UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

Gabriel Araujo Langer

ANÁLISE DE DADOS APLICADA PARA CONSCIÊNCIA SITUACIONAL DE PEDESTRES

Florianópolis

2018

Gabriel Araujo Langer

ANÁLISE DE DADOS APLICADA PARA CONSCIÊNCIA SITUACIONAL DE PEDESTRES

Trabalho de conclusão de curso apresentado como parte dos requisitos para a obtenção do Grau de Bacharel em ciências da computação.
Orientador: Prof. Dr. Elder Rizzon Santos

Florianópolis

2018

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Langer, Gabriel Araujo
Análise de dados aplicada para consciência situacional de pedestres / Gabriel Araujo Langer ; orientador, Elder Rizzon Santos, 2018.
84 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, , Graduação em , Florianópolis, 2018.

Inclui referências.

1. . 2. Inteligência artificial. 3. Ciência de dados. 4. Estatística. 5. Consciência situacional. I. Santos, Elder Rizzon. II. Universidade Federal de Santa Catarina. Graduação em . III. Título.

Gabriel Araujo Langer

ANÁLISE DE DADOS APLICADA PARA CONSCIÊNCIA SITUACIONAL DE PEDESTRES

Trabalho de conclusão de curso apresentado como parte dos requisitos para a obtenção do grau de "Bacharel em ciências da computação".

Florianópolis, 05 de dezembro 2018.

Banca Examinadora:

Prof. Dr. Elder Rizzon Santos
Orientador

Prof. Dr. André Wüst Zibetti

Me. Thiago Ângelo Gelaim

RESUMO

O número de acidentes próximos a vias urbanas, principalmente em ruas com grande movimento, tem aumentado consideravelmente. Este fato está sendo associado com o crescimento contínuo do uso de dispositivos móveis, como smartphones, tablets e smartwatches, em áreas urbanas. A fim de ajudar na diminuição dos acidentes causados pela distração por estes dispositivos, o projeto Awareness propõe um modelo de consciência situacional. Este modelo foi construído através de dados coletados em um ambiente de realidade virtual simulando um ambiente urbano, no qual um usuário é submetido a testes para avaliar sua consciência. Este trabalho terá uma abordagem de comparação entre diversas técnicas de análise com base nos dados obtidos no projeto, utilizando inteligência artificial e estatística. A partir dos resultados, realizou-se a inferência de um nível de consciência situacional do usuário de acordo com dados sobre o ambiente que está inserido, através de busca por padrões que originem distrações e ocasionam uma circunstância de risco e, então, foi construído um modelo preditivo para auxílio na tomada de decisões de pedestres. Outras análises foram realizadas anteriormente baseada nos dados do projeto, mas utilizaram o método de redes bayesianas. A proposta deste trabalho é compreender como outras abordagens estatísticas podem modelar esse problema e avaliar seus desempenhos.

Palavras-chave: Aprendizado estatístico. Modelos preditivos. Inteligência artificial, Aprendizado de máquina, Consciência situacional.

ABSTRACT

The quantity of accidents near urban roads, especially on busy roads, has increased considerably. This fact is being associated with continued growth in the use of mobile devices such as smartphones, tablets and smartwatches in urban areas. In order to help reduce the accidents caused by distraction by these devices, the Awareness project proposes a model of situational awareness. This model was constructed through data collected in a virtual reality environment simulating an urban environment in which a user is subjected to tests to assess his concentration. This work will have a comparative approach between several techniques of analysis based on the data obtained in the project, using artificial intelligence and statistics. From the results, it is desired to infer a level of situational awareness of the user according to data about the environment that is inserted, searching for patterns that originate distractions and cause a risk circumstance, and then construct a predictive model for assistance in making pedestrian decisions. Other analyzes were done previously based on the project data, but used the bayesian networks method. The purpose of this paper is to understand how other statistical approaches can model this problem and evaluate its performance.

Keywords: Statistical learning. Predictive models. Artificial intelligence. Machine learning. Situational awareness.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

SUMÁRIO

1 INTRODUÇÃO

Nas últimas décadas houve um progresso muito grande em relação às tecnologias na área de computação. Isso acarretou em uma revolução em diversas áreas, dentre elas a estatística. Segundo Hand (2015), quando a publicação científica "Statistics and Computing" foi lançada em 1991, ficou aparente que a estatística passou a ser uma disciplina mais computacional do que matemática. Além disso, era crescente a sinergia com domínios associados a ciências da computação, como o aprendizado de máquina. O impacto dessa mudança permitiu que cálculos e análises estatísticas complexas fossem feitas em segundos, além de viabilizar muitas tarefas como a inversão de matrizes muito grandes e métodos iterativos, que possibilitaram a criação de ferramentas como modelos lineares generalizados (HAND, 2015).

Há muito tempo, um grande desafio para a humanidade tem sido entender a maneira a qual pensamos. A inteligência artificial é uma ciência recente que busca compreender os pensamentos do ser humano e criar entidades inteligentes. Várias áreas de conhecimento, como filosofia, matemática, economia, neurociência, psicologia, engenharia da computação e linguística contribuíram para fundamentar esse conceito (RUSSELL; NORVIG, 2003). Com a relevância da área em um evidente crescimento, diversos pesquisadores acabaram se envolvendo e as linhas de pesquisa se dividiram entre duas abordagens: a simbólica e a conexionista.

A inteligência artificial simbólica tem como objetivo simular um comportamento inteligente através de um sistema simbólico, ou seja, é relacionada com o raciocínio do ser humano. Esse paradigma teve como precursores Allen Newell e Hebert Alexander Simon, que em 1976 definiram os sistemas de símbolos físicos (NEWELL; SIMON, 1976). Tais sistemas também são chamados de sistemas formais, uma vez que combinam símbolos em expressões e as manipulam para produzir outras. A abordagem mencionada é perfeitamente representada em sistemas especialistas, onde são utilizados dados específicos do problema para o funcionamento ideal.

A inteligência artificial conexionista funciona baseando-se nos componentes que formam o cérebro para modelar a inteligência humana. Foi desenvolvida através de estudos para modelar neurônios a fim de permitir que computadores pudessem ter capacidades consideradas inteligentes, como o reconhecimento de padrões e a aprendizagem. Esses estudos resultaram nas redes neurais artificiais, que são cole-

ções de unidades conectadas juntas, onde as propriedades da rede são determinadas pela topologia e propriedade dos "neurônios" (RUSSELL; NORVIG, 2003) e auxiliam na resolução de diversos problemas computacionais e estatísticos. Diferentemente da abordagem simbólica, não necessita um conhecimento especializado sobre o problema na qual está trabalhando, a própria máquina é capaz de entender o funcionamento do sistema e auxiliar na tomada de decisões.

Durante os estudos que levaram à evolução da estatística por meio de métodos computacionais, outras ferramentas foram criadas por cientistas da computação, dentre elas as redes neurais, os sistemas especialistas e as máquinas de vetores de suporte (HAND, 2015). Ao obterem conhecimento das redes neurais artificiais, os estatísticos alcançaram êxito ao incorporar seus procedimentos de estimação na teoria de estimação estatística, o que levou décadas para ser possível. Isso levou a um reconhecimento maior do poder das redes neurais e as aplicações nas quais seriam importantes. Além disso, outras intersecções com inteligência artificial foram notadas, como a utilização de redes bayesianas, o que levou à criação de uma série de conferências denotada "AI and Statistics", a partir de 1984 (HAND, 2015).

Segundo Russell e Norvig (2003), com os avanços na capacidade de sistemas reais, há um progresso na compreensão da base teórica da inteligência, acarretando maior integração entre subáreas da inteligência artificial e no aumento da concordância com outras disciplinas. Um caso a ser citado é a integração com conceitos matemáticos através da estatística inferencial, que segundo Ross (2004), supõe que importantes aspectos do fenômeno sob estudo podem ser descritos em termos de probabilidades. Logo, pode-se tirar conclusões utilizando dados para fazer inferências sobre tais probabilidades. Basicamente, a estatística inferencial baseia-se na criação de uma representação útil e confiável para os dados fornecidos por um problema, com fins de exploração de relacionamentos para geração de conhecimento, através de métodos como regressão e classificação. A regressão funciona estimando relações entre variáveis independentes, sendo utilizadas para realizar a predição de um resultado numérico, o qual é chamado de variável dependente, com o objetivo de encontrar o melhor valor para ser representado no problema. A classificação funciona de maneira similar à regressão, mas em vez de encontrar valores numéricos como resultado, identifica, dentre classes pré-definidas, qual melhor se aproxima de cada dado testado.

A união dos conceitos apresentados possibilita a resolução de problemas de diversas disciplinas e resulta numa área conhecida como ciência de dados. Apesar de sua primeira aparição na literatura ter

sido no ano de 1974 no livro "Concise Survey of Computer Methods" (NAUR, 1974), a discussão sobre a utilização da terminologia é mais recente. Em 1997, Jeff Wu sugeriu que estatística fosse renomeada para ciência de dados, e que os estatísticos fossem denominados cientistas de dados para mudar o foco de análises tradicionais para questões mais relacionadas ao futuro (CAO, 2017). Em 2001, William S, Cleveland, também sugeriu a alteração dos termos com o objetivo de englobar mais áreas técnicas no campo de estatística, fazendo parcerias com cientistas da computação (CAO, 2017).

Este trabalho propõe uma aplicação e comparação de técnicas que utilizam análise de dados associadas a conceitos como o aprendizado de máquina para exploração de conhecimento em uma pesquisa sobre consciência situacional no projeto *Awareness*. O trabalho contribui sugerindo modelos de predições e procura auxiliar no entendimento do comportamento humano quando submetido a situações de risco, verificando sua consciência em cada momento e auxiliando a tomar decisões, alertando ainda, sobre situações com periculosidade, buscando assim, diminuir a incidência de acidentes em vias urbanas. Ademais, a importância do trabalho está em apresentar a aplicação dos variados conceitos essenciais em ciência de dados e ajudar futuras pesquisas a escolherem abordagens e técnicas mais adequadas para diferentes desafios.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

Analisar os dados do projeto *Awareness*, avaliando abordagens e técnicas em um modelo computacional para representar a necessidade de notificar um pedestre dependendo do seu nível de consciência situacional.

1.1.2 Objetivos Específicos

- Analisar o estado da arte em ciência de dados
- Realizar o pré-processamento dos dados fornecidos pelo projeto
- Realizar uma modelagem estatística com base nas informações extraídas

- Analisar técnicas de análise de dados mais adequadas utilizando estatística e aprendizado de máquina

- Aplicar as técnicas escolhidas

- Demonstrar os pontos fracos e fortes de cada técnica inserida no contexto do trabalho

1.2 METODOLOGIA

A linha de pesquisa utilizada neste trabalho será a Design Science Research. Hevner et al. (2008) afirmam que este método é um processo para resolução de problemas, onde o princípio fundamental é que conhecimento e entendimento de um problema de projeto e sua solução são adquiridos construindo e aplicando um artefato. Isso resulta em um artefato inovativo e útil para um domínio específico. March e Smith (1995) afirmam que a fase de avaliação é essencial neste método, pois as métricas utilizadas definem o que uma área de pesquisa está buscando alcançar, comparando a performance das construções, dos modelos, métodos e instâncias para tarefas específicas.

Peffers et al. (2007) propõem um método derivado de diversas disciplinas, obedecendo os princípios de pesquisa de Design Science. Seu método é decomposto em seis etapas:

1. **Identificação do problema e motivação**: É a etapa responsável pela geração de um artefato que reduza o problema para que a solução possa capturar sua complexidade. Além disso, deve ser justificado o valor da solução, demonstrando a sua importância na resolução do problema identificado.

2. **Definir objetivos de uma solução**: Estágio em que é realizada a "inferência de objetivos da solução através da definição do problema e o conhecimento do que é possível e factível". Pode ser quantitativo, demonstrando que a solução pode ser melhor do que as existentes em certos aspectos ou pode ser qualitativa, descrevendo como será o suporte a problemas que até o momento não estão resolvidos.

3. **Projeto e desenvolvimento**: Etapa responsável por definir requisitos e funcionalidades, além de criar o artefato propriamente dito.

4. **Demonstração**: Aqui o objetivo é demonstrar o uso do artefato na resolução de uma ou mais instâncias do problema, envolvendo seu uso em experimentos, simulação, estudo de caso, prova de conceito, além de outras atividades. Os recursos necessários, nesse momento, incluem o conhecimento da utilização do artefato para resolver o problema da pesquisa.

5. **Avaliação**: Período em que deve ser observado e avaliado como o artefato auxilia na resolução do problema. Essa atividade envolve a comparação dos objetivos de uma solução com os resultados observados no uso do artefato na demonstração. Requer conhecimento de métricas relevantes e técnicas de análise.

6. **Comunicação**: Última etapa, onde deverá ser comunicado o problema e sua importância, o artefato, sua utilidade e suas inovações, o rigor do seu projeto e sua efetividade para outros pesquisadores e outros leitores relevantes.

Desta forma, no contexto deste trabalho, foram realizados os passos necessários para construção dos artefatos.

1.2.1 Problema e Motivação

Ensley (1995) definiu o termo Consciência Situacional (SA), como "a percepção dos elementos no ambiente através do volume de tempo e espaço, a compreensão dos seus significados e a projeção de seu estado no futuro próximo" (ENSLEY, 1995, tradução livre). Em um ambiente urbano, um nível de consciência situacional baixo pode acarretar riscos à integridade do pedestre, tendo em vista que seu poder de reação a situações adversas tende a diminuir. Nasar, Hecht e Wener (2008) evidenciam que assim como motoristas, pedestres podem ter sua consciência situacional reduzida pela distração cognitiva de telefones móveis, aumentando comportamentos inseguros e assim, tornando os pedestres mais vulneráveis a acidentes e crimes.

1.2.2 Objetivos da solução

Avaliar técnicas de aprendizado estatístico quanto ao potencial representativo em consciência situacional de pedestres.

1. Utilizar consolidadas bases de pesquisa como o dblp e Google Scholar

2. Analisar os dados coletados manualmente e desenvolver um script para extrair as informações mais relevantes sobre o problema
3. Criar uma tabela de dados organizada com eventos do experimento e as consequências em consciência situacional de pedestres
4. Avaliar técnicas utilizadas em trabalhos relacionados e selecionar algumas para serem analisadas no contexto desse trabalho.
5. (a) Utilizar ferramentas bibliotecas que realizam de forma consistente e customizável a aplicação das técnicas
   (b) Extrair as variáveis mais importantes para cada técnica selecionada
   (c) Usar métodos para representação e validação da integridade da aplicação de cada algoritmo
6. Apresentar resultados de forma quantitativa e criar visualizações para demonstrar a diferença entre cada técnica selecionada

2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção serão tratados os assuntos que embasam os estudos realizados na comparação entre as técnicas de análise de dados no contexto de consciência situacional. Por se tratar de um trabalho que abrange disciplinas como estatística e inteligência artificial, são inseridos os conceitos que fundamentam as aplicações, desde os mais abrangentes até os mais específicos. Primeiramente são introduzidos conceitos gerais em ciência de dados, seguido de definições a respeito de métodos de aprendizado estatístico que atuam como classificadores e as características a serem avaliadas após as aplicações.

2.1 CONCEITOS GERAIS EM CIÊNCIA DE DADOS

O termo ciência de dados é comumente definido como uma metodologia que extrai conclusões a partir de dados. A tarefa realizada almeja produzir opiniões a partir de dados para serem utilizados como base em tomada de decisões (IGUAL; SEGUÍ, 2017). Em diversos domínios existem perguntas que podem ser respondidas utilizando conjuntos de dados para construção de modelos nos quais conhecimentos importantes possam ser extraídos utilizando técnicas de aprendizado, através de predições e inferências. Técnicas de aprendizado são métodos orientados a dados combinando conceitos fundamentais em ciência da computação com ideias de estatística, probabilidade e otimização (MOHRI; ROSTAMIZADEH; TALWALKAR, 2012).

Na construção de um modelo que represente o contexto desejado, existem as variáveis independentes e as dependentes. As variáveis independentes são as informações de entrada do modelo, que são utilizadas para estimar o valor da variável de saída, que é chamada de variável dependente. As variáveis independentes podem ser denotadas por X, onde $\mathrm{X} = X_1, X_2, \ldots, X_n$, , de forma que cada n-ésimo elemento de X corresponde a uma característica, ou atributo, do conjunto de dados (JAMES et al., 2013). Já a variável dependente é denotada por Y, representada na Equação 2.1 (JAMES et al., 2013) e se refere ao evento resultante ou quantidade que está sendo predita (KUHN; JOHNSON, 2013). O símbolo ϵ se refere ao erro no mapeamento entre X e Y, e deve ser minimizado.

$$Y = f(X) + \epsilon. \tag{2.1}$$

James et al. (2013) apresentam o termo "aprendizagem estatís-

tica", em essência, como um conjunto de abordagens para estimação da função f da Equação 2.1 Kuhn e Johnson (2013) empregam a nomenclatura de "modelos preditivos" para demonstrar a utilização dessas abordagens com o objetivo de obter predições precisas, sem priorizar a interpretação do modelo (KUHN; JOHNSON, 2013). Quando o intuito principal é a predição, a função utilizada para estimação da variável independente pode ser considerada uma caixa preta, sem se preocupar com sua forma exata, mas apenas em criar um modelo que faça predições precisas para Y. Já na inferência, não basta fazer predições com boa acurácia, ao invés disso, a prioridade é entender o exato relacionamento entre X e Y através da função f da Equação 2.1. Muitas vezes o problema acaba resultando em escolher qual abordagem seguir, porém é muito comum a utilização de uma combinação das duas (JAMES et al., 2013).

Adota-se como exemplo uma situação onde se tem como variáveis independentes as informações acerca de um tumor (ex: tamanho, cor, textura, formato) e para estimar se o mesmo é benigno ou maligno, é preciso tomar uma decisão. Se o contexto não requer boa interpretação dos resultados, pode-se criar um modelo preditivo, o qual receberá um conjunto grande de informações e servirá como um classificador quanto ao tipo dos tumores. Kuhn e Johnson (2013) estabelecem que "para modelos preditivos, a preocupação é com a acurácia dos preditores acima de fazer inferências válidas" (KUHN; JOHNSON, 2013, pag.42, tradução livre). Outra opção seria analisar os dados e verificar quais características sobre o tumor impactam com maior magnitude na variável dependente, tornando-se um problema de estatística inferencial.

Os métodos de aprendizagem estatística seguem algumas abordagens, sendo as principais: o aprendizado supervisionado, que recebe como exemplos pares de entrada e saída para definir uma função que os mapeia; o não-supervisionado, que busca encontrar relacionamento através de padrões nos dados sem receber *feedback* explícito, geralmente utilizando técnicas de agrupamento (*clustering*); e por reforço, que utiliza um modelo retroalimentado para auxílio da aprendizagem através de recompensas e punições (RUSSELL; NORVIG, 2003).

Os métodos de aprendizado estatísticos podem seguir dois caminhos: utilizar ajuste de parâmetros pré-estabelecidos (método paramétrico), ou encontrar padrões utilizando diferentes algoritmos (método não-paramétrico). Segundo Murphy (2012), os métodos paramétricos têm a vantagem de facilitar o uso, porém contém a desvantagem de não permitir fazer premissas mais consistentes sobre a natureza da distribuição de dados. Já os modelos não paramétricos são mais flexíveis, porém

são computacionalmente intratáveis para grandes conjuntos de dados. Trevor, Robert e JH (2009) afirmam que os métodos não-paramétricos necessitam de uma quantidade muito maior de dados para a representação, pois não reduzem o problema à ajustar uma pequena quantidade de parâmetros (TREVOR; ROBERT; JH, 2009).

Outros termos são frequentemente utilizados no processo de criação de modelos preditivos utilizando métodos de aprendizagem estatística. As variáveis independentes também podem ser chamadas de preditores, rótulos, atributos ou descritores, enquanto a variável dependente a ser prevista é chamada de classe, objetivo, resultado ou resposta. Uma amostra, ou observação, representa uma instância única, independente, de todos os elementos. O conjunto de treinamento é definido como um subconjunto de todas as amostras, e é utilizado para a criação da função que relaciona as variáveis dependentes e independentes. Já o conjunto de teste (ou validação) representa as amostras restantes com o objetivo de avaliar a performance dos modelos apresentados. Os dados utilizados nessa etapa podem ser divididos em dois tipos: Contínuos ou quantitativos, onde a informação contida pode ser quantificada, obtendo escalas naturais e numéricas; e Categóricos ou nominais, que contêm informações discretas, sem escalas (KUHN; JOHNSON, 2013).

Para os problemas de aprendizado supervisionado, os métodos utilizados geralmente se dividem em duas categorias: regressão e classificação. Se a saída esperada Y da função f da equação 2.1 for um nível contínuo, trata-se de uma regressão e o modelo preditivo pode ser nomeado como regressor. Em contraste, se conjunto de valores possíveis para a saída é finito, então o utiliza-se como variável dependente dados categóricos e, portanto, configura um modelo preditivo de classificação, podendo ser chamado de classificador. Se existirem apenas duas saídas possíveis para a predição, trata-se de um problema de aprendizado binário (RUSSELL; NORVIG, 2003).

Quando o problema é de classificação, uma boa forma de visualizar o resultado é através da geração de um gráfico de dispersão com as amostras do conjunto de testes juntamente com os limites de decisão. Segundo Russell e Norvig (2003), um limite de decisão é uma linha (ou uma superfície, para dimensões maiores) que separa as classes disponíveis no modelo preditivo. Um exemplo é demonstrado na Figura 1, onde em (a) é possível uma separação linear das amostras pela classe equivalente, enquanto (b) não consegue dividir o conjunto através de uma função do primeiro grau.

O *bootstrap* é uma ferramenta extremamente poderosa quanto

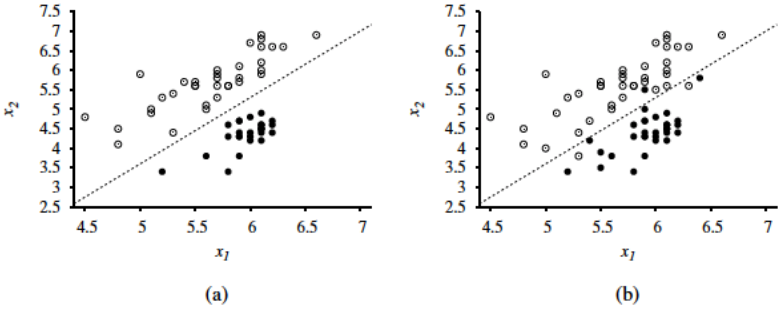

Figura 1 – Exemplo de gráficos relacionando dois parâmetros (x1 e x2). Em (a) a distribuição no espaço provê possibilidade de separação linear; (b) demonstra o problema com uma quantidade maior de dados, impedindo a separação linear (RUSSELL; NORVIG, 2003, pg. 723).

ao potencial de quantificar a incerteza associada com uma variável ou método de aprendizado estatístico (JAMES et al., 2013). Funciona dividindo o conjunto de dados original em subconjuntos denominados "*bootstraps*", os quais contêm a quantidade desejada de amostras, resultando em múltiplas divisões aleatórias distintas nos dados, utilizadas para avaliar a variância e o viés de predições. Pode avaliar essas características de praticamente todas as funções usadas, com o revés de precisar de maior poder computacional (IGUAL; SEGUÍ, 2017).

Em uma predição, o principal objetivo é encontrar um método estatístico de aprendizagem que represente os dados através do equilíbrio entre variância e viés. Como uma regra geral, quanto mais flexível o método, maior a variância e menor o viés (JAMES et al., 2013). A variância é definida como o erro de sensibilidade para flutuações no conjunto de dados, de forma que se ruídos estiverem presentes nos dados, a variância será incrementada no modelo resultante. O viés (ou bias) é a característica que diz respeito à suposições incorretas feitas no modelo, um nível de perda nas relações essenciais buscadas em uma predição (SKIENA, 2017).

Um alto nível de variância em um modelo acarreta numa propriedade denominada overfitting, a qual implica que o modelo busca melhorar sua acurácia baseado em ruídos enganosos nos dados, se ajustando muito bem apenas para o conjunto de treinamento (SKIENA, 2017). Na Figura 2, onde se visualizam modelos de regressão através de um gráfico de dispersão com amostras representadas através de pon-

tos verdes, demonstra-se que aumentar a complexidade de um modelo não significa deixar sua representação mais realística. A linha azul, que representa a regressão linear, contém taxa de erro maior por não passar por cima exatamente dos dados, como acontece com a linha verde, que foi gerada a partir de um polinômio de grau 9. Visualmente, é perceptível que amostras não utilizadas no treinamento não são generalizadas suficientes no modelo polinomial, característica que indica overfitting. Na Figura 3 são visualizados os limites de decisão em uma classificação, onde à esquerda ocorre uma generalização que minimiza variância e viés, enquanto à direita é se encontra um alto nível de variância, criando padrões inexistentes e obtendo *overfitting*. Em geral, se a taxa de acertos for muito maior avaliando como entrada o conjunto de treinamento em relação ao conjunto de testes, caracteriza-se *overfitting* (SKIENA, 2017).

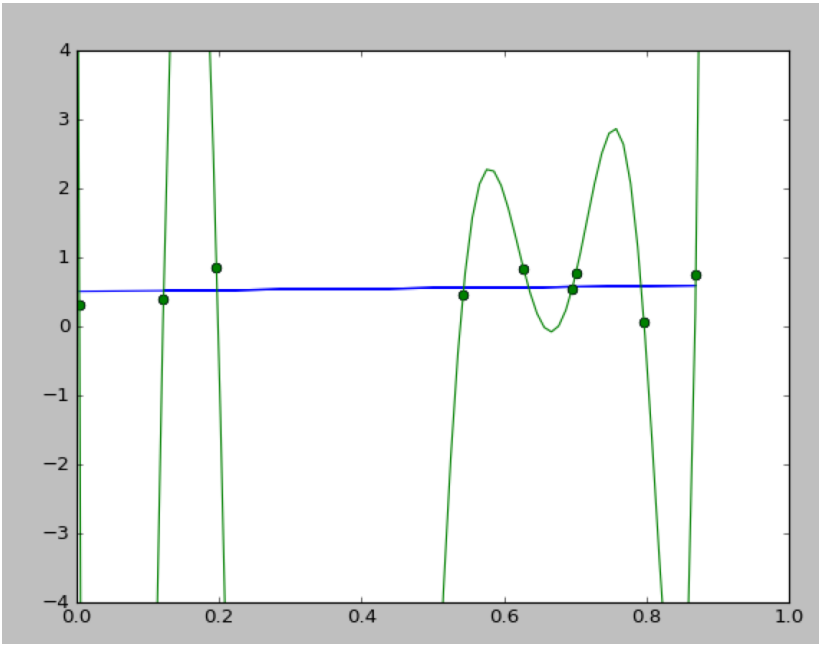

Figura 2 – Regressão Linear (em azul) e regressão polinomial de grau 9 (em verde). (Fonte: Elaboração própria).

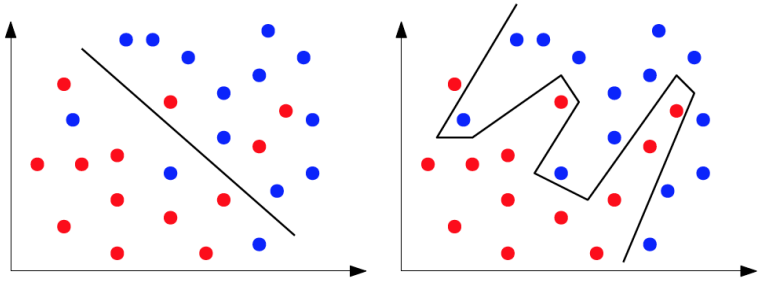

Figura 3 – *Overfitting* em classificação. Em (a) o limite de decisão representa de forma generalizada o padrão encontrado. Já em (b) é observado *overfitting*. (Fonte: Skiena (2017, pg. 304))

2.2 MÉTODOS DE APRENDIZADO ESTATÍSTICO COMO CLASSIFICADORES

Esta subseção abordará a implementação de métodos de aprendizado estatístico (JAMES et al., 2013) para utilização em modelos preditivos que atuam como classificadores, conforme esquema da Figura 4. Kuhn e Johnson (2013) definem que a modelagem preditiva é definida como "o processo de desenvolvimento de uma ferramenta matemática ou modelo que gera uma predição acurada" (KUHN; JOHNSON, 2013, pg. 2, tradução livre). Em análise de dados, não existe um método que terá garantia absoluta para qualquer domínio, apenas experimentos sistemáticos podem auxiliar na escolha do tipo do classificador. Esta característica foi provada por matemáticos sob uma prova rigorosa e ficou conhecida como teorema "*No-Free-Lunch*", que significa que não existe "almoço grátis" (ou facilidade) na escolha dessas abordagens (KUBAT, 2017).

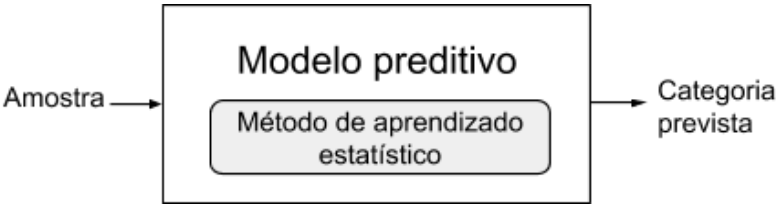

Figura 4 – Processo de predição em um classificador. (Fonte: Elaboração Própria)

2.2.1 Regressão Logística

Métodos de regressão são utilizados para obtenção de uma saída contínua através dos dados de entrada por meio de, por exemplo, funções lineares. Ao configurar um valor para atuar como threshold (ou limiar) em uma regressão linear, se configura uma divisão nos dados, obtendo um classificador linear. Porém, a natureza do *threshold* explícito pode causar alguns problemas, como a brusca decisão entre as classes de predição, como demonstrado na Figura 5 (a). Para evitar esse problema, é proposta a utilização da função logística, que suaviza a função de *threshold*, aproximando-a com uma função diferenciável. No caso da regressão logística, utiliza-se a função sigmóide, definida na equação 2.2 e demonstrada na Figura 5 (b). Um exemplo do resultado da aplicação da técnica com duas variáveis independentes, está demonstrado na Figura 5 (c) (RUSSELL; NORVIG, 2003).

$$\texttt{threshold}(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

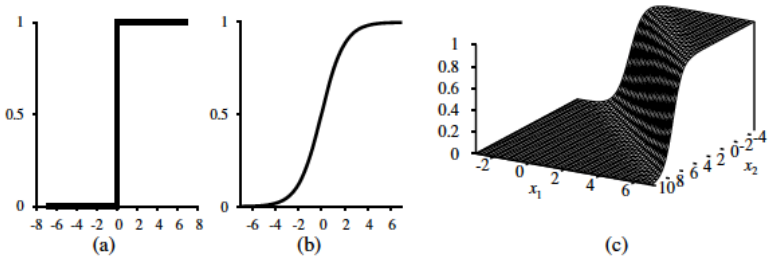

Figura 5 – Regressão logística. (a) representa a regressão linear com *threshold* explícito; (b) representa a função logística sigmóide; (c) representa um exemplo da utilização da regressão linear com 2 variáveis. (Fonte: Russell e Norvig (2003, pg. 726)).

Dentre os pontos positivos desse método, Akerkar e Sajja (2016) elencam que o mesmo: possui uma alta taxa de acertos para muitos conjuntos de dados simples; não faz suposições sobre a distribuição de classes; é resistente ao *overfitting*; além de ser facilmente extensível para múltiplas classes.

2.2.2 KNN - K vizinhos mais próximos

O método KNN é utilizado para classificar amostras baseado em proximidade no espaço das variáveis independentes em treinamento. É um algoritmo simples que armazena todos os casos disponíveis e classifica os novos através de uma votação, elegendo a classe majoritária a partir de seus k vizinhos. O caso mais simples é quando $k = 1$, onde a classe escolhida é encontrada avaliando a classe da amostra mais próxima encontrada no conjunto de treinamento. Incrementando o número de vizinhos a serem avaliados, melhora-se a resistência do método contra *overfitting*, porém dificulta a distinção de classes na visualização dos limites de decisão. A acurácia do algoritmo é fortemente degradada na presença de variáveis independentes irrelevantes ou mal tratadas, ou quando suas escalas não estão consistentes com a importância equivalente (AKERKAR; SAJJA, 2016).

Na Figura 6, pode-se analisar a diferença de k nos valores 1 e 100 para um exemplo. No caso de $k = 1$, verifica-se um nível baixo de viés e uma alta variância, enquanto com $k = 100$ a situação se inverte, ocasionando em um viés alto e uma baixa variância. Como deseja-se reduzir essas características, deve ser avaliado uma quantidade de vizinhos intermediária, buscando otimizar as predições. Ainda na Figura 6, a linha tracejada roxa indica um classificador bayesiano que gera limites de decisões similares ao KNN quando $k = 1$. (JAMES et al., 2013)

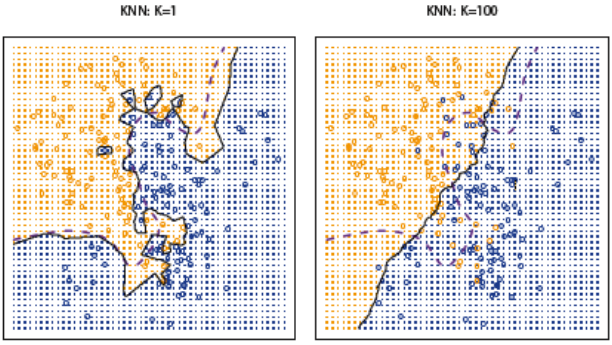

Figura 6 – Exemplo de aplicação do algoritmo KNN para k = 1 e k = 100. (Fonte: (JAMES et al., 2013, pg. 41))

2.2.3 SVM - Máquinas de vetores de suporte

As máquinas de vetores de suporte representam uma técnica eficiente para diversas aplicações de aprendizado supervisionado. Segundo Russell e Norvig (2003), na ausência de conhecimento prévio sobre o domínio do problema, esse método tende a ser excelente para começar. As SVM constroem um separador de margem máximo, isso é, limites de decisão com a maior distância possível entre as amostras, auxiliando na generalização, como demonstrado na Figura 7, onde em (a) são demonstrados separadores lineares candidatos, enquanto em (b) é evidenciado o caso onde a margem (demonstrada através da área entre linhas pontilhadas) que divide as classes é máxima (RUSSELL; NORVIG, 2003).

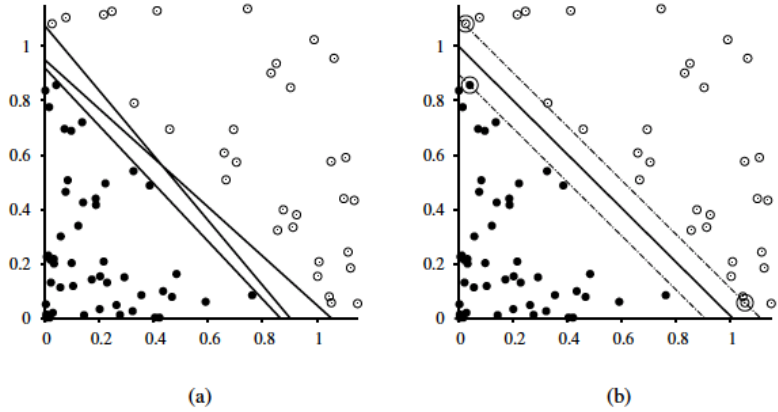

Figura 7 – Classificação em SVM: (a) Espaço amostral com três possíveis separadores. (b) Separador com maior margem entre as classes. (Fonte: Russell e Norvig (2003, pg. 745))

As SVM utilizam modelos *kernel*, que permitem a utilização da técnica em dados em espaço multidimensional, ocasionando em limites de decisão que fazem a separação entre classes de forma mais complexa, de maneira não linear. Ao contrário da classificação por regressão logística, a qual é feita utilizando uma função e modificando seus coeficientes, SVM são classificadores não-paramétricos, fornecendo flexibilidade sem ser suscetível à *overfitting* (RUSSELL; NORVIG, 2003).

Os modelos *kernel*, muito utilizados em aprendizado de máquina,

são funções definidas sob um produto intrínseco em um espaço multidimensional (MOHRI; ROSTAMIZADEH; TALWALKAR, 2012). Na Figura 8 são demonstrados dois casos de separação, em (a) os limites de decisão mostram que esse conjunto de dados não é separável linearmente, porém, através da utilização de funções kernel, o espaço amostral pode ser elevado a maiores dimensões, até que seja possível separar as classes linearmente e, então, reduzir as dimensões para o espaço original, demonstrado em (b), ocasionando em uma separação não linear (MOHRI; ROSTAMIZADEH; TALWALKAR, 2012). Esse processo é definido utilizando um mapeamento e extraindo uma medida de similaridade entre os elementos de entrada (MOHRI; ROSTAMIZADEH; TALWALKAR, 2012). A construção é feita através de diferentes técnicas e resultam em um kernel K, que pode ser linear (equação 2.3), ou expandido para polinomial (equação 2.4) ou radial (equação 2.5) por exemplo (IGUAL; SEGUÍ, 2017). A Figura 9 demonstra à esquerda uma separação utilizando kernel polinomial e à direita o uso de um kernel radial (JAMES et al., 2013). (MOHRI; ROSTAMIZADEH; TALWALKAR, 2012)

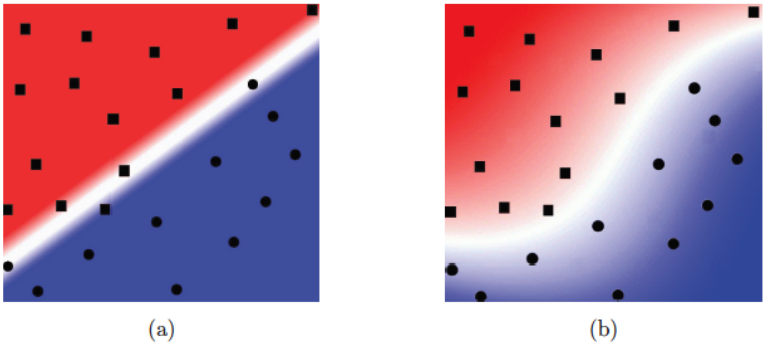

Figura 8 – Margens de separação: (a) Tentativa de separação linear. (b) Separação não linear. (Fonte: Mohri, Rostamizadeh e Talwalkar (2012, pg. 90))

$$K(xi, xj) = x_i^T x_j \tag{2.3}$$

$$K(xi, xj) = (1 + x_i^T x_j)^p \tag{2.4}$$

$$K(xi, xj) = e^{-\frac{\|x_i - x_j\|}{2\delta^2}} \tag{2.5}$$

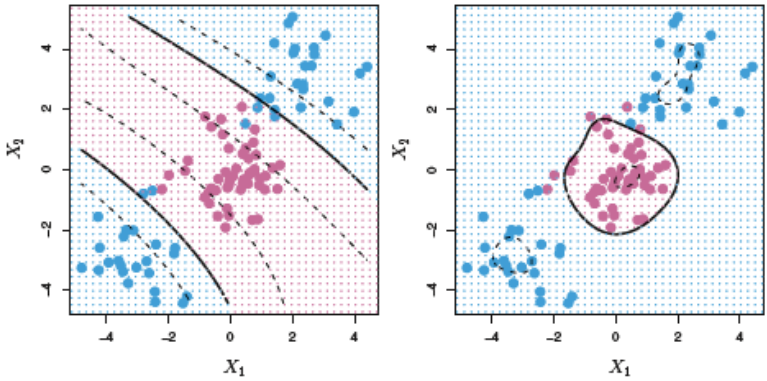

Figura 9 – Classificação utilizando kernels - À esquerda: Kernel polinomial de grau 3. À direita: Kernel radial. (Fonte: James et al. (2013, pg. 353))

2.2.4 Árvores de decisão

Russell e Norvig (2003) afirmam que a indução por árvores de decisão está entre as formas mais simples e bem sucedidas de aprendizado de máquina. Kuhn e Johnson (2013) definem que o seu objetivo, quando o intuito é classificação, é particionar os dados em pequenos grupos mais homogêneos, onde a homogeneidade nesse caso significa que os nodos para cada divisão são mais puros, ou seja, contendo uma proporção maior de uma classe em cada nodo. (JAMES et al., 2013) definem que a predição em cada amostra pertence à classe com ocorrências mais comuns no conjunto de treinamento, e que a criação da árvore é feita através de divisões binárias recursivas, utilizando alguns critérios pré-estabelecidos como a taxa de erros na classificação, e medidas de pureza, onde quanto valores mais baixos indicam a predominância de uma classe específica, como o índice Gini e a entropia. A árvore resultante das três fases pode ser analisada para entendimento da veracidade e precisão nos resultados. Em alguns problemas, o algoritmo pode gerar uma árvore muito grande, com muitos ramos e nodos, significando que um padrão não foi encontrado nos dados do conjunto (RUSSELL; NORVIG, 2003). Akerkar e Sajja (2016) dividem a elaboração das árvores de decisão em três fases:

1. **Construção**: Constrói-se uma árvore de decisão inicial, utilizando todas as amostras disponíveis. Requer o particionamento do conjunto de treinamento através dos critérios pré-estabelecidos anteriormente

2. **Poda**: A árvore resultante da fase anterior pode resultar em *overfitting*, não produzindo os melhores resultados possíveis. Essa fase remove ramificações e nodos para melhorar a performance do classificador

3. **Processamento**: A árvore podada é processada para melhoria na interpretabilidade.

Algumas das vantagens das técnicas de aprendizado utilizando árvores de decisão são: simplicidade, interpretabilidade, proximidade com a tomada de decisões do ser humano e utilização facilitada de variáveis categóricas, sem necessidade de discretização. Dentre as desvantagens, destacam-se a taxa de acertos (acurácia) menor do que de outras técnicas de classificação e a instabilidade gerada pelo fato de que pequenas mudanças nos dados podem acarretar modificações severas na árvore. Para mitigar esses problemas e melhorar a performance de predição, são utilizadas técnicas de aprendizado ensemble, como bagging, boosting e floresta aleatória (as quais serão abordadas na subseção 2.2.5) (JAMES et al., 2013).

2.2.5 Aprendizado ensemble

Segundo Russell e Norvig (2003), as técnicas de aprendizado ensemble ganharam popularidade por melhorar a performance de algoritmos de aprendizagem, tendo o primeiro método efetivo (*Bagging*) criado em 1996 por Leo Breiman. Essas técnicas utilizam a combinação de classificadores utilizando agregação, obtendo algumas propriedades no combate ao *overfitting* (IGUAL; SEGUÍ, 2017). Um conceito essencial na avaliação dessas técnicas é a importância de variáveis, tendo em vista que o aprendizado ensemble diminui a interpretabilidade do modelo, é interessante avaliar quantitativamente a influência de cada informação na classificação. Os métodos que utilizam árvores de decisão geralmente utilizam o índice de pureza, como o Gini, para calcular essa métrica (JAMES et al., 2013).

2.2.5.1 *Bagging* - Agregação *Bootstrap*

A alta variância causada na aplicação do método de árvores de decisão ocorre pois diferentes divisões no conjunto de treinamento ocasionam em classificações muito distintas. Como essa propriedade deve ser evitada, a técnica de *Bagging* foi inventada. Trata-se de um procedimento de propósito geral que, em suma, calcula a média de predições através de uma votação utilizando como dados de treinamento subconjuntos amostrais obtidos através do processo de *bootstrap*. A equação 2.6 demonstra o cálculo da média descrito anteriormente, onde B é a quantidade de subconjuntos (*bootstraps*), e f representa as predições para o bootstrap atual utilizando a árvores de decisão construída (JAMES et al., 2013).

$$\hat{f}_{bag(x)} = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{b}(x) \tag{2.6}$$

2.2.5.2 Floresta aleatória

Leo Breiman propôs em 2001 a adição de uma camada adicional de aleatoriedade ao método *Bagging*. Ao invés de construir as árvores de decisão utilizando a melhor divisão sob todas as variáveis, na Floresta aleatória, cada nodo seleciona a mais adequada a partir de um subconjunto aleatório de variáveis independentes do modelo (AKERKAR; SAJJA, 2016). (JAMES et al., 2013) explicam que a quantidade ideal de variáveis em um subconjunto aleatório é igual à raiz do número total de preditores. Dessa forma, uma possível influência de uma variável específica sobre a geração das árvores de decisão não ocorre, ocasionando em um processo que diminui a correlação na floresta (todas as árvores geradas), tornando a média das agregações resultantes menos variável e mais confiável (JAMES et al., 2013).

2.2.5.3 *Boosting*

Para entender a ideia do algoritmo de Boosting, é necessário introduzir o conceito de um conjunto de treinamentos ponderado. Para toda amostra em um conjunto de dados, é associado um peso $wj \geq 0$, e quanto maior o peso, maior a importância incorporada durante a aprendizagem (JAMES et al., 2013).

Dentre todos os métodos de aprendizado estatístico que utilizam a técnica de boosting, destacam-se o *Boosting* adaptativo (AdaBoost) e o *Boosting* gradiente (*Gradient Boosting*). O AdaBoost é um meta algoritmo, no sentido de que é utilizado para melhorar a performance de outros algoritmos. O processo inicia com todos os valores de w (para todos as amostras j) tendo atribuído o valor 1 e verifica quais amostras foram classificadas incorretamente, as quais são separadas e têm seu peso incrementado para a próxima iteração. Já os dados que foram classificados corretamente têm seu peso decrementado, e esse processo se repete K vezes, valor no qual foi inserido como entrada do algoritmo (RUSSELL; NORVIG, 2003). Ao invés de tentar representar todos os dados através de uma única grande árvore de decisão, que podem ocasionar em *overfitting* ou dificuldades na busca por padrões, a abordagem *Boosting* realiza o aprendizado de forma lenta, utilizando resíduos de árvores menores (JAMES et al., 2013). Já o *Boosting* gradiente funciona iniciando com um universo de pequenas árvores, com poucos nodos, implementando uma lógica suficiente para evitar *overfitting*, onde os pesos relacionados seguem de um procedimento de treinamento, tentando corrigir erros de iterações anteriores e aumenta o peso das árvores que classificaram corretamente os casos difíceis (SKIENA, 2017).

A técnica de *Boosting* contém três parâmetros de otimização (JAMES et al., 2013):

1. O número de árvores B, o qual deve ser controlado para não crescer, podendo ocasionar em overfitting

2. O parâmetro de contração λ, que controla a taxa de aprendizado do boosting e tem geralmente valores pequenos

3. O número **d** de divisões em cada árvore, que controla a complexidade do algoritmo.

Quando frações do conjunto de treinamento contêm dados que não estão bem representados ou rotulados, a técnica de Boosting pode ampliar erros através do seu potencial de encontrar padrões mesmo quando a quantidade de amostras é baixa. Por isso, os dados devem ter sido verificados e tratados anteriormente para uma melhor performance, evitando ruídos, coeficientes nulos ou com valores muito altos (SKIENA, 2017).

2.2.6 Redes neurais artificiais

Segundo Kubat (2017), a função de um neurônio é obter a soma dos sinais ponderados de entrada e submetê-los à uma função de transferência, que pode ser a sigmóide. As redes neurais, embora sejam um conceito antigo, ainda são muito popular e efetivas nas aplicações atuais. Segundo Russell e Norvig (2003), uma rede neural é "uma coleção de unidades conectadas de forma conjunta; as propriedades da rede são determinadas por sua topologia e pelas características dos neurônios" (RUSSELL; NORVIG, 2003, p. 728, tradução livre). Sua estrutura é dada pela composição de nodos conectados por links que são utilizados para propagação da função de ativação. Cada um desses links contém um peso associado, que determina a força e o sinal da conexão (RUSSELL; NORVIG, 2003).

Quando a função de ativação da rede neural é composta de um *threshold* explícito ou de uma função logística (como a sigmóide), a unidade é denominada *perceptron* (RUSSELL; NORVIG, 2003). A Figura 10 demonstra um exemplo do conceito *perceptron* multicamada, o qual consiste em múltiplas camadas contendo neurônios que utilizam uma função de transferência para analisar as entradas e mover para a camada seguinte. O processo ocorre iniciando dos sinais de entrada, passando pelas camadas ocultas, e obtendo os sinais resultantes através da camada de saída (KUBAT, 2017).

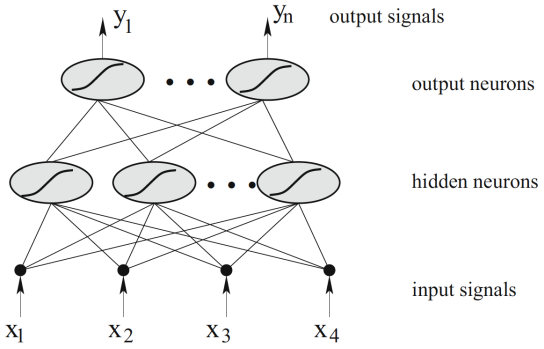

Figura 10 – Exemplo de rede neural com duas camadas. (Fonte: Kubat (2017, pg. 92))

2.3 INTERPRETABILIDADE DO MODELO

Existem muitos motivos para preferir um modelo mais restritivo, como por exemplo, quanto o objetivo principal é a realização de inferências estatísticas. Um modelo mais restritivo como o linear torna mais fácil o entendimento da relação entre cada variável independente com a variável dependente. No caso de métodos não-paramétricos, por exemplo, por serem muito flexíveis, a busca pela variável dependente que está mais associada com a variável independente se torna mais difícil. Como é possível analisar na Figura 11, em suma, quanto maior a flexibilidade do método de aprendizado estatístico, menor a interpretabilidade (JAMES et al., 2013). Embora pareça contra intuitivo, os modelos mais flexíveis nem sempre vão fazer predições mais acuradas. Isso se deve ao potencial de *overfitting* em métodos com alta flexibilidade (JAMES et al., 2013).

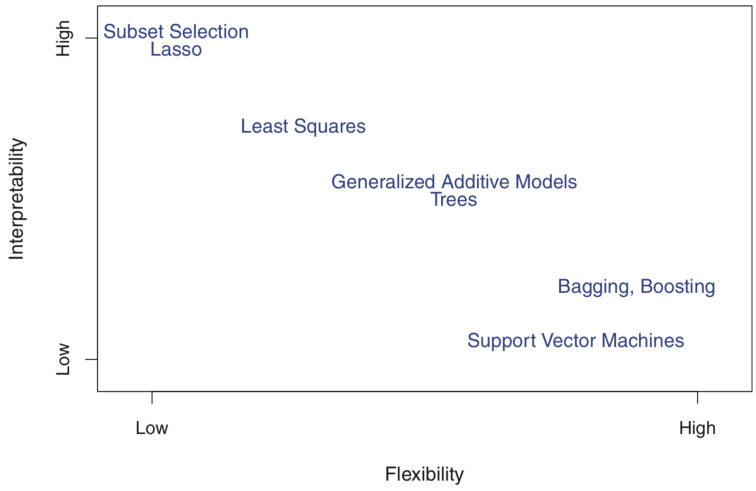

Figura 11 – Flexibilidade x Interpretabilidade do Modelo (JAMES et al., 2013).

2.4 QUALIDADE DO MODELO CLASSIFICADOR

Para decidir qual conjunto de técnicas melhor se adequa ao domínio do problema, algumas métricas devem ser extraídas das aplicações. A primeira medida a ser calculada através do resultado das predições é a acurácia, também chamada de taxa de acertos, que indica a proporção de amostras previstas corretamente de acordo com seu rótulo real. É possível avaliar a acurácia de um algoritmo através da curva de aprendizado, como demonstrado na Figura 12, onde o eixo vertical indica a taxa de acertos das amostras de teste de acordo com o tamanho do conjunto de treinamento, indicado no eixo horizontal (RUSSELL; NORVIG, 2003).

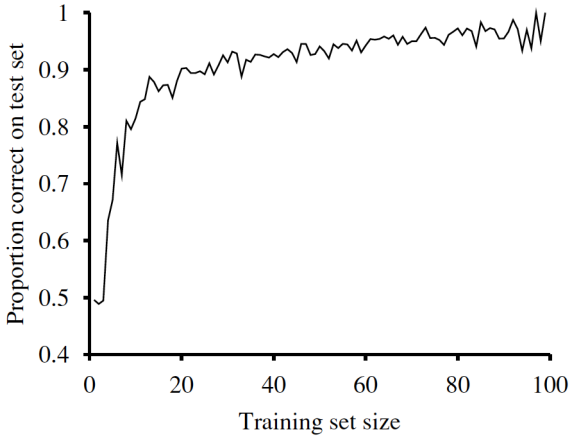

Figura 12 – Curva de aprendizado (Fonte: Russell e Norvig (2003))

Embora acurácia seja a métrica mais comum na validação de classificadores, existem casos onde a predição de elementos de uma classe contém um significado de domínio diferente das outras, exigindo uma avaliação mais específica dos resultados. Para resolver esse problema, a matriz de confusão foi criada, permitindo analisar outros conceitos dos modelos. Em um classificador binário quatro valores são extraídos (IGUAL; SEGUÍ, 2017):

- Verdadeiros positivos (**TP**): Quantidade de previsões de um classificador onde uma amostra com rótulo positivo é acertada
- Falsos positivos (**FP**): Quantidade de previsões de um classifica-

dor onde uma amostra com rótulo positivo é equivocada

- Verdadeiros negativos (**TN**): Quantidade de previsões de um classificador onde uma amostra com rótulo negativo é acertada

- Falsos negativos (**FN**): Quantidade de previsões de um classificador onde uma amostra com rótulo negativo é equivocada.

A partir da combinação desses elementos, uma série de novas métricas podem ser calculadas, como demonstrado na Tabela 1, onde também são apontados os aspectos da matriz de confusão. Além disso, pode-se reescrever a definição da acurácia através da equação 2.7, quanto menor for a quantidade de elementos presentes fora da diagonal principal da matriz, mais acurado será o modelo. Através das informações obtidas nas colunas, obtém-se a métrica de sensibilidade (equação 2.8), que indica a proporção de acertos nas predições de amostras positivas, e a especificidade (equação 2.9), que representa a proporção de acertos entre as amostras negativas. Ao analisar as linhas, obtém-se, conforme demonstrado na equação 2.10, a métrica de precisão (também chamada de taxa de predições positivas, ou TPR), a qual indica a proporção de acertos em todas as predições positivas, e a taxa de predições negativas (TNR) (IGUAL; SEGUÍ, 2017).

	Amostra positiva	Amostra negativa		Métricas
Predição positiva	TP	FP	→	Precisão (TPR)
Predição negativa	FN	TN	→	TNR
	↓	↓		
Métricas	Sensibilidade	Especificidade		

Tabela 1 – Matriz de confusão e métricas extraídas.(Fonte: Igual e Seguí (2017, pg. 73, tradução livre))

$$\texttt{acurácia} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.7}$$

$$\texttt{sensibilidade} = \frac{TP}{TP + FN} \tag{2.8}$$

$$\texttt{especificidade} = \frac{TN}{TN + FP} \tag{2.9}$$

$$\texttt{precisão}(TPR) = \frac{TP}{TP + FP} \tag{2.10}$$

$$\texttt{taxa de falsos negativos}(TNR) = \frac{TN}{TN + FN} \tag{2.11}$$

Uma ferramenta essencial na análise da qualidade de um classificador é a característica de operação do receptor, ou curva ROC. Essa é obtida através das taxas de precisão (TPR) e de falsos negativos (TNR), as quais são plotadas nos eixos x e y, respectivamente, ao aplicar variados níveis de *threshold*, ou seja, combinações diferentes entre quantidades de predições positivas e negativas. Um exemplo de curva ROC é demonstrado na Figura 13 (a), onde a curva A tem melhor desempenho que a B, pois nessa técnica a principal informação a ser extraída é a área sob a curva (AUC), onde o máximo valor possível é 1 e a curva passa pela margem superior esquerda, separando perfeitamente os falsos positivos e negativos (MURPHY, 2012).

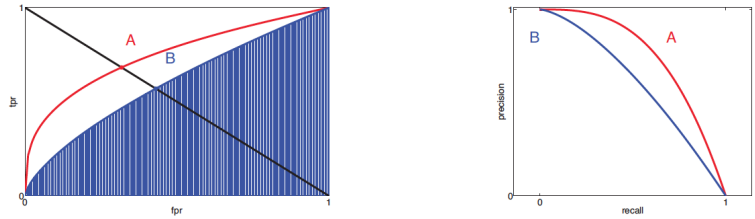

Figura 13 – Curvas ROC (a) e Precisão-sensibilidade (b). (Fonte: Murphy (2012, pg.182))

Quando o objetivo do classificador é realizar a predição de eventos que ocorrem raramente, ocorre uma alta quantidade de amostras negativas. Nesses casos, a curva ROC não consegue demonstrar o desempenho do modelo, pois a taxa de falsos positivos é muito pequena, o que é resolvido alterando a configuração do gráfico. Ao trocar a FPR pela métrica de sensibilidade (equação 2.5.2), obtém-se um gráfico similar à Figura 13 (b), o qual é denominado "curva precisão-sensibilidade" e funciona de maneira similar à ROC, analisando a AUC, porém a visualização é alterada, onde uma área ideal 1 acontece quando a curva passa pela margem superior direita. Nessa figura, também se observa que a curva A obteve um resultado melhor que a B (MURPHY, 2012).

2.4.1 Validação Cruzada

A partir do conjunto de dados, a utilização de um método de aprendizado estatístico é bem sucedida quando seu resultado provê uma taxa de erros baixa. No domínio do problema, se um conjunto de testes adequado estiver disponível, o processo é mais simples, porém geralmente não é o caso. Pode-se usar amostras do próprio conjunto de treinamento para avaliar a qualidade das predições, porém, ao seguir dessa forma, os resultados podem estar subestimados (JAMES et al., 2013).

Para suprir a ausência de um conjunto específico para testes, uma série de ferramentas são providenciadas para estimar a taxa de erros utilizando apenas os dados de treinamento. A validação cruzada é um método que funciona separando de maneira eficiente um subconjunto de treinamento para testes (JAMES et al., 2013). Para uma avaliação estatística avançada, a estratégia de validação cruzada utilizando N *folds* muitas vezes é escolhida, dividindo em N subconjuntos do mesmo tamanho (denominados *folds*). Dessa forma, são realizados N experimentos, onde em cada é retirado um *fold* para utilização como teste (isso garante que a cada execução, é feito um teste diferente) e o restante é utilizado para treinamento. Então, é feita uma média dos resultados e obtendo o desvio padrão calculado (KUBAT, 2017).

Uma vez que o conjunto de dados contém classes desbalanceadas (número de amostras rotuladas positivas muito maior que negativas, ou vice-versa), a divisão dos *folds* pode acarretar em testes com apenas amostras da classe majoritária, o que é indesejado. Para resolver esse problema, existe a abordagem estratificada, a qual força cada um dos N *folds* a conter a mesma representação de amostras de cada classe (KUBAT, 2017).

2.4.2 Análise de componentes principais (PCA)

Redução de dimensionalidade é o processo de converter um conjunto de dados contendo diversas dimensões em dados com dimensões menores, confirmando que representa informações relativamente similares. Essas técnicas são utilizadas na resolução de problemas de aprendizado de máquina para obter melhores características para uma predição (AKERKAR; SAJJA, 2016). A análise de componentes principais (PCA) é a técnica de redução de dimensionalidade mais utilizada atualmente (MOHRI; ROSTAMIZADEH; TALWALKAR, 2012) e se trata de um pro-

cedimento estatístico que transforma e converte dados em outros, os quais contêm variáveis não correlacionadas, denominadas componentes principais, que buscam capturar o máximo de informações possíveis do espaço amostral mais complexo (AKERKAR; SAJJA, 2016). Os componentes principais definem os eixos de um elipsóide que melhor se adequa aos pontos, onde a origem de cada eixo é o centróide dos pontos. A análise de componentes principais identifica a direção para projetar os pontos e representar o máximo de variância, através de uma linha similar à utilizada na regressão linear, e, projetando cada dado nessa linha, são definidas as novas dimensões na representação (IGUAL; SEGUÍ, 2017).

3 TRABALHOS RELACIONADOS

Este trabalho compara técnicas de análise de dados em consciência situacional, demonstrando o estado da arte em Ciência de dados e Aprendizado de máquina criando modelos preditivos aplicados num contexto específico. Muitos trabalhos utilizam esta metodologia, porém, englobam características distintas.

Diversos estudos avaliam a incidência de acidentes em vias urbanas apenas do ponto de vista do motorista. Pereira et al. (2017) realizou uma análise de consciência situacional de pedestres utilizando aprendizagem por reforço em redes bayesianas.

Estudos baseados em análise de dados tendem a oferecer uma alta diversidade de conceitos. O trabalho de Kotthaus (2018) comparou métodos em algoritmos de aprendizado de máquina estatísticos no contexto de utilização eficiente de recursos. Utilizou otimização baseada em modelos (abordagem essencial quando existem grandes demandas de recursos) para configuração automática de algoritmos paramétricos. Lhéritier (2015) utilizou métodos não-paramétricos para aprendizado e detecção de dissimilaridade estatística multivariada, incluindo técnicas de aprendizado ensemble. Bogojeska (2012) realizou um estudo sobre métodos de aprendizagem estatística no contexto de triagem terapêutica para pacientes com HIV de forma a utilizar tanto aprendizado supervisionado quanto não-supervisionado (busca por similaridades em históricos de terapias). Lu (2012) utilizou métodos de aprendizado estatístico para aplicações de contextos variados utilizando aprendizado de máquina e inteligência artificial, incluindo uma combinação de técnicas de árvores com boosting combinadas com processamento de sinais.

3.1 TABELA DE COMPARAÇÃO

Trabalhos	Demonstra estado da arte em ciência de dados?	Utiliza Aprendizado de máquina?	Utiliza modelos lineares?	Utiliza modelos com aprendizado ensemble?	Faz comparação de métodos de aprendizado estatístico?
(PEREIRA et al., 2017)	X	X			
(KOTTHAUS, 2018)		X	X		X
(LHÉRITIER, 2015)	X		X	X	X
(BOGOJESKA, 2012)	X		X		X
(LU, 2012)	X			X	X
(LANGER, 2018)	X	X	X	X	X

Tabela 2 – Tabela de comparação de trabalhos relacionados.

Este trabalho se diferencia pois analisa um conjunto maior de técnicas e faz uma comparação mais abrangente, porém com conceitos que podem ser aplicados em diversos contextos de análise de dados.

4 DESENVOLVIMENTO E ANÁLISE DE DADOS

Serão abordados, nesta seção, os procedimentos práticos deste trabalho desde os experimentos realizados, o pré-processamento dos dados, as abordagens para representação do nível de consciência situacional e a execução dos métodos de aprendizado estatístico.

4.1 CENÁRIO, DADOS E O EXPERIMENTO

O experimento teve como objetivo analisar o comportamento do ser humano quando inserido em um contexto que simula o tráfego urbano, onde acredita-se que o uso de smartphones pode influenciar na incidência de acidentes. Foram realizados dois experimentos diferentes através de uma plataforma de realidade aumentada. No primeiro, o usuário teria que perceber os carros e controlar uma sinaleira, procurando evitar atropelamentos de animais fictícios denominados 'lemmings' que constantemente atravessavam a faixa de pedestres (GELAIM et al., 2019).

Figura 14 – Participante controlando sinaleira para garantir a segurança dos 'lemmings' no primeiro experimento. (Fonte: Elaboração própria)

No segundo cenário, desenvolvido novamente em parceria com

a Universidade de Salford em 2017, o objetivo foi aumentar o estresse do participante tanto manipulando o dispositivo móvel, quanto na sua interação com o ambiente. Assim sendo, o participante ficava na rua, buscando evitar ser atropelado, enquanto interagia com o aparelho. Na figura 15 é apresentada uma perspectiva do ambiente utilizado para simulação deste experimento no sistema multi-modal Octave (GELAIM et al., 2019).

Figura 15 – Participante do segundo experimento. (Fonte: Elaboração própria)

Em todas as simulações, o ambiente contava com doze carros, sendo metade deles oriundos de trás do usuário e a outra metade provenientes de sua frente. É importante, ainda, salientar que seis veículos emitiam sons e todos se aproximavam separadamente. Os experimentos foram realizados por vinte participantes orientados a atravessarem a rua caso se sentissem ameaçados (GELAIM et al., 2019). Além disso, eram suscetíveis a três tipos de interação em seus aparelhos: Aplicativo com apenas um botão, onde o participante precisava pressionar assim que percebesse o carro se aproximando (simulação tipo 1); Aplicativo com o botão de percepção do carro e um jogo que necessitava interação constante (simulação tipo 2); Aplicativo com o botão de percepção, o jogo e um fone de ouvido com música (simulação tipo 3).

A ordem do aplicativo utilizado variou entre os três tipos de inte-

ração. Além disso, antes dos três experimentos, os usuários faziam um teste inicial com 5 carros para se familiarizar com o ambiente. Os dados coletados foram vídeos, áudios, posições dos objetos, ações e seus tempos. O jogo utilizado para distração do participante está demonstrado na figura 16, e baseia-se em controlar uma bola colorida procurando não a deixar cair no "chão" sequer encostar em obstáculos quando sua cor não for correspondente. É um desafio que exige um grande nível de concentração por parte do usuário, o que representa a distração em um contexto da consciência situacional dos pedestres nas vias urbanas (GELAIM et al., 2019).

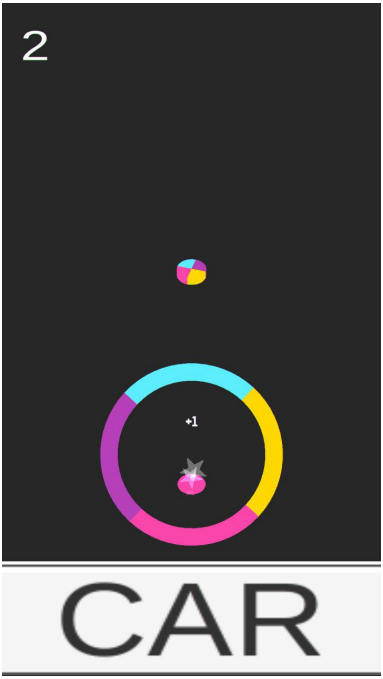

Figura 16 – Interface do aplicativo para simulações do tipo 2 e 3. (Fonte: (GELAIM et al., 2019))

Na conjuntura deste trabalho, foram utilizados apenas os dados do segundo experimento, por conter mais informações relevantes quanto ao nível de consciência situacional, com distrações maiores e, por consequência, representação de situações de risco ao pedestre.

4.2 PRÉ-PROCESSAMENTO

Os dados coletados foram agrupados por usuários do experimento, onde as informações sobre a interação com o aplicativo e todo o cenário foram separados em arquivos para cada tipo de interação no dispositivo móvel. Portanto, cada experimento conta com três arquivos sobre os dados do aplicativo e outros três arquivos sobre o cenário. Os arquivos foram salvos no formato *json*, contendo informações absolutas e descritivas sobre todo o ambiente e o aplicativo, incluindo coordenadas e tempo decorrente de cada evento. Os arquivos possuem uma grande quantidade de dados de tipos variados, dificultando sua análise e, para isto, foram necessárias transformações para facilitar a interpretação das informações.

Para realizar as análises deste estudo, foi necessário organizar as informações importantes distribuídas em eventos pertencentes aos intervalos entre o surgimento de um veículo no cenário e sua retirada. Desta forma, o propósito das análises utilizou eventos independentes, sem continuidade e sem distinção entre usuários. Dividindo o conjunto de dados coletados através do experimento desta forma, foram obtidas informações suficientes para encontrar padrões e classificar se um pedestre está correndo algum risco. Então, para cada veículo, foram armazenados no esquema os dados com um script na linguagem de programação *python*. Dentre as informações extraídas (GELAIM et al., 2019), as mais importantes para o contexto de consciência situacional são:

- *Added*: Tempo de execução do experimento no qual o carro foi adicionado; Removed: Tempo de execução do experimento no qual o carro foi removido;
- *Sound*: Variável binária que indica se o carro produzia sons;
- *Is Occluded*: Variável binária que indica se o carro vinha de uma posição oclusa ao usuário;
- *Critical AVG Speed*: Velocidade média registrada pelo carro em todo o trajeto, utilizando o tempo necessário para alcançar o usuário e a distância percorrida;
- *Direction*: Direção do veículo;
- *Time Critical*: Tempo de execução absoluto no qual o veículo e o usuário ocuparam a mesma posição no cenário;

- *Time Critical from Added*: Variável adicionada com o objetivo de obter o intervalo de tempo entre o surgimento do veículo no cenário e o momento que ocupasse a mesma posição que o usuário no cenário;

- *Safe Lane*: Lado da rua na qual o usuário está seguro;

- *Simulation Type*: Tipo de simulação (Button, para simulação simples, onde era apenas necessário apertar o botão de percepção do carro; SoundOff, que adicionava a distração do jogo no aplicativo do celular; e SoundOn, que além do aplicativo, adicionava música em um headphone ao participante como distração);

- *Moved To Current Lane*: Tempo no qual o usuário atravessou para a posição atual;

- *Moved To Next Lane*: Tempo no qual o usuário atravessou para a outra posição;

- *User Movement*: Somatório do deslocamento total absoluto de um usuário no cenário durante um evento;

- *Head Rotation*: Somatório da quantidade de rotação total absoluta da cabeça de um usuário no cenário durante um evento;

- *Time for Aware*: Tempo necessário para o usuário sinalizar a percepção do veículo desde seu surgimento no cenário através do botão de "CAR" no aplicativo, demonstrado na Figura 16;

- *Run Over*: Indica se o usuário não conseguiu trocar de pista em tempo e acabou sendo submetido à uma situação de atropelamento no contexto do experimento.

- *Is Aware*: Indica se o usuário apertou o botão "CAR" do aplicativo, o que indica ter percebido o veículo antes do mesmo ultrapassar sua posição atual.

Também foram adicionados os dados provenientes da utilização do aplicativo por parte do usuário em cada amostra do esquema criado. Estes estão presentes exclusivamente quando a simulação exigia mais do que apertar o botão de percepção do veículo, ou seja, continha distrações do jogo:

- *Points*: Total de pontos ganhos pelo usuário no jogo;

- *Deaths*: Quantidade de vezes que o usuário teve uma "morte" no jogo
- *Max Obstacle*: Nível máximo que o usuário alcançou no aplicativo durante o período.

Apesar de todas essas informações estarem disponíveis para a criação do modelo preditivo, deve ser analisado quais podem ser obtidas no contexto real de um ambiente urbano. A seção 4.3 apresenta o modelo desenvolvido para utilização no apoio à tomada de decisões, onde as leituras são continuamente atualizadas para a inteligência do sistema detectar situações de risco. Alguns dados foram simplesmente selecionados ou renomeados, enquanto outros foram agrupados por apresentarem um contexto maior.

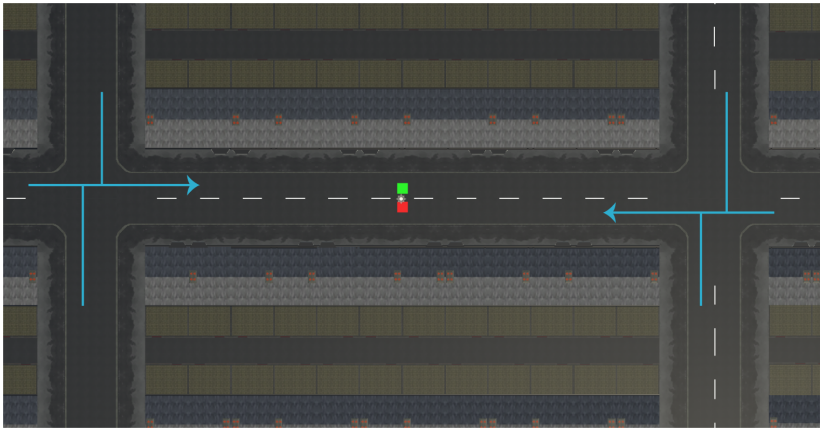

Figura 17 – Visão superior do cenário. (Fonte: (GELAIM et al., 2019)

Analisando somente os veículos, o que é possível de ser obtido no ambiente real e ser útil para predições é a presença de som, renomeado para car_has_sound, além de dados sobre sua velocidade média (denominado car_avg_speed) e origem no cenário, como sua oclusão ao usuário, que foi chamada is_car_ocluded. Informações específicas sobre sua direção também podem ser extraídas, porém uma variável enumerando as opções possíveis pode influenciar o modelo preditivo de forma negativa e não representar a realidade. Este problema pode ser corrigido aplicando a técnica de codificação *One-Hot*, criando 6 bits que representam cada direção do carro (demonstrados através da vista superior do cenário na Figura 17): car_direction_back,

car_direction_back_left, car_direction_back_right, car_direction_front, car_direction_front_left, car_direction_right. Por exemplo, quando a amostra representa uma origem da esquerda e por trás, car_direction_back_left é setado para 1, enquanto o resto é mantido em 0.

Para os dados relacionados ao pedestre e suas percepções, foram selecionadas algumas informações. Sua movimentação no cenário e rotação da cabeça foram renomeados para user_movement e head_rotation, respectivamente. Para representar a presença de distrações proveniente de smartphones, o tipo de simulação foi selecionado para determinar se o pedestre está utilizando aplicativos no celular (simulation_type_button), além da informação se o mesmo está escutando música em fones de ouvido (simulation_type_sound_off e simulation_type_sound_on). Além dessas variáveis binárias sobre a simulação, um nível contínuo de distração foi atribuído à variável app_distraction, a qual utiliza o desempenho no aplicativo (usando todas as características obtidas) para representar a imersão variando entre 0 e 1.

Para a construção da variável dependente, que representa o nível de consciência situacional do pedestre a ser previsto, foram utilizadas outras informações do experimento. A escolha desses parâmetros e abordagens utilizadas para o prosseguimento das análises, objetivo deste trabalho, estão relatadas na seção 4.4.

4.3 PROPOSTA DE APLICAÇÃO DO MODELO

Utilizando as informações extraídas dos dados do experimento, o objetivo é criar um modelo preditivo obtendo a configuração de fatores que interferem na percepção dos veículos por pedestres, aplicando métodos de aprendizado estatístico. Para isto, a estratégia utilizada foi verificar a correlação entre as variáveis e definir abordagens que levam a uma variável dependente a qual representa o nível de consciência situacional. Uma vez que essa variável é definida, o trabalho remanescente consiste em avaliar como as informações disponíveis em ambientes urbanos podem auxiliar na tomada de decisões. Em uma visão mais geral, necessita-se monitorar em tempo real o ambiente em que o pedestre está inserido e suas distrações. Portanto, a utilização prevista do modelo pode ser observada na na Figura 18, onde os dados são utilizados como entrada para o modelo preditivo responsável por verificar a necessidade de notificar o usuário, com o objetivo de cessar distrações e fornecer tempo para que o mesmo evite ser submetido à

situações de risco.

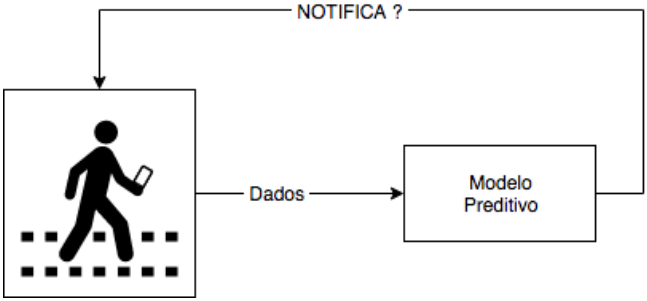

Figura 18 – Esquema de utilização de modelo preditivo para notificação de pedestres. (Fonte: Elaboração própria)

4.4 REPRESENTAÇÃO DE CONSCIÊNCIA SITUACIONAL PARA PEDESTRES

Como o intuito é obter a representação de situações com risco, foram selecionadas algumas informações dentre todas as extraídas do experimento para desenvolver uma métrica denominada *awareness*, a qual visa servir como referência para avaliação das correlações e fornecer um modelo computacional adequado aos objetivos do projeto, a ser utilizado futuramente. Utilizando como base o conceito de que um aplicativo derivado desse modelo pode utilizar as informações providas em tempo real para detectar situações de risco e alertar o pedestre, o grau de consciência situacional pode ser representado através de níveis pré-definidos, onde padrões comportamentais sugerem os possíveis avisos ao usuário. Outra forma de representação é criar um nível quantitativo e contínuo, do nível consciência situacional, que avalia se o usuário está completamente consciente dos riscos do seu entorno, onde o valor 0 (zero) representa o mínimo de atenção possível, e o valor 1 (um) representa um pedestre completamente atento.

Da maneira como foi projetado, o experimento provê informações suficientes para avaliar a percepção do pedestre em relação aos componentes do cenário e utilizar o tempo no qual o mesmo demorou para identificar um veículo se aproximando e, quando necessário, mover-se para evitar um possível acidente. As informações são provenientes das variáveis obtidas no pré-processamento, as quais se destacam:

- '*Is Aware*', que indica se o usuário pressionou o botão de percepção do veículo
- '*Time for Aware*', que fornece o tempo para percepção
- '*Run Over*', que indica se o usuário tomou a decisão correta para não ser submetido a uma situação de atropelamento, estando do lado correto da rua

Com base nessas informações, a consciência situacional foi modelada através de duas abordagens: atribuição por classes e por nível quantitativo.

4.4.1 Classes de consciência situacional

Através dos dados escolhidos para definir uma variável dependente para construção do modelo computacional de consciência situacional, a primeira estratégia é criar categorias que demonstram o nível de segurança do pedestre baseadas em informações sobre sua percepção. Como as variáveis *Is Aware* e *Run Over*, de forma integrada, podem fornecer uma resposta sobre os riscos do pedestre, foram selecionadas para compor as classes. Ambas as variáveis são binárias, o que culmina em quatro combinações diferentes para classificar cada amostra do experimento, proporcionando maior importância à variável que representa o atropelamento, considerando que o usuário prioriza evitar sofrer um acidente a apertar o botão de percepção do veículo. Desta forma as classes foram definidas como:

1. **danger**: Perigo, onde a situação do experimento tende a uma circunstância em que o usuário não percebe o carro, tampouco evita ser atropelado
2. **inattentive**: Desatento, onde o usuário percebe o carro através do aplicativo, porém não evita ser atropelado
3. **at_risk**: Em risco, onde o usuário não indica a percepção do carro, porém não é atropelado
4. **safe**: Seguro, onde o usuário percebe o carro e não é atropelado

A distribuição quantitativa de eventos baseados em suas classificações, foi organizada de acordo com a Figura 19, onde a grande maioria dos usuários esteve no estado safe (seguro). Do total de 732

dados disponíveis, 53 foram de situação de perigo (danger), 12 foram de situações de usuário desatento (Inattentive), 88 foram de situações em risco (at_risk) e as outras 579 amostras foram classificadas como situações seguras (safe).

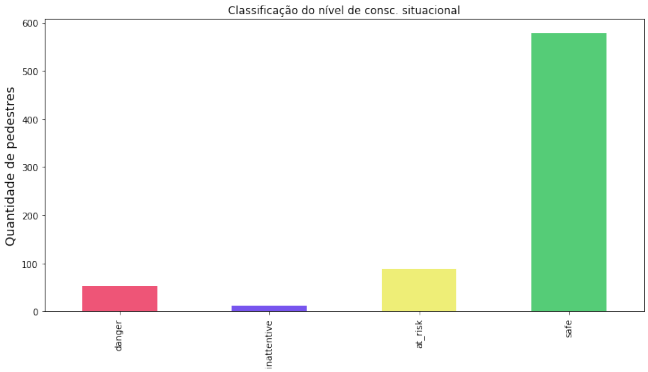

Figura 19 – Distribuição de quantidade de eventos por classe. (Fonte: Elaboração própria)

Utilizando os dados resultantes da formação das classes, técnicas de aprendizado estatístico que funcionam como classificadores podem ser utilizadas para criar um modelo que represente cada classe conforme as variáveis independentes que possuam maior importância. Isso fornece uma precisão para ativar possíveis notificações para auxílio na tomada de decisão pelos pedestres quando o modelo final resultante deste trabalho for implementado.

Porém, como o objetivo final do projeto é definir se o pedestre e o motorista deverão ser alertados quanto ao risco de um acidente, é interessante que a classificação se dê na forma binária. Portanto, é possível treinar o modelo baseando-se que uma notificação deve ser emitida caso a classe do usuário não seja safe, levando a uma nova variável booleana derivada, nomeada notify_user. A distribuição dos pedestres em cada classe de notificação está contida na figura 20.

4.4.2 Nível quantitativo de consciência situacional

A análise efetuada através da primeira abordagem provê uma modelagem do problema utilizando variáveis dependentes discretas sobre o resultado, negativo ou positivo, de cada evento de veículo no

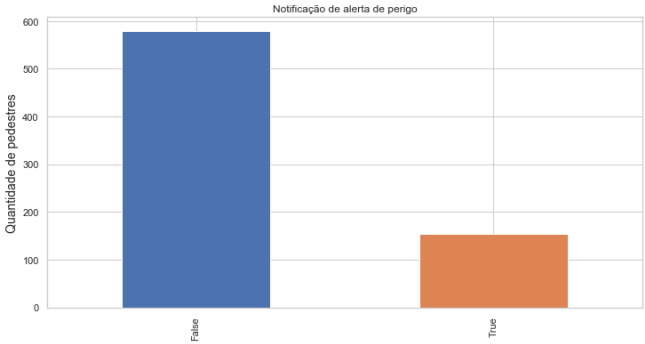

Figura 20 – Distribuição dos dados por classe de notificação. Os dados em azul representam as situações seguras (classe *safe*). (Fonte: Elaboração própria)

experimento. Como foram utilizadas as quatro classes para entendimento do comportamento e separação entre uma situação onde seria necessária uma notificação com alerta de perigo ao usuário ou não, o modelo tende a ficar mais restrito, porém não significa piora na performance. A segunda abordagem utiliza a informação referente ao tempo de segurança do usuário, isto é, o usuário percebeu o carro em tempo suficiente para evitar uma situação de perigo?

Utilizando como base as variáveis geradas na etapa de pré-processamento dos dados, a variável *Time for Aware* (que indica o tempo necessário para percepção do veículo desde seu surgimento no cenário) juntamente com a informação de Run Over (que indica uma situação de atropelamento) podem ser utilizadas em conjunto para demonstrar um nível quantitativo de consciência situacional de pedestres em cada ocasião.

Uma vez que *Time for Aware* é uma variável com valores contínuos e *Run Over* é uma variável binária, uma função deve ser utilizada levando em consideração um peso para cada medida. Partindo do pressuposto que o tempo para um pedestre perceber o veículo é inversamente proporcional ao seu nível de atenção, pode-se utilizar essa informação para criar uma fórmula do nível de consciência situacional do usuário. Um fator multiplicativo n é usado para indicar a importância da ausência de atropelamento no evento atual do experimento, demonstrado na equação 4.1.

$$\texttt{awareness} = (\texttt{maxTFA} - \texttt{userTFA}) \\ * (n - \texttt{runOver} * n + \texttt{runOver}) \tag{4.1}$$

Onde *awareness* é o nível de consciência situacional final de cada usuário; *maxTFA* representa o tempo máximo que os usuários levaram para perceber o carro; *userTFA* simboliza o tempo que o usuário demorou para perceber o carro; n é o fator multiplicativo de segurança; e *runOver* é um valor binário, onde 1 indica que o mesmo foi atropelado no experimento.

Assim, para cada evento no conjunto de dados, é aplicada a função e seus valores são normalizados, bem como todas as variáveis independentes quantitativas que poderão ser utilizadas posteriormente no modelo de aprendizado estatístico. O valor utilizado para o fator multiplicativo n foi 2, o que representa uma duplicação na atribuição do nível de consciência situacional do pedestre caso evite o atropelamento. Se esse fator for aumentado, o sistema tende a incrementar o nível de concentração na construção da variável dependente quando o veículo tiver atropelado o usuário, reduzindo a importância do tempo para de percepção. A distribuição do nível normalizado da variável awareness se deu de acordo com a figura 21.

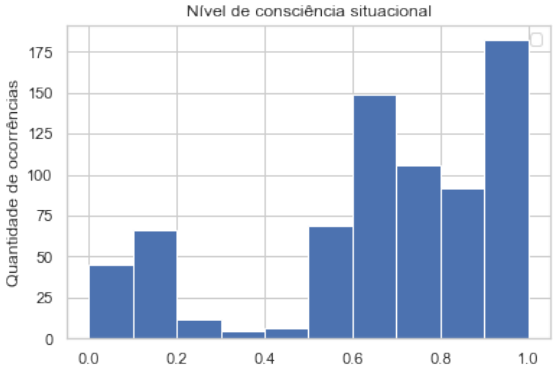

Figura 21 – Histograma da quantidade ocorrências de acordo com o nível de consciência situacional. (Fonte: Elaboração Própria)

Isso confere uma precisão maior no nível de consciência situacional. Porém, o objetivo ainda é realizar uma classificação binária para verificação da necessidade de alerta ao usuário para auxílio na tomada

de decisão. Para tanto, foi escolhido um valor chamado *threshold*, que indica um divisor variável para o nível que é considerado seguro para o pedestre em uma via urbana. Assumindo o *threshold* como 0.7, a distribuição de eventos com necessidade de notificação ao usuário se deu conforme a figura 22.

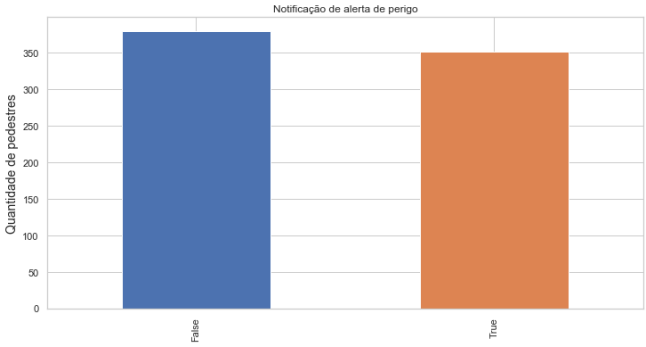

Figura 22 – Classes de notificação utilizando *threshold* de 0.7. Os dados em azul representam as situações seguras (classe safe). (Fonte: Elaboração própria)

Uma configuração de *threshold*, dessa forma, pode acarretar notificações desnecessárias, minimizando a importância perante situações realmente perigosas. Diminuindo o valor para 0.5, obtém-se uma distribuição que converge melhor de acordo com o contexto, evitando notificações desnecessárias, porém implicando em classes mais desbalanceadas, conforme figura 23. Contudo, dependendo do método de aprendizagem estatística utilizado para criar o modelo de predição de riscos a pedestres, esse desbalanceamento pode dificultar a busca por padrões exclusivos da classe com menor número de amostras.

4.4.3 Discussão sobre as representações

Nessa subseção foram apresentadas duas abordagens para representar quais situações devem ser notificadas ao pedestre no sistema proposto. Na primeira, onde a tomada de decisão é influenciada pela categoria na qual o usuário está inserido, assume-se que as circunstâncias do experimento que o sujeito foi atropelado e não sinalizou percepção do veículo são padrões que devem levar a notificações no sistema. Porém essa hipótese ainda deve ser testada em experimentos

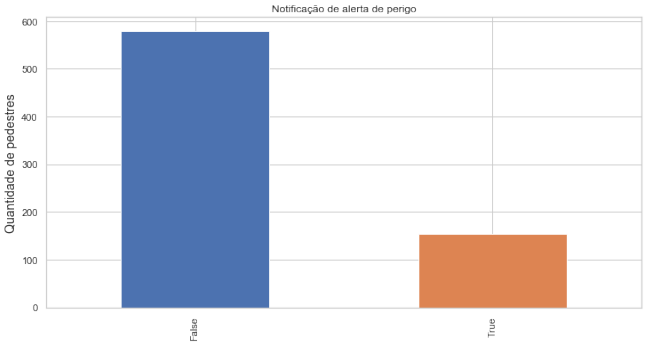

Figura 23 – Classes de notificação utilizando *threshold* de 0.5. Os dados em azul representam as situações seguras (classe safe). (Fonte: Elaboração própria)

futuros, isto é, deve-se analisar se esses padrões estão representando os riscos com precisão. Como só foram separados 4 níveis nesta representação, os ajustes possíveis podem não ser suficientes. A segunda abordagem, como utiliza uma função para determinar um valor quantitativo de consciência situacional, contém o valor denominado *threshold* para fornecer mais versatilidade ao modelo, onde ajustes podem ser feitos com o objetivo de aumentar ou diminuir a incidência de notificações. A diminuição desse valor é feita para reduzir as notificações desnecessárias (o que leva à perda de importância quando a situação for perigosa). Aumentá-lo significa que as notificações não estão suficientes para manter o pedestre seguro.

A efetividade de cada uma das abordagens criadas só poderá ser avaliada após a aplicação de ambas as técnicas de aprendizado estatístico selecionadas. Por mais que a segunda abordagem forneça mais versatilidade, se a primeira fornecer melhores resultados nas métricas selecionadas para avaliação, deve ser considerada para escolha de um modelo final na evolução do projeto.

4.5 ANÁLISE EXPLORATÓRIA DE DADOS

Estabelecido o conjunto de dados para aplicar técnicas e estudar os resultados, observou-se a dependência entre todas as variáveis da primeira abordagem, na Figura 24, e da segunda abordagem utilizando como *threshold* o valor 0.5, na Figura 25, aplicando o método de Corre-

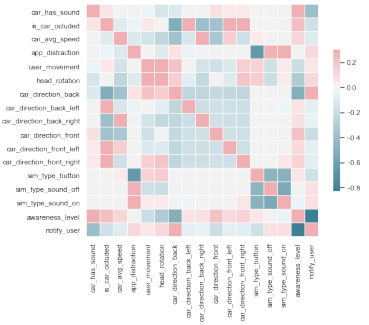

Figura 24 – Correlação entre variáveis da primeira abordagem. (Fonte: Elaboração própria)

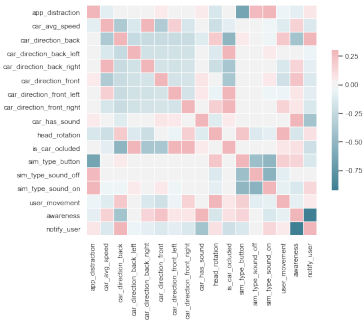

Figura 25 – Correlação entre variáveis da segunda abordagem. (Fonte: Elaboração própria)

lação de *Pearson*, onde o valor está demonstrado no indicador de cores (à direita da imagem). Quanto mais azulado, tende ao negativo, e mais rosado, tende mais para positivo. Se forem observadas apenas as relações das variáveis utilizadas como dependentes na criação do modelo, pode-se perder resultados cruciais, pois um conjunto de informações de diferentes origens podem ter influências que não são demonstradas de forma linear direta.

As variáveis binárias que indicam a presença de som no carro e se o mesmo é proveniente de uma posição oclusa demonstram conter grande influência no nível de consciência situacional do pedestre. Conforme demonstrado na Figura 26, a presença de som no carro (caixa laranja) indica um valor médio e desvio padrão tendendo para valores próximos ao máximo valor para atenção do usuário. Essa análise foi feita utilizando o nível quantitativo de awareness normalizado.

Para conseguir visualizar a influência do nível de distração por parte do aplicativo na consciência situacional do pedestre, o que é a premissa inicial dessa pesquisa, foi desenhado um gráfico (Figura 27) de três dimensões avaliando a variável app_distraction, que compõem a imersão do usuário no aplicativo, com a variável head_rotation, que indica se o usuário movimentou muito a cabeça durante o uso do smartphone, com a variável contínua de *awareness*. As amostras demonstram-se esparsas e não indicam correlação direta visível. Porém, com a criação do modelo preditivo no prosseguimento do trabalho, a importância da variável de distração do aplicativo no conjunto com

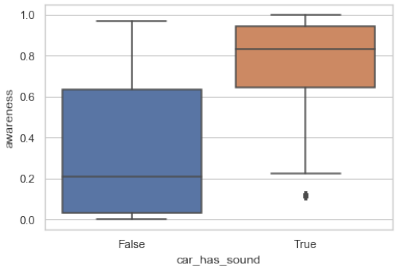

Figura 26 – Gráfico de diagrama de caixas (BoxPlot) relacionando o nível de consciência situacional do pedestre com a presença de som do veículo. (Fonte: Elaboração Própria)

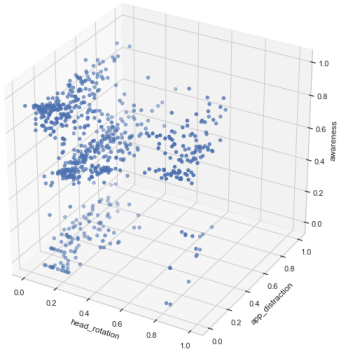

Figura 27 – Gráfico 3D relacionando as variáveis independentes app_distraction e head_rotation com a variável dependente awareness. (Fonte: Elaboração Própria)

outros fatores utilizados como variáveis independentes será avaliada.

4.6 SELEÇÃO E APLICAÇÃO DOS MÉTODOS DE APRENDIZADO ESTATÍSTICO

A seleção dos métodos de aprendizado estatístico para avaliação quanto ao potencial de representação em um modelo preditivo, foi feita de acordo com as características encontradas nos dados extraídos do experimento. O objetivo é abranger variadas estratégias que obtenham um bom funcionamento quando submetidas à treinamento utilizando uma quantidade relativamente pequena de dados (menos de 1000 amostras). Sendo assim, foram escolhidas técnicas lineares e de vizinhança, além de métodos de árvores de decisão e de ensemble learning, além de redes neurais. Os métodos escolhidos para a classificação foram:

- Regressão Logística
- K Nearest Neighbors (KNN)
- Máquinas de Vetores de Suporte (SVM)

Aplicação	Abordagem	Threshold	Notificar	Não notificar
1	1 (categórica)	-	173	579
2	2 (contínua)	0.5	134	598
3	2 (contínua)	0.7	352	380

Tabela 3 – Distribuição de classes para cada aplicação. (Fonte: Elaboração Própria)

- Árvores de decisão
- Adaptive Boosting (AdaBoost)
- Bagging
- Gradient Boosting
- Floresta aleatória (Random Forest)
- MLP - Perceptron Multicamadas.

Para cada técnica, avaliou-se três aplicações, uma com a primeira abordagem (categorias) e as outras duas com a segunda abordagem (nível quantitativo) com o valor de *threshold* 0.5 e 0.7. A distribuição de classes para cada aplicação está demonstrada na Tabela 3.

Para cada método utilizado, foi empregado o método de validação cruzada (aplicando *Folds* estratificados), juntamente com seleção de variáveis para uma classificação mais precisa, conforme demonstrado no esquema da Figura 28. Então, foram extraídas métricas de avaliação, que serão detalhadas na subseção 4.6.1, envolvendo acurácia, matriz de confusão para quantidade de falsos negativos e positivos, além das curvas ROC e Precisão-Sensibilidade para verificação de integridade e confiabilidade dos algoritmos. Ao fim, 3 tabelas foram extraídas das análises, onde as colunas contêm informações de parâmetros escolhidos para determinar as características que qualificam o resultado das aplicações das técnicas.

4.6.1 Métricas de Avaliação

O primeiro parâmetro utilizado como métrica é o número de variáveis selecionadas no processo de seleção, o qual demonstra o número de dimensões utilizadas no modelo. O valor máximo é 15, onde todas as informações são utilizadas na etapa de predição, ocasionando na maior

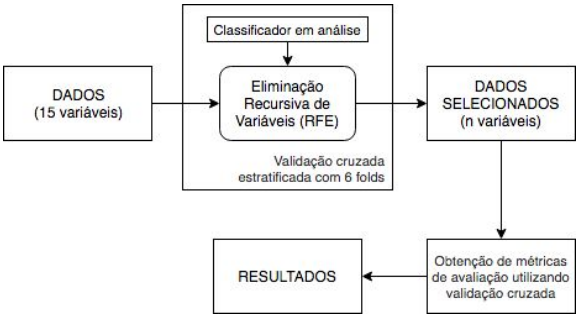

Figura 28 – Estratégia para obtenção de resultados através métricas para cada classificador. (Fonte: Elaboração Própria).

complexidade possível. Um número menor implica em um resultado mais interpretável, facilitando a visualização das inferências realizadas, onde pode-se demonstrar através de gráficos em até três dimensões sem utilização de métodos de análise de componentes principais. Essa quantidade foi selecionada utilizando o método de eliminação recursiva de variáveis independentes, onde, em conjunto com o método de validação cruzada, é aplicado para encontrar um ponto de máximo global na acurácia com todas as combinações possíveis. No exemplo demonstrado na figura 29, a acurácia é máxima quando são selecionadas 12 variáveis, as quais são mantidas no proceder da aplicação da técnica.

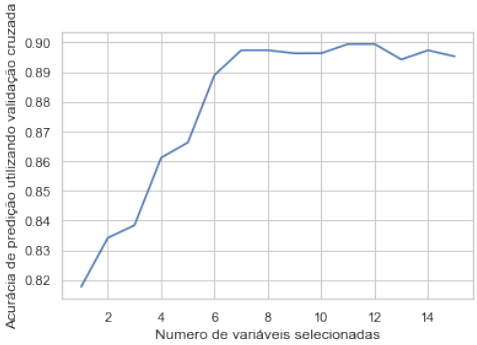

Figura 29 – Exemplo de acurácia de predição por número de variáveis. (Fonte: Elaboração própria)

Outro parâmetro é a acurácia de predição, que é resultado da

aplicação do método de validação cruzada com 6 folds, dividindo o conjunto de dados de formas distintas e obtendo a média das taxas de acerto. Esse valor é qualificado como informação chave no processo de análise das técnicas e abordagens a serem utilizadas, porém não fornece detalhes suficientes para avaliação do modelo quanto ao potencial de prever novas amostras, sequer se os acertos estão sendo influenciados por um possível desbalanceamento de classes.

Aplicando o método de matriz de confusão com validação cruzada de 6 *folds*, foi extraída a média de predições corretas e equivocadas, dando enfoque nas variáveis onde ocorreram erros na escolha de notificar, ou não, o pedestre. O primeiro parâmetro é o número de falsos positivos, onde é verificada a proporção de amostras que, na etapa de teste, foram qualificadas como situações de perigo, enquanto na verdade seriam seguras, sem necessidade de notificações. O segundo parâmetro extraído é o número de falsos negativos, os quais funcionam de forma similar, porém demonstram situações de perigo não notificadas ao pedestre. Um exemplo de matriz de confusão é demonstrado na Figura 30, onde existem 142 falsos positivos (que correspondem a 25% das amostras cujo rótulo esperado é verdadeiro, ou seja, a notificação se faz necessária, porém não é prevista) e 105 falsos negativos (que correspondem a 27% das amostras cujo rótulo esperado é falso e a notificação é considerada desnecessária, porém ocorre). Os falsos negativos são uma informação primordial na seleção do método, pois não notificar uma situação real de perigo invalida a ideia do sistema proposto.

Outras métricas são extraídas através da análise de Curva ROC, na qual se obtém informações sobre acurácia de predição utilizando as taxas de positivos verdadeiros TPR e de falsos positivos FPR para diversas configurações. Através desses valores, traça-se um gráfico e avalia-se sua área sob a curva (AUC), esperando sempre o máximo valor possível. O método de validação cruzada é utilizado novamente nesse caso, aplicando a curva em 6 *Folds*, extraindo a área e desvio padrão médios como métricas adicionais na tabela final de análise. Um exemplo é demonstrado na Figura 31, observando-se, no gráfico da esquerda, uma boa curva, em que todas as variações de conjunto de treinamento e testes culminam em valores semelhantes (desvio padrão médio baixo) e sua área sob a curva média tem o valor de 91%. Já no gráfico da direita desta mesma figura, observa-se uma curva mais próxima a linha vermelha tracejada, indicando que as predições podem estar sendo influenciadas por fatores como overfitting, não fornecendo uma boa representação dos dados de consciência situacional.

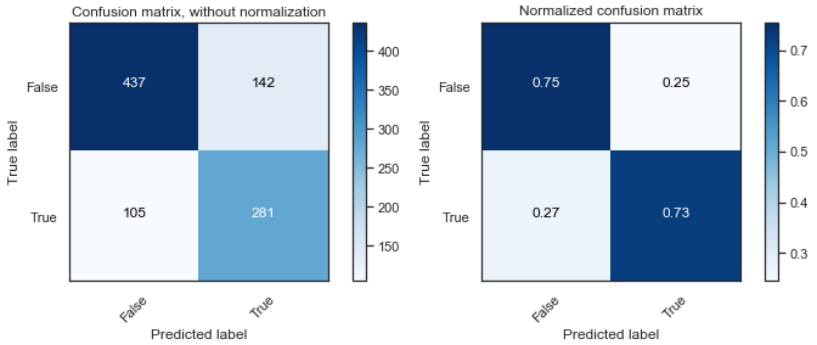

Figura 30 – Exemplos de matriz de confusão utilizando quantidades absolutas (a esquerda) e utilizando valores normalizados, de forma proporcional à quantidade de amostras (a direita). (Fonte: Elaboração própria)

A partir das informações de precisão e sensibilidade, é possível construir uma nova curva, similar à ROC, para representar a relevância dos elementos selecionados quanto ao potencial de classificação de notificações, funcionando de forma mais realística em classes desbalanceadas. A métrica extraída é a área sob a curva (AUC) média utilizando validação cruzada de 6 Folds e assim como na curva ROC, se esperam valores próximos a 100%. Conforme demonstrado no exemplo da Figura 32, ao lado direito, tem-se área sob a curva de 79,48%, o que não é o ideal. Já ao lado esquerdo a área sob a curva é de 92,52%, o que se aproxima do esperado de um modelo.

É necessário conseguir visualizar a maneira na qual as técnicas utilizadas em cada aplicação estão avaliando a necessidade de notificação no sistema de apoio à decisão. O método de análise de componentes principais (PCA) foi utilizado em cada técnica das aplicações para as variáveis específicas selecionadas com o objetivo de reduzir a quantidade de dimensões originais para duas, possibilitando a criação de um gráfico conforme exemplo da Figura 33, onde os pontos indicam amostras de teste e a cor do fundo indica as margens de decisão propostas por cada classificador. Nessa imagem, os quadrados e áreas azuis representam amostras em situações seguras, enquanto os triângulos e áreas laranjas representam situações onde a notificação é necessária. Apesar deste gráfico seguir os mesmos princípios das métricas anteriores, servem apenas como exemplos de visualização, pois informações podem ser perdidas na diminuição das dimensões, ocasionando num processo

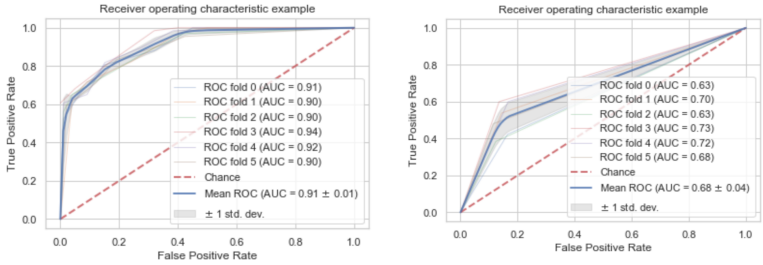

Figura 31 – Exemplos de gráficos de curvas ROC. (Fonte: Elaboração Própria)

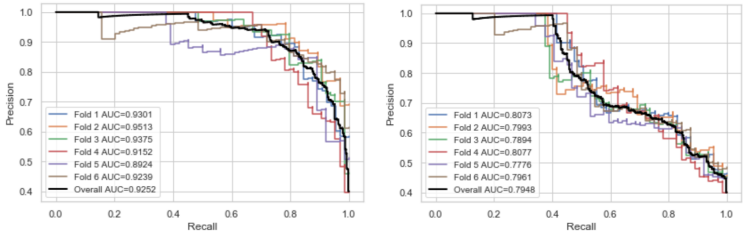

Figura 32 – Exemplos de gráficos de curvas precisão-sensibilidade (*Precision-Recall*). (Fonte: Elaboração Própria)

de treinamento diferenciado.

4.6.2 Aplicação das técnicas escolhidas e resultados

Nesta subseção serão demonstrados os resultados e destacadas as características analisadas que representam o grau de confiabilidade do modelo.

4.6.2.1 Primeira aplicação

Na tabela 4, apresenta-se a primeira abordagem (classes de consciência situacional) e observa-se que a maioria dos métodos acarretam quantidade proporcional de falsos negativos muito alta (cerca de 50%

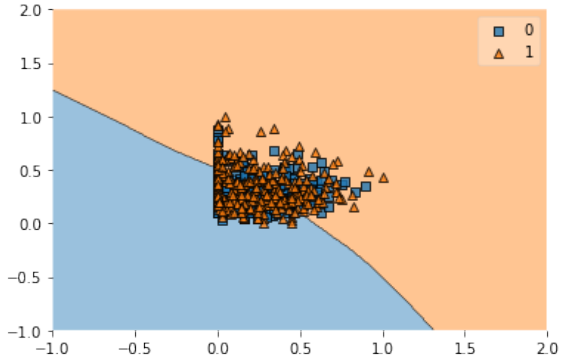

Figura 33 – Exemplo de visualização de regiões de classificação. (Fonte: Elaboração própria).

para as técnicas de *Bagging* e *Gradient Boosting*, e cerca de 60% nas restantes), o que significa que muitas das situações que necessitavam de notificações, acabaram não ocasionando em alertas ao pedestre, implicando em possíveis riscos à sua integridade. Com isso, pode-se perceber que pelo fato das classes estarem desbalanceadas, os valores de acurácia e área sob a curva ROC não demonstram o quão insatisfatório é o resultado das predições. Ao se avaliarem os dados de área sob a curva de precisão-sensibilidade, que qualificam melhor a confiabilidade em classes desbalanceadas, a hipótese é confirmada, onde o valor ideal 1 está longe de ser alcançado.

Classificador	Quantidade de variáveis selecionadas	Acurácia	Falsos Positivos (%)	Falsos Negativos (%)	Área sob a curva ROC	Desvio padrão curva ROC	Área sob a curva precisão-sensibilidade
Regressão logística	2	0,871621	0,172712	60,784314	0,766228	0,051427	0,590833
KNN	15	0,844229	4,317789	58,169935	0,764363	0,020336	0,636061
SVM	2	0,871621	0,172712	60,784314	0,777578	0,053669	0,61752
Árvore de decisão	13	0,766389	17,098446	50,980392	0,677493	0,043787	0,525814
AdaBoost	8	0,812907	8,117444	58,823529	0,765289	0,041788	0,501967
Bagging	8	0,845607	4,145078	58,823529	0,76491	0,05047	0,644221
Gradient Boosting	11	0,853804	3,454231	57,51634	0,792841	0,037511	0,640883
Florestas aleatórias	7	0,848384	2,763385	60,130719	0,740854	0,020925	0,596525
Perceptron Multicamadas	15	0,868888	0,172712	60,784314	0,813714	0,045654	0,641027

Tabela 4 – Aplicação de técnicas com abordagem categórica. (Fonte: Elaboração própria)

Analisando os gráficos gerados no processo de criação das regiões de classificação obtidas no treinamento de cada técnica (figura 34, verifica-se que essa aplicação não generaliza suficientemente as in-

formações fornecidas pelas amostras. Por conter classes desbalanceadas e utilizarem apenas variáveis binárias, técnicas como regressão logística e máquinas de suporte à vetor (SVM) ignoram situações onde existem poucos casos de perigo. As técnicas que utilizam *ensemble learning* e usam variáveis contínuas possuem uma versatilidade maior, aplicando diversas árvores de decisão, obtendo resultados mais favoráveis. O método de *perceptron* multicamada (MLP) utiliza todas as variáveis e contém ótimo desempenho, mas na visualização utilizando PCA informações são perdidas, implicando que o treinamento não convergiu para apenas 2 dimensões, portanto a visualização não corresponde à realidade do modelo proposto.

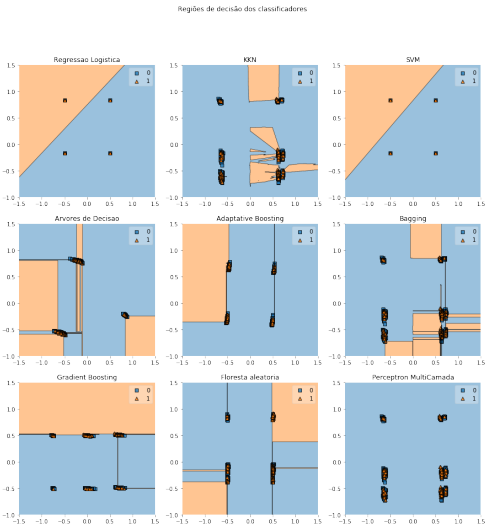

Figura 34 – Regiões de classificação para classificadores da primeira aplicação. (Fonte: Elaboração Própria)

4.6.2.2 Segunda aplicação

Na tabela 5 é demonstrada a segunda abordagem com o valor de *threshold* em 0.5. Observa-se que os resultados foram similares à primeira aplicação pela questão do desbalanceamento de classes. A técnica de Árvore de decisão obteve uma pequena melhora na diminuição de falsos negativos em relação a todos as técnicas aplicadas até

então, porém ainda é insatisfatório.

Classificador	Quantidade de variáveis selecionadas	Acurácia	Falsos Positivos (%)	Falsos Negativos (%)	Área sob a curva ROC	Desvio padrão curva ROC	Área sob a curva precisão-sensibilidade
Regressão logística	2	0,897525	0,167224	55,223881	0,786495	0,044548	0,616854
KNN	15	0,882486	2,006689	55,223881	0,782533	0,046055	0,662518
SVM	2	0,897525	0,167224	55,223881	0,794748	0,050261	0,644691
Árvore de decisão	8	0,808745	12,207358	44,029851	0,704202	0,014876	0,541396
AdaBoost	7	0,853841	4,347826	60,447761	0,767682	0,037736	0,492267
Bagging	14	0,879709	2,006689	53,731343	0,778348	0,023782	0,631694
Gradient Boosting	7	0,883852	1,337793	54,477612	0,789672	0,047045	0,643603
Florestas aleatórias	6	0,892049	2,508361	55,970149	0,772422	0,044905	0,63803
Perceptron Multicamadas	15	0,89617	0,167224	55,223881	0,813724	0,039327	0,643703

Tabela 5 – Abordagem 2 - *Threshold* 0.5 (Elaboração própria)

As regiões de classificação demonstradas na Figura 35 mostram que os padrões encontrados nas técnicas de *ensemble learning* demonstram explorar de forma mais versátil as informações extraídas. Essa busca é dificultada pelo fato do método utilizado para geração da visualização, o PCA, reduz muitas informações quando contém variáveis binárias. Novamente o *Perceptron* multicamadas não convergiu em duas dimensões.

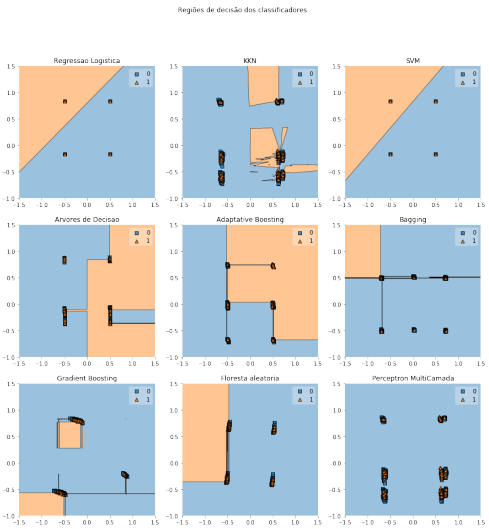

Figura 35 – Regiões de classificação para classificadores da segunda aplicação. (Fonte: Elaboração Própria)

4.6.2.3 Terceira aplicação

Na tabela 6 está demonstrada a segunda abordagem com o valor de *threshold* em 0.7. Observa-se que os resultados estão melhores que nas aplicações anteriores, pelo fato das classes estarem balanceadas. Os métodos tendem a selecionar mais variáveis para treinamento, obter maior acurácia e diminuição de falsos negativos para cerca de 25%. Informações obtidas através de área sob as curvas ROC e Precisão-Sensibilidade confirmam a efetividade das análises obtidas, obtendo padrões que ocasionam em resultados mais íntegros.

Classificador	Quantidade de variáveis selecionadas	Acurácia	Falsos Positivos (%)	Falsos Negativos (%)	Área sob a curva ROC	Desvio padrão curva ROC	Área sob a curva precisão-sensibilidade
Regressão logística	6	0,851142	5,263158	25,284091	0,907751	0,011941	0,922824
KNN	15	0,806013	13,157895	26,136364	0,884459	0,017252	0,905584
SVM	3	0,841579	1,052632	31,818182	0,865386	0,00813	0,88418
Árvore de decisão	7	0,777279	22,368421	22,159091	0,782913	0,023677	0,827318
AdaBoost	10	0,785476	20,263158	22,727273	0,864159	0,030671	0,858976
Bagging	11	0,800616	12,105263	26,420455	0,874743	0,010532	0,899597
Gradient Boosting	15	0,821053	10,526316	25	0,888681	0,018791	0,903351
Florestas aleatórias	7	0,804558	11,052632	27,272727	0,876238	0,021823	0,892408
Perceptron Multicamadas	15	0,841646	8,947368	25	0,902976	0,00621	0,921107

Tabela 6 – Abordagem 2 - *Threshold* 0.7 (Elaboração própria)

Nas visualizações geradas utilizando PCA na Figura 36, observa-se que os classificadores contêm mais informações sobre os padrões a serem encontrados para notificação. Cada método encontra uma maneira de aumentar sua acurácia de predição, e os melhores resultados são obtidos nas técnicas de KNN, Árvore de Decisão, *Gradient Boosting*. O *Perceptron* Multicamadas, que não havia conseguido convergir nas outras aplicações utilizando duas dimensões, generaliza grande parte dos dados obtidos, mas consegue obter uma classificação razoável nessa etapa.

4.6.3 Análise específica de classificadores

Dados os resultados da etapa anterior, esta subseção explora de forma mais profunda a interpretação de algumas aplicações, restringindo à terceira, onde a segunda abordagem com *threshold* 0.7 foi utilizada. A análise de componentes principais demonstrada após os resultados na subseção anterior não fornece precisão na visualização, por não especificar que informações estão sendo demonstradas.

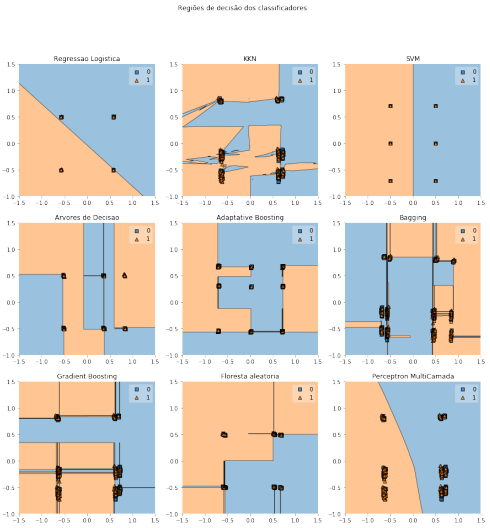

Figura 36 – Regiões de classificação para classificadores da terceira aplicação. (Fonte: Elaboração Própria)

A técnica de árvores de decisão é interessante pois permite a visualização facilitada da estratégia utilizada no processo de classificação. A Figura 37 mostra a árvore gerada no processo da terceira aplicação, após a seleção de variáveis, onde os nodos em azul representam situações de risco onde o pedestre deve ser notificado. O resultado demonstra que, pela quantidade de nodos utilizadas ser muito alta, as variáveis utilizadas acabaram acarretando *overfitting* no classificador, causado pelas constantes decisões acerca de variáveis contínuas.

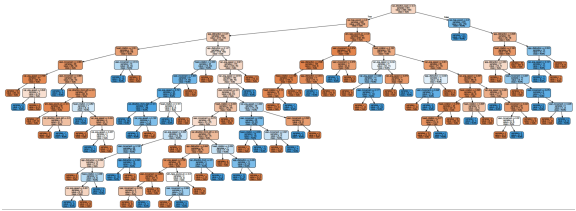

Figura 37 – Árvore de decisão na segunda abordagem. (Fonte: Elaboração própria)

Na Figura 38, o número de variáveis utilizadas na árvore de de-

cisão foi reduzido para 2, além de serem apenas dados binários. Foram utilizadas informações sobre o carro conter ou não som (car_has_sound) e se sua direção é proveniente da uma posição oclusal (is_car_ocluded), o que é suficiente para classificar de forma correta cerca 75% de todas as amostras do experimento, porém prevê apenas aproximadamente metade das notificações necessárias (50% de falsos negativos). Analisando a árvore, verifica-se que só são notificados os pedestres que foram submetidos a uma situação onde o carro não contém som e vem de uma posição oclusal.

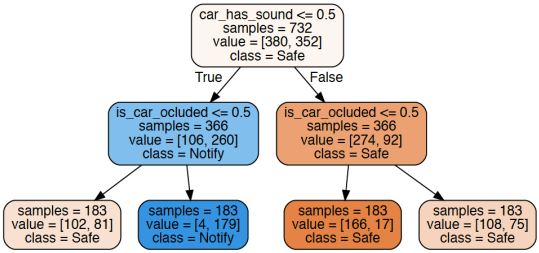

Figura 38 – Árvore de decisão com 2 variáveis. (Fonte: Elaboração própria)

Para melhorar o desempenho do classificador, foram incluídas na árvore as informações sobre o usuário (se ele estava ouvindo música enquanto jogava um jogo no smartphone) (sim_type_sound_on), e se a direção do carro é rotulada como "back" (car_direction_back). Como demonstrado na Figura 39, a árvore cresce apenas alguns nodos e aumenta sua acurácia para aproximadamente 85%, notificando 78% das situações de perigo (22% de falsos negativos).

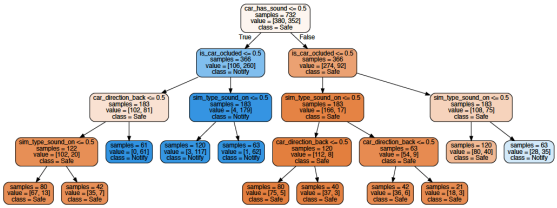

Figura 39 – Árvore de decisão com 4 variáveis binárias. (Fonte: Elaboração própria)

Utilizando variáveis contínuas, é possível visualizar as regiões de classificação de forma mais compreensível separando duas variáveis por vez. Dessa forma, todos os classificadores podem ser avaliados de acordo com as informações do contexto. Com isso, pode-se inferir relações para descoberta de conhecimento e testar sua visualização. Como este projeto visa verificar a relação entre a utilização de smartphones por pedestres e a incidência de acidentes em vias urbanas, analisar o potencial de classificação utilizando variáveis específicas pode levar a conclusões importantes.

No exemplo da Figura 40, são avaliadas duas técnicas para classificação de notificações utilizando o nível de distração no aplicativo por parte do pedestre, além da velocidade média do carro. A primeira é o *Perceptron* Multicamada, que obteve 71,44% de acertos, e 74% das situações de notificação. A segunda é a KNN, que acertou 77% das amostras, encontrando um padrão de vizinhança apenas com 2 dimensões.

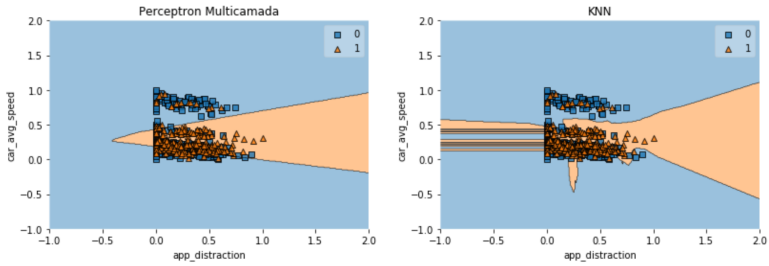

Figura 40 – Análise de regiões classificação utilizando nível de distração do aplicativo e velocidade média do carro para duas técnicas. (Fonte: Elaboração própria)

4.7 ANÁLISE DOS RESULTADOS

Dados os resultados da etapa anterior, observou-se que, por obter melhores métricas em mais variadas técnicas, e por conter um *threshold* para calibragem de notificações em uma aplicação futura em novos experimentos para verificação da eficácia no auxílio na tomada de decisões, a segunda abordagem com *threshold* 0.7 foi escolhida como a principal linha a ser seguida. Em geral, as técnicas demonstraram aspectos semelhantes nos resultados, embora as técnicas de Regressão

Carro tem som?	Carro em posição oclusal?	Velocidade média do carro	Direção do carro	Rotação da cabeça do pedestre	Deslocamento total do pedestre	Distração do aplicativo	Tipo de simulação	Notificar?
0	1	5.523	back_left	4.912	4.912	1.124	sound_off	Sim
1	0	1.322	front	1.900	2.031	150	button	Não
0	0	5.023	back	5.942	6.021	3.829	sound_on	Sim

Tabela 7 – Exemplos de predições do modelo. (Fonte: Elaboração própria)

Logística e *Perceptron* Multicamada forneceram maior confiabilidade e acurácia de predição. As técnicas de Árvore de decisão e *Adaptative Boosting* tiveram melhor desempenho ao avaliar a importante métrica de falsos negativos, mas a diferença para as restantes não foi significativa.

Trata-se de um problema onde os casos em que a notificação não é necessária são a maioria absoluta. Portanto, as classes tendem a ficar desbalanceadas, não importa quantos experimentos sejam realizados. Um aumento considerável na dificuldade poderia mitigar estas situações, porém isso não representa uma situação real, onde as pessoas geralmente não estão em constante perigo. Como as situações de perigo estão representadas em características muito específicas, uma predição para o valor de "não notificar" de forma equivocada pode acarretar problemas para a integridade física do pedestre. Portanto, devem ser priorizados os falsos positivos à falsos negativos.

Técnicas de *undersampling* e *oversampling* podem ser utilizadas em conjunto com a análise de componentes principais e redução de dimensionalidade. Sendo assim, amostras sintéticas podem ser criadas para balancear as classes de forma a não replicar os mesmos dados diversas vezes, aumentando a confiabilidade da classificação para novos dados.

4.8 EXEMPLOS DE APLICAÇÃO DOS MODELOS

Através de um modelo treinado utilizando a segunda abordagem com *threshold* 0.7 e o método de regressão logística com as 6 variáveis selecionadas, são testadas algumas amostras e documentadas na Tabela 7. A predição é demonstrada na coluna "Notificar?".

5 CONCLUSÃO

Ao longo desse trabalho de conclusão de curso foram estudadas e aplicadas abordagens, estratégias, técnicas e ferramentas auxiliares em um processo de análise de dados. A construção de modelos preditivos capazes de avaliar diversas informações sobre um ambiente urbano e decidir a necessidade de notificações para mitigar acidentes se mostrou possível, apesar de estar em estágio inicial. Para conseguir verificar o potencial de representação de situações de risco através de métodos de aprendizado estatístico, algumas hipóteses sobre o domínio do problema foram desenvolvidas e investigadas, através de modelos empíricos com os dados obtidos, mas o ambiente necessário para validações não é de fácil acesso. A quantidade e origem das amostras disponíveis extraídas dos experimentos não possibilita afirmações sobre padrões de comportamento do ser humano de forma geral.

Tendo em vista que o objetivo geral deste trabalho é analisar os dados do projeto *Awareness*, avaliando abordagens e técnicas em um modelo computacional para representar a necessidade de notificar um pedestre dependendo do seu nível de consciência situacional, o mesmo foi cumprido com êxito, realizando uma versão a ser testada de um modelo preditivo. Foi analisado o estado da arte em ciência de dados, feito um pré-processamento, modelagem estatística, aplicação de técnicas estatísticas e de aprendizado de máquina e, então, foram analisados os resultados, que devem ser aprimorados.

Com a evolução da tecnologia, envolvendo conceitos como *Big Data* e *Smart Cities*, as cidades tendem a fornecer cada vez mais informações sobre ambiente urbano, possibilitando um avanço no potencial do modelo em representar os eventos do contexto que se relacionam com o nível de consciência situacional de pedestres. Além disso, se a obtenção de mais dados de distração por meio de smartphones for possível, o modelo pode tornar-se progressivamente mais preciso no auxílio de tomada de decisões, uma vez que representará mais satisfatoriamente o nível de concentração dos usuários.

A contribuição principal deste trabalho se dá no estabelecimento de uma métrica inicial, que deverá ser aprimorada e calibrada, para o nível de consciência situacional de pedestres em vias urbanas. Tendo como base os princípios levantados e aspectos gerais que influenciam nas decisões de pessoas através das fases de percepção de elementos do ambiente, compreensão das informações e projeção do estado futuro, é possível modelar situações reais para contextos variados relacionados à

atenção. Além disso, as simulações realizadas no desenvolvimento permitem uma análise qualitativa para auxiliar na mudança de comportamento de maneira geral nos componentes do trânsito, como controle de tráfego, motoristas, carros autônomos e conscientizando pedestres a estarem cada vez mais atentos.

Este trabalho está inserido numa classe de trabalhos de ciência de dados aplicados e engloba diversos conceitos que são comuns a diversos outros contextos, como predição em classes desbalanceadas com variáveis contínuas e categóricas. Além disso, a aplicação é feita utilizando como problema as situações de riscos vividas por pedestres em ambientes urbanos, porém os estudos realizados podem ser adaptados para outras problemáticas, tendo em vista que o conceito de consciência situacional é utilizado para diversas áreas, como esporte e aeronáutica. As duas abordagens criadas têm potencial para representar de forma confiável a realidade da consciência situacional dos pedestres, o melhor caminho deverá se sobressair no evoluir da pesquisa.

5.1 TRABALHOS FUTUROS

Apesar deste trabalho abranger estratégias variadas, existem muitas outras abordagens a serem seguidas e avaliadas. Como o contexto é o comportamento humano, a complexidade dos modelos é extremamente alta e difícil de ser prevista. Além disso, é imprescindível em qualquer pesquisa não trivial em análise de dados, que sejam feitos diversos ciclos de verificações nos resultados. Em resumo, para possibilitar a melhoria na descoberta de conhecimento, uma quantidade maior de informações deve ser obtida e as hipóteses geradas devem ser constantemente testadas.

Sugere-se modelar um novo experimento a partir dos resultados desse trabalho, de forma que os pedestres se submetam a situações cada vez mais realísticas, com o intuito de obter dados mais confiáveis. Com o sistema proposto na subseção 4.3 e as tecnologias utilizadas no decorrer desse trabalho, é possível implementar, de forma relativamente simples, um serviço em tempo real responsável por receber amostras e prever possíveis padrões de situações perigosas. Dessa forma, além do smartphone servir como distração, pode ser utilizado como assistente em tomada de decisões numa aplicação retroalimentada que fornece notificações a serem julgadas pelo usuário (verificando se foi útil), possibilitando diferentes configurações para calibração do modelo preditivo. Esses ajustes podem ser feitos utilizando, entre outras variáveis, os va-

lores de n e de *threshold*, definidos no capítulo 4 para representação da consciência situacional. Além disso, pela diversidade imposta pelo contexto de comportamento humano, a utilização de lógica nebulosa (fuzzy) deve auxiliar nas incertezas que podem vir a surgir. Outras sugestões para a nova versão do experimento são:

- Quantidade maior de usuários
- Coletar dados sobre perfil do usuário
- Usuários de faixas etárias variadas
- Adição de sons típicos de ambientes urbanos, como conversas, buzinas e ambulâncias
- Alteração dinâmica de iluminação
- Alteração dinâmica na intensidade do tráfego (mais de um veículo por vez)
- Veículos com mais variação de velocidade e cores
- Rua mais larga, exigindo mais esforço na troca de pista
- Obtenção de mais variáveis sobre o cenário
- Obtenção de mais informações sobre o pedestre:
  - Dilatação da pupila
  - Ondas cerebrais
  - Movimentação corporal
  - Batimentos cardíacos
  - Pressão arterial
  - Tempo de resposta para percepções

Em relação às medidas extraídas e às métricas modeladas, a utilização de métodos que vão além da teoria clássica das medidas deve ser avaliada, isto é, utilizar características latentes através de modelagens estatísticas como a teoria da resposta ao item (TRI) para obter um nível de consciência situacional, tendo em vista que essa medida não é facilmente obtida, realizando uma discriminação efetiva acerca dos participantes do experimento.

Quanto à abordagem de nível contínuo de consciência situacional, onde foi elaborado um modelo empírico responsável por quantificar

a atenção dos usuários do experimento, melhorias nos componentes da fórmula devem ser feitas para as próximas análises. Quanto ao patamar estabelecido como tempo máximo de percepção para todos os pedestres, o *maxTFA* da equação 4.1, o valor fixo inserido não provê uma representação confiável para a análise de todos os participantes. Além do tempo máximo ser influenciado por fatores diversos nos quais os carros são submetidos, existe também o componente de conhecimento prévio do participante no experimento, ou seja, a experiência de cada usuário e ambientação deve ser levada em consideração (como citado no modelo de Ensley (1995)). Com isso, além de repetir o experimento com os mesmos usuários para observar essas alterações, uma variável aleatória pode ser utilizada no auxílio a atribuição do tempo de percepção máximo, usando distribuições probabilísticas no modelo empírico da consciência a ser prevista.

Na aplicação dos métodos de análise de componentes principais (PCA), a utilização de menos variáveis discretas deve ser avaliada, tendo em vista que certas características podem ter níveis intermediários, provendo uma possibilidade maior de conseguir encontrar padrões conclusivos a respeito de situações de risco aos pedestres.

Além disso, para o sistema se tornar contínuo e incremental, a criação de uma aplicação independente do ambiente de simulação utilizado pode fornecer um imenso potencial de crescimento ao sistema proposto. Isso pode ser feito através de um jogo portátil utilizando tecnologias como o *Google VR* e *Oculus Rift*, e implementando aprendizado por reforço ao modelo preditivo.

Pelo problema ser caracterizado como desbalanceado, independente da aquisição de novas amostras, outro trabalho futuro envolve a utilização de métodos avançados em balanceamento de classes, para representar os padrões de situações realmente perigosas através da classe minoritária.

REFERÊNCIAS

AKERKAR, R.; SAJJA, P. S. *Intelligent techniques for data science.* [S.l.]: Springer, 2016.

BOGOJESKA, J. *Statistical learning methods for bias-aware HIV therapy screening.* Tese (Doutorado) — Saarland University, 2012.

CAO, L. Data science: a comprehensive overview. *ACM Computing Surveys (CSUR)*, ACM, v. 50, n. 3, p. 43, 2017.

ENSLEY, M. Toward a theory of situation awareness in dynamic systems. *Human factors*, v. 37, p. 85–104, 1995.

GELAIM, T. A. et al. A hybrid intelligent agent for notification of users distracted by mobile phones in an urban environment. In: *ICAART*. [S.l.: s.n.], 2019.

HAND, D. J. Statistics and computing: the genesis of data science. *Statistics and Computing*, Springer, v. 25, n. 4, p. 705–711, 2015.

HEVNER, A. R. et al. Design science in information systems research. *Management Information Systems Quarterly*, v. 28, n. 1, p. 6, 2008.

IGUAL, L.; SEGUÍ, S. *Introduction to Data Science - A Python Approach to Concepts, Techniques and Applications.* Springer, 2017. (Undergraduate Topics in Computer Science). ISBN 978-3-319-50016-4. <https://doi.org/10.1007/978-3-319-50017-1>.

JAMES, G. et al. *An introduction to statistical learning.* [S.l.]: Springer, 2013.

KOTTHAUS, H. *Methods for efficient resource utilization in statistical machine learning algorithms.* Tese (Doutorado) — Technical University of Dortmund, Germany, 2018.

KUBAT, M. *An Introduction to Machine Learning.* [S.l.]: Springer, 2017.

KUHN, M.; JOHNSON, K. *Applied predictive modeling.* [S.l.]: Springer, 2013.

LHÉRITIER, A. *Nonparametric methods for learning and detecting multivariate statistical dissimilarity.* Tese (Doutorado) — Université Nice Sophia Antipolis, 2015.

LU, Y. *Statistical methods with application to machine learning and artificial intelligence*. Tese (Doutorado) — Georgia Institute of Technology, 2012.

MARCH, S. T.; SMITH, G. F. Design and natural science research on information technology. *Decision support systems*, Elsevier, v. 15, n. 4, p. 251–266, 1995.

MOHRI, M.; ROSTAMIZADEH, A.; TALWALKAR, A. Foundations of machine learning. adaptive computation and machine learning. *MIT Press*, v. 31, p. 32, 2012.

MURPHY, K. P. *Machine Learning: A Probabilistic Perspective*. [S.l.]: MIT Press, 2012.

NASAR, J.; HECHT, P.; WENER, R. Mobile telephones, distracted attention, and pedestrian safety. *Accident analysis & prevention*, Elsevier, v. 40, n. 1, p. 69–75, 2008.

NAUR, P. Concise survey of computer methods. Studentlitteratur, 1974.

NEWELL, A.; SIMON, H. A. Computer science as empirical inquiry: Symbols and search. *Commun. ACM*, v. 19, n. 3, p. 113–126, 1976. <https://doi.org/10.1145/360018.360022>.

PEFFERS, K. et al. A design science research methodology for information systems research. *Journal of management information systems*, Taylor & Francis, v. 24, n. 3, p. 45–77, 2007.

PEREIRA, R. L. et al. *Aplicação de aprendizagem por reforço para um modelo bayesiano de consciência situacional*. [S.l.]: Florianópolis, SC, 2017.

ROSS, S. M. *Introduction to probability and statistics for engineers and scientists*. [S.l.]: Elsevier, 2004.

RUSSELL, S. J.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. 2. ed. [S.l.]: Pearson Education, 2003. ISBN 0137903952.

SKIENA, S. S. *The Data Science Design Manual*. [S.l.]: Springer, 2017.

TREVOR, H.; ROBERT, T.; JH, F. *The elements of statistical learning: data mining, inference, and prediction*. [S.l.]: New York, NY: Springer, 2009.

6 ANEXOS

6.1 ANEXO A - CÓDIGOS-FONTE DO PRÉ-PROCESSAMENTO, DESENVOLVIMENTO DAS TÉCNICAS E ANÁLISES

data.py: Pré-processamento

```
1
import json
3 import os
import re
5 import csv
import numpy as np
7
carsPaths = {"path1":{
9     "wayPoints":{
          "wp1":{ "x": 142.7,"y": 0,"z": -64.3},
11        "wp2":{ "x": 142.7,"y": 0,"z": -15.4},
          "wp3":{ "x": -61,"y": 0,"z": -15.4},
13        "wp4":{ "x": -149.5,"y": 0,"z": -15.4},
          "wp5":{ "x": -149.5,"y": 0,"z": -64.8}
15        },
      "direction": "Front"
17 },
"path2":{
19    "wayPoints":{
          "wp1":{ "x": -267.1,"y": 0,"z": 36.2},
21        "wp2":{ "x": -267.1,"y": 0,"z": -13.0},
          "wp3":{ "x": -61,"y": 0,"z": -13.0},
23        "wp4":{ "x": 26.5,"y": 0,"z": -13.0},
          "wp5":{ "x": 26.5,"y": 0,"z": 82}
25        },
      "direction": "Back"
27 },
"path3":{
29    "wayPoints":{
          "wp1":{ "x": -30.6,"y": 0,"z": 212.3},
31        "wp2":{ "x": -30.6,"y": 0,"z": -15.4},
          "wp3":{ "x": -61,"y": 0,"z": -15.4},
33        "wp4":{ "x": -149.5,"y": 0,"z": -15.4},
          "wp5":{ "x": -149.5,"y": 0,"z": -64.8}
35        },
      "direction": "FrontLeft"
37 },
"path4":{
39    "wayPoints":{
          "wp1":{ "x": -32.84,"y": 0,"z": -241},
41        "wp2":{ "x": -32.84,"y": 0,"z": -15.4},
          "wp3":{ "x": -61,"y": 0,"z": -15.4},
```

```
43          "wp4":{ "x": -148.2,"y": 0,"z": -15.4},
            "wp5":{ "x": -148.2,"y": 0,"z": -92}
45          },
        "direction": "FrontRight"
47  },
    "path5":{
49      "wayPoints":{
            "wp1":{ "x": -90.17,"y": 0,"z": 212.3},
51          "wp2":{ "x": -90.17,"y": 0,"z": -13.0},
            "wp3":{ "x": -61,"y": 0,"z": -13.0},
53          "wp4":{ "x": 26.5,"y": 0,"z": -13.0},
            "wp5":{ "x": 26.5,"y": 0,"z": 82}
55          },
        "direction": "BackLeft"
57  },
    "path6":{
59      "wayPoints":{
            "wp1":{ "x": -90.4,"y": 0,"z": -241},
61          "wp2":{ "x": -90.4,"y": 0,"z": -13.0},
            "wp3":{ "x": -61,"y": 0,"z": -13.0},
63          "wp4":{ "x": 26.5,"y": 0,"z": -13.0},
            "wp5":{ "x": 26.5,"y": 0,"z": 58}
65          },
        "direction": "BackRight"
67  }}

69  defaultSafeSide = {"Safe":"Front","Time":0,"x":0.1992155,"y"
        :0.4543561,"z":1.405861,"Rx":-0.2583797,"Ry":0.2267305,"
        Rz":-0.6287609}

71  criticalDistance = {
        "path1": 252.4,
73      "path2": 255.3,
        "path3": 258.1,
75      "path4": 253.76,
        "path5": 254.47,
77      "path6": 257.4,
    }
79
    def getAllPaths(path):
81      r = []
        for(dirpath, dirnames, filenames) in os.walk(path):
83          if(filenames):
                for filename in filenames:
85                  if filename.endswith('.json'):
                        r.append([dirpath, filename])
87      return r

89  def createUserPaths(paths):
        uids = set()
91      userPaths = {}
        for(dirpath, filename) in paths:
```

```
93          if('uid' in dirpath):
                uid = re.search('uid\w*', dirpath).group(0)
95              uids.add(uid)

97      for ID in uids:
            userPaths[ID] = []
99          for (dirpath, filename) in paths:
                if '/'+ID in dirpath:
101                 userPaths[ID].append((dirpath,filename))


103     return userPaths
105
    def fixSimulationPaths(simulationPaths):
107
        iterator = sorted(list(simulationPaths.keys()))
109
        deadIDs = []
111
        for ID in iterator:
113         if ID not in deadIDs:
                if str(int(ID)+1) in iterator:
115                 simulationPaths[ID].extend(simulationPaths[
                        str(int(ID)+1)])
                    del simulationPaths[str(int(ID)+1)]
117                 deadIDs.append(str(int(ID)+1))

119             if (str(int(ID)+2)) in iterator:
                    simulationPaths[ID].extend(simulationPaths[
                        str(int(ID)+2)])
121                 del simulationPaths[str(int(ID)+2)]
                    deadIDs.append(str(int(ID)+2))
123
        return simulationPaths
125


127
    def createSimulationPaths(userPaths):
129
        simulationIDs = set()
131
        simulationPaths = {}
133
        for (dirpath, filename) in userPaths:
135         simulationIDs.add(re.search('\d\d\d\d',filename).
                group(0))
            uid = re.search('uid\w*', dirpath).group(0)
137
        for ID in simulationIDs:
139         simulationPaths[ID] = []
            for (dirpath, filename) in userPaths:
141
```

```
143             if ID in filename:
                    simulationPaths[ID].append((dirpath,filename
                        ))
145
147
    simulationPaths = fixSimulationPaths(simulationPaths)
149
    return simulationPaths
151
153 def createObjectPaths(simulationPaths):
155     objectPaths = {}
157     types = [ 'App', 'Scene']
159     for tag in types:
            for (dirpath, filename) in simulationPaths:
161             if tag in filename:
                    objectPaths[tag] = dirpath+'/'+filename
163
    return objectPaths
165
167 def createExperimentPaths():
169     paths = getAllPaths(os.getcwd() )
171     paths = [list(tuples) for tuples in paths]
173     # userPaths = {'uid':[(dirpath,filename)...]...}
    userPaths = createUserPaths(paths)
175     # simulationPaths = {'uid': {'simulation': [(dirpath,
        filename)...]...}... }
    simulationPaths = {}
177
    for user in userPaths:
179         simulationPaths[user] = createSimulationPaths(
                userPaths[user])
181     objectPaths = {}
183     for user in simulationPaths:
            objectPaths[user] = {}
185         for simulation in simulationPaths[user]:
                objectPaths[user][simulation] =
                    createObjectPaths(simulationPaths[user][
                    simulation])
187
```

```
189     return objectPaths

191 def fixSceneTime(scene, firstAware):
        rightTime = scene['AwareCars'][0]['Time'] - firstAware
193     for htd in scene['HeadTracking']:
          if htd:
195           htd['Time'] = htd['Time'] - rightTime
        for ss in scene['SafeSide']:
197       if ss:
              ss['Time'] = ss['Time'] - rightTime
199     for ac in scene['AwareCars']:
          if ac:
201           ac['Time'] = ac['Time'] - rightTime
        for car in scene['Cars'].values():
203         car['timeAdded'] = car['timeAdded'] - rightTime
            car['timeRemoved'] = car['timeRemoved'] - rightTime
205         car['timeCritical'] = car['timeCritical'] -
                rightTime

207     return scene

209 def fixAppTime(app, firstAware):
        # Filter button app case
211     if 'Death' in app.keys():
            deaths = app['Death']
213         for deathID, death in deaths.items():
                death['time'] = (firstAware['Time'] - app['
                    CarAware'][0]) + death['time']
215     return app

217 def getCarsOrdered():
        return ["Car0", "Car1", "Car2", "Car3", "Car4", "Car5",
            "Car6", "Car7", "Car8", "Car9",
219             "Car10", "Car11"]

221 def getAVGSpeed(timeAdded, timeCritical, path):
        return 3.6 * (criticalDistance[path]/(timeCritical-
            timeAdded))
223

225 def createSimulationDataObject(simulationPaths):


227
        dirpath = simulationPaths.items()[0][1]['App']
229     user = re.search('uid\w*', dirpath).group(0)


231
        simulations = []
233     myCars = []
        app = ""
235     for simulation in simulationPaths:
            scene = json.load(open(simulationPaths[simulation]['
```

```
            Scene']))
237
        if 'App' in simulationPaths[simulation]:
239         app = json.load(open(simulationPaths[simulation
                ]['App']))
            app = fixAppTime(app, scene['AwareCars'][0])
241
        #scene = fixSceneTime(scene, app['CarAware'][0])
243
        simID = simulation
245     if app:
            simType = app['SimType']
247     else:
            simType = "SoundOff"
249     if simType != "Button":
            if app:
251             appDeaths = len(app['Death'])
                points = max(app['Death'].values(),key=
                    lambda item:item['points'])['points'];
253
        head_tracking = scene['HeadTracking']
255

257     lastCriticalTime = 0
        lastPoints = 0
259     for name in getCarsOrdered():
            car = scene['Cars'][name]
261         aCar = []
            aCar.append(user)
263         aCar.append(name)
            #aCar.append(car['name'])
265         aCar.append(car['timeAdded'])
            aCar.append(car['timeRemoved'])
267         aCar.append(car['sound'])
            aCar.append(getAVGSpeed(car['timeAdded'], car['
                timeCritical'], car['route']))
269         aCar.append(car['route'] in ('path3', 'path4', '
                path5')) # is ocluded?

271         timeForAware = '-'
            for awareCar in scene['AwareCars']:
273             if awareCar:
                    if awareCar['name'] == car['name']:
275                     timeForAware = awareCar['Time'] -
                            car['timeAdded']
                        break
277
            aCar.append(timeForAware)
279         direction = carsPaths[car['route']]['direction']
            aCar.append(direction)
281         timeCritical = car['timeCritical']
            aCar.append(timeCritical)
```

```
283             aCar.append(timeCritical-car['timeAdded'])

285

                laneChangesSmaller = []
287             laneChangesBigger = []
                safeWithDefault = []
289             safeWithDefault.append(defaultSafeSide)
                safeWithDefault.extend(scene['SafeSide'])
291             #a = np.argmax(x['Time'] for x in
                    safeWithDefault if x['Time'] < car['
                    timeCritical'])


293

                for side in safeWithDefault:
295                 if side:
                        if side['Time'] < car['timeCritical']:
297                         laneChangesSmaller.append(side)
                        else:
299                         laneChangesBigger.append(side)
                laneChangesSmaller = sorted(laneChangesSmaller,
                    key=lambda k: k['Time'])
301             laneChangesBigger = sorted(laneChangesBigger,
                    key=lambda k: k['Time'])


303



305             if laneChangesSmaller:
                    aCar.append(laneChangesSmaller[-1]['Time'])
307                 aCar.append(laneChangesSmaller[-1]['Safe'])
                if laneChangesBigger:
309                 aCar.append(laneChangesBigger[0]['Time'])
                    aCar.append(laneChangesBigger[0]['Safe'])
311             else:
                    aCar.append('-')
313                 aCar.append('-')


315

                runOver = False
317             if laneChangesSmaller:
                    if direction in ('Front', 'FrontLeft', '
                        FrontRight') and laneChangesSmaller[-1][
                        'Safe'] == 'Back':
319                     runOver = True
                    elif direction in ('Back', 'BackLeft', '
                        BackRight') and laneChangesSmaller[-1]['
                        Safe'] == 'Front':
321                     runOver = True

323             aCar.append(runOver)
                aCar.append(simType)
325

                points = "-"
```

```
327             numDeaths = "-"
                maxObstacle = "-"
329             # App Data
                if 'Death' in app.keys():
331                 points = 0
                    numDeaths = 0
333                 maxObstacle = 1
                    deaths = app['Death']
335                 for deathID, death in deaths.items():
                        total_points = death['points']
337                     deathTime = death['time']

339                     if (deathTime > lastCriticalTime) and (
                            deathTime <= timeCritical):
                            numDeaths += 1
341
                        if (total_points - lastPoints) >
                            points:
343                         points = total_points -
                                lastPoints

345                     obstacleNumber = death['
                            obstacleNumber']
                        if obstacleNumber > maxObstacle:
347                         maxObstacle = obstacleNumber


349
                ''' Head Tracking variables '''
351             total_movement = 0
                total_rotation = 0
353             car_head_tracking = []
                # [:-1] porque o ultimo = {} (vazio)
355             for track in head_tracking[:-1]:
                    ht_time = track['Time']
357                 if (ht_time >= car['timeAdded']) and (
                        ht_time <= car['timeRemoved']):
                        car_head_tracking.append(track)
359
                for i, row in enumerate(car_head_tracking):
361                 if i > 0:
                        # User Moviment
363                     total_movement += abs(car_head_tracking[
                            i]['x'] - car_head_tracking[i - 1]['
                            x'])
                        total_movement += abs(car_head_tracking[
                            i]['y'] - car_head_tracking[i - 1]['
                            y'])
365                     total_movement += abs(car_head_tracking[
                            i]['z'] - car_head_tracking[i - 1]['
                            z'])

367                     # Head Rotation
```

```
                        total_rotation += abs(car_head_tracking[
                            i]['Rx'] - car_head_tracking[i - 1][
                            'Rx'])
                        total_rotation += abs(car_head_tracking[
                            i]['Ry'] - car_head_tracking[i - 1][
                            'Ry'])
                        total_rotation += abs(car_head_tracking[
                            i]['Rz'] - car_head_tracking[i - 1][
                            'Rz'])

                aCar.append(points)
                aCar.append(numDeaths)
                aCar.append(maxObstacle)
                aCar.append(timeCritical - lastCriticalTime)
                aCar.append(total_movement)
                aCar.append(total_rotation)

                myCars.append(aCar)


    return myCars


def createHeadTrackingPlot(simulationPaths):

    for simulation in simulationPaths:
        scene = json.load(open(simulationPaths[simulation]['
            Scene']))


objects = createExperimentPaths()




toCSV = []
toCSV.append(["User", "Car", "Added", "Removed", "Sound", "
    Critical AVG Speed" , "Is Ocluded", "Time for Aware",
    "Direction", "Time Critical", "Time Critical from Added"
        , "Moved to Current Lane", "Safe Lane", "Moved to
        next Lane",
    "Next Lane", "Run Over", "Simulation Type", "Points", "
        Deaths", "Max Obstacle", "Time between Critical
        times", "User Movement", "Head Rotation"])

for user in objects:
    toCSV.extend(createSimulationDataObject(objects[user]))



with open( os.getcwd() +'/'+ 'experimentDataParser.csv', 'w'
    ) as f:
```

```
407     writer = csv.writer(f, delimiter=';')
        writer.writerows(toCSV)
```

approaches_creation.py: Criação das abordagens

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(style="whitegrid")
cmap = sns.diverging_palette(220, 10, as_cmap=True)
get_ipython().magic(u'matplotlib inline')


df = pd.read_csv('data_with_aware.csv', sep=';')

df.Points = pd.to_numeric(df.Points, errors="coerce")
df.Deaths = pd.to_numeric(df.Deaths, errors="coerce")
df["Max Obstacle"] = pd.to_numeric(df["Max Obstacle"],
    errors="coerce")

df.Points = df.Points.fillna(0)
df.Deaths = df.Deaths.fillna(0)
df["Max Obstacle"] = df["Max Obstacle"].fillna(0)

df = df.assign(
    app_distraction = lambda x: 6 * df.Deaths + df.Points +
        2 * df["Max Obstacle"]
)
app_distraction = df.app_distraction
df = df.assign(
    app_distraction = lambda x: (app_distraction-
        app_distraction.min())/(app_distraction.max()-
        app_distraction.min())
)

speed = df['Critical AVG Speed']
df = df.assign(
    car_avg_speed = lambda x: (speed-speed.min())/(speed.max
        ()-speed.min())
)
plt.figure(figsize=[12,12])
df[['car_avg_speed']].hist(figsize=[12,6])


cols = ['Sound', 'Is Ocluded', 'Direction', 'Safe Lane', '
    Max Obstacle',
        'Simulation Type', 'Is Aware', 'Time for Aware','Run
            Over']
df_to_plot = df[cols]
```

```
df_to_plot['Time for Aware'][df_to_plot['Time for Aware'] ==
    '-'] = df_to_plot['Time for Aware'].max()
df_to_plot[df_to_plot['Time for Aware'] == '-']
df_to_plot['Time for Aware'] = df_to_plot['Time for Aware'].
    astype('float64')
df_to_plot = pd.concat([df_to_plot, pd.get_dummies(
    df_to_plot[['Direction']])], axis=1)
df_to_plot = df_to_plot.drop(['Direction'], axis=1)
df_to_plot = pd.concat([df_to_plot, pd.get_dummies(
    df_to_plot[['Safe Lane']])], axis=1)
df_to_plot = df_to_plot.drop(['Safe Lane'], axis=1)
df_to_plot = pd.concat([df_to_plot, pd.get_dummies(
    df_to_plot[['Simulation Type']])], axis=1)
df_to_plot = df_to_plot.drop(['Simulation Type'], axis=1)
df_to_plot


plt.figure(figsize=[10,10])

sns.heatmap(df_to_plot.fillna(0).corr(), cmap=cmap, vmax=1,
    center=0, vmin=-1,
            square=True, linewidths=.5, cbar_kws={"shrink":
                .5})


movement = df['User Movement']
rotation = df['Head Rotation']
df = df.assign(
    user_movement = lambda x: (movement-movement.min())/(
        movement.max()-movement.min()),
    head_rotation = lambda x: (rotation-rotation.min())/(
        rotation.max()-rotation.min())
)
plt.figure(figsize=[12,12])
df.user_movement.hist(figsize=[12,6])
plt.figure(figsize=[12,12])
df.head_rotation.hist(figsize=[12,6])


def get_level_is_aware(row):
    if row:
        return 1
    return 0

def get_level_safety(row):
    if row:
        return 0
    return 1


awareness_labels = {0: 'danger',
                    1: 'inattentive',
```

```
                    2: 'at_risk',
                    3: 'safe'}


def get_awareness_label(row):
    return awareness_labels[row]


df_1 = df.copy()
df_1 = df_1.assign(
    level_is_aware = df_1['Is Aware'].apply(
        get_level_is_aware),
    level_safety = df_1['Run Over'].apply(get_level_safety),
)
df_1 = df_1.assign(awareness_level = df_1.level_is_aware + 2
     * df_1.level_safety)
df_1 = df_1.assign(awareness = df_1.awareness_level.apply(
    get_awareness_label))

df_1.awareness.value_counts()

plt.figure(figsize=[12,6])

(   df_1.awareness.value_counts()[['danger','inattentive', '
    at_risk', 'safe']]
    .plot(kind='bar', title=u'Classifica  o do n vel de
        consc. situacional', cmap='Pastel2_r'
#           colors = ['orange','blue','green']
    ).set_ylabel(u'Quantidade de pedestres', fontsize='14')
)

plt.show()


plt.figure(figsize=[12,6])
plt.hist(
    df_1.awareness_level,
    stacked=False,
    label='Quantidade de eventos',
#     color = '#99DD66',
    bins=4)

plt.legend()
plt.title(u'Classes de consci ncia situacional')
plt.xlabel('')
plt.xlim(-0.1,3.1)
plt.ylim(0,700)
plt.ylabel(u'Quantidade de eventos')
plt.show()

y = np.array([len(df_1[df_1.awareness_level == 0]),
              len(df_1[df_1.awareness_level == 1]),
```

```
                len(df_1[df_1.awareness_level == 2]),
                len(df_1[df_1.awareness_level == 3])]
                )

x = ["danger","inattentive", "at_risk","safe"]

df_1.head()


df_1_dummies = pd.get_dummies(df_1)


df_1_dummies.corr().abs().Deaths.sort_values(ascending=False
    )


data_1 = pd.DataFrame()
data_1 = data_1.assign(car_has_sound = df_1.Sound)
data_1 = data_1.assign(is_car_ocluded = df_1['Is Ocluded'])
data_1 = data_1.assign(car_avg_speed = df_1.car_avg_speed)
data_1 = data_1.assign(app_distraction = df_1.
    app_distraction)
data_1 = data_1.assign(user_movement = df_1.user_movement)
data_1 = data_1.assign(head_rotation = df_1.head_rotation)


data_1 = data_1.assign(car_direction_back = df_1_dummies.
    Direction_Back)
data_1 = data_1.assign(car_direction_back_left =
    df_1_dummies.Direction_BackLeft)
data_1 = data_1.assign(car_direction_back_right =
    df_1_dummies.Direction_BackRight)
data_1 = data_1.assign(car_direction_front = df_1_dummies.
    Direction_Front)
data_1 = data_1.assign(car_direction_front_left =
    df_1_dummies.Direction_FrontLeft)
data_1 = data_1.assign(car_direction_front_right =
    df_1_dummies.Direction_FrontRight)

data_1 = data_1.assign(sim_type_button = df_1_dummies['
    Simulation Type_Button'])
data_1 = data_1.assign(sim_type_sound_off = df_1_dummies['
    Simulation Type_SoundOff'])
data_1 = data_1.assign(sim_type_sound_on = df_1_dummies['
    Simulation Type_SoundOn'])

data_1 = data_1.assign(awareness_level = df_1.
    awareness_level)
data_1 = data_1.assign(awareness = df_1.awareness)

data_1.head()

```

```
plt.figure(figsize=[12,12])
sns.heatmap(data_1.corr(), cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink":
                .5})



filename = 'data_1_awareness.csv'
data_1.to_csv(filename, encoding='utf-8', index=False)



df_2 = df.copy()
max_time_for_aware = pd.to_numeric(df_2[df_2['Time for Aware
    '] != "-"]['Time for Aware']).max()

def set_no_aware_to_max(row):
    if row == "-":
        return max_time_for_aware
    return pd.to_numeric(row, errors=max_time_for_aware)

df_2["Time for Aware"] = df_2["Time for Aware"].apply(
    set_no_aware_to_max)

n = 2

df_2 = df_2.assign(
    awareness = ((max_time_for_aware + 10) - df_2['Time for
        Aware']) * (n - df_2['Run Over'] * n + df_2['Run
        Over'])
)
df_2.awareness.describe()


df_2 = df_2.assign(
    awareness = ((max_time_for_aware) - df_2['Time for Aware
        ']) - df_2['Run Over'] * df_2['Time for Aware'].
        median()
)
df_2.awareness.describe()


awareness = df_2.awareness
df_2 = df_2.assign(
    awareness_normalized = lambda x: (awareness-awareness.
        min()))/(awareness.max()-awareness.min())
)
# df_2[['awareness_normalized']].hist(color='#99DD66')
plt.hist(df_2.awareness_normalized)
```

```
plt.legend()
plt.title(u'N vel de consci ncia situacional')
plt.xlabel('')
plt.ylabel(u'Quantidade de ocorr ncias')
plt.show()


# In[102]:
plt.plot([x for x in range(len(df_2))],df_2.
    awareness_normalized.sort_values())
# In[103]:
df_2.assign(notify_user=df_2.awareness_normalized<0.5).
    notify_user.value_counts()
# ### Correla  o entre vari veis
# In[99]:
list(df_2.columns)
# In[36]:
df_2.info()
# ### N vel de consci ncia situacional do pedestre em
    rela  o a condi  o de carro com som

sns.boxplot(x="awareness_normalized", y="Sound", data=df_2,
    orient='h')


# ### N vel de consci ncia situacional do pedestre em
    rela  o    oclus o do carro


sns.boxplot(x="awareness_normalized", y="Is Ocluded", data=
    df_2, orient='h')


# ### N vel de consci ncia situacional do pedestre em
    rela  o    dire  o do carro

sns.boxplot(x="awareness_normalized", y="Direction", data=
    df_2, orient='h')


# ### N vel de consci ncia situacional do pedestre em
    rela  o ao tipo de simula  o

sns.boxplot(x="awareness_normalized", y="Simulation Type",
    data=df_2, orient='h')

# ## App distraction

# #### Points

# In[41]:
```

```
sns.boxplot(x="awareness_normalized", y="Points", data=df_2,
      orient='h')


# #### Deaths

# In[42]:


sns.boxplot(x="awareness_normalized", y="Deaths", data=df_2,
      orient='h')


# #### Max Obstacle

# In[43]:


sns.boxplot(x="awareness_normalized", y="Max Obstacle", data
    =df_2, orient='h')


# #### App distraction variable

# In[44]:


# sns.boxplot(x="awareness_normalized", y="app_distraction",
      data=df_2, orient='h')
plt.scatter(df_2.awareness_normalized, df_2.app_distraction)


# #### Head Tracking variables

# In[45]:


plt.scatter(df_2.head_rotation, df_2.user_movement)


# In[46]:


import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_trisurf(df_2.head_rotation, df_2.user_movement, df_2
    .awareness_normalized)
```

```
302 # ## Correla  es

304 # In[47]:


306
# plt.scatter([x for x in range(len(df_2.awareness))], df_2.
    awareness)
308 pd.get_dummies(df_2).corr().awareness_normalized.abs().
    sort_values(ascending=False)[:30]


310
# In[48]:

312


314 df_2_dummies = pd.get_dummies(df_2)
df_2_dummies.corr().awareness.sort_values(ascending=False)
316

318 # #### Com valores absolutos em ordem decrescente

320 # In[49]:


322
df_2.corr(method="pearson").awareness.abs().sort_values(
    ascending=False)
324


326 # ## Cria  o de Dataframe final para an lise de
    Regress o

328 # In[50]:


330
# selected_variables = (['car_has_sound', 'is_car_ocluded',
    'car_avg_speed', 'app_distraction'])
332 data_2 = pd.DataFrame()
data_2 = data_2.assign(
334     car_has_sound = df_2.Sound,
    is_car_ocluded = df_2['Is Ocluded'],
336     car_avg_speed = df_2.car_avg_speed,
    app_distraction = df_2.app_distraction,
338     user_movement = df_2.user_movement,
    head_rotation = df_2.head_rotation,
340     car_direction_back = df_2_dummies.Direction_Back,
    car_direction_back_left = df_2_dummies.
        Direction_BackLeft,
342     car_direction_back_right = df_2_dummies.
        Direction_BackRight,
    car_direction_front = df_2_dummies.Direction_Front,
344     car_direction_front_left = df_2_dummies.
        Direction_FrontLeft,
    car_direction_front_right = df_2_dummies.
```

```
        Direction_FrontRight,
    sim_type_button = df_2_dummies['Simulation Type_Button'],
    sim_type_sound_off = df_2_dummies['Simulation Type_SoundOff'],
    sim_type_sound_on = df_2_dummies['Simulation Type_SoundOn'],
)

data_2 = data_2.assign(
    awareness = df_2.awareness_normalized
)

data_2.tail()

corr = data_2.corr()

plt.figure(figsize=[12,12])
sns.heatmap(corr, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})

abs(corr.awareness).sort_values(ascending=False)

filename = 'data_2_awareness.csv'
data_2.to_csv(filename, encoding='utf-8', index=False)
```

results.py: Aplicação das técnicas e extração de resultados

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import util

from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import AdaBoostClassifier, BaggingClassifier
from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier
from sklearn.neural_network import MLPClassifier

from sklearn.feature_selection import RFECV
from sklearn.model_selection import StratifiedKFold, cross_val_score, cross_val_predict

from sklearn.decomposition import PCA
```

```
pca = PCA(n_components=2)

def plot_2d_space(X, y, label='Classes'):
    colors = ['#1F77B4', '#FF7F0E']
    markers = ['.', '.']
    fig = plt.figure(figsize=[12,6])
    for l, c, m in zip(np.unique(y), colors, markers):
        plt.scatter(
            X[y==l, 0],
            X[y==l, 1],
            c=c, label=l, marker=m, alpha=0.5
        )
    plt.title(label)
    plt.legend(loc='upper right')
    fig.show()

class RandomForestClassifierWithCoef(RandomForestClassifier):
    def fit(self, *args, **kwargs):
        super(RandomForestClassifierWithCoef, self).fit(*args, **kwargs)
        self.coef_ = self.feature_importances_

class BaggingClassifierWithCoef(BaggingClassifier):
    def fit(self, *args, **kwargs):
        super(BaggingClassifierWithCoef, self).fit(*args, **kwargs)
        self.coef_ = np.mean([tree.feature_importances_ for tree in self.estimators_], axis=0)

from mlxtend.plotting import plot_decision_regions
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split

def plot_boundary(clf, X, y, ax=None, image_title=u'Fronteiras de Classifica  o de notifica  es ao pedestre'):
#     X_transformed = pca.transform(X)
#     plt.figure(figsize=[6,4])
    X_transformed = np.array(X.astype('float64'))
    y_transformed = np.array(y.astype('int64'))
    X_train, X_test, y_train, y_test = train_test_split(X_transformed, y_transformed)
    clf.fit(X_train, y_train)
    print accuracy_score(clf.predict(X_test), y_test)
#     scatter_kwargs = {'s': 60, 'edgecolor': None, 'alpha': 0.6}
    plot_decision_regions(X_transformed, y_transformed, clf=clf, ax=ax,
                          filler_feature_values={2: -5},
                          filler_feature_ranges={2: 0.75})
```

```
#     plt.title(image_title)
#     plt.show()


def get_analysis_report(X,y):
    classifiers = ([LogisticRegression(),
                    KNeighborsClassifier(),
                    SVC(kernel="linear", probability=True),
                    DecisionTreeClassifier(),
                    AdaBoostClassifier(),
                    BaggingClassifierWithCoef(),
                    GradientBoostingClassifier(max_features=1.0),
                    RandomForestClassifier(),
                    MLPClassifier()
                   ])
#     classifiers = ([LogisticRegression(),
    DecisionTreeClassifier(), LogisticRegression(),
#                     DecisionTreeClassifier(),
    LogisticRegression(), DecisionTreeClassifier(),
#                     LogisticRegression(),
    DecisionTreeClassifier(), LogisticRegression()])

    classifiers_names = ([
                    'Regressao Logistica',
                    'KNN',
                    'SVM',
                    'Arvores de Decisao',
                    'Adaptative Boosting',
                    'Bagging',
                    'Gradient Boosting',
                    'Floresta aleatoria',
                    'Perceptron MultiCamada'
                ])
    n_all_features = len(list(X.columns))
    n_features = []
    scores = []
    false_positives = []
    false_negatives = []
    roc_aucs = []
    roc_stds = []
    precision_recall_auc = []
    pca_transform = []
    for i, clf in enumerate(classifiers):
        clf_name = classifiers_names[i]
        print "Classifier: {}".format(clf_name)
        all_params = X.columns
        try:
            rfecv = RFECV(estimator=clf, step=1, cv=6,
                          scoring='accuracy')
            rfecv.fit(X, y)
            X_transformed = rfecv.transform(X)
            selected_params = all_params[rfecv.get_support()
```

```
                ]
109             n_features.append(rfecv.n_features_)
#               print pd.DataFrame(data=zip(list(X.columns),
    rfecv.grid_scores_), columns=["Variable","Scores"])
111         except:
                print "Feature elimination is not possible on
                    this method."
113             X_transformed = X
                selected_params = all_params
115             n_features.append(n_all_features)

117         score = cross_val_score(clf, X_transformed, y, cv=
                StratifiedKFold(6)).mean()
            y_pred = cross_val_predict(clf, X_transformed, y, cv
                =StratifiedKFold(6))
119         cfm = util.show_confusion_matrix(y, y_pred,
                show_images=False)
            a_roc_auc, a_roc_std = util.make_roc_curve(clf,
                X_transformed, y,show_images=False)
121         pr_auc = util.make_precision_recall_curve(clf,
                X_transformed, y,show_images=False)
            scores.append(score)
123         false_positives.append((cfm[0][1] * 100.0) / (cfm
                [0][0] + cfm[0][1]))
            false_negatives.append((cfm[1][0] * 100.0) / (cfm
                [1][1] + cfm[1][0]))
125         roc_aucs.append(a_roc_auc)
            roc_stds.append(a_roc_std)
127         precision_recall_auc.append(pr_auc)
            print "Features selected: {}\n".format(
                selected_params)
129         pca_transform.append(pca.fit_transform(X[
                selected_params]))


131
    # Plotting decision regions
133 fig, axarr = plt.subplots(3, 3, figsize=(15,15), sharex=
        True, sharey=True)
#   values = [-4.0, -1.0, 1.0, 4.0]
135 width = 0.75
    for i, ax in enumerate(axarr.flat):
137     plot_boundary(classifiers[i], pca_transform[i], y,
            image_title=(classifiers_names[i].encode('utf-8'
            )), ax=ax)
        ax.set_title(classifiers_names[i])
139     ax.set_xlim(-1.5, 1.5)
        ax.set_ylim(-1.0, 1.5)
141 #       ax.set_xlabel('x')
#           ax.set_ylabel('Feature 2')
143 #       ax.set_title('Feature 3 = {}'.format(value))

145 # Adding axes annotations
```

```
    fig.suptitle(u'Regi es de decis o dos classificadores')
    plt.show()
#     for i, clf_name in enumerate(classifiers_names):
#         plot_boundary(classifiers[i], pca_transform[i], y, image_title=(clf_name.encode('utf-8')))


    df_classifiers = pd.DataFrame(data=classifiers_names, columns=['Classifier'])
    df_classifiers['n_features'] = pd.DataFrame(data=n_features)
    df_classifiers['scores'] = pd.DataFrame(data=scores)
    df_classifiers['false_positives'] = pd.DataFrame(data=false_positives)
    df_classifiers['false_negatives'] = pd.DataFrame(data=false_negatives)
    df_classifiers['roc_auc'] = pd.DataFrame(data=roc_aucs)
    df_classifiers['roc_std'] = pd.DataFrame(data=roc_stds)
    df_classifiers['precision_recall_auc'] = pd.DataFrame(data=precision_recall_auc)
    return df_classifiers


# Primeira abordagem

df_1 = pd.read_csv('data-ap-1-notification.csv')
X = df_1.iloc[:, :-3]
y = df_1.notify_user
result_1 = get_analysis_report(X,y)
print result_1

# Segunda abordagem
# Threshold 0.5

df_2 = pd.read_csv('data-ap-2-notification.csv')
X = df_2.iloc[:, :-2]
y = df_2.assign(notify_user=(df_2.awareness < 0.5)).notify_user
result_2 = get_analysis_report(X,y)
print result_2

# Segunda abordagem
# Threshold 0.7
X = df_2.iloc[:, :-2]
y = df_2.assign(notify_user=(df_2.awareness < 0.7)).notify_user
result_3 = get_analysis_report(X,y)
print result_3
```

6.2 ANEXO B - ARTIGO DO TCC

A Hybrid Intelligent Agent for Notification of Users Distracted by Mobile Phones in an Urban Environment

Thiago Â. Gelaim[1], Gabriel A. Langer[1], Elder R. Santos[1], Ricardo A. Silveira[1], John O'Hare[2], Paul Kendrick[2], Bruno M. Fazenda[2]

[1]Department of Informatics and Statistics, Federal University of Santa Catarina
Florianópolis – SC – Brazil

[2]Acoustics Research Centre, University of Salford
Salford M5 4WT, UK

t.gelaim@posgrad.com, gabriel.langer@grad.ufsc.br

{elder.santos, ricardo.silveira}@ufsc.br

{J.OHare, p.kendrick, b.m.fazenda}@salford.ac.uk

Abstract. *Mobile devices are now ubiquitous in daily life and the number of activities that can be performed using them is constantly growing. This implies increased attention being placed on the device and diverted away from events taking place in the surrounding environment. The impact of using a smartphone on pedestrians in the vicinity of urban traffic has been investigated in a multimodal, fully immersive, virtual reality environment. Based on experimental data collected, an agent to improve the attention of users in such situations has been developed. The proposed agent uses explicit, contextual data from experimental conditions to feed a Bayesian network and two kinds of black-box predictive models. The agent's decision process is aimed at notifying users when they become unaware of critical events in their surroundings.*

1. Introduction

Doing activities such as payment of bills, organization of the agenda and reading articles used to be performed in different environments can now be done on smartphones. These and other flexibilities provided by smartphones, make users pay more attention to their devices than to the surrounding environment.

This research aims to analyze the use of smartphones by pedestrians in urban environments. Our work presents contributions in two aspects: (i) using a CAVE environment, we analyze the impact of the use of mobile devices on pedestrian situational attention; (ii) from this analysis, we developed an agent with Bayesian and Predictive reasoning to act in support of pedestrian decision making.

This paper is organized as follows: Section 2 presents the related work. Section 3 presents the base experiment of Situational Awareness. Section 4 presents our decision support agent. Section 5 presents conclusions and further work.

2. Research Context

Studies concerning mobile devices in urban environments have considered both the pedestrian and drivers' perspectives. Considering the driver's perspective, (Choudhary and Ve-

laga, 2017) presents the state-of-the-art on distraction effects considering reaction time, caused by conversation and/or texting, when using a mobile phone.

The work of Jiang et al. (Jiang et al., 2018), presents the use of mobile devices in a crossing environment by pedestrians who are college students. To achieve this, the experiment is performed in a real-life, outdoor environment, while subjects are distracted by texting, listening to music and talking on the phone. Data is collected from videos and an eye tracker.

Lin and Huang (Lin and Huang, 2017) also evaluate the use of smartphones in the roadside environment. The experiments are performed in a 'semivirtual walking environment'. Distractions in the mobile phone are texting, news-reading, or a picturedragging task. In the environment, the participants had to respond with a designed hand gesture to roadside events. The data was collected from eye tracker and video.

Modeling situational awareness of pedestrians can also be viewed from the perspective of the car and pedestrians. Neogi et al. (Neogi et al., 2017) and Kooij et al.(Kooij et al., 2018) present an approach to predict the intention of pedestrians crossing the street based on contextual information.

It is clear, particulary from the works of Jiang et al. and Lin and Huang, that mobile devices are becoming an increasing problem in urban environments and solutions are needed. In the research presented here we are interested in distraction from the point of view of a pedestrian, when engaged in the use of a mobile device near traffic and how this might affect their attention and situational awareness.

In our experiment, we have developed a scene in a fully immersive virtual reality environment, where the subject is required to perform a particular task, such as crossing the road, or move away from a moving car. A virtual environment enables us to have more control over experimental variables and also acquire a larger amount of data. As a mobile phone distraction we have used a game requiring constant attention. We propose a model of situational awareness of the pedestrian based on the data obtained from their interaction with the smartphone together with environmental, contextual data. The model aims output is then used to alert pedestrians and redirect their attention back to the roadside environment. To the best of our knowledge, this is the first time such approach is considered in this context.

3. Situational Awareness

In this section, we present the study about the impact of pedestrians using smartphones in the urban environment. This study and the data collected are the foundation for the development of the agent presented in section 4.

There are many theoretical perspectives to approach situational awareness. The Three Level Model (Endsley, 1995) is composed of a chain of information processing: perception, comprehension and projection. The Perceptual Cycle Model (Smith and Hancock, 1995) consists of the interaction between the agent and the environment. The Theory of Activity (Bedny and Meister, 1999) uses the activities to transform a current situation to the desired situation. In this work, we followed the Three Level model by Endsley as the theoretical background for the design of our experiment.

3.1. Experimental Design

The experiment was designed using Octave, a fully immersive, multimodal (audio and visual) CAVE environment, where we could create a safe testing environment allowing control of experimental variables and robust data collection. The goal of the experiment is to analyze the behavior of smartphone users in a scenario representing urban traffic. The testing environment consists of two-way lanes in a residential area, where cars can travel in any direction along these lanes. Figure 2 shows the urban scenarion from above and the six possible travel directions for the cars. Participants have been instructed to stay attentive to traffic whilst standing in one designated area and; to 'move to the safe lane' if they need to avoid being hit by travelling vehicles. Figure 1 shows a participant in the scenario. The green square on the ground represents the lane of the street in which the participant is considered safe and the red square is the lane that the participant believes that the car will be passing. The square colors change according to the participant's position. During the distracted conditions, the participant is additionaly asked to interact with a smartphone game that demands continuous attention.

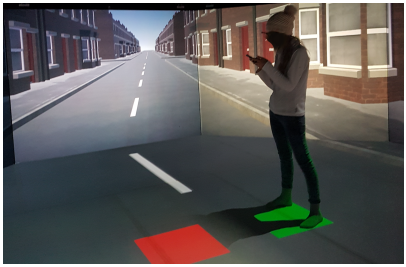

Figure 1. A participant in the environment.

Every simulation was composed of twelve cars, two cars for each of the six possible directions, one with sound and one without. This is so we can test car sound as a variable. Car instances are generated randomly from each possible direction with only one instance possible at any one time. Twenty participants took part, each performing the 'move to safety task' over three conditions with the following interaction on the smartphone:

- A "CAR" button to indicate awareness of an incoming car; no other distractions present (type 1).
- A "CAR" button to indicate awareness of an incoming car and; a game that required constant interaction (see 3) (type 2).
- A button to indicate awareness of an incoming car; a game that required constant interaction and; wearing headphones with music playing (type 3).

Before the main experiment, participants could practice on an identical scenario where five car instances were generated. This allowed subjects to get familiar with the test environment. The data collected consists of videos, audios, object positions, actions, and event elapsed time. The smartphone game used as a distraction is a variation of a game called color switch and is presented in figure 3. This game consists of a colored ball that has to be controlled to avoid hitting an obstacle that has a different color. It is a challenge that demands a high level of focus by the user.

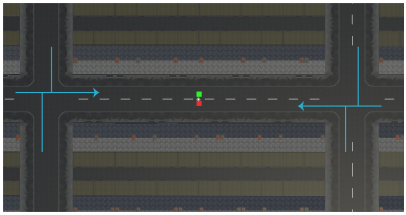

Figure 2. Superior view of the environment.

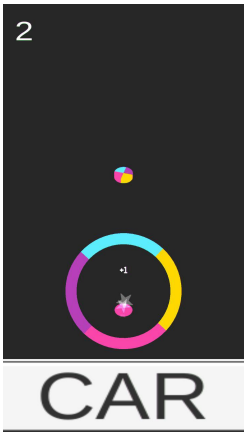

Figure 3. Game color switch, used for simulations 2 and 3.

3.1.1. Data Collection and preprocessing

As only one car exists in the scenario at any one time, we can analyze the events occurring between the addition and removal of each vehicle. Taking the car information into consideration, the analyses use independent events, without continuity, and users distinctions. For each vehicle, the information obtained from the environment are:

- Added: Execution time in which the car was added;
- Removed: Experiment execution time in which the car was removed;
- Sound: A binary variable that indicates if the car produces sounds;
- Is Occluded: A Binary variable that indicates if the car comes from an occluded position from the point of view of the participant (i.e.the side streets);
- Critical AVG Speed: Average speed computed by the car during the whole path, using the required time to reach the participant, and the traveled distance;
- Direction: Vehicle direction;
- Critical Time: Absolute execution time in which the vehicle and user occupied the same position in the scenario;
- Critical Time from Added: Variable added with the main goal of obtaining the time of the interval between the vehicle appearing in the scenario and the moment

that it occupies the same position of the user in the scenario;
- Safe Lane: Lane in the street where the user is safe;
- Simulation Type: Each of the three distraction conditions tested;
- SoundOff: using only the game as a distraction with no music on headphones;
- SoundOn: In addition to the distraction game, activates the music played in the headphones;
- Moved to Current Lane: Time in which the user crossed to the current position;
- Moved to Next Lane: Time in which the user crossed to other position;
- Time for Aware: Required time for the user to press the "CAR" button on the phone signalling awareness of a vehicle; measured from the inception time of the car into the scenario. Figure 3 presents this feature;
- Run Over: Indicates if a user did not change lanes in time and was run over by the car;
- Is Aware: Indicates if user pushed the "CAR" button in time. This state shows that the user was aware of the vehicle before it reached the critical time (see above). The information extracted from the app usage during type 2 or 3 conditions are:
- Points: The total amount of points earned by the user from playing the smartphone (distraction) game;
- Max Obstacle: Maximum level reached by the user during the period playing the smartphone (distraction) game.
- Deaths: The number of times that the user was run over during the game;

4. Decision Support Agent

In this section, we propose an agent in which its sensors capture the data from the environment and uses it to feed a Bayesian network and two kinds of black-box predictive models. Based on the output that is voted by these models, the agent decides if the pedestrian should be notified or not. Gathering information about the pedestrian, such as their behavior and perceptions, and contextual knowledge about the environment, danger can be inferred using previously obtained data. In figure 4 we present a general view of how our agent uses perceived information for pedestrian decision making. The models use artificial and statistical intelligence using different methods to find enough patterns to suggest an alert to the person. The three models together work as a black box, therefore, the agent receives the sensor parameters and indicates if it would be necessary to notify the user.

4.1. Bayesian Model

The Bayesian methods are used to reason about partial beliefs under the presence of uncertainty (Pearl, 1988)[pag.29]. The Bayes theorem states that the probability of a hypothesis h conditioned by some evidence e equals its probability P(ejh) multiplied by a priory probability for any evidence P(h) divided by the probability of evidence P(e) (Zelterman, 2005).

$$P(h|e) = \frac{P(e|h)\,P(h)}{P(e)} \tag{1}$$

Bayesian nets are used to simplify conditional, to planning decision on the situation in which uncertain is present, and to explain the results of a stochastic process

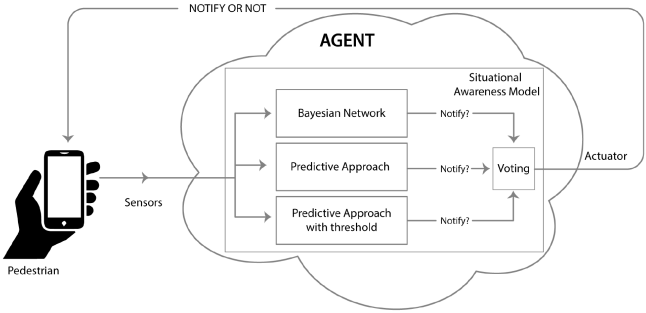

Figure 4. Decision Making on pedestrian notifications.

(Zelterman, 2005). In this work, Bayesian nets are adopted to model a situation of a pedestrian in an urban environment. To estimate the probability of a user being aware of a car and model it on a Bayesian Network, we use the collected data from the experiment presented in section 3. The proposed Naive Bayesian network is composed by the nodes aware, direction, sound, simulation type, is occluded, run over, and max obstacle. The structure was defined by the algorithm TAN. This algorithm is implemented on Netica, and is used to classify the 'aware' node based on the other nodes. Figure 5 shows the correlation between the variable used during the developing of the Bayesian net. The Naive Bayes model is presented in figure 6. To estimate the probability of the user being aware of the car, this net uses the data collected by the agent's sensors. When the node variable 'true' in the node 'aware' is less than 60% the Bayesian component sends to the voting component that the user must be notified.

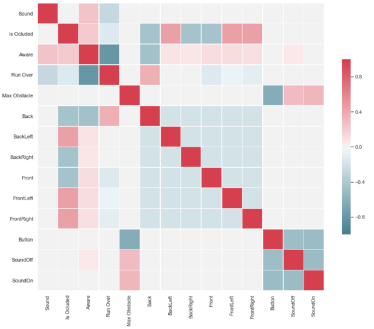

Figure 5. Correlation between variables - Naive Bayes.

To exemplify the working of our proposal, the following scenario consist of the following situation:

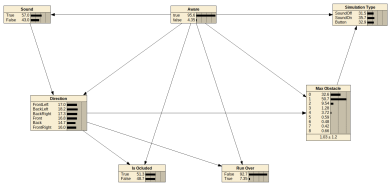

Figure 6. Bayesian Network built with Netica

The agent perceives that the user is listening to music on his smartphone and his interaction level with smartphone is 4 on a scale of 0 to 8. In the environment, there's a car approaching and, as the music is playing, it is unlikely the user heard the car sound. Figure 7 states that there's a 51.6% chance of the user not being aware of the car. In according with our proposal, this kind of situation requires an attention notification be sent to the user.

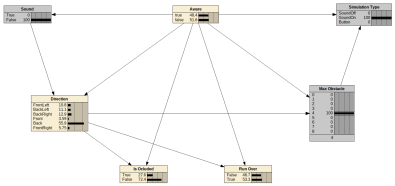

Figure 7: Example 1 Bayesian network

Figure 7. Example 1 Bayesian network

4.2. Predictive Model

With the objective of evaluating statistical learning techniques regarding the representative potential in situational awareness of pedestrians, the construction of a predictive model can be used to assign characteristics about the safety level of users submitted to plausible danger situations.

In order to perform satisfactorily the analysis of this study, it was necessary to create a scheme that would gather important information distributed in metrics of events that begin in the appearance of a vehicle in the scenario and finalize in its withdrawal. In this way the purpose of the model uses independent events, without continuity and without distinction between users. By dividing the data set obtained through the experiment in this way, we were able to obtain enough information to find patterns and classify if a pedestrian is at risk.

Using the information extracted from the experimental data, the objective is make the model able to obtaining the configuration of factors that interfere in the perception of the vehicles by pedestrians, applying methods of statistical learning. The strategy used was to verify the correlation between the variables and define approaches that lead to a dependent variable that represents the level of situational awareness. Once this variable is defined, the remaining work consists of evaluating how information available in urban

environments can aid in decision making, helping to reduce the amount of pedestrian accidents on urban roads. Some variables were selected on the pedestrian, such as their level of immersion in the application, variation of head rotation and movement in the environment as independent variables (features) of the model. Vehicle data were also used, such as average speed, direction and presence of sound. As a dependent variable (response), variables of car perception and run over were combined to classify the decision to notify the pedestrian about a possible risk of an accident in the urban road. Two approaches were used to create the response.

As the intention is to obtain the representation of risky situations, some information from all extracted to create a new metric called awareness, which serves as a reference to evaluate the correlations and provide a computational model appropriate to the project objectives, to be used in the future. Based on the concept that an application derived from this model can use the information provided in real time to detect risk situations and alert the pedestrian, the level of situational awareness can be represented through pre-defined levels, where behavioral patterns suggest the possible warnings to the user. Another form of representation is to create a quantitative and continuous level of the situational awareness level, which assesses whether the user is fully aware of the risks of their surroundings, where 0 (zero) represents the minimum possible attention, and the value 1 (one) represents a completely watchful pedestrian.

As it was designed, the experiment provides enough information to evaluate the pedestrian's perception of the components from scenario and to use the time it took to identify the presence of a vehicle coming toward him, to avoid a possible accident. This information comes from the variables obtained in the pre-processing.

4.2.1. Assignment of situational awareness categories

Through the data chosen to define a dependent variable for constructing the situational awareness computational model, the first strategy is to create categories that demonstrate the level of pedestrian safety based on information about their perception. Because the 'Is Aware' and 'Run Over' variables can provide an answer to the pedestrian risks, they have been selected to compose the classes. Both variables are binary, culminating in four different combinations to classify each sample of the experiment, giving greater importance to the variable that represents the run over occurrences, based on the premise that the user will prioritize avoiding an accident by moving lane, to pushing the "CAR" button to state that he is aware of the car. In this way the classes were defined as:

- danger: Category where the situation of the experiment tends to a circumstance where the user does not notice the car, nor does it avoid being run over;
- inattentive: Category where the user notices the car through the application, but does not avoid being run over;
- at risk: Category where the user does not indicate the perception of the car, but is not run over;
- safe: Category where the user notices the car and is not run over.

The quantitative distribution of events based on their classifications was given according to Figure 8, where the vast majority of users were in the safe state. Of the total of 732 data available, 53 were in danger, 12 were inattentive user situations, 88

were at risk and the other 579 were classified as safe situations. Using the data resulting from category creation, statistical learning techniques that work as classifiers can be used to create a model that represents each class according to the independent variables that are of the highest importance. This provides precision to enable possible notifications to aid in pedestrian decision making when the final model resulting from that work is implemented.

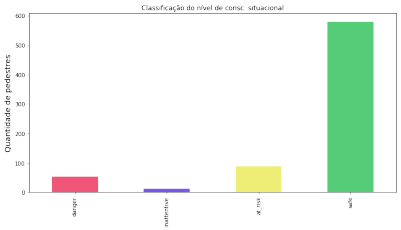

Figure 8. Distribution of pedestrian by situational awareness categories.

However, since the final objective of the project is to define whether the pedestrian and the driver should be warned about the risk of an accident, it is interesting that the classification is given in a binary way. Therefore, we can train the model based on whether a notification should be issued if the user's class is not safe, leading to a new boolean derived variable, named notify user. The distribution of pedestrians in each notification class is contained in Figure 9. In this way, the mentioned method can be used to create the final model. This approach does not take into account the time required for car aware perception by the user.

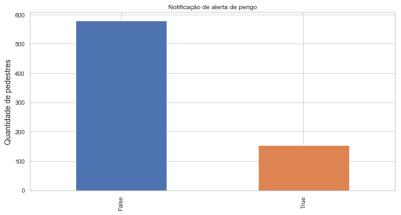

Figure 9. Distribution of pedestrians by notify user decision.

4.2.2. Assignment of situational awareness continuous level

The analysis performed through the first approach provides a modeling of the problem using discrete dependent variables on the negative or positive result of each vehicle event in the experiment. As the four classes were used to understand the behavior and separation between a situation where a notification with danger alert would be necessary to the user or not, the model tends to be more restricted, but does not mean worse performance. The second approach uses the information regarding the user's safety time, i.e. the user noticed

the car in time enough to avoid a dangerous situation? Based on the variables generated in the data preprocessing step, the Time for Aware variable (which indicates the time required for vehicle perception since its inception in the scenario) together with the Run Over information (which indicates a crash situation) can be used together to demonstrate a quantitative level of situational awareness of pedestrians in each situation. Since Time for Aware is a variable with continuous values and Run Over is a binary variable, a formula must be used taking into account a weight for each measure. Assuming that the time for a pedestrian to perceive the vehicle is inversely proportional to their level of attention, we can use this information to create a formula of the level of situational awareness of the user and using as a multiplicative factor of weight n as the data that the user was not run over in the experiment, as demonstrated in equation 2.

$$\begin{aligned} \texttt{awareness} = (\texttt{maxTFA} - \texttt{userTFA}) \\ * (n - \texttt{runOver} * n + \texttt{runOver}) \end{aligned} \tag{2}$$

- awareness: is the final situational awareness level of each user;
- maxTFA: represents the maximum time users took to perceive the car (constant);
- userTFA: represents the time the current user took to perceive the car;
- n: It's the security multiplicative factor;
- runOver: It is a binary value, where 1 indicates that the user was run over in the experiment;

Then, for each event in the data set, the formula is applied and its values are normalized, as are all independent quantitative variables that can be used later in the statistical learning model. The value used for the multiplicative factor n was 2, which represents a duplication in the assignment of the level of situational awareness of the pedestrian case to avoid trampling. The distribution of the normalized level of the awareness variable occurred according to Figure 10.

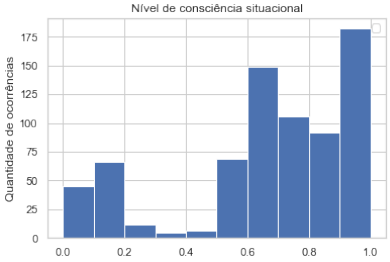

Figure 10. Pedestrian distribution by situational awareness continuous level.

4.2.3. Applying statistical learning techniques

Following the guidelines found in the theoretical basis, some learning methods were chosen according to the characteristics found in the data extracted from the experiment, cov-

Input	Bayesian Network	Predictive Model	Pred. Model w/ Threshold	Output
A	Not Notify(100%)	Not Notify (level 3)	Not Notify (0.7876)	Not notify
B	Not Notify(99%)	Not Notify (level 3)	Notify (0.4887)	Not notify
C	Notify(37.7%)	Notify (level 0)	Notify (0.0707)	Notify
D	Not Notify(96.2%)	Notify (level 2)	Not Notify (0.708979)	Not notify

Table 1. Awareness model output

ering techniques such as linear and neighborhood, as well as ensemble learning methods, which can improve prediction efficiency. The methods chosen for classification were:

- Logistic Regression;
- K Nearest Neighbors (KNN);
- Support Vector Machines (SVM);
- Decision Trees;
- Adaptive Boosting (AdaBoost);
- Bagging;
- Gradient Boosting;
- Random Forest.

For each method used, the cross-validation method was used (using Stratified K Folds, where each fold contains the same number of samples representing the classes), together with selection of variables for a more accurate classification. Then, accuracy metrics, confounding matrix for false negative and positive numbers were extracted, besides the ROC and precision-recall curves for verification of the integrity and reliability of the algorithms. The best-performing techniques were those of ensemble learning, especially Bagging and Random Forest, which were more accurate and lower false negative rate than others, which is a very important metric since the absence of a notification can cause damage to the pedestrian.

4.3. How these models influence pedestrian decision making

Let the sensors inputs be: A = car has sound = true, is car ocluded = true, car avg speed = 0.227491, app distraction = 0, user movement = 0.397433, head rotation = 0.225124, car direction = back left, sim type = button. B = car has sound = false, is car ocluded = false, car avg speed = 0.0564026, app distraction = 0.272997, user movement = 0.2551, head rotation = 0.2004, car direction = front, sim type = sound on. C = car has sound = false, is car ocluded = false, car avg speed = 0.0564026, app distraction = 0.3038, user movement = 0.4137, head rotation = 0.5741, car direction = back, sim type = sound on. D = car has sound = false, is car ocluded = false, car avg speed = 0.825519, app distraction = 0.35905, user movement = 0.177664, head rotation = 0.107857, car direction = back right, sim type = sound on. In table 1 we present a set of behavior of our agent according with these inputs.

The bayesian network will send a message to notify only in case C, because the variable 'true' in aware node is smaller than our bayesian threshold of 60%. The categorical predictive model uses the Bagging Classifier technique, which provides an aggregation of decision trees with random samples from the training dataset in a setting that notifies the pedestrian only in case C and D. The predictive model with threshold of 50% uses the Random Forests Classifier technique, in a way that less predictors are used to each split in

the aggregation of decision trees, providing reduced variance. Predicted the notification requirement on B and C inputs.

4.4. Exploratory Data Analysis (EDA)

Using continuous data for the dependent variable awareness obtained in approach number 2, we can analyze how much it is influenced by the presence of sound in the car through a BoxPlot type graph, as shown in Figure 11. The influence of sound on the level of situational awareness of a pedestrian is clear, the orange box indicates the vehicles that produce sounds, and the median is close to 90 % of the maximum level measured, besides having the lower and upper quartiles near this value. Analyzing the blue box, we can observe that pedestrians with lower level of situational awareness are more present.

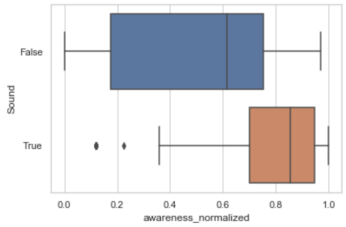

Figure 11. BoxPlot of info that car has sound and measured awareness.

A larger view of correlation between all variables is shown in Figure 12, where the reddest color indicates positive correlation, and more bluish negative. The data such as direction, safe lane, and simulation type were transformed into multiple columns for the reason of being categorical. One can not simply enumerate the values from these categories because they do not have an order of magnitude that sorts them.

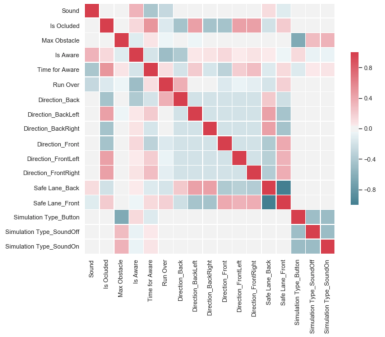

Figure 12. Correlation between variables

5. Conclusion

The use of mobile devices by pedestrians and drivers can increase the incidence of traffic accidents. In this research, we investigate the use of mobile devices by pedestrians and propose an agent to act as a notification system for critical distraction levels. The aims of this research were twofold: 1) to develop an understanding of the impact of mobile device usage on pedestrians' situational awareness and; 2) to develop an agent that can predict the level of awareness of a pedestrian who is using a mobile device in critical zones such as near roads. Using a Cave Automatic Virtual Environment (CAVE), an urban environment has been designed and calibrated to simulate the interaction between a pedestrian user of smartphone and moving traffic. Based on the data collected three models were developed and with its outputs, a voting system defines if the user must be notified. In the Bayesian model it is easier to add a new behavior through nodes and CPT, but in this version, it is more restrictive to send notifications. On both predictive models, the use of statistical learning methods gives a whole set of different tools to enable finding data patterns that indicate dangerous situations. Despite the satisfactory results, a larger quantity of samples is likely to have a positive influence on the knowledge discovery for pedestrian situational awareness.

5.1. Future Work

Our results with the Bayesian Network must be improved. An approach we may try is to extend it as a Dynamic Bayesian Network (DBN). This approach already exists for the driver's view of pedestrians (Kooij et al., 2018). We also are planning to add online learning to fit the participant profile. Another interesting project is to develop a smartphone application with this agent. At this time we only define that the user should be notified, not specifying such notification. Depending on the level of attention, or lack thereof, the agent may have a set of actions. For example, if the user's attention is very low, and the user is listening to music, a beep may be applied. Other possible actions may be to interrupt texting, showing a warning message or blocking the display altogether.

6. References

Bedny, G. and Meister, D. (1999). Theory of activity and situation awareness. International Journal of cognitive ergonomics, 3(1):63–72.

Choudhary, P. and Velaga, N. R. (2017). Modelling driver distraction effects due to mobile phone use on reaction time. Transportation Research Part C: Emerging Technologies, 77:351–365.

Endsley, M. R. (1995). Toward a theory of situation awareness in dynamic systems. Human Factors: The Journal of the Human Factors and Ergonomics Society, 37(1):32–64.

Jiang, K., Ling, F., Feng, Z., Ma, C., Kumfer, W., Shao, C., and Wang, K. (2018). Effects of mobile phone distraction on pedestrians' crossing behavior and visual attention allocation at a signalized intersection: An outdoor experimental study. Accident Analysis & Prevention, 115:170–177.

Kooij, J. F., Flohr, F., Pool, E. A., and Gavrila, D. M. (2018). Context-based path prediction for targets with switching dynamics. International Journal of Computer Vision, pages 1–24.

References

Lin, M.-I. B. and Huang, Y.-P. (2017). The impact of walking while using a smartphone on pedestrians' awareness of roadside events. Accident Analysis & Prevention, 101:87–96.

Neogi, S., Hoy, M., Chaoqun, W., and Dauwels, J. (2017). Context based pedestrian intention prediction using factored latent dynamic conditional random fields. In Computational Intelligence (SSCI), 2017 IEEE Symposium Series on, pages 1–8. IEEE.

Pearl, J. (1988). Probabilistic reasoning in intelligent systems: networks of plausible inference. Morgan Kaufmann. Smith, K. and Hancock, P. A. (1995). Situation awareness