidência para o *cluster* mais próximo, seu respectivo identificador (*nearest_cluster_id*) e o identificador do cluster atual (*current_cluster_id*). Logo após a computação das distâncias, a regra da Figura 24 (a) verifica se alguma evidência mudou de *cluster*, ou seja, se *nearest_cluster_id* é diferente de *current_cluster_id*. Se houve mudança, é criado o evento *LoopEvent*. A linha 6 desta regra limita o número de eventos, pois basta saber se ao menos uma evidência mudou de *cluster* para iniciar

outra iteração. Por conseguinte, as regras das Figura 24 (b) e Figura 24 (c) verificam se houve o padrão (*Pattern Matching*) – um evento *min_distance_matrix* seguido por um intervalo de tempo e por um *LoopEvent.* A regra necessita deste intervalo de tempo porque precisa esperar todo o processamento do *LoopEvent*. Caso verdadeiro, a primeira regra atualiza o valor dos centroides dos *clusters* e a segunda regra reinsere as evidências no fluxo.

```
INSERT INTO MIN_DISTANCE_MATRIX
SELECT SQRT(SUM(POW(e.rawValue - c.centroid),2)) AS distance,
       e.cluster_id AS current_cluster_id,
       c.id AS nearest_cluster_id
FROM  EvidenceStreamEvent.win:time_batch(timeWindowLength sec) AS e,
      ClusterStreamEvent.win:time_batch(timeWindowLength sec) AS c
HAVING MIN(SQRT(SUM(POW(e.rawValue - c.centroid),2)))) =
           SQRT(SUM(POW(e.rawValue - c.centroid),2)))
```

Figura 23. Cálculo da distância para o *Cluster* mais próximo.

a

```
INSERT INTO LoopEvent
SELECT TRUE AS loop
FROM MIN_DISTANCE_MATRIX.win:time_batch(timeWindowLength sec) d
WHERE d.AnyOf( i=> i.current_associate_cluster_id != i.cluster_id)
HAVING COUNT(*) > 0
LIMIT 1
```

b

```
INSERT INTO ClusterStreamEvent
SELECT mdm.cluster_id AS id,
       avg( mdm.rawValue) AS centroid
FROM pattern[every mdm=MIN_DISTANCE_MATRIX -> (timer:INTERVAL(timeWindowLength sec) AND
     LoopEvent(loop=TRUE))].win:time_batch(timeWindowLength sec)
GROUP BY mdm.cluster_id
HAVING  COUNT(*) > 0
```

c

```
INSERT INTO EvidenceStreamEvent
SELECT  *
FROM pattern[every mdm=MIN_DISTANCE_MATRIX -> (timer:INTERVAL(timeWindowLength sec) AND
     LoopEvent(loop=TRUE))]
```

Figura 24. Instruções de *Loop* do K-Means.

Caso haja um padrão no fluxo de dados correspondente a um evento *min_distance_matrix* seguido por um intervalo de tempo e pela ausência do evento *LoopEvent*, então isso significa que a iteração chegou ao fim e, portanto, o K-Means convergiu. Esta regra EPL é mostrada na Figura 25. O resultado é enviado para a EPA de análise, cujos detalhes são discutidos na Seção 6.3.1, que explica o processo de pontuação e classificação dos motoristas.

```
1. INSERT INTO K_MEANS_EVENT
2. SELECT  *
3. FROM pattern[every mdm=MIN_DISTANCE_MATRIX ->
4.     (timer:INTERVAL(timeWindowLength sec) AND
5.     NOT LoopEvent)].win:time_batch(timeWindowLength sec)
```

Figura 25. Regra EPL que identifica o fim da iteração.

4.4 Limitações e Discussão

Embora CEP forneça vários benefícios para o processamento de fluxo de dados, tais como consulta contínua, detecção de padrões e janelas temporais, é difícil expressar controles iterativos, isto é, laços de repetição (*loop*) *while*, *repeat* e *for*, utilizando as suas primitivas. Os motores CEP geralmente seguem uma topologia de estados de processamento (*pipeline*), onde os dados/eventos fluem em uma determinada direção, para um ou mais estados de processamento, mas sem retornar aos estágios anteriores. Isso pode ser problemático ao descrever algoritmos iterativos, como K-Means, que requer iterações para convergir. Para superar esse problema, uma verificação de *loop* foi realizada como visto nas regras da Figura 24. Se a verificação de *loop* não existir, isto é, se as evidências não tiverem alterado o seu centroide, encaminha-se o evento para as fases de processamento seguintes. No entanto, se foi gerado o evento *LoopEvent*, ou seja, uma evidência mudou seu centroide, reinsere-se esse evento no fluxo e a iteração continua a partir do cálculo das distâncias dos centroides. Isto é feito reinserindo as evidências em *EvidenceStreamEvents*.

Embora útil, a independência de cada etapa de processamento do CEP pode dificultar a coordenação entre elas. Por exemplo, por um lado, o algoritmo K-Means proposto faz a *bufferização* das evidências recebidas em lotes de tempo Δ, que são enviadas para a próxima fase de processamento, como mostrado na Figura 21. Isso é necessário para que, durante as interações, o algoritmo analise o mesmo conjunto de evidências agrupando-as em *clusters*. Assim, mesmo que os eventos sejam armazenados em *buffer*s, os eventos pertencentes a um lote são analisados um por um nos estágios sequenciais de processamento. No entanto, os próximos estados de processamento não recebem todo o lote (*buffer*) em um único evento. Na verdade, recebem as evidências do lote individualmente e, portanto, também requerem que uma outra janela de lote seja criada para que seja efetuado um

processamento que considere todo o lote, ocasionando um tempo maior para realizar o processamento. Logo, este comportamento influencia diretamente o tempo de convergência do K-Means. A especificação deste tempo está associada principalmente ao poder de processamento do dispositivo móvel. Durante os testes de desenvolvimento usou-se dispositivos móveis com capacidade distintas de processamento e quanto maior a capacidade, menor foi o tempo. Já durante a fase de preparação do estudo de caso (Seção 6), este tempo sempre foi inferior ou igual a 0,5 segundo. Um exemplo desta limitação é mostrado na regra EPL Figura 24 (a). O parâmetro *timeWindowLength* é o tempo mínimo necessário para fazer o *buffer* de todo o fluxo de saída no evento *min_distance_matrix* e verificar se alguma evidência mudou seu centroide.

Como discutido da Seção 2.1, a detecção de anomalia visa descobrir instâncias que desviam significativamente do restante das amostras. Por exemplo, na Figura 26 ficam evidenciadas três mudanças significativas no valor do dado. A primeira entre os tempos 5 e 6, a segunda entre 11 e 12 e a terceira entre 12 e 13. Aggarwal [22] usa este exemplo para explicar que na detecção de anomalia em fluxo de dados e séries temporais os dados não podem ser analisados isoladamente, pois são altamente influenciados pelos valores adjacentes. No entanto, pode ocorrer um fenômeno em que os valores tendem a mudar lentamente durante o tempo. Este fenômeno é referenciado como *concept drift* [22], [107], [108]. Este fenômeno torna-se especialmente desafiador na análise *online* do fluxo de dados visto que fluxo de dados são considerados uma sequência infinita (i.e., *unbounded*) [19], [107] e consequentemente, impraticável armazenar todos os dados para comparação. Em outras palavras, na abordagem desta tese, caso o motorista mude o comportamento de forma muito lenta, os algoritmos adaptados não seriam capazes de identificar anomalias. Apesar de não ser um problema explorado no escopo da tese, há na literatura trabalhos propostos que abordam o problema de *concept drift* em fluxo de dados [109], [110]. Além disso, outra característica do fluxo de dados é o conceito de evolução (*concept evolution*). Isso ocorre quando novas classes surgem no fluxo. Por exemplo, considerando o problema de classificação do comportamento de motoristas (i.e., prudente e imprudente), então o conceito de evolução ocorre quando surge uma nova classe de modo de condução, tal como, a que é praticada por motoristas com déficit técnico (popularmente conhecidos como “barbeiros”). Este problema é abordado

de forma limitada pelas técnicas de classificação em fluxo de dados atualmente disponíveis [107], [108], [111].

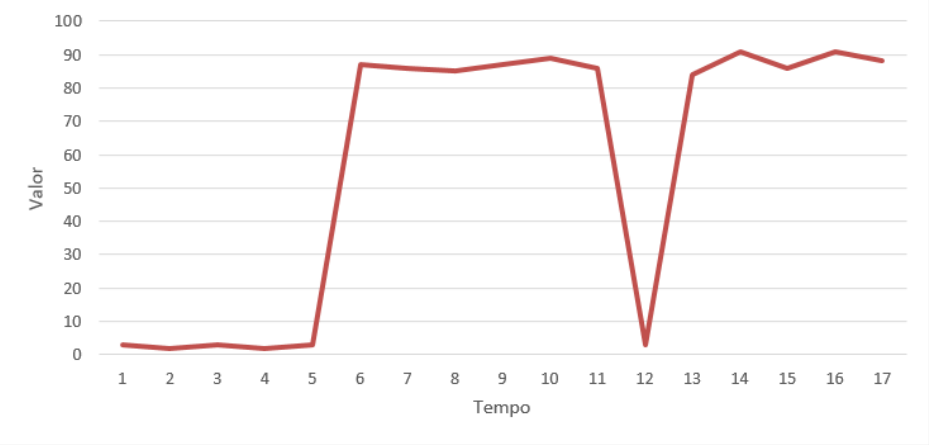

Figura 26. Mudanças na série temporal.

5
Definição do Estudo de Caso

Conforme Cervo e Bervian [112], o método em seu sentido mais geral é a ordem que se deve impor aos diferentes processos necessários para atingir um resultado desejado. Dessa forma, quando cuidadosamente selecionado, é o que confere rigor científico a uma pesquisa. Em função da natureza do problema a ser estudado, da complexidade do tema, e das características do universo analisado, esta tese adotará como método central de avaliação o estudo de caso, por investigar fenômenos dentro de seu contexto real. A respeito do estudo de caso, Martins [113] e Yin [114] enfatizam que fontes múltiplas favorecem não somente uma maior variedade de evidências, mas principalmente, aumentam a confiabilidade das informações obtidas e o grau de credibilidade da pesquisa.

5.1
Objetivos e Contribuições

O objetivo deste estudo de caso é avaliar os algoritmos projetados e implementados, conforme Seção 4, a fim de identificar o comportamento de condução dos motoristas com base na detecção de anomalias. Mais especificamente, os objetivos são:

- Responder as questões de pesquisa elencadas na Seção 1.4 sobre o desempenho dos algoritmos, ou seja, avaliar o desempenho dos algoritmos de detecção de anomalia *online* em um contexto de recursos computacionais limitados. As métricas utilizadas são explicadas da Seção 5.4. Os algoritmos foram avaliados e comparados com três trabalhos relacionados usando como *Ground truth* o *dataset* disponibilizado por Romera, Bergasa e Arroyo [115];
- Realizar uma avaliação do comportamento de condução dos motoristas, em um ambiente real, através do algoritmo de detecção de anomalia, proposto nesta tese, com melhor desempenho nas métricas indicadas na Seção 5.4. Para verificar a eficiência e eficácia

na classificação do estilo de condução dos motoristas, o *Ground truth* utilizado foi a avaliação de um motorista especialista;

- Fornecer um conjunto de dados a respeito do comportamento de condução tais como velocidade, rotação do motor por minuto (rpm), posição do acelerador, acelerômetro e giroscópio. Estes dados podem ser utilizados em novas pesquisas sobre detecção de anomalias em fluxos de dados e classificação do estilo de condução.

5.2 Seleção dos Motoristas e Planejamento da Rota

Devido à dificuldade de recrutamento de motoristas e aos custos financeiros associados à avaliação do comportamento de condução (i.e., custo com combustível), o processo de seleção dos motoristas ocorreu por conveniência, tornando o tipo de amostragem por cota [116]. No entanto, procurou-se estabelecer uma amostra que representasse o universo dos motoristas, preservando as mesmas características comportamentais (estilo de condução). Assim, foram escolhidos 25 motoristas para o estudo de caso. Dezesseis eram do sexo masculino e nove do sexo feminino, com idade variando de 20 a 60 anos. Outra questão importante para o estudo de caso é a experiência dos motoristas. Na amostra, os motoristas possuíam de 2 a 42 anos de experiência. Todos os motoristas estavam devidamente habilitados e, portanto, familiarizados com as condições de tráfego local e a legislação de trânsito.

Quanto à seleção da rota, foram avaliados vários locais para o teste. Entretanto, definimos um percurso pavimentado em Aracaju-SE totalizando aproximadamente 14,5 km, cujas ruas e avenidas possuíam de uma a três faixas. Além disso, a rota que é mostrada na Figura 27, contém rotatórias, semáforos, faixa de pedestres, cruzamentos, interseções (parada total) e curvas de aproximadamente 45º e 90º. O limite de velocidade em todo o percurso é de 60 km/h.

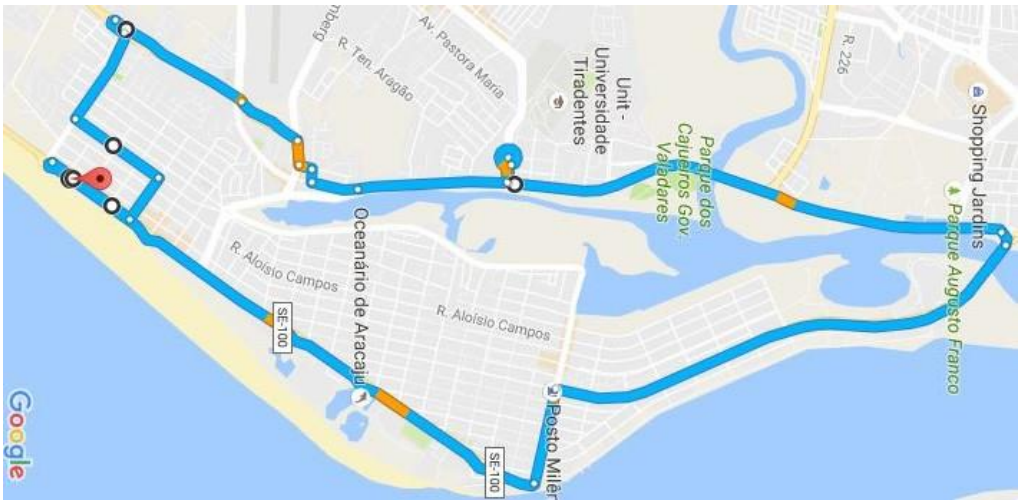

Figura 27. A rota de estudo escolhida para a coleta de dados (Mapa © 2016 Google)

5.3
Instrumentação

O processo de instrumentação[5] começou com a implementação dos algoritmos, por meio das regras CEP, conforme descrito na Seção 4. Os algoritmos foram implementados em EPL. No estudo de caso utilizamos o ASPER, um motor de processamento CEP baseado no ESPER, e adaptado para *Android*.

A avaliação e a comparação dos algoritmos relatadas nas Seções 6.1.1 e 6.1.2 utilizaram um Samsung Galaxy SIII 1,4 GHz Quad Core com 1GB de RAM, executando os nossos algoritmos. Já na avaliação descrita na Seção 6.3, a versão brasileira do Citröen C3 (câmbio manual) foi equipada com o *smartphone* supracitado, e um dispositivo OBD-II. A comunicação entre o *smartphone* e o dispositivo OBD-II deu-se via Bluetooth. O protótipo foi instalado no *smartphone* executando o algoritmo Z-Score Online. Mais detalhes sobre a escolha do algoritmo são dados na Seção 6.3. Os dados coletados pelo protótipo e processados pelo Z-Score foram armazenados no SQLite [117], um banco de dados SQL livre projetado para funcionar em dispositivos como *smartphones*, automóveis, televisões, câmeras, sensores [117], ou seja, qualquer objeto inteligente.

5 O processo de instrumentação refere-se ao processo de implementação dos algoritmos, procedimentos de configuração das ferramentas utilizadas e planejamento da coleta dos dados.

5.4
Métricas de desempenho

Dentre as muitas maneiras de avaliar a capacidade preditiva de um algoritmo, a *matriz de confusão* é uma técnica simples [118] e útil para analisar o quão bem um classificador pode reconhecer instâncias de dados de diferentes classes [119], [120]. Para *n* classes, a matriz de confusão é uma tabela de *n* x *n*. Como mostrado na Tabela 2, a coluna *classe correta* corresponde às classificações corretas para cada instância e a *classe prevista* representa a classificação dada pelo algoritmo de classificação. No caso dessa pesquisa há apenas duas classes, então uma é a positiva (condução prudente) e a outra é a negativa (condução imprudente) [119].

Tabela 2. Matriz de confusão.

Classe Correta	**Classe Predita**	
	Positiva	Negativa
Positiva	Verdadeiro Positivo (VP)	Falso Negativo (FN)
Negativa	Falso Positivo (FP)	Verdadeiro Negativo (VN)

Tabela 3. Métricas de desempenho usadas na avaliação e comparação dos algoritmos [119].

Métrica	**Equação**
Acurácia é o percentual de instâncias (evidências) corretamente classificadas.	$\frac{VP + VN}{VP + VN + FP + FN}$
Cobertura é o percentual de instâncias classificadas como positiva.	$\frac{VP}{VP + FN}$
Precisão é o percentual de instâncias corretamente classificadas como positiva (evidências).	$\frac{VP}{VP + FP}$
Medida F é a média harmônica da precisão e cobertura.	$\frac{2x\ precisão\ x\ cobertura}{precisão + cobertura}$
Taxa de erro é a proporção de instâncias que são incorretamente classificadas.	$\frac{FP + FN}{VP + FN + FP + VN}$

Assim, VP significa que uma instância da classe positiva foi classificada corretamente como positiva (evidência de condução prudente). FN significa que uma instância da classe positiva foi erroneamente classificada como negativa (evidência de condução imprudente). VN significa que uma instância da classe negativa é classificada corretamente como negativa e, finalmente, FP significa que uma instância da classe negativa foi erroneamente classificada como positiva.

Com base na matriz de confusão, pode-se calcular cinco métricas de desempenho dos algoritmos, mostradas na Tabela 3, que serão usadas para avaliar os algoritmos. Além disso, como métrica de qualidade, utilizou-se o **tempo de execução médio**, ou seja, a média aritmética dos tempos de execução para um determinado algoritmo e o **consumo médio de recursos** dos algoritmos. Enquanto as métricas de desempenho avaliam a eficácia dos algoritmos, as métricas de qualidade avaliam a eficiência. Como os dispositivos móveis possuem poder de processamento e capacidade de armazenamento limitados, o algoritmo que atingir maior nível de eficiência e eficácia é mais adequado para resolver o problema proposto nesta tese.

6
Operação do Estudo de Caso

Esta Seção descreve a preparação e execução do estudo de caso em um contexto real. A fase de preparação visa avaliar e comparar os algoritmos antes da execução do estudo de caso. Para esse propósito, utilizou-se um *dataset* público que fornece uma grande quantidade de dados condução veicular de seis condutores e seis veículos distintos em diferentes tipos via. O algoritmo baseado em regras CEP com melhor desempenho foi utilizado na execução do estudo de caso.

6.1
Preparação

Para a avaliação e comparação dos algoritmos, foi utilizado o *dataset* disponibilizado por Romera, Bergasa e Arroyo [115]. O *dataset* possui dados do acelerômetro (três eixos) rotulados em prudente e imprudente de acordo com os limiares definidos por Paefgen *et al.* [50] para aceleração lateral e longitudinal. Este *dataset* contém dados da condução de seis condutores. Cada motorista utilizou um veículo de pequeno porte distinto. Os dados foram coletados em duas rotas: a primeira, cuja velocidade máxima permitida é 120 km/h, possui 25 km e é uma estrada com três pistas em cada direção na maior parte do trajeto. A segunda, cuja velocidade máxima permitida é 90 km/h, possui cerca de 16 km e é uma estrada secundária com uma pista em cada sentido na maior parte do trajeto. Cada motorista dirigiu duas vezes nas rotas definidas. Na primeira vez foi solicitado que dirigisse de forma prudente e na segunda de forma imprudente.

Nesta fase do estudo de caso adotou-se a técnica de validação cruzada (*cross-validation*) [121] por dar validade estatística e por ser amplamente usada para avaliar o desempenho dos algoritmos. Para realizar a validação cruzada, o *dataset* é dividido em subconjuntos (*folds*). Todos os subconjuntos, exceto um, são usados para treinamento, e o subconjunto não usado para treino serve para avaliar o desempenho do modelo aprendido. Os subconjuntos são rotacionados e o

desempenho médio é utilizado como medida de desempenho do algoritmo. O processo de validação que utiliza três subconjuntos, *3-fold cross-validation*, foi adotado. Neste processo os dados dos motoristas foram aleatoriamente divididos em duas partes de 35% para treino e uma parte de 30% para teste e checagem do subconjunto de dados que gerou os melhores resultados para cada algoritmo. Para o algoritmo Z-Score Online, uma evidência foi considerada anômala com base no limiar (Z-Score maior que módulo de três) proposto por Chandola, Banerjee e Kumar [14]. Já para o Box Plot Online, uma evidência foi considerada anômala com base na heurística (1,5 x IQR) proposta por Laurikkala, Juhola e Kentala [61]. Por fim, para o K-Means foram definidos dois *clusters*, um para agrupar evidências normais e outro para agrupar evidências anômalas. Além disso, um motorista do *dataset* foi escolhido aleatoriamente, extraíram-se os dados da condução (prudente e imprudente) e os centroides dos *clusters* foram calculados. Assim, o K-Means começa com conhecimento previamente aprendido em vez de escolher valores aleatórios para os centroides dos *clusters*. Por considerar um *cluster* com evidências anômalas, a implementação do K-Means sempre atribuiu uma evidência ao *cluster* mais próximo, desconsiderando o parâmetro *dMax* da Figura 5.

Com o objetivo de ser executável em um dispositivo móvel, as aplicações precisam variar as taxas de envio dos dados dos sensores de acordo com recursos computacionais disponíveis. Dessa forma, os algoritmos precisam adaptar seu comportamento para realizar a detecção das anomalias com boa precisão mesmo com variação na taxa de envio dos dados. Assim, com base nesse cenário, foram definidas duas configurações. Na primeira, a taxa de envio dos dados dos sensores foi h = 100Hz e a janela de tempo Δ = 10s (configuração 1). Já na segunda, a taxa de envio dos dados dos sensores foi h = 50Hz e a janela de tempo Δ = 20s (configuração 2). A Seção 6.1.1 mostra os resultados dos algoritmos para esta configuração.

6.1.1
Avaliação Intrínseca do Modelo de Conhecimento

Nesta subseção, apresentamos os resultados da avaliação dos algoritmos usando o *dataset* citado na Seção 6.1. Como dito anteriormente, foi adotado o processo de validação *3-fold cross-validation* e o nível de confiança é de 95%.

Como mostrado na Tabela 4 e Tabela 5, é possível notar que o *dataset* contém um número bem maior de instância positivas que negativas. Isto corrobora o primeiro pressuposto salientado na Seção 1.7. A partir da matriz de confusão, as métricas de desempenho foram calculadas como definido na Tabela 3. Os três algoritmos tiveram um excelente desempenho, pois o pior resultado classificou corretamente 93,71% das evidências e não houve diferenças significativas (maiores que 1% para o mesmo algoritmo em ambas configurações e maiores que 5% para diferentes algoritmos) nos resultados apresentados na Tabela 6. Apesar do bom desempenho dos algoritmos, o Z-Score Online e o K-Means Online destacaram-se com a maior acurácia média.

Tabela 4. Matriz de confusão por algoritmo e motorista (VP e VN) para configuração 1.

Matriz de Confusão						
Motorista	**Prudente (VP)**			**Imprudente (VN)**		
	K-Means	**Box Plot**	**Z-Score**	**K-Means**	**Box Plot**	**Z-Score**
D1	5990	5004	5143	17	49	40
D2	5642	5439	5593	4	52	28
D3	5876	5667	5817	26	75	39
D4	13465	5618	5716	123	191	45
D5	6114	5024	5201	5	58	41

Tabela 5. Matriz de confusão por algoritmo e motorista (FP e FN) para configuração 1.

Matriz de Confusão						
Motorista	**Prudente (FP)**			**Imprudente (FN)**		
	K-Means	**Box Plot**	**Z-Score**	**K-Means**	**Box Plot**	**Z-Score**
D1	3	19	28	110	175	36
D2	0	34	58	83	201	47
D3	14	40	76	61	191	41
D4	12	14	160	288	131	33
D5	10	27	44	70	206	29

Na configuração 1, Box Plot e Z-Score tiveram praticamente a mesma precisão e K-Means obteve um resultado expressivo com uma precisão média de 99,89%. Por outro lado, na segunda configuração, o Box Plot classificou corretamente 100% das evidências. Apenas o K-Means piorou a precisão como

mostrado na Tabela 7. De acordo com a A Tabela 10 mostra o tempo de execução, em milissegundos, de cada algoritmo, ou seja, o *delay* para classificar uma instância. É possível notar que o Z-Score requer consideravelmente menos tempo de processamento em relação aos outros algoritmos. Como o K-Means requer computação das distâncias entre todas as evidências e os centroides dos *clusters*, possui complexidade quadrática, em contraste com uma complexidade linear do Box Plot e Z-Score. Isso explica o tempo de execução bem maior. Além disso, enquanto Box Plot e Z-Score executam uma única análise do fluxo de dados, o K-Means requer, em média, três iterações por janela de tempo para convergir. O tempo de execução dos algoritmos não mudou significativamente (mais que 1%) nas configurações escolhidas.

Tabela 8, o Z-Score obteve um melhor desempenho da cobertura em ambas as configurações. Isto significa que, na média, o Z-Score atingiu melhores taxas de verdadeiro positivo. O algoritmo K-Means também obteve um excelente resultado. Quanto à Medida F, mais uma vez K-Means e Z-Score obtiveram desempenho semelhante e o Box Plot obteve um desempenho ligeiramente inferior na configuração 1. No entanto, na configuração 2, o Z-Score destacou-se com a Medida F média mais alta, como mostrado na Tabela 9.

Tabela 6. Comparação da acurácia dos algoritmos.

Motorista	**Configuração 1**			**Configuração 2**		
	K-Means	**Box Plot**	**Z-Score**	**K-Means**	**Box Plot**	**Z-Score**
D1	98,15%	96,30%	98,78%	98,05%	93,71%	98,58%
D2	98,55%	95,90%	98,17%	98,16%	96,30%	98,80%
D3	98,75%	96,13%	98,04%	98,40%	96,61%	98,59%
D4	97,84%	97,59%	96,76%	96,12%	97,80%	98,94%
D5	98,71%	95,62%	98,63%	97,86%	97,40%	98,57%
Média	**98,40%**	96,30%	98,07%	97,72%	96,36%	**98,70%**

Tabela 7. Comparação da precisão dos algoritmos.

Motorista	**Configuração 1**			**Configuração 2**		
	K-Means	**Box Plot**	**Z-Score**	**K-Means**	**Box Plot**	**Z-Score**
D1	99,95%	96,22%	99,46%	99,93%	100,00%	99,52%
D2	100,00%	99,38%	98,97%	99,60%	100,00%	100,00%

D3	99,76%	99,30%	98,71%	99,61%	100,00%	100,00%
D4	99,76%	99,75%	97,28%	99,89%	100,00%	100,00%
D5	99,84%	99,47%	99,16%	99,71%	100,00%	100,00%
Média	**99,89%**	98,82%	98,72%	99,75%	**100,00%**	99,90%

A Tabela 10 mostra o tempo de execução, em milissegundos, de cada algoritmo, ou seja, o *delay* para classificar uma instância. É possível notar que o Z-Score requer consideravelmente menos tempo de processamento em relação aos outros algoritmos. Como o K-Means requer computação das distâncias entre todas as evidências e os centroides dos *clusters*, possui complexidade quadrática, em contraste com uma complexidade linear do Box Plot e Z-Score. Isso explica o tempo de execução bem maior. Além disso, enquanto Box Plot e Z-Score executam uma única análise do fluxo de dados, o K-Means requer, em média, três iterações por janela de tempo para convergir. O tempo de execução dos algoritmos não mudou significativamente (mais que 1%) nas configurações escolhidas.

Tabela 8. Comparação da cobertura dos algoritmos.

Motorista	**Configuração 1**			**Configuração 2**		
	K-Means	**Box Plot**	**Z-Score**	**K-Means**	**Box Plot**	**Z-Score**
D1	98,20%	96,62%	99,30%	98,10%	96,35%	98,91%
D2	98,55%	96,44%	99,17%	98,52%	96,30%	98,80%
D3	98,97%	96,74%	99,30%	98,77%	98,86%	98,60%
D4	97,91%	97,72%	99,43%	96,19%	97,80%	98,94%
D5	98,87%	96,06%	99,45%	98,12%	97,40%	98,63%
Média	98,50%	96,72%	**99,33%**	97,94%	97,34%	**98,78%**

Tabela 9. Comparação da medida F dos algoritmos.

Motorista	**Configuração 1**			**Configuração 2**		
	K-Means	**Box Plot**	**Z-Score**	**K-Means**	**Box Plot**	**Z-Score**
D1	99,07%	98,10%	99,38%	99,00%	98,17%	99,45%
D2	99,27%	97,89%	99,07%	99,06%	98,06%	99,30%
D3	99,37%	98,00%	99,00%	99,19%	98,06%	99,20%
D4	98,90%	98,73%	98,34%	98,00%	99,19%	99,42%
D5	99,35%	97,73%	99,30%	98,91%	98,68%	99,30%

Média	**99,19%**	98,09%	99,02%	98,83%	98,43%	**99,33%**

Tabela 10. Comparação dos tempos de execução dos algoritmos em milissegundos (ms).

Motorista	**Configuração 1**			**Configuração 2**		
	K-Means	**Box Plot**	**Z-Score**	**K-Means**	**Box Plot**	**Z-Score**
D1	769,6 ms	132,0 ms	100,2 ms	712,0 ms	129,8 ms	98,3 ms
D2	700,0 ms	187,0 ms	103,1 ms	709,4 ms	186,4 ms	99,0 ms
D3	706,4 ms	189,9 ms	100,9 ms	705,5 ms	190,2 ms	99,7 ms
D4	705,3 ms	186,0 ms	101,5 ms	702,3 ms	186,8 ms	100,6 ms
D5	787,0 ms	188,0 ms	102,7 ms	780,4 ms	190,1 ms	100,3 ms
Média	733,6 ms	176,4 ms	**101,6** ms	721,9 ms	176,6 ms	**99,6** ms

Para verificar o consumo dos recursos dos algoritmos em ambas as configurações, primeiro verificou-se a memória do *smartphone* e o uso da unidade central de processamento (*Central Processing Unit* – CPU) em duas situações: em espera (*standby*) e coletando dados dos sensores do próprio *smartphone* e do dispositivo OBD-II, no entanto, sem processá-los. Através da Tabela 11 é possível observar que somente a coleta de dados aumenta o consumo da memória em 12,54% e o uso da CPU em 60,83%, mas ainda com um baixo consumo geral de recursos, visto que a utilização de memória em espera e em coleta de dados equivale, respectivamente, 0,55% e 0,62% da memória disponível. Em seguida, analisando os dados da Tabela 12, evidencia-se que o Z-Score supera positivamente o K-Means e o Box Plot em ambas as configurações. O K-Means destacou-se negativamente em termos de uso da CPU. Além disso, uma janela de tempo maior resultou em um maior uso de memória e CPU.

Tabela 11. Consumo de recursos do smartphone.

Situação	**RAM (MB)**	**CPU (%)**
Em espera	4,53 MB	1,41%
Coletando	5,18 MB	3,60%

Tabela 12. Comparação do consumo de recursos dos algoritmos.

Algoritmo	**Configuração 1**		**Configuração 2**	
	RAM (MB)	**CPU (%)**	**RAM (MB)**	**CPU (%)**
K-Means	6,75 MB	20,20%	7,83 MB	23,35%
Box Plot	6,30 MB	11,40%	7,30 MB	11,45%

Z-Score	***6,15 MB***	***6,61%***	***6,40 MB***	***7,46%***

Tabela 13. Comparação da taxa de erro dos algoritmos.

Motorista	**Configuração 1**			**Configuração 2**		
	K-Means	**Box Plot**	**Z-Score**	**K-Means**	**Box Plot**	**Z-Score**
D1	1,85%	3,70%	1,22%	1,35%	3,20%	1,00%
D2	1,45%	4,10%	1,83%	1,50%	3,40%	0,93%
D3	1,25%	3,87%	1,96%	1,17%	2,80%	1,80%
D4	2,16%	2,44%	3,24%	1,50%	2,00%	1,20%
D5	1,29%	4,38%	1,37%	1,45%	3,80%	1,10%
Média	**1,60%**	3,70%	1,93%	1,39%	3,04%	**1,21%**

Após revisão dos resultados supracitados e calcular a taxa média de erro dos algoritmos, mostrada na Tabela 13, o K-Means e Z-Score destacaram-se com taxas de erro médias mais baixas. Ambos tiveram taxas médias semelhantes, mas, por um lado, o K-Means obteve melhor desempenho na configuração 1 e, por outro lado, o Z-Score obteve melhor desempenho na configuração 2. Portanto, em relação às métricas de desempenho, o K-Means e o Z-Score obtiveram resultados semelhantes (diferença entre 0,1% e 1,1%). No entanto, em geral, o K-Means obteve melhores resultados na configuração 1 e Z-Score na configuração 2. A única exceção foi na cobertura em que o Z-Score obteve melhor resultado em ambas as configurações. No que se refere às métricas de qualidade, o tempo de execução médio de Z-Score foi cerca de sete vezes menor do que o K-Means e duas vezes menor que o Box Plot. Além disso, o Z-Score requereu, em média, 14% menos de memória que o K-Means e 8% menos que Box Plot. Por fim, o Z-Score requereu, em média, uso da CPU 68% menor que o K-Means e 38,5% menor que o Box Plot.

6.1.2 Comparação

Com o intuito de comparar a abordagem desta tese para classificação do comportamento de condução e, portanto, mostrar seus benefícios, três outros trabalhos relacionados foram implementados. Uma versão simplificada do VEDAS [94] foi desenvolvida, no entanto, a versão *online* do algoritmo *Piecewise Linear Approximation* [95] foi implementado, através de regras CEP. Já o

trabalho de Aljaafreh, Alshabatat e Najim [99] que utiliza inferência por Lógica Fuzzy para identificação *online* de dados de condução anormais foi implementado utilizando o Fuzzylite [122], um motor de controle de lógica fuzzy livre e de código aberto programada em C++ para múltiplas plataformas, inclusive *Android*. Por fim, a proposta de Quintero, Lopez e Cuervo [16] foi implementada. Contudo, diferentemente da proposta inicial cujo processamento das variáveis do sistema Fuzzy é feito *offline* por uma rede neural (RN) em um servidor remoto, todo o processamento ocorreu no *smartphone* através da implementação de uma rede neural utilizando o Neuroph [123], um *framework lightweight* que facilita a implementação de redes neurais de código aberto e com versão para *Android*. O processo de avaliação do desempenho destes trabalhos seguiu os passos descritos na Seção 6.1.

A Tabela 14 exibe as métricas de desempenho dos algoritmos. Para o K-Means, Box Plot, Z-Score e Piecewise Linear Approximation (VEDAS) que foram implementados por meio de regras CEP, mostra-se a média obtida na configuração 1 e configuração 2. Comparando os resultados nota-se que a abordagem utilizando redes neurais obteve um desempenho superior em todas as métricas avaliadas, exceto a precisão. Além disso, com rede neural obteve-se uma taxa de erro extremamente baixa. Contudo, o Z-Score teve um desempenho muito próximo da rede neural visto que, em termos percentuais, para cada métrica a diferença sempre foi menor ou igual a 1%.

Tabela 14. Comparação das métricas de desempenho.

Algoritmo	**Métricas Desempenho**				
	Acurácia	Cobertura	Precisão	Medida F	Taxa de Erro
K-Means	98,06%	98,22%	**99,82%**	99,01%	**1,50%**
Box Plot	96,33%	97,03%	99,41%	98,26%	3,37%
Z-Score	**98,39%**	**99,06%**	99,31%	**99,18%**	1,57%
VEDAS [94]	97,61%	98,55%	99,02%	98,77%	2,38%
Fuzzy [99]	98,22%	99,84%	98,36%	99,10%	1,78%
RN [16]	**99,34%**	**100,00%**	99,43%	**99,72%**	**0,57%**

Tabela 15. Comparação das métricas de qualidade.

Algoritmo	**Métricas Qualidade**		
	RAM(MB)	CPU(%)	Tempo (ms)

K-Means	7,29 MB	21,78%	727,75 ms
Box Plot	6,80 MB	11,43%	176,50 ms
Z-Score	**6,27 MB**	**7,04%**	**100,60 ms**
VEDAS [94]	6,34 MB	26,39%	501.00 ms
Fuzzy [99]	6,99 MB	26,67%	**10.11 ms**
RN [16]	13,72 MB	27,22%	11.21 ms

Comparando-se as métricas de qualidade da Tabela 15, nota-se que o consumo de memória do Z-Score foi 1,10%, 10,30% e 54,30% mais eficiente que VEDAS, Fuzzy e Redes Neurais, respectivamente. Das abordagens que analisam um conjunto de instâncias pertencentes a uma janela de tempo para determinar quais delas são anômalas, o Z-Score indubitavelmente obteve o melhor resultado.

Tabela 16. Comparação das características dos trabalhos relacionados.

Característica	VEDAS	Lógica Fuzzy	Redes Neurais	CEP + Anomalia
Característica do Algoritmo	Algoritmo de aproximação simples e intuitivo [95]	A formulação de regras *fuzzy* são baseadas na experiência anterior e então projeta-se o modelo com base nas expectativas do projetista.	Para identificar tipos de comportamentos de condução necessita de uma quantidade relativamente grande de dados [33]. A qualidade dos parâmetros é crucial para a precisão da RN.	CEP fornece um modelo de processamento em memória de fluxos de dados com primitivas que permitem filtrar, agregar e transformar eventos. É preciso definir a região que representa dados normais. As técnicas de detecção de anomalia isolam o ruído naturalmente.
Acurácia do Modelo	Alta	Alta	Muito Alta	Alta
Desempenho em Tempo-Real	Baixo com janela deslizante e Alto usando janela em lote [95]	Razoável	Razoável [35]	Excelente
Adaptação aos dados	Não	Não	Não	Sim
Desvantagens	O desempenho do algoritmo depende do limiar de erro especificado. Uma tarefa subjetiva e dependente de aplicação	As regras *fuzzy* são formuladas apenas com base em um conhecimento prévio [33]. Pode ser complexo a formulação de regras em aplicações com	Não há um método viável unificado para ajustar os parâmetros (por exemplo, o número de camadas), possui tempo de treinamento longo e requer número	É difícil expressar controles iterativos utilizando suas primitivas.

	[95]	um grande número de parâmetros de entrada.	fixo de parâmetros de entrada [33], [36].	

As abordagens com Lógica Fuzzy e Redes Neurais classificaram uma instância em anômala ou normal, respectivamente, em ≈10 e ≈11 milissegundos. No entanto, considerando uma taxa de atualização de um sensor a 100Hz e uma janela de tempo de 20s tem-se 2.000 instâncias a serem analisadas. O Z-Score leva ≈100 milissegundos para analisar todos estes dados e classificar uma instância em normal ou anômala – o que dá uma média de 0,05 milissegundo por instância analisada. Através desse raciocínio, o Z-Score necessita de menos tempo para processar uma instância. No tocante ao uso da CPU, Box Plot e Z-Score obtiveram os melhores desempenhos. Entretanto, o Z-Score consumiu 73,32%, 73,60% e 74,13% menos CPU, respectivamente, em relação ao VEDAS, Fuzzy e Rede Neural. A Tabela 16 exibe um resumo comparativo das características dos modelos de detecção do comportamento de condução usados nos trabalhos relacionados e na abordagem desta tese.

6.2 Execução

O *smartphone* foi colocado no centro do vidro frontal do veículo, como mostrado na Figura 28. O leitor OBD-II foi conectado à porta OBD-II para coletar uma variedade de dados a partir do barramento de dados (CAN) do veículo. O leitor OBD-II envia os fluxos de dados via Bluetooth para o *smartphone*. A execução do estudo de caso consistiu na realização do processo de detecção de anomalia dos fluxos de dados gerados à partir da condução dos motoristas voluntários.

Figura 28. Posição do *smartphone*

6.2.1
Coleta dos Dados

Os dados de comportamento dos motoristas foram coletados em sete dias ensolarados e os motoristas dirigiram entre as 9 e as 20 horas. Cada motorista dirigiu uma vez na rota escolhida. Portanto, um total de 362,5 km foram percorridos compreendendo 12,5 horas de condução. Além disso, antes de cada motorista voluntário começar a dirigir foi explicado o propósito do estudo de caso e que as informações pessoais seriam tratadas com absoluta confidencialidade. Em seguida, a rota foi explicada detalhadamente e foi solicitado ao motorista dirigir como habitualmente o faria. Também foi informado ao motorista voluntário que um motorista especialista com 15 anos de experiência o acompanharia durante o estudo de caso – de forma semelhante ao teste para aquisição da carteira nacional de trânsito – mas enfatizamos que o objetivo era analisar e classificar o comportamento do motorista em prudente ou imprudente para fins acadêmicos e não aprová-lo ou desaprová-lo. Esta classificação serviu como *ground truth*.

O protótipo coleta dados dos sensores do *smartphone* (i.e., acelerômetro, giroscópio, bússola magnética e GPS) e dos sensores do veículo (velocidade, RPM e posição relativa do acelerador) através do leitor OBD-II. O Mobile Hub [101], um middleware móvel genérico para comunicação de curto alcance, foi usado para identificar os sensores disponíveis a bordo do veículo e realizar a coleta dos dados.

6.3
Avaliação Extrínseca do Modelo de Conhecimento

De acordo com Bramer [119], não há maneira certa e infalível para encontrar o melhor classificador de dados para uma dada aplicação. No entanto, no caso dessa pesquisa, o Z-Score obteve um desempenho melhor com relação às métricas de qualidade de classificação apresentadas na Seção 5.4 e com relação às métricas de desempenho, o Z-Score e o K-Means alcançaram um desempenho próximo como mostrado na Seção 6.1.1. Por esta razão, o algoritmo Z-Score Online foi utilizado para analisar os comportamentos dos condutores no estudo de caso. Além disso, os dados foram analisados apenas quando a velocidade foi maior ou igual a 5 km/h – abaixo desta velocidade espera-se que não haja mudanças significativas nos dados dos sensores.

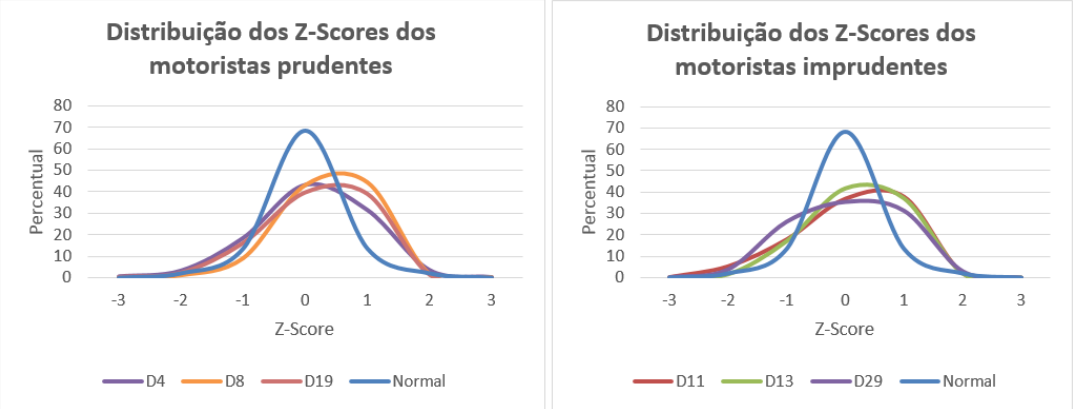

Figura 29. Comparação da distribuição dos Z-Scores da velocidade.

Hong, Margines e Dey [52], coletaram dados de condução de 22 motoristas classificando-os de acordo com uma escala de agressividade. Os autores concluíram que não há diferenças significativas em relação à velocidade adotada entre motoristas prudentes e imprudentes. Analisando a distribuição dos Z-Scores dos motoristas (prudente e imprudente) de forma *offline*, ou seja, analisando todo o *dataset* através do Z-Score clássico, realmente não foi possível constatar diferenças significativas como mostrado na Figura 29. A Figura 29 exibe a distribuição dos Z-Scores de três motoristas prudentes e imprudentes. Vale ressaltar que o gráfico exibe apenas três motoristas para ficar visualmente mais fácil identificar a distribuição, mas para os demais motoristas o padrão de distribuição é similar. No entanto, através de uma análise do Z-Score Online, ou seja, analisando o fluxo de dados através de regras CEP, é possível identificar

manobras imprudentes que resultam em mudanças significativas na distribuição dos Z-Scores, como mostrado na Figura 30.

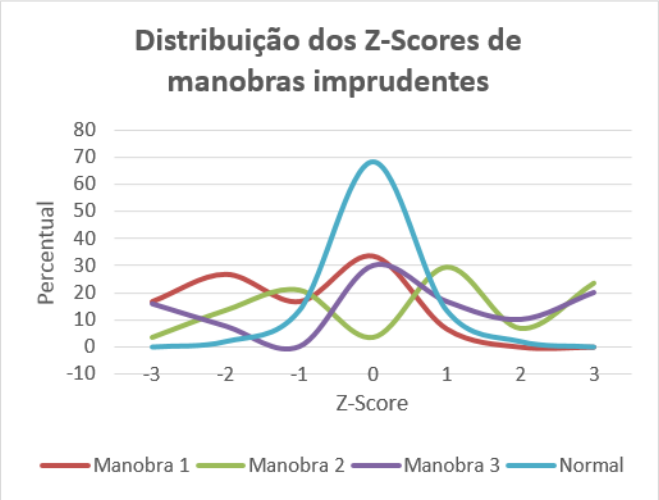

Figura 30. Comparação da distribuição da velocidade do Z-Score Online.

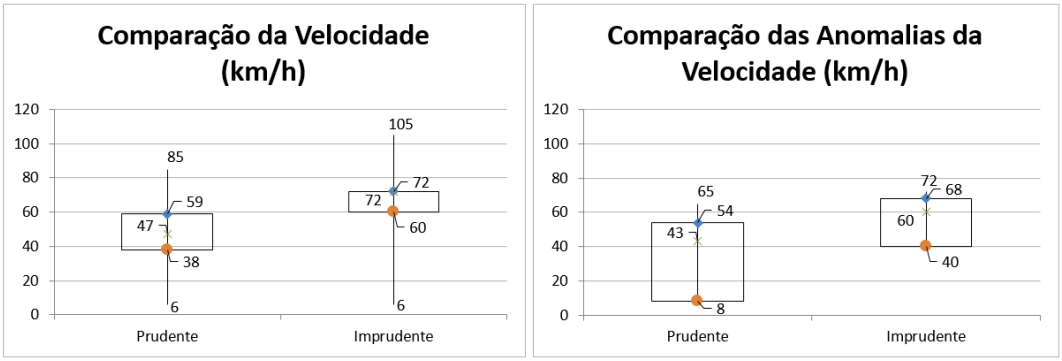

Figura 31. Comparação *offline* da velocidade (esquerda) e comparação *online* da velocidade (direita).

Conforme mostrado na Figura 31, motoristas prudentes e imprudentes geram evidências anômalas. Contudo, ao realizar análise *offline* das evidências de velocidade, como mostrado na Figura 31 (à esquerda), é possível notar que mais de 75% das evidências dos motoristas prudentes ocorreram em velocidades inferiores a 60 km/h (limite de velocidade) e 50% foram inferiores a 47 km/h. Isto significa que os motoristas classificados como prudentes pelo especialista permaneceram abaixo do limite de velocidade na maior parte do tempo. Embora excesso de velocidade seja uma violação grave, o especialista não percebeu ou não considerou este comportamento como imprudência em alguns momentos. Por outro lado, os motoristas imprudentes permaneceram acima do limite de velocidade na maior parte do tempo, isto é, apenas 25% das evidências de velocidade estavam abaixo de 60 km/h. Um ponto interessante reside no fato de que IQR dos motoristas prudentes é maior do que os imprudentes. Isto significa

que os motoristas prudentes tiveram uma variabilidade maior da velocidade e os imprudentes, mesmo permanecendo acima do limite de velocidade em 75% das evidências coletadas, tiveram pouca variabilidade na velocidade.

Além disso, a análise *online* da velocidade, isto é, avaliando a saída do algoritmo Z-Score *Online* (Figura 31 – à direita), observa-se que mudanças repentinas na velocidade (anomalias) realizadas pelos motoristas prudentes ocorreram em velocidades relativamente baixas. Metade destas mudanças ocorreram em uma faixa entre 8 e 43 km/h e 25% ocorreram em uma faixa entre 43 e 54 km/h. Por outro lado, motoristas imprudentes não tiveram mudanças súbitas em velocidades inferiores a 40 km/h. No entanto, em 50% anomalias relacionadas à velocidade ocorreram acima de 60 km/h. Os dados mostram que a velocidade é boa opção para classificar o comportamento do motorista e a abordagem de detecção *online* de anomalia é uma boa alternativa para classificar motoristas em prudentes ou imprudentes visto que há um padrão significativamente diferente na distribuição das velocidades como visto na Figura 31 (à direita).

A Figura 32 mostra a distribuição média dos Z-Scores da rpm de três motoristas prudentes (à esquerda) e imprudentes (à direita). Este é o comportamento comum para ambas as classes de motoristas. Os demais motoristas foram omitidos para facilitar a identificação do padrão da distribuição. Além disso, em relação à rpm, analisando todo o *dataset* com o Z-Score clássico, não há diferenças significativas entre as duas classes de motoristas. No entanto, há uma notável diferença de distribuição dos Z-Scores nas manobras imprudentes durante a análise do Z-Score Online (abordagem com CEP), como mostrado na Figura 33. Por exemplo, nas manobras um e dois, o total de evidências anômalas é de 17% e 23,5%, respectivamente.

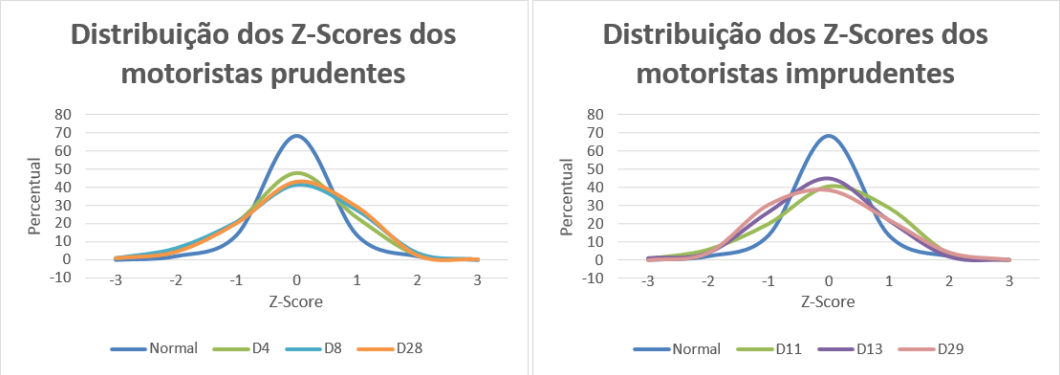

Figura 32. Comparação da distribuição dos Z-Scores da rpm.

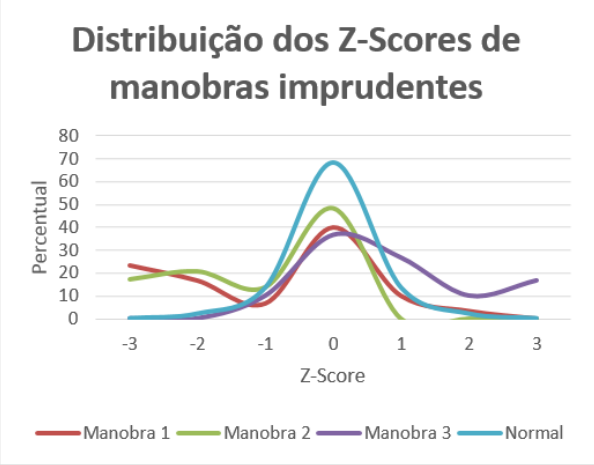

Figura 33. Comparação da distribuição da rpm do Z-Score Online.

Além disso, na análise *offline* mostrada na Figura 34 (à esquerda), as rpm dos condutores prudentes variaram de 1.678 a 1.960 rpm em 50% das vezes, e em 75% das evidências de rpm não excedeu 2.091. Os motoristas imprudentes tiveram uma variação minúscula da rpm e o valor do *Q1* dos imprudentes é 18,55% maior do que o valor do *Q3* dos prudentes. Provavelmente, em uma tentativa de poupar tempo, os motoristas imprudentes ocasionaram rpm mais altas para alcançar velocidades mais elevadas o mais rapidamente possível. No entanto, a Figura 34 (à direita) mostra que as evidências de rpm dos motoristas imprudentes, classificadas pelo algoritmo Z-Score Online como anômalas, têm uma distribuição mais ampla. Vinte e cincos por cento das mudanças bruscas da rpm dos motoristas imprudentes ocorreram entre 3.171 e 4.622 rpm e 50% das evidências anômalas de rpm variou de 1.250 a 3.171 rpm. Estes são valores elevados de acordo com a documentação do fabricante do veículo e demonstram que os motoristas imprudentes têm um estilo de condução inconsistente.

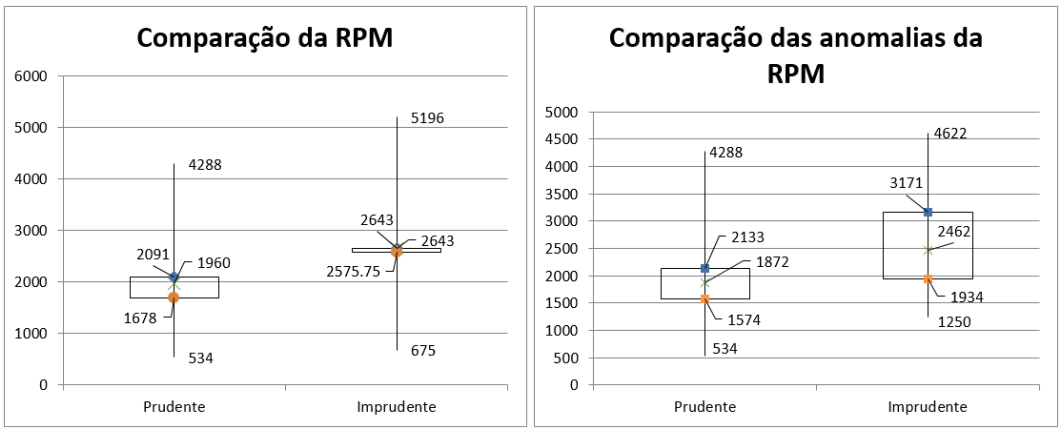

Figura 34. Comparação da rpm.

A Figura 35 mostra que a distribuição da posição (em percentual) do uso do acelerador dos motoristas prudentes e imprudentes está próxima da distribuição normal. Assim, não é possível identificar diferenças entre prudente e imprudente com uma visão *offline* das evidências. No entanto, o algoritmo Z-Score Online identificou uma distribuição bastante diferente para manobras imprudentes, como mostra a Figura 36. Por exemplo, as manobras 1 e 2 tiveram, respectivamente, 32,67% e 31,14% das evidências classificadas como anômalas.

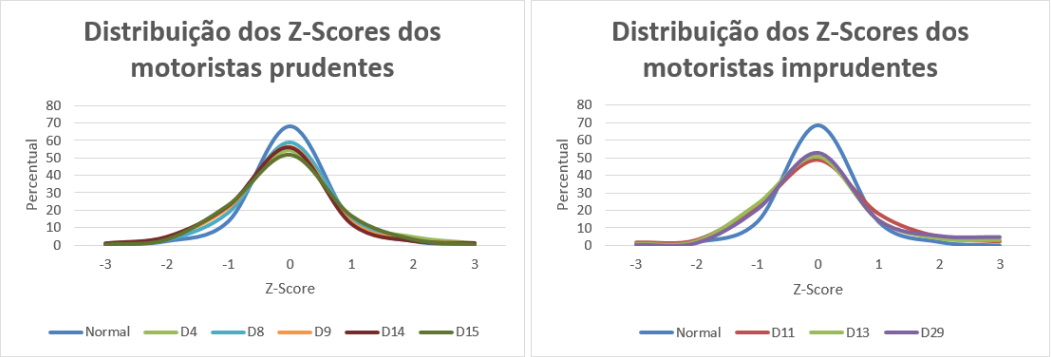

Figura 35. Comparação da distribuição dos Z-Scores da posição relativa do acelerador.

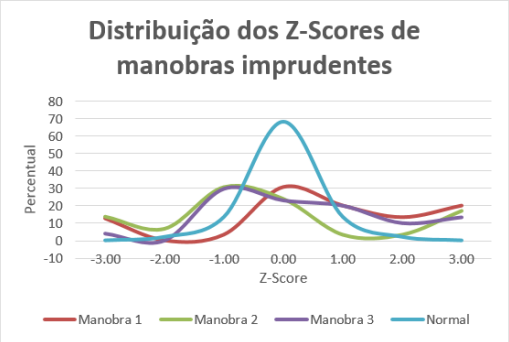

Figura 36. Comparação da distribuição do uso do acelerador do Z-Score Online.

Corroborando com os dados supracitados, a Figura 37 (à esquerda) mostra uma análise *offline* do uso do acelerador. Observa-se um uso consistente, especialmente os motoristas imprudentes, já que o valor do *IQR* é extremamente pequeno. No entanto, identificando anomalias da posição do acelerador através do algoritmo Z-Score Online, é possível notar, na Figura 37 (à direita), que mudanças suaves dos motoristas prudentes foram identificadas como anomalias. Contudo, é importante destacar que aproximadamente 0,1% das evidências da posição do acelerador foram superiores a 60% do uso do acelerador. Além disso, os motoristas imprudentes pressionam o acelerador de forma mais agressiva uma vez que o *Q1* dos imprudentes tem valor próximo ao *Q3* dos prudentes e 50% das

anomalias ocorreram entre 35,88% e 87,45% de uso. Certamente, este comportamento terá impacto no consumo de combustível e na emissão de CO_2.

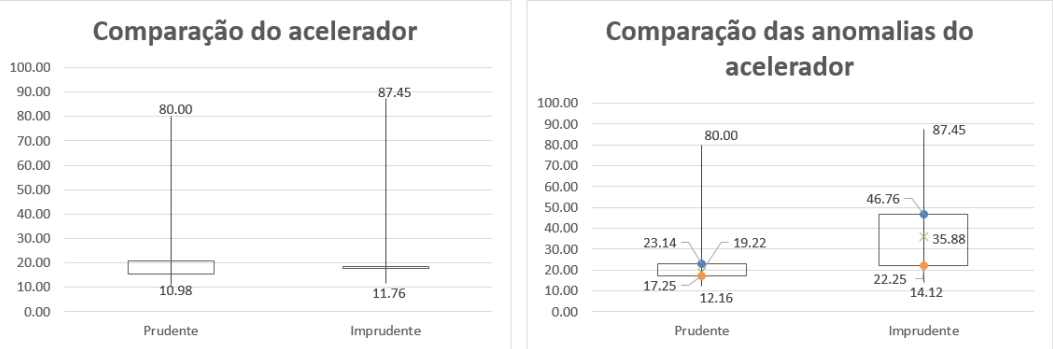

Figura 37. Comparação do uso do acelerador.

Analisando a aceleração (considerando os três eixos), nota-se que a distribuição dos Z-Scores dos motoristas prudentes e imprudentes é praticamente igual à curva normal, como mostrado na Figura 38. Ao contrário de outras pesquisas, como a de Hong, Margines e Dey [52] que consideram apenas a aceleração lateral e longitudinal para classificação do comportamento de condução dos motoristas, esta tese considera os três eixos porque, no Brasil, boa parte das vias têm má qualidade e, portanto, acredita-se retratar mais fielmente o cenário brasileiro do estado das ruas. No entanto, indo contra aos resultados obtidos por Hong, Margines e Dey [52], na avaliação do comportamento do motorista, não foram notadas mudanças significativas na distribuição dos Z-Scores da aceleração, mesmo em manobras imprudentes, como mostrado na Figura 39. Por exemplo, ao contrário dos dados supracitados, as manobras 1 e 2 tiveram, respectivamente, apenas 9,37% e 7,86% de evidências classificadas como anomalias. No entanto, Zhao *et al.* [51] propuseram um mecanismo avaliar os motoristas, com base em parâmetros estabelecidos pela ISO 2631-1-1997 [96], que avalia os efeitos da exposição humana à aceleração, medindo o nível de conforto/desconforto dos passageiros. À partir destes parâmetros, e considerando que no estudo de caso desta tese um veículo parado tem aceleração igual a 9,8 m/s², assim, os passageiros sentem-se confortáveis enquanto a aceleração estiver entre 8,9 m/s² e 11,2 m/s². A Figura 40 mostra uma comparação entre a análise *offline* (à esquerda) e a detecção anomalia *online* (à direita). Na análise *offline* é possível notar que todos os motoristas estiveram dentro do intervalo confortável em mais de 50% das evidências uma vez que os *Q1* e *Q3* estão dentro da faixa confortável.

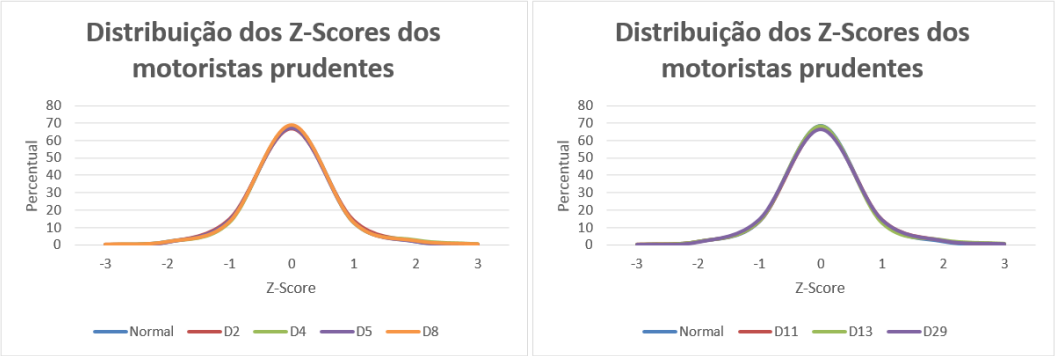

Figura 38. Comparação distribuição dos Z-Scores da aceleração.

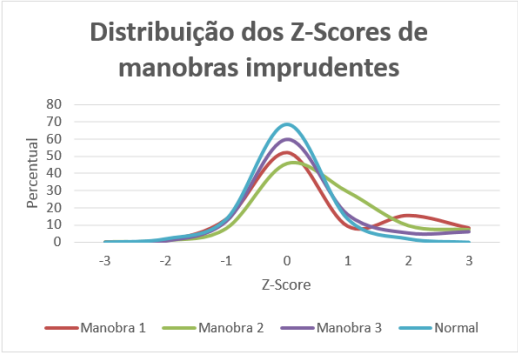

Figura 39. Comparação da aceleração do Z-Score Online.

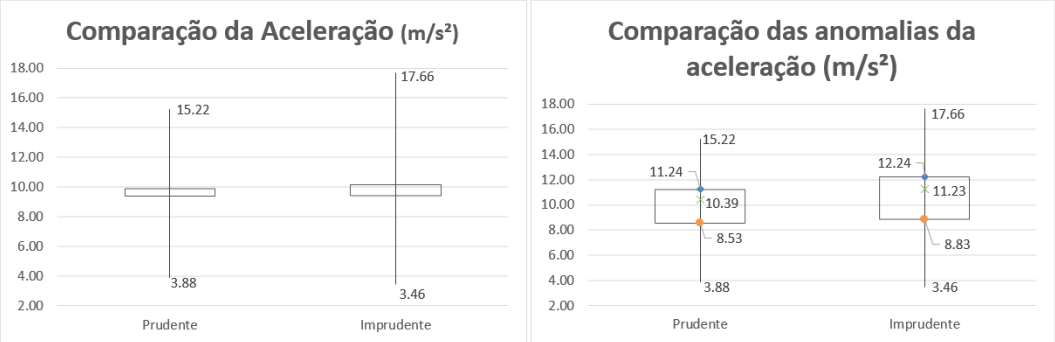

Figura 40. Comparação da aceleração.

No entanto, através da análise *online* é possível notar que eventos dos motoristas imprudentes, como mudanças bruscas de faixa, acelerações/ desacelerações abruptas ou qualquer movimento brusco, geraram desconforto para os passageiros (reportado pelo motorista especialista), uma vez que 75% das anomalias relacionadas à aceleração estavam no intervalo considerado desconfortável. Por outro lado, para os motoristas prudentes, praticamente 50% das anomalias estavam na faixa considerada confortável. Além disso, os valores das evidências anômalas dos motoristas imprudentes são, em média, 36,8% maiores que as dos motoristas prudentes. A Figura 41 mostra 20 segundos de dados de aceleração em que é possível notar 9 pontos anômalos, isto é, Z-Score

superior a +3 ou inferior a −3. Destes, dois pontos estavam no intervalor confortável supracitado. No entanto, houve uma variação da aceleração que embora dentro do intervalo considerado confortável, foi relatada como desconfortável pelo motorista especialista. Este comportamento repetiu-se em outros conjuntos de evidências. Esta é uma das vantagens da abordagem desta tese – o algoritmo analisa todo o contexto da janela de tempo e não apenas os valores individuais.

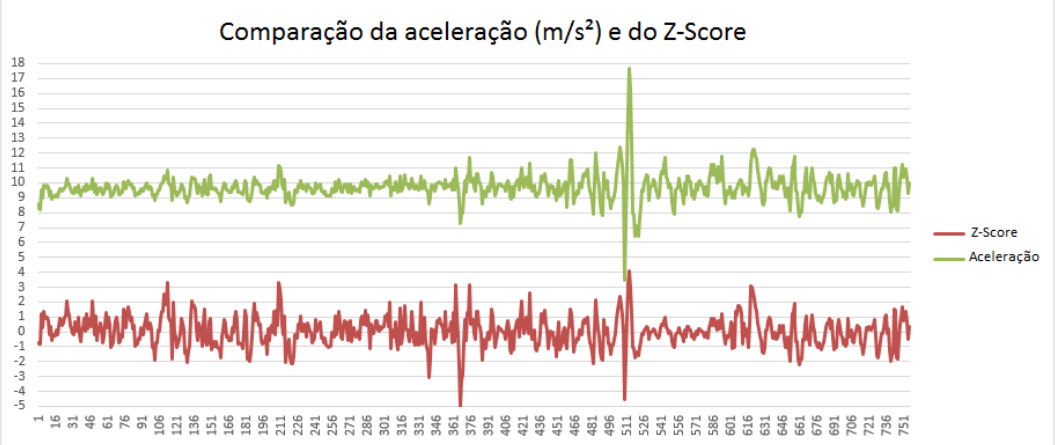

Figura 41. Análise da aceleração e Z-Score.

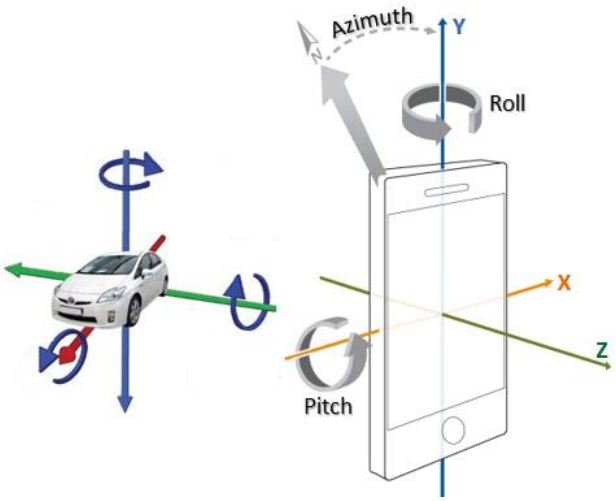

Figura 42. Orientação do smartphone no veículo.

Além dessas informações, foram analisadas mudanças nos ângulos de orientação do veículo a partir da combinação dos dados da bússola magnética e do acelerômetro *smartphone*. Usando estes dois sensores, obtiveram-se três ângulos de orientação: *Pitch* (grau de rotação em torno do eixo x), *Roll* (grau de rotação em torno do eixo y) e *Azimuth* (grau de rotação em torno do eixo z) como mostrado na Figura 42. A Figura 43, Figura 44 e Figura 45 mostram, respectivamente, a distribuição dos Z-Scores em manobras imprudentes do *Pitch*,

Roll e *Azimuth*. No entanto, em relação às evidências anteriormente citadas, percebeu-se um acréscimo significativo no número de evidências anômalas por janela de tempo independente da classificação do motorista. Por exemplo, *Azimuth*, *Roll* e *Pitch* geraram, respectivamente, aproximadamente 53%, 43% e 5% mais evidências anômalas que as demais evidências. Portanto, precisam ter um peso menor no processo de pontuação, explicado na Seção 6.3.1, do comportamento dos motoristas.

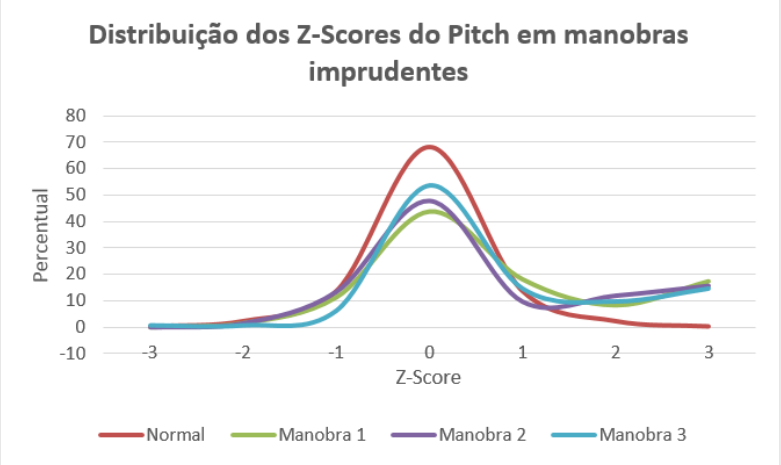

Figura 43. Comparação do Pitch do Z-Score Online.

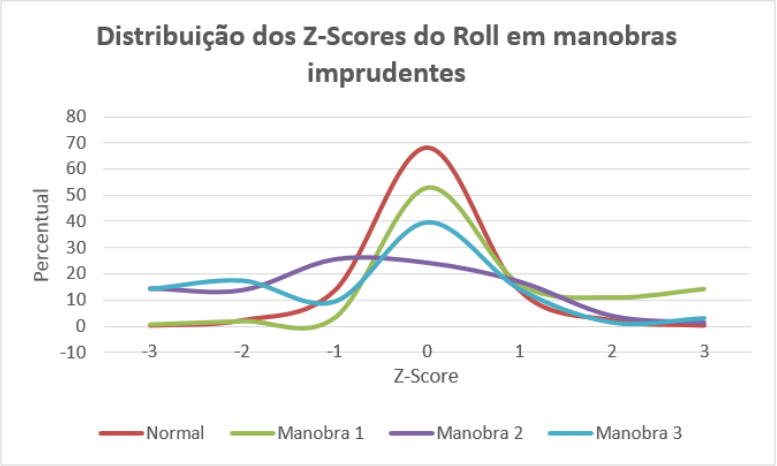

Figura 44. Comparação do Roll do Z-Score Online.

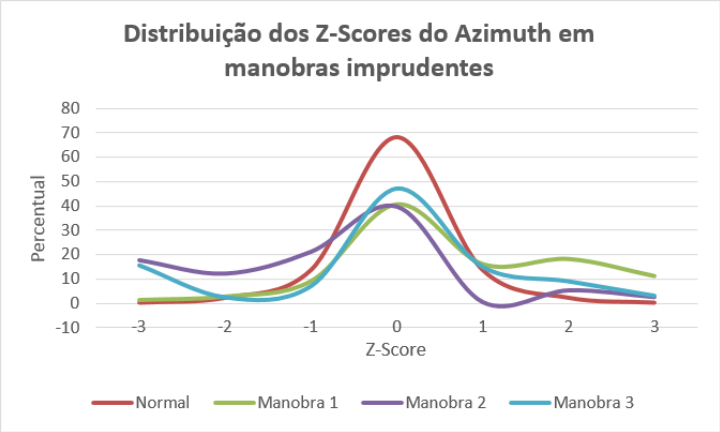

Figura 45. Comparação do Azimuth do Z-Score Online.

6.3.1
Pontuação dos Comportamentos de Condução

A fim de pontuar o comportamento dos motoristas é necessário considerar que (i) os sensores têm taxas de aquisição diferentes. Por exemplo, neste estudo de caso, a taxa de aquisição média do dispositivo OBD-II, do acelerômetro e da bússola magnética do *smartphone* foi, respectivamente, 8 Hz, 140 Hz e 100 Hz. Assim, durante o processamento do fluxo de dados teremos, por exemplo, 17,5 vezes mais evidências de aceleração do que de velocidade, (ii) certas evidências podem ter pouco poder para discriminar o comportamento do motorista e (iii) a aplicação pode diminuir a taxa de envio dos dados dos sensores de acordo com os recursos disponíveis. Para considerar esses três itens, um mecanismo estatístico usado na mineração de documentos para avaliar a importância de uma palavra em um documento, chamado de frequência inversa de documento (*inverse document frequency*) [124], foi adaptado para identificar a importância de uma anomalia no fluxo de dados.

A frequência de uma anomalia (*outlier frequency* – of_d) é definida como o número de anomalias que ocorrem em uma dimensão *d*. Além disso, a frequência inversa de anomalia (*inverse outlier frequency* – iof_d) na dimensão *d* é definida de acordo com a equação (5), onde n_d é o total de evidências pertencentes a uma janela de tempo e uma dimensão *d*.

$$iof_d = \log\left(\frac{n_d}{of_d}\right) \tag{5}$$

Assim, a iof_d de uma evidência anômala rara é alta, enquanto a iof_d de uma evidência anômala frequente é baixa. A fim de ponderar cada evidência

pertencente a uma janela de tempo, combinou-se a definição de frequência de anomalia e frequência inversa de anomalia (*outlier frequency e inverse outlier frequency – ofiof*) como mostrado na equação (6), onde *v* é o *valor da evidência anômala* e *d* é a *dimensão*. Portanto, a pontuação de uma viagem de um motorista é dada pela média ponderada de todas *ofiof* como mostrado na equação (7), onde *t* é o número de janelas de tempo durante a viagem.

$$ofiof_{v,d} = v_d * iof_d \tag{6}$$

$$Score = average\left(\left(\sum_{i=1}^{t}\left(ofiof_{v,d}\right)\right) \Big/ \sum_{i=1}^{t}\left(iof_d\right)\right) \tag{7}$$

A Figura 46 e Figura 47 mostram, respectivamente, a implementação destas equações por meio de regras EPL e a pontuação dos motoristas (*reckless_score*) com base nas evidências anômalas na rota escolhida. Para este estudo de caso, os motoristas com pontuação superior a 50 foram classificados como imprudentes. Este limiar foi escolhido através da análise dos dados de outros seis motoristas que dirigiram na mesma rota, mas para três deles foi solicitado que dirigisse prudentemente e os outros três imprudentemente. A pontuação máxima para os motoristas prudentes foi 35 e o mínimo para os imprudentes foi 65. Portanto, considerou-se o limiar igual a 50 como o limite superior na classificação dos motoristas prudentes e como o limite inferior na classificação dos imprudentes. Comparando a classificação do algoritmo com o *ground truth*, nota-se um excelente desempenho como mostrado Tabela 17.

```
1.  INSERT INTO SCORING_EVENT
2.  SELECT rawValue, z_score,
3.        (Math.log10(CASE WHEN COUNT(*, z_score > 3 OR z_score < -3) = 0 THEN 1 ELSE
4.                 CAST( COUNT(*) AS FLOAT) / COUNT(*, z_score > 3 OR  z_score < -3)
5.            END)) AS iof
6.  FROM Z_SCORE_EVENT.win:time_batch(windowLength sec)
7.  GROUP BY dimension                                                              a
8.  ------------------------------------------------------------------------------
9.  SELECT avg(SUM(rawValue  * iof, z_score > 3 OR z_score < -3 ) / (CASE WHEN SUM(iof) = 0 THEN 1
10.                                                                    ELSE SUM(iof)
11.                                                                END)) AS reckless_score,
12.
13.       avg(SUM(rawValue  * iof, z_score <= 3 AND z_score >= -3 ) / (CASE WHEN SUM(iof) = 0 THEN 1
14.                                                                    ELSE SUM(iof)
15.                                                                END)) AS cautious_score
16. FROM SCORING_EVENT.win:time_batch(windowLength sec)                             b
```

Figura 46. Regras EPL para computação da pontuação do motorista

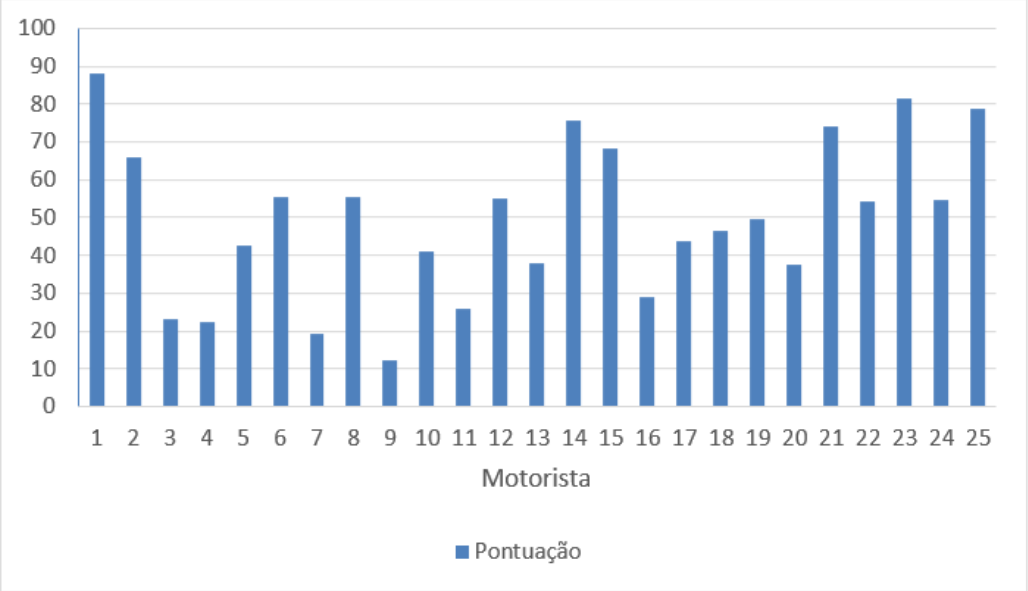

Figura 47. Pontuação dos motoristas.

Tabela 17. Desempenho do Z-Score Online durante o estudo de caso.

Métrica	**Valor**
Acurácia	84,00%
Cobertura	76,47%
Precisão	100,00%
Medida F	86,67%
Taxa de Erro	16,00%
RAM(MB)	6,35MB
CPU(%)	7,24%

Além das métricas supracitadas para a avaliação do desempenho do algoritmo, utilizou-se o teste de Kolmogorov-Smirnov (a.k.a KS) [125]. A vantagem do KS reside no fato de não assumir que os dados possuem uma distribuição fixa (e.g., normal) e pode ser aplicado em qualquer distribuição contínua arbitrária [126]. O teste compara a função de distribuição cumulativa observada para uma variável com uma distribuição teórica especificada, que pode ser, por exemplo, normal, uniforme ou exponencial. O Kolmogorov-Smirnov é calculado a partir da maior diferença (em valor absoluto) entre as funções de distribuição cumulativa observada e teórica [127]. Como mostrado na Figura 48, foi aceito que os dados são normais com um valor de P de 0,27, acima do nível de significância.

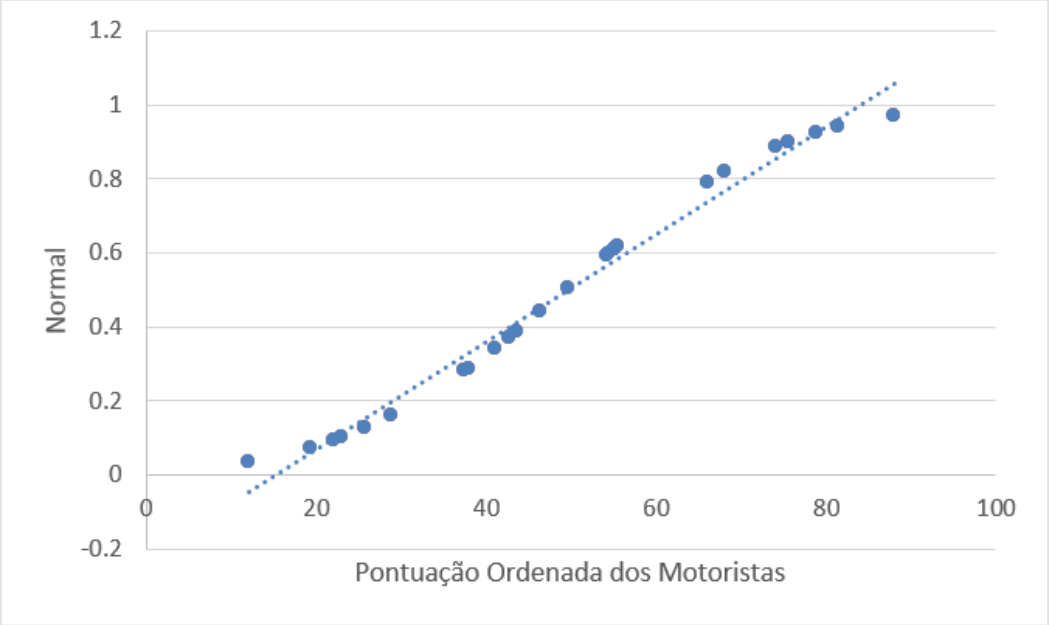

Figura 48. Probabilidade da pontuação.

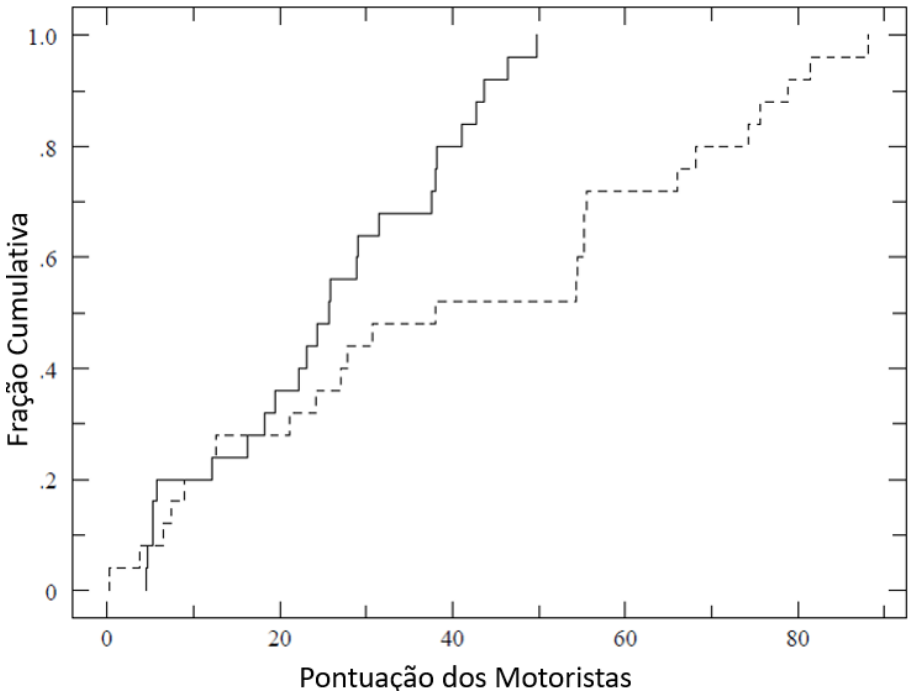

Figura 49. Comparação do teste KS2.

O KS quando usado para medir a habilidade discriminante do classificador, é chamado de teste KS2 [127]. O teste KS2 avalia o quão bem o modelo pode distinguir predições negativas (evidências imprudentes) de positivas (evidências prudentes) [125]. Assim, quanto maior o valor de KS2, melhor o desempenho do modelo. Com base nas pontuações (*cautious_score* e *reckless_score*) da regra da Figura 46 (b), a probabilidade do motorista ser prudente ou imprudente foi calculada. A Figura 49 mostra o gráfico KS2, no qual o KS máximo foi 48%.

Finalmente, embora os algoritmos tenham apresentado excelentes resultados, não conseguem identificar alguns comportamentos imprudentes destacados por Tasca [2], tais como não manter distância segura do veículo à frente, impedir uma ultrapassagem, realização de ultrapassagem imprópria (por exemplo, ultrapassagem pela direita), não obediência à sinalização (por exemplo, luz vermelha dos semáforos. Além disso, foi observado que alguns motoristas têm o hábito de dirigir com apenas uma mão ao volante. A Figura 28 mostra um dos motoristas dirigindo com uma das mãos sobre marcha por um longo período de tempo. Este comportamento aumenta o risco de colisão, visto que diminui a capacidade do motorista de realizar manobras evasivas e, portanto, é considerado um comportamento imprudente por definição e passível de multa pela legislação brasileira de trânsito [128].

6.3.2
Ameaças à Validade

Nesta Seção, os vieses que ameaçam a validade do estudo de caso são destacados. As ameaças à *validade interna* do estudo de caso são: a seleção da rota e do veículo. A qualidade das ruas e avenidas da rota pode ter gerado viés na classificação dos motoristas. A Figura 50 mostra duas de várias seções na rota escolhida com má qualidade do asfalto – infelizmente uma realidade brasileira. Nesses pontos, os motoristas precisaram executar tanto frenagens bruscas quanto mudanças de faixas abruptas. Assim, estas manobras repetidas podem ter ocasionado um viés na análise do padrão de condução dos motoristas. Outro ponto que deve ser enfatizado é o fato de que os motoristas não dirigiram em seus próprios veículos. Isso pode ter levado os motoristas a não dirigirem como diariamente dirigem. Além disso, um único veículo foi usado. É importante que veículos de diferentes portes (e.g., veículos populares, sedans, pick-ups, ônibus e caminhões) sejam usados na avaliação do estilo de condução. Uma ameaça de instrumentação é a posição do *smartphone* no veículo, que no estudo de caso sempre foi a mesma. Tanto o tipo do veículo, quanto a localização do *smartphone* podem gerar resultados diferentes.

Alguns *efeitos aditivos* podem ser ameaças à *validade interna* tais como a situação do tráfego, que é específica para Aracaju – SE, utilização do *smartphone* e horário de execução do estudo de caso. No entanto, acredita-se que os resultados

do estudo de caso não devem mudar substancialmente se fosse realizado em outra cidade ou outro trajeto. Além disso, uma situação ressaltada por Paefgen *et al*. [50] e não considerada foi o uso do *smartphone* durante o estudo de caso. A manipulação do *smartphone* durante uma ligação ou execução de outras aplicações poderia ter introduzido ruído adicional. Por exemplo, caso um passageiro utilize o *smartphone* em um jogo, poderia gerar ruído na leitura de sensores tais como o acelerômetro e o giroscópio. Por último, o horário que os motoristas dirigiram pode ter levado a um viés na análise do comportamento. Outros estudos devem ser realizados para avaliar como o horário de pico influencia o comportamento dos motoristas.

Figura 50. Seções da rota escolhida.

As expectativas do motorista especialista em relação ao modo de condução e criação do *ground truth* é uma *ameaça de constructo* que pode ter gerado distorções na classificação dos motoristas voluntários. Por fim, como *ameaça à valida externa* do estudo de caso, destacamos a não utilização de variáveis externas como condições do tráfego e do clima.

7
Conclusão e Trabalhos Futuros

Esta tese propõe e valida uma abordagem de detecção de anomalia *online* baseada em CEP para detecção do comportamento de direção de veículos privados. Diferentemente de outros trabalhos que fornecem uma detecção rápida para grandes *datasets* ou realizam o processamento distribuído, a proposta desta teste realiza o processamento de múltiplos fluxos de dados em um dispositivo móvel com recursos computacionais limitados. Além disso, assume-se um conjunto variável de dados de entrada. Portanto, as principais contribuições desta tese são:

- Adaptação de três algoritmos *offline* clássicos para realizarem a detecção *online* de anomalias. Além disso, os algoritmos para serem operacionais nos dispositivos móveis, são capazes de adaptar seu comportamento com base em recursos computacionais disponíveis, ou seja, alterar a taxa de atualização dos sensores e janela de tempo sem variar significativamente a precisão do algoritmo;

- Mecanismo de pontuação do comportamento do motorista que considera o poder discriminante de cada evidência e a taxa de atualização dos sensores;

- Desenvolvimento de um protótipo, na plataforma *Android*, que além da identificação *online* de anomalias, em um ambiente real, a partir de dados provenientes dos sensores do *smartphone* do motorista e de um dispositivo OBD, classifique motoristas em prudentes e imprudentes;

- Criação e disponibilização de um *dataset* com dados significativos da condução dos motoristas.

De acordo com a avaliação realizada, verifica-se que a abordagem proposta nesta teste tem potencial para ser utilizada na detecção de anomalia a partir de

múltiplos fluxos de dados em dispositivos com recursos computacionais limitados. Na comparação das métricas de desempenho com os trabalhos relacionados, verificou-se uma eficiência semelhante. No entanto, comparando as métricas de qualidade constatou-se um desempenho superior. Principalmente em relação à utilização da CPU que chegou a ser pouco mais que 74% mais eficiente. Além disso, no estudo de caso, cujo algoritmo utilizado foi o Z-Score, obteve uma acurácia de 84% e uma precisão de 100%.

Apesar dos benefícios da abordagem proposta, a detecção de *concept drift* e *concept evolution* não são abordados nesta tese. Além disso, existem várias direções para trabalhos futuros promissores na detecção de anomalias – técnicas de detecção de anomalias contextuais e coletivas encontram aplicabilidade crescente em vários domínios aplicação. Por exemplo, no domínio desta tese, identificar mudanças de comportamento de condução devido a um incidente de trânsito, mudança na via (instalação de radares ou semáforos), eventos, condições meteorológicas e dirigir sob efeito de álcool ou outra substância psicoativa. Técnicas para detecção *online* de anomalias em redes de sensores sem fio são especialmente interessantes. Implementações simplistas exigiriam a coleta/envio de todos os dados em um nó central (*gateway*) e execução do algoritmo de detecção de anomalia neste nó. Contudo, o consumo de energia devido ao envio de dados na rede e o *delay* entre a coleta e o processamento constituem duas desvantagens. Além disso, os nós da rede de sensores normalmente possuem poder de processamento menor que o dispositivo avaliado nesta tese.

7.1
Resultados da Tese

Para que fosse possível identificar oportunisticamente os sensores disponíveis a bordo do veículo, realizar descobertas e conexões rápidas e gerenciar os fluxos de dados de forma eficiente e escalável nos dispositivos móveis, esta tese deu origem a uma pesquisa de mestrado do Luis Eduardo Talavera Ríos, cujo título foi "*An Energy-aware IoT Gateway, with Continuous Processing of Sensor Data*". Além disso, esta tese deu origem a um projeto de iniciação científica no curso de Bacharelado em Sistemas de Informação do Instituto Federal de Sergipe (IFS), cujo título foi "Um estudo teórico e prático

sobre detecção de estilo de condução veicular". Este projeto de iniciação científica concedeu ao aluno Leonardo Nascimento Souza uma bolsa do Programa Institucional de Bolsas de Iniciação Cientifica PIBIC/PROPEX/IFS.

Esta tese também deu origem às seguintes publicações:

- L. E. Talavera, M. Endler, I. Vasconcelos, R. Vasconcelos, M. Cunha and F. J. d. S. e. Silva, "The Mobile Hub concept: Enabling applications for the Internet of Mobile Things", 2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops), St. Louis, MO, 2015, pp. 123-128;
- T. G. R. SOUSA., I. Vasconcelos, G. P. Santos, F. H. Santos, L. N. Souza, "Um estudo teórico e prático sobre detecção de estilo de condução veicular", in CONNEPI, 2016, Maceió. Anais/XI Congresso Norte Nordeste de Pesquisa e Inovação: artigos e resumos, 2016;
- I. Vasconcelos, R. Vasconcelos, B. Souza, M. Endler and M. Colaço, "Smart Driving Behavior Analysis Based on Online Outlier Detection", in *13th International Conference on Autonomic and Autonomous Systems (ICAS 2017)*, 2017, p. 6;
- I. Vasconcelos, R. Vasconcelos, B. Souza, M. Junior, M. Endler and M. Colaço, "A mobile and online outlier detection over multiple data streams: A complex event processing approach for driving behavior detection", *Journal of Internet Services and Applications (JISA 2017)*, vol. 8 (Em processo de revisão – 3º *round* de revisões).

8
Referências bibliográficas

[1] C. OWSLEY, “Driving Mobility, Older Adults, and Quality of Life,” *Gerontechnology*, vol. 1, pp. 220–230, 2002.

[2] L. Tasca, *A review of the literature on aggressive driving research*. Ontario Advisory Group on Safe Driving Secretariat, Road User Safety Branch, Ontario Ministry of Transportation, 2000.

[3] W. H. O. Who, “Global status report on road safety 2013: supporting a decade of action,” Geneva, 2013.

[4] G. Jacobs, A. Aeron-Thomas, A. Astrop, and G. Britain, “Estimating global road fatalities,” TRL Report 445, Crowthorne, Transport Research Laboratory, 2000.

[5] D. A. Johnson and M. M. Trivedi, “Driving style recognition using a smartphone as a sensor platform,” in *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2011, pp. 1609–1615.

[6] J. S. Hickman and E. S. Geller, “Self-Management to Increase Safe Driving Among Short-Haul Truck Drivers,” *Journal of Organizational Behavior Management*, vol. 23, no. 4, pp. 1–20, Apr. 2005.

[7] V. Moosavi and L. Hovestadt, “Modeling urban traffic dynamics in coexistence with urban data streams,” in *Proceedings of the 2nd ACM SIGKDD International Workshop on Urban Computing - UrbComp ’13*, 2013, p. 1.

[8] J. Aslam, S. Lim, X. Pan, and D. Rus, “City-scale traffic estimation from a roving sensor network,” in *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems - SenSys ’12*, 2012, p. 141.

[9] Z. Yang, Y. Song, T. Wang, and Y. Li, “Detecting expressway traffic incident by traffic flow and robust statistics,” in *2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet)*, 2012, pp. 2537–2540.

[10] F. Terroso-Saenz, M. Valdes-Vela, C. Sotomayor-Martinez, R. Toledo-Moreo, and A. F. Gomez-Skarmeta, “A Cooperative Approach to Traffic Congestion Detection With Complex Event Processing and VANET,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 2, pp. 914–929, Jun. 2012.

[11] C. R. Berger and E. Smith, “Intelligent Transportation Systems Provide Operational Benefits for New York Metropolitan Area Roadways: A Systems Engineering Approach,” in *2007 IEEE Long Island Systems, Applications and Technology Conference*, 2007, pp. 1–8.

[12] C. Perera, A. B. Zaslavsky, P. Christen, and D. Georgakopoulos, “Sensing as a Service Model for Smart Cities Supported by Internet

of Things," *CoRR*, vol. abs/1307.8, 2013.

[13] Q. Xie, J. Zhu, M. A. Sharaf, X. Zhou, and C. Pang, "Efficient buffer management for piecewise linear representation of multiple data streams," in *Proceedings of the 21st ACM international conference on Information and knowledge management - CIKM '12*, 2012, p. 2114.

[14] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection," *ACM Computing Surveys*, vol. 41, no. 3, pp. 1–58, Jul. 2009.

[15] S. Krishnaswamy, J. Gama, and M. M. Gaber, "Mobile Data Stream Mining: From Algorithms to Applications," in *2012 IEEE 13th International Conference on Mobile Data Management*, 2012, pp. 360–363.

[16] C. G. Quintero M., J. O. Lopez, and A. C. Cuervo Pinilla, "Driver behavior classification model based on an intelligent driving diagnosis system," in *2012 15th International IEEE Conference on Intelligent Transportation Systems*, 2012, pp. 894–899.

[17] Y. Leng and L. Zhao, "Novel design of intelligent internet-of-vehicles management system based on cloud-computing and Internet-of-Things," in *Proceedings of 2011 International Conference on Electronic & Mechanical Engineering and Information Technology*, 2011, vol. 6, pp. 3190–3193.

[18] W. He, G. Yan, and L. Xu, "Developing Vehicular Data Cloud Services in the IoT Environment," *Industrial Informatics, IEEE Transactions on*, vol. PP, no. 99. p. 1, 2014.

[19] H. CUI, "Online Outlier Detection Over Data Streams," Master of Science, Simon Fraser University, 2005.

[20] M. Elahi, K. Li, W. Nisar, X. Lv, and H. Wang, "Efficient Clustering-Based Outlier Detection Algorithm for Dynamic Data Stream," in *2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery*, 2008, vol. 5, pp. 298–304.

[21] M. Elahi, K. Li, W. Nisar, X. Lv, and H. Wang, "Detection of Local Outlier over Dynamic Data Streams Using Efficient Partitioning Method," in *2009 WRI World Congress on Computer Science and Information Engineering*, 2009, vol. 4, pp. 76–81.

[22] C. C. Aggarwal, *Outlier Analysis*. New York, NY: Springer New York, 2013.

[23] M. Gupta, J. Gao, C. C. Aggarwal, and J. Han, "Outlier Detection for Temporal Data: A Survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 9, pp. 2250–2267, Sep. 2014.

[24] A. Bouchachia, "Incremental learning with multi-level adaptation," *Neurocomputing*, vol. 74, no. 11, pp. 1785–1799, May 2011.

[25] K. Li, M. Lu, F. Lu, Q. Lv, L. Shang, and D. Maksimovic, "Personalized Driving Behavior Monitoring and Analysis for Emerging Hybrid Vehicles," in *Proceedings of the 10th International Conference on Pervasive Computing*, Newcastle, UK: Springer-Verlag, 2012, pp. 1–19.

[26] H. Ferdowsi, S. Jagannathan, and M. Zawodniok, "An Online Outlier Identification and Removal Scheme for Improving Fault Detection Performance," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 5, pp. 908–919, 2014.

[27] D. M. Hawkins, *Identification of Outliers*. Dordrecht: Springer Netherlands, 1980.
[28] S. Agrawal and J. Agrawal, “Survey on Anomaly Detection using Data Mining Techniques,” *Procedia Computer Science*, vol. 60, pp. 708–713, 2015.
[29] V. J. Hodge and J. Austin, “A Survey of Outlier Detection Methodologies,” *Artificial Intelligence Review*, vol. 22, no. 2, pp. 85–126, Oct. 2004.
[30] M. M. Gaber, A. Zaslavsky, and S. Krishnaswamy, “Mining data streams: A Review,” *ACM SIGMOD Record*, vol. 34, no. 2. ACM, New York, NY, USA, p. 18, 2005.
[31] X. Deng, M. Ghanem, and Y. Guo, “Real-Time Data Mining Methodology and a Supporting Framework,” in *Network and System Security, 2009. NSS ’09. Third International Conference on*, 2009, pp. 522–527.
[32] X. Tang, G. Li, and G. Chen, “Fast Detecting Outliers over Online Data Streams,” in *2009 International Conference on Information Engineering and Computer Science*, 2009, pp. 1–4.
[33] N. Lin, C. Zong, M. Tomizuka, P. Song, Z. Zhang, and G. Li, “An Overview on Study of Identification of Driver Behavior Characteristics for Automotive Control,” *Mathematical Problems in Engineering*, vol. 2014, pp. 1–15, 2014.
[34] M. Čermák, D. Tovarňák, M. Laštovička, and P. Čeleda, “A performance benchmark for NetFlow data analysis on distributed stream processing systems,” in *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, 2016, pp. 919–924.
[35] W. Chang, “Research on driving intention identification based on hidden markov model [Ph.D. thesis],” College Automotive Engineering, Jilin University, Changchun, China, 2011.
[36] W. Wang, J. Xi, and H. Chen, “Modeling and Recognizing Driver Behavior Based on Driving Data: A Survey,” *Mathematical Problems in Engineering*, vol. 2014, pp. 1–20, 2014.
[37] M. Kontaki, A. Gounaris, A. N. Papadopoulos, K. Tsichlas, and Y. Manolopoulos, “Efficient and Flexible Algorithms for Monitoring Distance-based Outliers over Data Streams,” *Inf. Syst.*, vol. 55, no. C, pp. 37–53, 2016.
[38] C. C. Aggarwal and K. Subbian, “Event Detection in Social Streams,” in *Proceedings of the 2012 SIAM International Conference on Data Mining*, Philadelphia, PA: Society for Industrial and Applied Mathematics, 2012, pp. 624–635.
[39] S. Sadik and L. Gruenwald, “Online outlier detection for data streams,” in *Proceedings of the 15th Symposium on International Database Engineering & Applications - IDEAS ’11*, 2011, p. 88.
[40] D. C. Luckham, *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2001.
[41] O. Etzion and P. Niblett, *Event processing in action*. Manning Publications Co., 2010.
[42] A. Arasu, S. Babu, and J. Widom, “The CQL continuous query

language: semantic foundations and query execution," *The VLDB Journal*, vol. 15, no. 2, pp. 121–142, Jun. 2006.

[43] R. Kalsoom and Z. Halim, "Clustering the driving features based on data streams," in *INMIC*, 2013, pp. 89–94.

[44] C.-Y. Chan, "On the detection of vehicular crashes-system characteristics and architecture," *Vehicular Technology, IEEE Transactions on*, vol. 51, no. 1, pp. 180–193, Jan. 2002.

[45] P. Chaovalit, C. Saiprasert, and T. Pholprasit, "A method for driving event detection using SAX on smartphone sensors," in *2013 13th International Conference on ITS Telecommunications (ITST)*, 2013, pp. 450–455.

[46] E. Ayapana, "Motor Trend - 2016 Chevrolet Malibu Will Give Teens a Driving Report Card," 2015. [Online]. Available: http://www.motortrend.ca/en/news/1503-2016-chevrolet-malibu-will-give-teens-a-driving-report-card/. [Accessed: 16-Oct-2015].

[47] G. R. Singh and S. S. Dongre, "Crash Prediction System for Mobile Device on Android by Using Data Stream Minning Techniques," in *2012 Sixth Asia Modelling Symposium*, 2012, pp. 185–190.

[48] S. Singh, S. Nelakuditi, Y. Tong, and R. R. Choudhury, "Your Smartphone Can Watch the Road and You : Mobile Assistant for Inattentive Drivers," in *MobiHoc*, 2012, pp. 261–262.

[49] J. White, C. Thompson, H. Turner, B. Dougherty, and D. C. Schmidt, "WreckWatch: Automatic Traffic Accident Detection and Notification with Smartphones," *Mobile Networks and Applications*, vol. 16, no. 3, pp. 285–303, Mar. 2011.

[50] J. Paefgen, F. Kehr, Y. Zhai, and F. Michahelles, "Driving Behavior Analysis with Smartphones: Insights from a Controlled Field Study," in *Proceedings of the 11th International Conference on Mobile and Ubiquitous Multimedia*, 2012, p. 36:1--36:8.

[51] Hongyang Zhao, Huan Zhou, Canfeng Chen, and Jiming Chen, "Join driving: A smart phone-based driving behavior evaluation system," in *2013 IEEE Global Communications Conference (GLOBECOM)*, 2013, pp. 48–53.

[52] J.-H. Hong, B. Margines, and A. K. Dey, "A smartphone-based sensing platform to model aggressive driving behaviors," in *Proceedings of the 32nd annual ACM conference on Human factors in computing systems - CHI '14*, 2014, pp. 4047–4056.

[53] L. M. Bergasa, D. Almeria, J. Almazan, J. J. Yebes, and R. Arroyo, "DriveSafe: An app for alerting inattentive drivers and scoring driving behaviors," in *2014 IEEE Intelligent Vehicles Symposium Proceedings*, 2014, pp. 240–245.

[54] J. Dai, J. Teng, X. Bai, Z. Shen, and D. Xuan, "Mobile phone based drunk driving detection," in *Proceedings of the 4th International ICST Conference on Pervasive Computing Technologies for Healthcare*, 2010, pp. 1–8.

[55] L. Portnoy, E. Eskin, and S. Stolfo, "Intrusion detection with unlabeled data using clustering," in *In Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001*, 2001, pp. 5–8.

[56] E. M. Knorr and R. T. Ng, "Finding Intensional Knowledge of

Distance-Based Outliers," in *Proceedings of the 25th International Conference on Very Large Data Bases*, 1999, pp. 211–222.

[57] C. C. Aggarwal and P. S. Yu, "Outlier detection for high dimensional data," *ACM SIGMOD Record*, vol. 30, no. 2, pp. 37–46, Jun. 2001.

[58] L. X. Pang, S. Chawla, W. Liu, and Y. Zheng, "On detection of emerging anomalous traffic patterns using GPS data," *Data & Knowledge Engineering*, vol. 87, pp. 357–373, Sep. 2013.

[59] G. W. Heiman, *Basic Statistics for the Behavioral Sciences*, 6th ed. Cengage Learning, 2006.

[60] G. R. Cutter, M. L. Baier, R. A. Rudick, D. L. Cookfair, J. S. Fischer, J. Petkau, K. Syndulko, B. G. Weinshenker, J. P. Antel, C. Confavreux, and others, "Development of a multiple sclerosis functional composite as a clinical trial outcome measure," *Brain*, vol. 122, no. 5, pp. 871–882, 1999.

[61] J. Laurikkala, M. Juhola, and E. Kentala, "Informal Identification of Outliers in Medical Data," in *Fifth International Workshop on Intelligent Data Analysis in Medicine and Pharmacology IDAMAP-2000 Berlin, 22 August. Organized as a workshop of the 14th European Conference on Artificial Intelligence ECAI-2000*, 2000.

[62] M. K. Pakhira, "A Linear Time-Complexity k-Means Algorithm Using Cluster Shifting," in *2014 International Conference on Computational Intelligence and Communication Networks*, 2014, pp. 1047–1051.

[63] S. Basu, M. Bilenko, and R. J. Mooney, "A Probabilistic Framework for Semi-supervised Clustering," in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004, pp. 59–68.

[64] M. Ester, H. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," 1996, pp. 226–231.

[65] S. Guha, R. Rastogi, and K. Shim, "Rock: A robust clustering algorithm for categorical attributes," *Information Systems*, vol. 25, no. 5, pp. 345–366, 2000.

[66] L. Ertöz, M. Steinbach, and V. Kumar, "Finding Topics in Collections of Documents: A Shared Nearest Neighbor Approach," 2004, pp. 83–103.

[67] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.

[68] J. A. Hartigan and M. A. Wong, "A {K}-Means Clustering Algorithm," *Applied Statistics*, vol. 28, pp. 100–108, 1979.

[69] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, vol. 39, no. 1, pp. 1–38, 1977.

[70] A. Vinueza and G. Grudic, "Unsupervised outlier detection and semi-supervised learning," 2004.

[71] L. Cao, M. Wei, D. Yang, and E. A. Rundensteiner, "Online Outlier Exploration Over Large Datasets," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015, pp. 89–98.

[72] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF," *ACM*

SIGMOD Record, vol. 29, no. 2, pp. 93–104, Jun. 2000.

[73] A. Likas, N. Vlassis, and J. J. Verbeek, “The global k-means clustering algorithm,” *Pattern Recognition*, vol. 36, no. 2, pp. 451–461, 2003.

[74] R. Bruns, J. Dunkel, H. Billhardt, M. Lujak, and S. Ossowski, “Using Complex Event Processing to support data fusion for ambulance coordination,” in *Information Fusion (FUSION), 2014 17th International Conference on Information Fusion (FUSION 2014)*, 2014, pp. 1–7.

[75] V. Govindasamy, V. Akila, S. Hariharan, R. S. Pandian, P. V. M. S. Naidu, and P. Haridev, “Novel Complex Event Processing,” in *Proceedings of the 2015 International Conference on Advanced Research in Computer Science Engineering & Technology (ICARCSET 2015) - ICARCSET ’15*, 2015, pp. 1–6.

[76] J. Dunkel, “On complex event processing for sensor networks,” *2009 International Symposium on Autonomous Decentralized Systems*, 2009.

[77] A. Adi and O. Etzion, “Amit - the Situation Manager,” *The VLDB Journal*, vol. 13, no. 2, pp. 177–203, 2004.

[78] G. Cugola and A. Margara, “Processing Flows of Information: From Data Stream to Complex Event Processing,” *ACM Comput. Surv.*, vol. 44, no. 3, p. 15:1--15:62, 2012.

[79] Esper, “EsperTech,” 2014. [Online]. Available: http://www.espertech.com/. [Accessed: 11-Dec-2014].

[80] L. M. Bergasa, J. Nuevo, M. A. Sotelo, R. Barea, and M. E. Lopez, “Real-time system for monitoring driver vigilance,” *Intelligent Transportation Systems, IEEE Transactions on*, vol. 7, no. 1, pp. 63–77, 2006.

[81] W. Rongben, G. Lie, T. Bingliang, and J. Lisheng, “Monitoring mouth movement for driver fatigue or distraction with one camera,” in *Intelligent Transportation Systems, 2004. Proceedings. The 7th International IEEE Conference on*, 2004, pp. 314–319.

[82] J. van den Berg, “Indicators and predictors of sleepiness,” Faculty Med., Umeå Univ., Umeå, Sweden, 2006.

[83] I. Mohamad, M. A. M. Ali, and M. Ismail, “Abnormal driving detection using real time Global Positioning System data,” in *Proceeding of the 2011 IEEE International Conference on Space Science and Communication (IconSpace)*, 2011, pp. 1–6.

[84] W. Hailin, L. Hanhui, and S. Zhumei, “Fatigue Driving Detection System Design Based on Driving Behavior,” in *Optoelectronics and Image Processing (ICOIP), 2010 International Conference on*, 2010, vol. 1, pp. 549–552.

[85] T. Imkamon, P. Saensom, P. Tangamchit, and P. Pongpaibool, “Detection of hazardous driving behavior using fuzzy logic,” in *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, 2008. ECTI-CON 2008. 5th International Conference on*, 2008, vol. 2, pp. 657–660.

[86] E. Berdoulat, D. Vavassori, and M. T. M. Sastre, “Driving anger, emotional and instrumental aggressiveness, and impulsiveness in the prediction of aggressive and transgressive driving,” *Accident*

Analysis & Prevention, vol. 50, pp. 758–767, 2013.

[87] J. M. Houston, P. B. Harris, and M. Norman, “The Aggressive Driving Behavior Scale: Developing a Self-Report Measure of Unsafe Driving Practices,” *North American Journal of Psychology*, vol. 5, no. 2, p. 269, 2003.

[88] D. Wiesenthal, D. Hennessy, and P. Gibson, “The Driving Vengeance Questionnaire (DVQ): the development of a scale to measure deviant drivers’ attitudes,” *Violence and Victims*, vol. 15, no. 2, pp. 115–136, 2000.

[89] J. Deffenbacher, E. Oetting, and R. Lynch, “Development of a driving anger scale,” *Psychological Reports*, vol. 74, no. 1, pp. 83–91, 1994.

[90] J. L. Deffenbacher, R. S. Lynch, E. R. Oetting, and R. C. Swaim, “The Driving Anger Expression Inventory: a measure of how people express their anger on the road,” *Behaviour Research and Therapy*, vol. 40, no. 6, pp. 717–737, 2002.

[91] J. T. Reason, A. S. R. Manstead, S. G. Stradling, J. S. Baxter, and K. Campbell, “Errors and violations on the road: a real distinction?,” *Ergonomics*, vol. 33, pp. 1315–1332, 1990.

[92] O. Etzion, “Towards an Event-Driven Architecture: An Infrastructure for Event Processing Position Paper,” in *Rules and Rule Markup Languages for the Semantic Web: First International Conference, RuleML 2005, Galway, Ireland, November 10-12, 2005. Proceedings*, A. Adi, S. Stoutenburg, and S. Tabet, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 1–7.

[93] I. Assent, P. Kranen, C. Baldauf, and T. Seidl, “AnyOut: Anytime Outlier Detection on Streaming Data,” in *Database Systems for Advanced Applications: 17th International Conference, DASFAA 2012, Busan, South Korea, April 15-19, 2012, Proceedings, Part I*, S. Lee, Z. Peng, X. Zhou, Y.-S. Moon, R. Unland, and J. Yoo, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 228–242.

[94] H. Kargupta, H., Bhargava, R., Liu, K., Powers, M., Blair, P., Bushra, S., Dull, J., Sarkar, K., Klein, M., Vasa, M., “Vedas: A Mobile and Distributed Data Stream Minig System for Real-Time Vehicle Monitoring,” in *Proceedings of the SIAM International Data Mining Conference, Orlando*, 2004, p. 13.

[95] E. Keogh, S. Chu, D. Hart, and M. Pazzani, “An online algorithm for segmenting time series,” in *Proceedings 2001 IEEE International Conference on Data Mining*, pp. 289–296.

[96] I. O. for Standardization, “ISO 2631-1-1997: Mechanical vibration and shock-Evaluation of human exposure to whole-body vibration-Part 1: General requirements,” 1997.

[97] H. Sakoe and S. Chiba, “Dynamic programming algorithm optimization for spoken word recognition,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, no. 1, pp. 43–49, Feb. 1978.

[98] R. Musculo, S. Conforto, M. Schmid, P. Caselli, and T. D’Alessio, “Classification of motor activities through derivative dynamic time warping applied on accelerometer data,” in *Annual International Conference of the IEEE Engineering in Medicine and Biology -*

Proceedings, 2007, pp. 4930–4933.

[99] A. Aljaafreh, N. Alshabatat, and M. S. Najim Al-Din, “Driving style recognition using fuzzy logic,” in *2012 IEEE International Conference on Vehicular Electronics and Safety (ICVES 2012)*, 2012, pp. 460–463.

[100] B. S. Dangra, M. V. Bedekar, and S. S. Panicker, “User Profiling of Automobile Driver and Outlier Detection,” *International Journal of Innovative Research & Development*, vol. 3, no. 12, pp. 49–55, 2014.

[101] L. E. Talavera, M. Endler, I. Vasconcelos, R. Vasconcelos, M. Cunha, and F. J. Da Silva E. Silva, “The Mobile Hub concept: Enabling applications for the Internet of Mobile Things,” in *2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, 2015, pp. 123–128.

[102] V. Karthik, “Driver Telematics Analysis. Master’s Thesis in Computer Science,” San José State University, 2015.

[103] L. Brenna, J. Gehrke, M. Hong, and D. Johansen, “Distributed event stream processing with non-deterministic finite automata,” in *Proceedings of the Third ACM International Conference on Distributed Event-Based Systems - DEBS ’09*, 2009, p. 1.

[104] B. Kim, S. Lee, Y. Lee, I. Hwang, Y. Rhee, and J. Song, “Mobiiscape: Middleware support for scalable mobility pattern monitoring of moving objects in a large-scale city,” *Journal of Systems and Software*, vol. 84, no. 11, pp. 1852–1870, Nov. 2011.

[105] I. Flouris, N. Giatrakos, M. Garofalakis, and A. Deligiannakis, “Issues in Complex Event Processing Systems,” in *2015 IEEE Trustcom/BigDataSE/ISPA*, 2015, pp. 241–246.

[106] G. L. dos Santos, “Máquinas de Estados Hierárquicas em Jogos Eletrônicos,” Disstertação de Mestrado, Pontifícia Universidade Católica do Rio de Janeiro, 2004.

[107] M. M. Masud, Q. Chen, L. Khan, C. Aggarwal, J. Gao, J. Han, and B. Thuraisingham, “Addressing Concept-Evolution in Concept-Drifting Data Streams,” in *2010 IEEE International Conference on Data Mining*, 2010, pp. 929–934.

[108] C. C. Aggarwal, *Data Streams: Models and Algorithms (Advances in Database Systems)*, vol. 31. Boston, MA: Springer US, 2007.

[109] J. Z. Kolter and M. A. Maloof, “Using additive expert ensembles to cope with concept drift,” in *Proceedings of the 22nd international conference on Machine learning - ICML ’05*, 2005, pp. 449–456.

[110] H. Wang, W. Fan, P. S. Yu, and J. Han, “Mining concept-drifting data streams using ensemble classifiers,” in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD ’03*, 2003, p. 226.

[111] G. S. Gurjar and S. Chhabria, “A review on concept evolution technique on data stream,” in *2015 International Conference on Pervasive Computing (ICPC)*, 2015, pp. 1–3.

[112] F. N. Kerlinger and H. M. Rotundo, *Metodologia da pesquisa em ciências sociais: um tratamento conceitual*. Epu, 1980.

[113] C. SELLTIZ, L. WRIGHTSMAN, and S. COOK, “Métodos de

Pesquisa nas Relações Sociais - Delineamentos de Pesquisa v.1." EPU, São Paulo, 1987.

[114] F. M. Raupp and I. M. Beuren, "Metodologia da pesquisa aplicável às ciências sociais," *Como elaborar trabalhos monográficos em contabilidade: teoria e prática*, vol. 3, pp. 76–97, 2003.

[115] E. Romera, L. M. Bergasa, and R. Arroyo, "Need data for driver behaviour analysis? Presenting the public UAH-DriveSet," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, 2016, pp. 387–392.

[116] L. Zamberlan, *Pesquisa de mercado*, 1ª Edição. Ijuí, Rio Grande do Sul, 2008.

[117] M. Owens, "Embedding an SQL Database with SQLite," *Linux J.*, vol. 2003, no. 110, p. 2--, Jun. 2003.

[118] B. S. Santos, M. Colaço Júnior, B. C. da Paixão, R. M. Santos, A. V. R. P. Nascimento, H. C. dos Santos, H. L. Wallace Filho, and A. S. L. de Medeiros, "Comparing Text Mining Algorithms for Predicting Irregularities in Public Accounts," in *Proceedings of the annual conference on Brazilian Symposium on Information Systems: Information Systems: A Computer Socio-Technical Perspective-Volume 1*, 2015, p. 89.

[119] M. Bramer, *Principles of Data Mining*. London: Springer London, 2013.

[120] J. Han, J. Pei, and M. Kamber, *Data mining: concepts and techniques (3rd. ed.)*. Morgan Kaufmann Publishers, San Francisco, CA, 2011.

[121] Y. Zhang, W. C. Lin, and Y. K. S. Chin, "A Pattern-Recognition Approach for Driving Skill Characterization," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 4, pp. 905–916, 2010.

[122] J. Rada-Vilela, "fuzzylite: a fuzzy logic control library." 2014.

[123] C. Sevarac, Z., Goloskokovic, I., Tait, J., Morgan, A and L, "Neuroph," 2010. [Online]. Available: http://neuroph.sourceforge.net/. [Accessed: 19-Jan-2017].

[124] C. D. Manning, P. Raghavan, and H. Schütze, *An Introduction to Information Retrieval*, Online edi. Cambridge, England: Cambridge University Press, 2008.

[125] W. J. Conover, *Pratical Nonparametric Statistics*. Chap. 6, Third Edition, John Wiley & Sons, 1999.

[126] A. Lall, "Data streaming algorithms for the Kolmogorov-Smirnov test," in *2015 IEEE International Conference on Big Data (Big Data)*, 2015, pp. 95–104.

[127] P. J. L. Adeodato, D. S. P. Salazar, L. S. Gallindo, A. G. Sa, and S. M. Souza, "Continuous variables segmentation and reordering for optimal performance on binary classification tasks," in *2014 International Joint Conference on Neural Networks (IJCNN)*, 2014, pp. 3720–3725.

[128] J. M. de Araujo, "Artigo 252 da Legislação Brasileira de Trânsito," 2016. [Online]. Available: http://www.ctbdigital.com.br/?p=Comentarios&Registro=75. [Accessed: 28-Dec-2016].

9
Glossário

Mobile Crowd Sensing: é o paradigma de sensoriamento baseado em dispositivos móveis, tais como *smartphones,* veículos inteligentes, dispositivos vestíveis (*wearable devices*) que envolve geração de dados, processamento destes dados para derivar dados de nível de abstração maior e disseminação do conhecimento.

Mobile Activity Recognition: é uma área popular de pesquisa em computação pervasiva devido a sua importância para aplicações sensíveis ao contexto. O objetivo geral é analisar dados contínuos de sensores e identificar a ocorrência de atividades de interesse com alta acurácia.

Mobile Healthcare: é um termo usado para a prática da medicina e saúde pública suportada por dispositivos móveis. Os aplicativos móveis de saúde (*mobile healthcare applications*) gozam de popularidade crescente e proporcionam benefícios significativos aos usuários que podem aproveitar a disponibilidade geral e crescente conjunto de sensores em dispositivos móveis.

Intelligent Transportation Systems: são definidos como aqueles sistemas que utilizam tecnologias e conceitos de engenharia de sistemas para desenvolver e melhorar sistemas de transporte de todos os tipos.