Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Especialização em Informática: Área de Concentração Gestão em Tecnologia da Informação

Vítor Carneiro Curado

Fatores sociais e climáticos e a ocorrência de epidemias de dengue: Uma análise sob a perspectiva da mineração de dados

Brasília - DF
3 de maio de 2019

Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Especialização em Informática: Área de Concentração Gestão em Tecnologia da Informação

Vítor Carneiro Curado

Fatores sociais e climáticos e a ocorrência de epidemias de dengue: Uma análise sob a perspectiva da mineração de dados

Monografia apresentada como requisito parcial para conclusão do curso de pós-graduação *Lato Sensu* em Informática, área de concentração em Gestão de Tecnologia da Informação.

Orientador: Prof. Dr. Wagner Meira Júnior

Brasília - DF
3 de maio de 2019

Ficha catalográfica elaborada pela Biblioteca do ICEx - UFMG

Curado, Vitor Carneiro

C975f Fatores sociais e climáticos e a ocorrência de epidemias de dengue: uma análise sob a perspectiva da mineração de dados / Vitor Carneiro Curado – 2019.
85 f.

Monografia (especialização em informática) – Universidade Federal de Minas Gerais. Departamento de Ciência da Computação.

Orientador: Prof. Dr. Wagner Meira Júnior.

1. Computação. 2. Mineração de dados I. Orientador. II. Título.

CDU 519.6*

UNIVERSIDADE FEDERAL DE MINAS GERAIS

INSTITUTO DE CIÊNCIAS EXATAS
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
ESPECIALIZAÇÃO EM INFORMÁTICA: ÁREA DE CONCENTRAÇÃO GESTÃO EM TECNOLOGIA DA INFORMAÇÃO

Fatores Sociais e Climáticos e a ocorrência de epidemias de dengue

VITOR CARNEIRO CURADO

Monografia apresentada aos Senhores:

Prof. Wagner Meira Júnior
Orientador
DCC - ICEx – UFMG

Prof. José Nagib Cotrim Árabe
DCC - ICEx - UFMG

Prof. José Marcos Silva Nogueira
DCC - ICEx - UFMG

Belo Horizonte, 15 de março de 2019

À Deus, por sempre me guiar. À minha esposa, companheira de vida, Helena, por sempre me apoiar. Ao meu filho, Henrique, que nasceu ao longo dessa pós-graduação e que me mostrou o que é, verdadeiramente, amar.

Resumo

Entre as doenças virais transmitidas por mosquitos, a dengue é a que apresenta a situação epidemiológica mais alarmante do mundo(WHO, 2012). Diversos estudos já mostraram que a transmissão do vírus da dengue está intimamente relacionada ao clima(HALES et al., 2002; HU et al., 2012; LOWE et al., 2014). Entretanto, a Organização Mundial de Saúde recomenda que mecanismos de vigilância e controle de epidemias de dengue utilizem, além de dados climáticos, informações de caráter social e econômico para entender melhor a distribuição espaço-temporal da incidência da doença(WHO, 2012). Não obstante essa recomendação, há, atualmente, uma carência por estudos que analisem a influência de fatores socioeconômicos na taxa de incidência de dengue(HU et al., 2012).

Considerando o exposto, o presente trabalho analisou a influência de fatores climáticos e sociais na taxa de incidência de dengue no Brasil nos anos de 2011 a 2013. Todos os dados foram obtidos através de fontes oficiais. Foi criada uma metodologia que permitiu derivar regras de associação relacionadas a alta incidência de dengue. A comparação das métricas obtidas para essas regras sugere que fatores sociais exercem, de fato, significativa influência na taxa de incidência do vírus.

Abstract

Dengue ranks, today, as the most important mosquito-borne viral disease in the world(WHO, 2012). There are plenty of studies that show that dengue virus transmission is closely related to climate conditions(HALES et al., 2002; HU et al., 2012; LOWE et al., 2014). In spite of the importance of climate factors, it is a recommendation of the World Health Organization that surveillance strategies utilize information related to social and economic conditions to better understand the spatial and temporal distribution of dengue cases(WHO, 2012). Nevertheless, existing models for dengue fever do not account for socioecological factors(HU et al., 2012).

Therefore, the present work analysed the influence of socioecological and climate factors in dengue fever transmittion rates in Brazil during the years from 2011 to 2013. All the data used in the work was obtained from brazilian government oficial sources. During the work, a methodology was developed to mine association rules related to the high incidence of dengue fever. The results obtained showed that social factos play a significant role in dengue fever's incidence rate.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

BDMEP	Banco de Dados Meteorológicos para Ensino e Pesquisa
CNES	Cadastro Nacional de Estabelecimentos de Saúde
IIQ	Intervalo Interquartil
INMET	Instituto Nacional de Meteorologia
OMS	Organização Mundial de Saúde
SNIS	Sistema Nacional de Informações sobre Saneamento
WHO	*World Health Organization*

Sumário

1 Introdução

Doenças infecciosas representam uma constante ameaça à saúde pública. Segundo a Organização Mundial de Saúde, entre as doenças virais transmitidas por mosquitos, a Dengue é a que apresenta a situação epidemiológica mais alarmante do mundo (WHO, 2012). O vírus da dengue é transmitido pelas fêmeas dos mosquitos *Aedes Aegypti* e, em menor extensão, *Aedes albopictus*. A incidência de dengue cresceu drasticamente ao redor do mundo, afetando centenas de milhares de pessoas todo ano (WHO, 2018). O Brasil concentra a maior parte dos casos de dengue do continente americano. Em 2016, por exemplo, dos 2,38 milhões de casos registrados em todo o continente americano, 1,5 milhão foi registrado no Brasil (WHO, 2018).

A dengue ocorre, principalmente, em áreas tropicais e subtropicais, onde as condições do meio ambiente favorecem a proliferação dos mosquitos que são os vetores de transmissão da doença (MS, 2018). Essa sensitividade às condições climáticas ocorre devido ao fato de os mosquitos precisarem de água parada para procriar e, também, de um ambiente quente para favorecer o desenvolvimento da larva e aumentar a velocidade de replicação do vírus (HALES et al., 2002).

A compreensão dos fatores que influenciam a transmissão do vírus da Dengue pode contribuir para diminuir a ocorrência de epidemias. Diversos estudos já mostraram que a transmissão do vírus da dengue está intimamente relacionada ao clima. Hales et al. (2002) conseguiu 92% de acurácia na construção de um modelo que utiliza a umidade relativa do ar para determinar áreas com risco de transmissão da dengue. Hu et al. (2012) mostrou que a transmissão do vírus em Queensland, na Austrália, está diretamente relacionada ao aumento do índice pluviométrico e da temperatura máxima local. Por sua vez, Lowe et al. (2014) utilizou dados de previsão do clima para construir um modelo capaz de prever a ocorrência de dengue em algumas cidades do Brasil com 3 meses de antecedência.

Não obstante a importância do clima, existem outros fatores que influenciam a ocorrência de epidemias de dengue. Gubler e Clark (1996) aponta que o crescimento populacional aliado à urbanizações não planejadas contribuem para o aumento da incidência da doença. Nesse sentido, a Organização Mundial de Saúde recomenda que mecanismos de vigilância e controle de epidemias de dengue utilizem informações de caráter social e econômico para entender melhor a distribuição espaço-temporal da taxa de incidência da doença(WHO, 2012). Entretanto, conforme apontado por Hu et al. (2012), a maioria dos modelos de previsão de incidência de dengue consideram apenas fatores climáticos, havendo uma carência por estudos que analisem a influência de fatores socioeconômicos na taxa de incidência da doença.

Considerando o exposto, o presente trabalho analisou a influência de fatores climáticos e sociais na taxa de incidência de dengue em 79 municípios do Brasil nos anos de 2011,

2012 e 2013. Buscou-se encontrar regras de inferência associadas a alta incidência de dengue, separando-as em duas classes: regras envolvendo apenas fatores climáticos e regras envolvendo fatores climáticos e sociais. O objetivo foi avaliar a influência que fatores sociais exercem na incidência da dengue, comparando as métricas obtidas para essas duas classes de regras.

Procurou-se, na realização deste trabalho, utilizar dados de fontes oficiais, porém que estivessem disponibilizados abertamente na rede mundial de computadores. A única exceção foram os dados relacionados a incidência da dengue, que não estavam disponíveis na internet mas foram obtidos com o Ministério da Saúde. Dados relacionados a condições climáticas, por sua vez, foram obtidos através do Banco de Dados Meteorológicos para Ensino e Pesquisa (BDMEP/INMET). Por fim, dados relativos a condições sociais foram obtidos através do Sistema Nacional de Informações sobre Saneamento (SNIS/Ministério das Cidades) e, também, através do Cadastro Nacional de Estabelecimentos de Saúde (CNES/Ministério da Saúde).

O capítulo 2 apresenta uma breve fundamentação teórica, citando, inclusive, trabalhos relacionados ao tema na seção 2.6. A metodologia adotada na realização do trabalho é apresentada no capítulo 3. Por sua vez, o capítulo 4 detalha a análise realizada e os resultados obtidos. Por fim, o capítulo 5 conclui o trabalho, apresentando sugestões de trabalhos futuros.

2 Fundamentação Teórica

O desenvolvimento da tecnologia da informação e, em especial, da internet, levou a um aumento drástico na quantidade de dados disponíveis das mais diferentes fontes. Esse recente crescimento na quantidade de dados criou a necessidade de se utilizar novas técnicas e novas ferramentas que permitissem extrair conhecimento dessa imensidão de dados. De acordo com Han Micheline Kamber (2012), essa mudança foi responsável pelo desenvolvimento do ramo da Ciência da Computação chamado de mineração de dados (*data mining*).

Nesse mesmo sentido, Tan Michael Steinbach (2014) aponta os principais fatores que motivaram o desenvolvimento da área de mineração de dados, a saber:

- **Escalabilidade:** cada vez mais aumenta a quantidade de dados disponíveis para análise. Esse rápido crescimento na quantidade de dados exige que os algoritmos de análise sejam escaláveis. Os algoritmos de mineração de dados empregam estratégias especiais para lidar com essa quantidade crescente no volume de dados;

- **Alta Dimensionalidade:** Hoje em dia é habitual encontrar conjuntos de dados com centenas, ou milhares, de atributos. Técnicas tradicionais de análise de dados não lidam muito bem com dados com muitas dimensões. Além disso, a complexidade computacional de alguns algoritmos de análise de dados aumenta significativamente à medida que a dimensão do dado (quantidade de atributos) aumenta. Os algoritmos de mineração de dados foram desenvolvidos para lidar com essa complexidade;

- **Dados Complexos e Heterogêneos:** Métodos de análise de dados tradicionais, muitas vezes, lidavam com conjuntos de dados contendo atributos do mesmo tipo. Entretanto, nos últimos anos, houve um aumento significativo na quantidade de objetos de dados complexos, cujos atributos são heterogêneos. Para analisar esses conjuntos de dados complexos, as técnicas de mineração de dados levam em consideração relacionamentos entre os dados que não eram levados em consideração pelas técnicas tradicionais, como auto-correlação temporal e espacial, relacionamentos pai-filho entre elementos de documentos de texto semi-estruturados, entre outros;

- **Distribuição Geográfica e Propriedade dos Dados:** É cada vez mais comum que os dados necessários para uma análise estejam distribuídos geograficamente em bases de dados de propriedade de diferentes organizações. Para tratar essa questão, algumas técnicas de mineração de dados distribuída reduzem a quantidade de comunicação necessária além de criar métodos para consolidar de forma eficiente os resultados obtidos de diferentes origens;

- **Técnicas de Análise não Tradicionais:** Métodos de análise tradicionais têm como base um paradigma estatístico de elaborar e testar hipóteses. Nesse tipo de paradigma, primeiro uma hipótese é elaborada e, então, um experimento é criado para coletar e analisar dados em respeito à hipótese proposta. Entretanto, hoje em dia é comum que as tarefas de análise de dados exijam a validação de milhares de hipóteses, tornando esse processo muito trabalhoso. Deste modo, algumas técnicas de mineração de dados foram criadas com o objetivo de automatizar esse processo de gerar e testar hipóteses.

As técnicas de mineração de dados surgiram, portanto, com o objetivo de resolver as dificuldades acima expostas. Essas técnicas de mineração de dados utilizam conhecimento de diferentes áreas como estatística, recuperação de informação e aprendizado de máquina, entre outras.

Tarefas de mineração de dados são, normalmente, divididas em dois grandes grupos: tarefas preditivas e tarefas descritivas. Tarefas preditivas têm o objetivo de prever o valor de um atributo tendo como base os valores de outros atributos. Nesse tipo de tarefa, o atributo cujo valor tanta-se prever é chamado de variável dependente (variável alvo), enquanto os outros atributos são chamados de variáveis independentes. Tarefas preditivas podem ser modeladas como tarefas de classificação (vide seção 2.1) ou de regressão (seção 2.2). Por sua vez, tarefas descritivas têm o objetivo de encontrar padrões (correlações, agrupamentos, tendências e anomalias) que exemplifiquem o relacionamento entre os dados. Tarefas descritivas são exploratórias por natureza e, frequentemente, precisam utilizar técnicas de pós-processamento para validar ou explicar os resultados. Essas tarefas podem utilizar técnicas de análise de associações (vide seção 2.3), análise de agrupamentos (seção 2.4) ou de detecção de anomalias (seção 2.5).

2.1 Classificação

Em modelos preditivos, a variável dependente (variável alvo) é modelada como uma função das variáveis independentes. O objetivo é criar um modelo de aprendizado que minimize o erro entre o valor previsto e o valor real da variável dependente. Nesses modelos, a técnica de classificação pode ser utilizada quando a variável dependente é discreta.

2.2 Regressão

A regressão, de modo semelhante à classificação, também é utilizada em modelos preditivos. Entretanto, a regressão é, normalmente, utilizada quando a variável dependente é contínua.

2.3 Análise de associações

A análise de associações é uma técnica descritiva utilizada para descobrir padrões que descrevam características dos dados que estejam fortemente associadas entre si. Os padrões descobertos são, normalmente, representados na forma de regras de associação. A análise de associações possui, normalmente, um espaço de busca muito extenso, de modo que o objetivo é extrair os padrões e regras mais interessantes de uma maneira eficiente.

Para se extrair regras de associação é necessário, antes, minerar os itens frequentes do conjunto de dados. Um item X é considerado frequente se ele ocorre com uma determinada frequência mínima dentro do domínio analisado. Essa frequência mínima é, também, chamada de nível de suporte mínimo (*minsup*). O suporte de um item corresponde a quantidade de vezes em que ele ocorre no domínio e esse item é considerado frequente se seu suporte for maior ou igual a *minsup*.

Uma vez minerados os itens frequentes, a atividade de extrair regras de associação consiste em derivar regras da forma $X \rightarrow Y$, onde X e Y são itens do domínio analisado, $X \cup Y$ é frequente e $X \cap Y = \emptyset$.

Um desafio da atividade de minerar itens frequentes de um grande conjunto de dados é que o resultado invariavelmente conterá uma grande quantidade de itens que satisfazem o nível de suporte mínimo (HAN MICHELINE KAMBER, 2012). Essa grande quantidade de itens frequentes irá gerar uma grande quantidade de regras de associação. Deste modo, uma dificuldade dessa técnica consiste em estabelecer critérios para filtrar os itens e as regras mineradas de modo a encontrar regras que possam ser consideradas interessantes.

Conforme apontado por Zaki e Meira (2014), não existe um modo objetivo de se verificar se uma regra é interessante. Existem, entretanto, métodos para se eliminar regras estatisticamente insignificantes. As fórmulas apresentadas a seguir, retiradas de Zaki e Meira (2014), apresentam alguns desses métodos. Nessas fórmulas o item $X \cup Y$ está representado como XY.

2.3.1 Métricas para avaliação das regras

Suporte

O suporte de uma regra corresponde à quantidade de transações que contem X e Y e é igual a quantidade de itens frequentes que contem X e Y. O suporte relativo, por sua vez, é o percentual de regras que contem X e Y, isto é, o suporte relativo corresponde ao suporte dividido pela quantidade de total de itens frequentes do conjunto de dados. A equação 2.1 apresenta a fórmula para cálculo do suporte relativo da regra.

$$rsup(X \rightarrow Y) = P(XY) = rsup(XY) = \frac{sup(XY)}{|\mathbf{D}|} = \frac{|\mathbf{t}(XY)|}{|\mathbf{D}|} \tag{2.1}$$

O nível de suporte de uma regra é, segundo Zaki e Meira (2014), frequentemente utilizado para eliminar regras que não têm significância. Nesses casos, um nível de suporte mínimo (*minsup*) é estabelecido de modo a considerar interessantes apenas regras tais que $sup(X \rightarrow Y) \geq minsup$.

Confiança

A confiança de uma regra corresponde à probabilidade condicional de uma transação conter o consequente Y dado que ela contem o antecedente X. A equação 2.2 apresenta a fórmula de cálculo da confiança de uma regra.

$$conf(X \rightarrow Y) = P(Y|X) = \frac{P(XY)}{P(X)} = \frac{rsup(XY)}{rsup(X)} = \frac{sup(XY)}{sup(X)} \tag{2.2}$$

Normalmente, durante a atividade de minerar regras de associação, estabelece-se um nível mínimo de confiança *minconf* de modo a procurar por regras tais que $conf(X \rightarrow Y) \geq minconf$.

Lift

Lift é uma medida de correlação definida como a razão entre a probabilidade conjunta de X e Y e a probabilidade esperada se X e Y fossem estatisticamente independentes. Conforme pode ser observado pela equação 2.3, o valor de *lift* será sempre maior ou igual à confiança da regra, pois o *lift* corresponde à confiança da regra dividida pela probabilidade do consequente.

$$lift(X \rightarrow Y) = \frac{P(XY)}{P(X) \times P(Y)} = \frac{rsup(XY)}{rsup(X) \times rsup(Y)} = \frac{conf(X \rightarrow Y)}{rsup(Y)} \tag{2.3}$$

Um valor de *lift* igual a 1 indica que X e Y são variáveis independentes. Um valor menor do que 1, por sua vez, indica que X e Y possuem uma correlação negativa, isto é, a presença de um implica na ausência do outro. Por fim, um *lift* maior do que 1 indica uma correlação positiva entre X e Y. Normalmente, ao minerar regras de associação, procura-se por valores de *lift* que sejam muito maiores, ou muito menores, do que 1.

Convicção (*Conviction*)

A convicção é uma métrica do erro esperado da regra, isto é, ela indica a frequência na qual X ocorre em transações sem que Y ocorra junto. Analisando a equação 2.4 é possível observar que a convicção tende ao infinito quando a confiança é igual a 1. Por outro lado, se X e Y forem variáveis independentes, então a convicção será igual a 1.

$$conv(X \rightarrow Y) = \frac{P(X) \times P(\neg Y)}{P(X) - P(XY)} = \frac{P(\neg Y)}{1 - \frac{P(XY)}{P(X)}} = \frac{1 - rsup(Y)}{1 - conf(X \rightarrow Y)} \tag{2.4}$$

Razão de possibilidades (*odds ratio*)

A razão de possibilidades (*odds ratio*) de uma regra mede a probabilidade do consequente (Y) ocorrer na presença do antecedente (X) em contraponto a possibilidade de Y ocorrer na ausência de X. A razão de possibilidades é calculada através da equação 2.5.

$$oddsratio(X \rightarrow Y) = \frac{sup(XY) \times sup(\neg X \neg Y)}{sup(X \neg Y) \times sup(\neg XY)} \tag{2.5}$$

Segundo Zaki e Meira (2014), a razão de possibilidades é uma medida simétrica e, caso X e Y sejam independentes, ela será igual a 1. Deste modo, valores próximos a 1 podem indicar que existe pouca dependência entre X e Y. De modo semelhante, uma razão de possibilidades maior que 1 indica que a possibilidade de Y ocorrer na presença de X é maior que a de Y ocorrer na ausência de X.

2.4 Análise de agrupamentos (*Clustering*)

A análise de agrupamentos (*clustering*) visa agrupar os dados de modo que os dados de um mesmo grupo (*cluster*) guardem mais similaridades entre si do que com os dados de outros grupos.

2.5 Detecção de anomalias *outliers*

A detecção de anomalias, por sua vez, é a tarefa de identificar dados que possuam características significativamente diferentes do restante da base. Esses dados são chamados de *outliers*. O objetivo é eliminar os *outliers* antes da aplicação de outra técnica (como classificação ou regressão) e, com isso, aprimorar a acurácia do resultado. Outra aplicação comum da técnica de detecção de anomalias é a de detectar fraudes na utilização de cartões de crédito, identificando compras que não correspondam ao padrão do titular do cartão.

2.6 Trabalhos relacionados

O desenvolvimento da mineração de dados vem permitindo resolver problemas que técnicas tradicionais de análise de dados não conseguiam resolver. Nesse sentido, diversos estudos vêm utilizando técnicas de mineração de dados para analisar fatores que influenciam a transmissão ou a ocorrência de surtos e epidemias de doenças em diferentes regiões do mundo.

Souza et al. (2016) analisou postagens do Twitter para encontrar focos de infecção de dengue no Brasil, criando um método eficiente de vigilância epidemiológica através

da monitoração de redes sociais. Os dados utilizados nesse estudo eram postagens geolocalizadas. Essas postagens eram analisadas por um algoritmo de análise de sentimentos que as classificava como sendo indicativa de um caso individual de infecção de dengue ou não. Posteriormente, os dados geolocalizados eram utilizados para construir um mapa das trajetórias dos indivíduos. Essa metodologia permitiu identificar os pontos de infecção de dengue, isto é, as áreas nas quais as pessoas eram infectadas pelo vírus.

Banu et al. (2014), por sua vez, utilizou um algoritmo de regressão para construir um modelo que utilizava dados de temperatura e umidade locais para prever a ocorrência de dengue na cidade de Dhaka, em Bangladesh. O modelo em questão foi construído utilizando dados dos anos de 2000 a 2008 e validado com dados dos anos de 2009 e 2010, apresentando uma acurácia de 89%. Por fim, o estudo utilizou uma estimativa de aumento na temperatura média anual de 3,3°C para projetar um aumento de mais de 16 mil casos de dengue por ano da cidade de Dhaka até o final deste século.

Chowell et al. (2011) utilizou dados populacionais, geográficos e climáticos de um período de 15 anos para estudar a dinâmica de transmissão do vírus da dengue entre a região costeira, a região de montanhas e a as florestas do Peru. Nesse estudo foi utilizado um algoritmo de análise de séries temporais que constatou uma diferença significativa entre o tempo necessário para disseminação de uma epidemia de dengue entre as regiões analisadas. Além disso, o estudo em questão sugeriu que o vírus da dengue, no Peru, é transmitido das regiões de florestas, que são endêmicas, para as regiões costeiras. Deste modo, ações voltadas a controlar o vírus da dengue em regiões de florestas levariam, como consequência, a uma diminuição da incidência de epidemias de dengue em regiões costeiras.

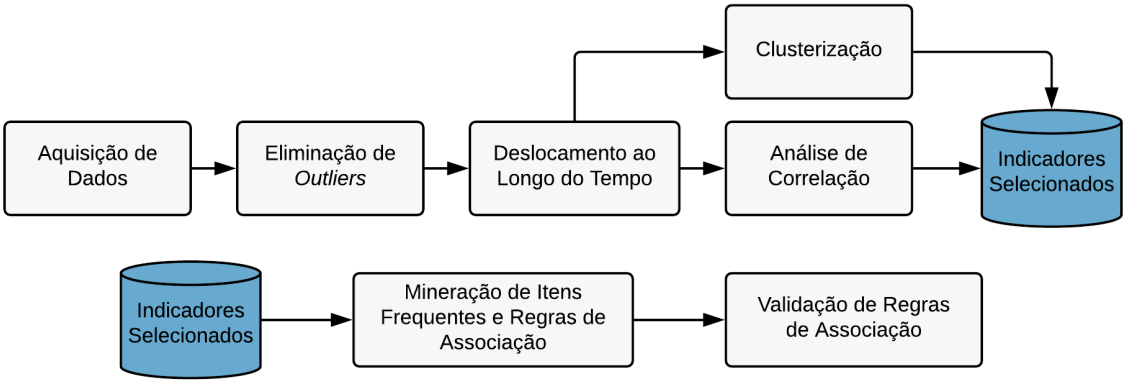
Zhou et al. (2004), por sua vez, estudou a associação entre a variação climática e a incidência de epidemias de malária no leste africano. Nesse estudo foi criado um modelo de regressão que conseguiu explicar até 81% da variação no número mensal de pacientes detectados com malária.

Os artigos supracitados estão diretamente relacionados ao trabalho proposto nesta monografia, visto que utilizam dados de diferentes origens, disponíveis na internet, para analisar (ou prever) a transmissão de doenças ou a ocorrência de epidemias. A metodologia, entretanto, adotada nesses estudos é diferente da proposta nessa monografia. Essa diferença se deve, principalmente, aos dados utilizados, à forma de análise dos dados e às técnicas utilizadas para correlacionar os dados com a taxa de incidência da doença.

3 Metodologia

A metodologia proposta neste trabalho utiliza diferentes técnicas de mineração de dados a fim de analisar a influência de condições climáticas e sociais na taxa de incidência de dengue dos municípios avaliados. A figura 3 apresenta a visão geral da metodologia. As seções subsequentes detalham cada uma das etapas do trabalho.

Figura 1 – Visão geral da metodologia do trabalho



Fonte: Elaboração própria, 2018

3.1 Aquisição de dados

O rápido desenvolvimento da internet e de mecanismos de coleta e armazenamento de dados permitiu a disponibilização de uma quantidade cada vez maior de dados. Segundo (TAN MICHAEL STEINBACH, 2014), entretanto, extrair informação útil dessa quantidade de dados disponíveis é, normalmente, uma tarefa desafiadora.

Na realização do presente trabalho, procurou-se obter dados climáticos e de condições sociais dos municípios do Brasil. O objetivo foi o de correlacionar esses dados com a taxa de incidência de dengue nos anos de 2011, 2012 e 2013. Procurou-se trabalhar com dados de fontes oficiais, porém que estivessem disponibilizados abertamente na rede mundial de computadores. A única exceção foram os dados relacionados a taxa de incidência de dengue, que não estavam disponíveis na internet, mas foram obtidos com o Ministério da Saúde.

Dados relacionados a condições climáticas foram obtidos através do Banco de Dados Meteorológicos para Ensino e Pesquisa (BDMEP) do Instituto Nacional de Meteorologia (INMET)(INMET, 2018). O BDMEP disponibiliza uma série histórica de dados coletados por estações meteorológicas que o INMET possui em diversos municípios do Brasil.

Através do BDMEP foi possível obter dadas meteorológicos de 240 municípios para todo o período pesquisado. A quantidade de dados disponibilizados no BDMEP, portanto, implicou em uma limitação inicial à quantidade de municípios incluídos no escopo do trabalho.

Por sua vez, a obtenção de dados relacionados a condições sociais representou uma dificuldade à realização do trabalho. Os dados do Censo[1] e da Pnad Contínua[2], ambos do IBGE, não serviram aos propósitos almejados, pois estes dados são segmentados em setores censitários e não em municípios. Deste modo, não seria possível relacionar os dados do Censo, ou da Pnad Contínua, com os dados de incidência de dengue.

Não obstante a dificuldade apresentada, foi possível obter dados de caráter social através do Sistema Nacional de Informações sobre Saneamento (SNIS), do Ministério das Cidades[3]. O SNIS coleta dados sobre a prestação de serviços de Água e Esgoto e sobre os serviços de manejo de Resíduos Sólidos Urbanos. Os dados de cada ano são disponibilizados para consulta através do Sistema de Série Histórica do SNIS[4](CIDADES, 2018).

Por fim, foram acrescentados à análise dados provenientes do Cadastro Nacional de Estabelecimentos de Saúde (CNES)[5]. O CNES é o sistema oficial do Ministério da Saúde que mantém o cadastro de todos os estabelecimentos de saúde do país, sejam estabelecimentos públicos, conveniados ou privados, que realizem qualquer tipo de serviço de atenção à saúde no âmbito do território nacional(DATASUS, 2018).

3.2 Eliminação de *outliers*

Um *outlier* é um dado que é significativamente diferente da expectativa, se diferenciando do restante dos dados, como se tivesse sido gerado por um mecanismo diferente. O processo de detecção de *outliers* consiste, portanto, de encontrar esses dados anômalos, que diferem significativamente do restante dos dados.

A eliminação de *outliers* foi necessária pois o algoritmo KMeans, utilizado no procedimento de agrupamento (vide seção 3.4), é do tipo *hard clustering*, o que significa que ele atribui um grupo (e apenas um) a cada um dos pontos. Com isso, a presença de *outliers* pode afetar significativamente a formação geral dos grupos. Deste modo, a eliminação de *outliers* foi realizada antes do processo de agrupamento.

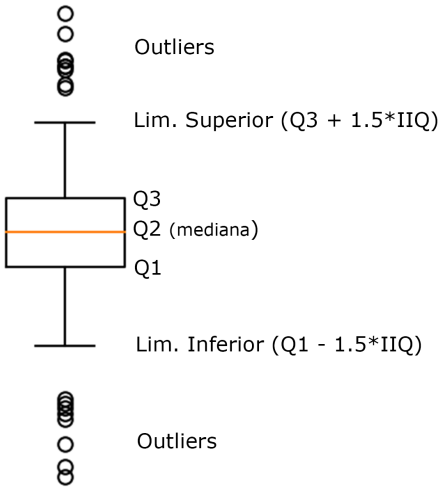
Uma das formas de eliminação de *outliers* tem como base a utilização de métodos estatísticos. Conforme detalhado por Han Micheline Kamber (2012), a ideia por trás desses métodos estatísticos é a identificação de um modelo que se encaixe no conjunto de

[1] <https://censo2010.ibge.gov.br/resultados.html>
[2] <https://www.ibge.gov.br/estatisticas-novoportal/sociais/populacao/9171-pesquisa-nacional-por-amostra-de-domicilios-
html>
[3] <http://www.snis.gov.br/aplicacao-web-serie-historica>
[4] <http://app4.cidades.gov.br/serieHistorica/>
[5] <http://cnes.datasus.gov.br>

dados e, então, os dados localizados nas regiões de baixa probabilidade desse modelo são identificados como *outliers*.

Han Micheline Kamber (2012) apresenta um método estatístico relativamente simples para detecção de *outliers*. Esse método assume que os dados seguem uma distribuição normal e tem como base a divisão dos dados em quartis: o primeiro quartil ($Q1$), a mediana ($Q2$) e o terceiro quartil ($Q3$). A intervalo interquartil (IIQ) é, então, definido como $Q3 - Q1$. Assim, qualquer dado que for menor que $Q1 - 1.5 \times IIQ$ ou maior que $Q3 + 1.5 \times IIQ$ é classificado como *outlier*. Segundo Han Micheline Kamber (2012), em uma distribuição normal, 99.3% dos dados estão contidos nesse intervalo. A figura 3.2 ilustra o resultado desse método para eliminação de *outliers*.

Figura 2 – *Boxplot* ilustrando a eliminação de *outliers* pelo método estatístico



Fonte: Han Micheline Kamber (2012)

3.3 Deslocamento ao longo do tempo

Os indicadores climáticos foram, no presente trabalho, relacionados com a taxa de incidência de dengue através de um deslocamento ao longo do tempo. Esse deslocamento temporal teve o objetivo de simular o tempo necessário para que mudanças climáticas se reflitam na atividade do mosquito da dengue.

Segundo Chowell et al. (2011) esse deslocamento temporal deve levar em consideração o tempo necessário para que o mosquito procrie e se desenvolva. Além disso, deve ser considerado o tempo de incubação extrínseca do vírus no vetor de transmissão e, também, o tempo de incubação intrínseca no hospedeiro humano.

Considerando o exposto, os dados climáticos na semana i foram correlacionados com a taxa de incidência de dengue na semana $i + \tau$, onde τ corresponde ao deslocamento temporal em semanas. O deslocamento temporal variou de 1 a 4 semanas, de modo que

os dados climáticos na semana i foram correlacionados com a taxa de incidência de dengue nas semanas $i+1$, $i+2$, $i+3$ e $i+4$.

3.4 Agrupamento

Após eliminar os *outliers*, todos os indicadores foram discretizados com a utilização do algoritmo de agrupamento KMeans. A qualidade do agrupamento foi verificada utilizando o coeficiente de Silhouette.

O coeficiente de Silhouette mede a coesão e a separação dos grupos e é baseado na diferença entre a distância média dos pontos de um grupo ao grupo mais próximo e a distância média dos pontos ao seu próprio grupo (ZAKI; MEIRA, 2014). O coeficiente de Silhouette varia de $[-1, +1]$, sendo que um valor próximo de $+1$ indica que os pontos de um grupo são mais próximos de outros pontos do mesmo grupo do que de pontos de outros grupos. De modo análogo, um valor próximo a -1 indica que os pontos de um grupo estão, na média, mais próximos a pontos de grupos vizinhos do que a pontos de seu próprio grupo. Deste modo, um valor para o coeficiente de Silhouette perto de $+1$ é um indicativo de um bom agrupamento.

O algoritmo KMeans recebe como parâmetro um número k que indica a quantidade de grupos que devem ser formados. Assim, o algoritmo agrupa os pontos, separando-os em k grupos. Um problema é, então, saber qual o melhor valor de k. Para isso, o KMeans foi, neste trabalho, executado variando k de 2 até 5. Para cada valor de k foi calculado o coeficiente de Silhouette. Além disso, foi adotada uma faixa de tolerância de 1% para o coeficiente de Silhouette. Com isso, foi escolhido o menor valor de k cujo coeficiente de Silhouette estivesse dentro da faixa de tolerância de 1% em relação ao maior coeficiente de Silhouette encontrado. Esse procedimento adotado para discretizar os indicadores está detalhado no algoritmo 1.

3.5 Análise de correlação

Medidas de correlação medem como um atributo varia em relação outro. Neste sentido, para verificar como se dá a relação entre dois atributos numéricos, um método muito utilizado é calcular o coeficiente de correlação de *Pearson*.

O coeficiente de correlação de *Pearson*, cujo cálculo é realizado de acordo com a equação 3.1, mede a relação linear entre dois conjuntos de dados. Na equação 3.1, $x_1, x_2, ..., x_n$ e $y_1, y_2, ..., y_n$ são os valores medidos para as variáveis cuja correlação se deseja calcular. O coeficiente varia de -1 a $+1$, sendo que o resultado igual a 0 (zero) significa que não existe correlação. Por sua vez, -1 e $+1$ indicam uma relação linear perfeita, com correlações negativas indicando que um indicador aumenta a medida em que o outro diminui.

input : Dataset
output: Clusters

```
1  Clusters ← ∅;
2  foreach Indicador ∈ Dataset do
3      MelhorSilhouette ← −1;
4      MelhorCluster ← ∅;
5      k ← 2;
6      while k ≤ 5 do
7          IndCluster ← KMeans(k, Indicador);
8          IndSilhouette ← Silhouette(Indicador, IndCluster);
9          if (IndSilhouette × 0.99 > MelhorSilhouette) then
10             MelhorSilhouette ← IndSilhouette;
11             MelhorCluster ← IndCluster;
12         k ← k + 1;
13     Clusters ← (Clusters, MelhorCluster);
```

Algoritmo 1: Clusterização dos indicadores

Correlações positivas, por sua vez, indicam que, quando um indicador aumenta, o outro também aumenta.

$$\rho = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2} \times \sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{3.1}$$

3.6 Mineração de itens frequentes e regras de associação

Conforme apontado por Zaki e Meira (2014), para derivar regras de associação é essencial, antes, enumerar todos os itens frequentes juntamente com seus respectivos níveis de suporte. Segundo esse mesmo autor, dados uma base de dados **D** e um nível mínimo de suporte *minsup*, a tarefa de minerar itens frequentes corresponde à tarefa de enumerar todos os itens da base de dados **D** que ocorrem com uma frequência mínima igual a *minsup*. A frequência de um item corresponde a quantidade de transações que contêm esse item e é, também, chamada de nível de suporte do item.

Segundo detalhado na seção 2.3.1, todas as métricas necessárias para avaliar a qualidade de regras de associação podem ser derivadas do nível de suporte dos itens frequentes que compõem a regra. Sendo assim, Han Micheline Kamber (2012) afirma que a complexidade da atividade de encontrar regras de associação pode ser reduzida à complexidade da tarefa de encontrar itens frequentes.

A atividade de encontrar itens frequentes é computacionalmente onerosa. Uma vez, entretanto, identificados os itens frequentes, a consulta a essa tabela de itens deve ser rápida. Caso essa consulta seja lenta, a atividade de derivar regras de associação também se tornará lenta, já que a derivação dessas regras pode implicar em várias consultas a tabela de itens frequentes.

A identificação de itens frequentes foi realizada, neste trabalho, utilizando o algoritmo *Apriori*. Os itens frequentes foram, então, armazenados em uma estrutura de dados chamada de dicionário[6]. A construção desse dicionário possui tempo de execução $O(n)$, mas o tempo para consulta é $O(1)$. Essa estratégia permitiu significativo ganho de performance na derivação de regras de associação.

Além disso, a derivação de regras de associação se limitou a encontrar regras que tivessem, no consequente, o indicador de *Alta* incidência de dengue. Essa estratégia permitiu, mais uma vez, significativo ganho de performance pois, conforme pode ser observado na tabela 4.2, os itens frequentes que possuem o indicador de *Alta* incidência de dengue correspondem apenas a 0.68% do total de itens encontrados. O algoritmo 2 apresenta as etapas realizadas para derivar as regras de associação.

input : ItensFreq, Consequent, MinThreshold
output: Regras

1 $Y \leftarrow Consequent$;
2 $Regras \leftarrow \emptyset$;
3 $A \leftarrow \{I | (I \subset ItensFreq) \wedge (Y \subset I)\}$;
4 **foreach** $Z \in A$ *tal que* $|Z| \geq 2$ **do**
5 $X \leftarrow Z \setminus Y$; `// retiramos Y de Z e atribuímos a X`
6 $conf = sup(XY)/sup(X)$;
7 **if** $conf \geq MinThreshold$ **then**
8 $lift = conf/sup(Y)$;
9 $leverage = sup(XY) - sup(X) \times sup(Y)$;
10 $conviction = \frac{1-sup(Y)}{1-conf}$;
11 $productive, pvalue, oddsratio = FisherTest(ItensFreq, X, Y)$;
12 $Regras \leftarrow (Regras, X \rightarrow Y)$;

Algoritmo 2: Derivação de regras de associação com base no consequente

3.7 Validação de regras de associação

Existem diversas métricas que visam quantificar as propriedades das regras de associação mineradas. Conforme apontado por Zaki e Meira (2014), não existe um modo objetivo para verificar se uma determinada regra é interessante. Existem, entretanto, métodos para eliminar regras que não são estatisticamente significantes. Alguns desses métodos estão detalhados na seção 2.3.1.

Considerando o exposto, o presente trabalho realizou uma comparação entre as regras de associação relacionadas a alta incidência de dengue, dividindo-as em duas classes: regras que possuíam, no antecedente, apenas fatores climáticos e regras que envolviam fatores climáticos e sociais. O objetivo era de verificar a influência que fatores sociais exercem na taxa de incidência de dengue.

[6] <https://docs.python.org/2/library/stdtypes.html#typesmapping>

A influência exercida pelos fatores sociais foi, portanto, apurada comparando as métricas das duas classes de regras.

4 Experimentos e Resultados

4.1 Engenharia de dados

Conforme relatado na seção 3.1, a execução deste trabalho utilizou dados do Banco de Dados Meteorológicos para Ensino e Pesquisa (BDMEP), do INMET(INMET, 2018), do Cadastro Nacional de Estabelecimentos de Saúde (CNES), do Ministério da Saúde(DATASUS, 2018) e do Sistema Nacional de Informações sobre Saneamento (SNIS), do Ministério das Cidades(CIDADES, 2018).

A primeira parte do trabalho envolveu o tratamento dos dados coletados. Sendo assim, os dados passaram por um processo que envolveu o tratamento de dados errados, ausentes ou incoerentes. A tabela 4.1 apresenta um resumo do resultado do tratamento desses dados.

Tabela 1 – Resumo dos indicadores

Base de Dados	**Qtde. Indicadores**	**Qtde. Ausente ou Incoerente**
SNIS - Água e Esgoto	75	36,16%
SNIS - Resíduos Sólidos	46	70,05%
CNES	21	9,26%
BDMED	9	0%
Dengue	1	0%

Fonte: Dados da pesquisa

O apêndice A apresenta cada um dos indicadores coletados. Conforme pode ser observado na tabela 4.1, a base de dados do SNIS possui uma quantidade de dados ausentes significativa. Considerando o exposto, adotou-se a estratégia de retirar os dados ausentes, ou incoerentes, da base.

Além disso, para reduzir o esforço de avaliação do modelo, foi adotada uma estratégia semelhante à proposta por Albinati et al. (2017). No trabalho citado, Albinati et al. (2017) analisou apenas cidades com mais de 100 mil habitantes. Considerando, então, essa restrição adicional, o escopo do presente trabalho ficou restrito à análise de 79 cidades do Brasil pois, entre os 240 municípios cujos dados meteorológicos foram obtidos através do BDMEP (vide seção 3.1), apenas 79 possuem mais de 100 mil habitantes.

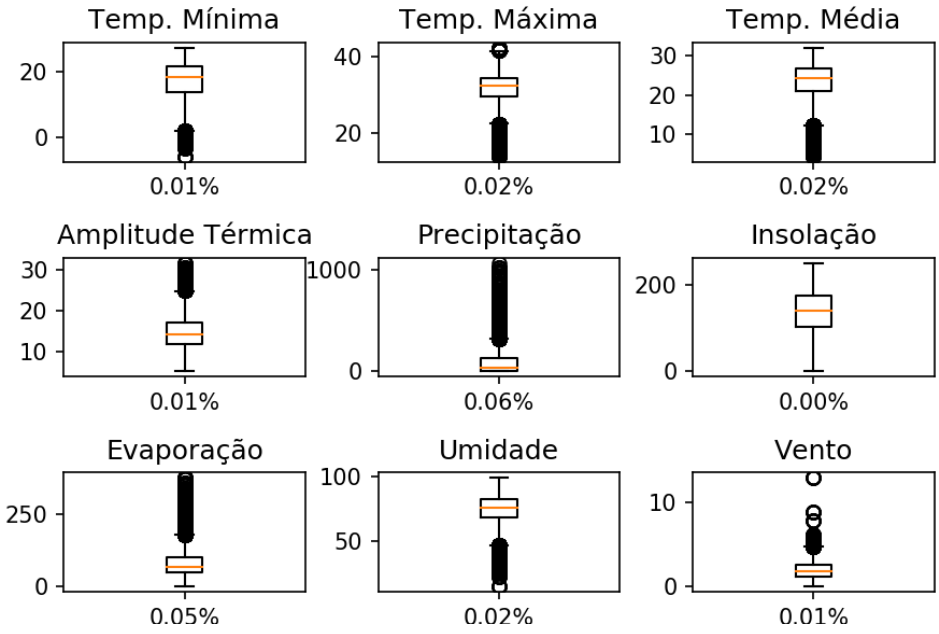
4.1.1 Eliminação de *outliers*

Um *outlier* é um dado que de diferencia significativamente do restante dos dados, como se tivesse sido gerado por um mecanismo diferente. Um processo de detecção de *outliers* tem, portanto, o objetivo de encontrar esses dados que são significativamente diferente

da expectativa. A seção 2.5 apresentou um resumo das diferentes técnicas de detecção de *outliers*. Na realização do presente trabalho foi adotado o método estatístico de detecção de *outliers*. Este método está detalhado na seção 3.2.

Conforme explicado na seção 3.2, todos os dados passaram pelo processo de eliminação de *outliers* antes de serem discretizados com o *K-means*. A figura 4.1.1 ilustra o resultado da eliminação de *outliers* para os dados climáticos. O número abaixo de cada *boxplot* indica o percentual de *outliers* do indicador.

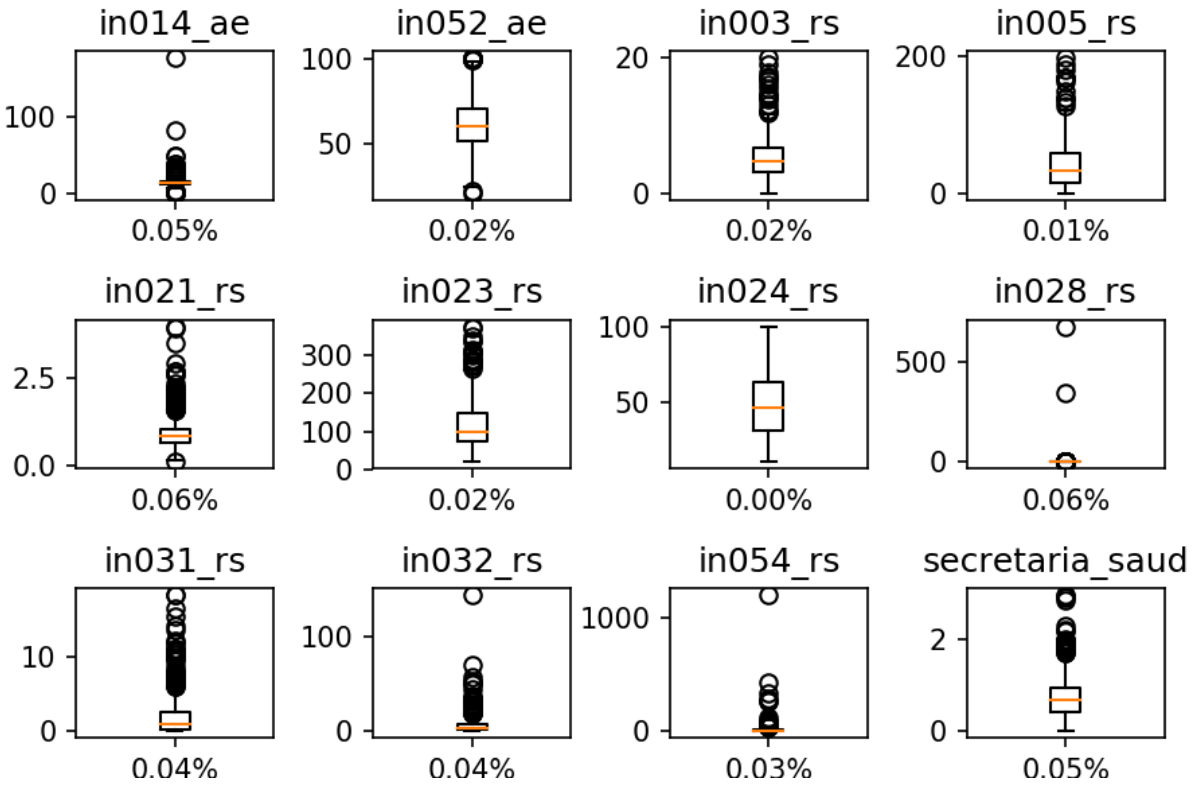
Figura 3 – Eliminação de *outliers* dos indicadores climáticos



Fonte: Dados da pesquisa

Além dos indicadores climáticos, foram analisados um total de 156 indicadores relacionados a condições sociais. Conforme detalhado na seção 4.2, a mineração de itens frequentes para todos esses indicadores se mostrou uma tarefa computacionalmente inviável. Sendo assim, esses indicadores passaram por um processo de análise de correlação que verificou a relação entre cada um desses indicadores e a taxa de incidência de dengue nos municípios analisados. Após realizada a análise de correlação, dos 156 indicadores relacionados a condições sociais, apenas 12 foram considerados relevantes para a realização do trabalho. A figura 4.1.1 ilustra o resultado da eliminação de *outliers* para os dados de caráter social. O processo de eliminação de *outliers* foi realizado para todos os 156 indicadores de caráter social mas, para simplificar, a figura 4.1.1 apresenta apenas o resultado para os 12 indicadores considerados relevantes. O resultado completo, que inclui o significado de cada sigla dos indicadores, está disponível no apêndice A. O número abaixo de cada *boxplot* indica o percentual de *outliers* do indicador.

Figura 4 – Eliminação de *outliers* dos indicadores sociais



Fonte: Dados da pesquisa

4.1.2 Análise de Correlação

A análise de correlação foi realizada buscando detectar a interdependência entre a taxa de incidência de dengue e os indicadores climáticos e sociais. Os dados relativos a taxa de incidência de dengue e os indicadores climáticos tinham periodicidade semanal. Deste modo, a análise de correlação entre esses indicadores foi mais intuitiva.

Os indicadores sociais, por outro lado, tinham periodicidade anual. Deste modo, para realizar a análise de correlação, foi calculada a taxa de incidência de dengue anual do município. Essa taxa anual foi obtida somando os valores relativos às taxas semanais de incidência do vírus.

O coeficiente de *Pearson* foi, então, calculado para verificar a correlação da taxa de incidência de dengue com cada um indicadores analisados. A fim de evitar distorções causadas pela presença de *outliers*, antes de calcular a correlação, os dados passaram pelo processo de eliminação de *outliers* descrito na seção 4.1.1.

4.1.3 Agrupamento

Todos os indicadores eram, originalmente, variáveis contínuas. Antes, então, de procurar por itens frequentes, esses indicadores foram discretizados de acordo com o procedimento

descrito na seção 3.4. Conforme explicado, o algoritmo KMeans foi executado variando k de 2 até 5. Para cada execução, foi calculado o coeficiente de Silhouette do agrupamento. Além disso, foi adotada uma margem de 1% para o coeficiente de Silhouette, de modo que o agrupamento escolhido foi o de menor valor de k cujo coeficiente de Silhouette estivesse dentro da faixa de tolerância de 1% em relação ao maior coeficiente de Silhouette encontrado. O algoritmo 1 detalhou esse procedimento.

Os indicadores climáticos, obtidos através do BDMEP, passaram pelo processo de deslocamento temporal descrito na seção 3.3. Sendo assim, passou-se a trabalhar com 9 indicadores climáticos da semana 0 (sem deslocamento temporal), 9 da semana 1 (deslocamento temporal de 1 semana) e assim sucessivamente, de modo que os 9 indicadores climáticos originais se transformaram em 45 indicadores.

O SNIS, por sua vez, forneceu 121 indicadores. Destes, 75 eram referentes a condições de saneamento de água e esgoto e os outros 46 referentes a saneamento de resíduos sólidos dos municípios. Esses indicadores tinham periodicidade anual e, portanto, não passaram pelo processo de deslocamento temporal.

Por fim, o CNES forneceu dados relativos a quantidade de estabelecimentos de saúde nos municípios. Esses dados estavam segmentados por tipo de estabelecimento de saúde. Os dados do CNES indicavam a quantidade absoluta de estabelecimentos de saúde do município. Esses dados foram, então, transformados de modo a obtermos um indicador da quantidade de estabelecimentos de saúde por 100 mil habitantes. A segmentação por tipo de estabelecimento foi mantida. Deste modo, o CNES forneceu um total de 21 indicadores, cada um desses indicando a quantidade de um determinado tipo de estabelecimento de saúde por grupo de 100 mil habitantes.

Cada grupo gerado pelo processo de agrupamento foi identificado pela sigla do indicador seguida de um número. Por exemplo, o indicador *in036_rs*, que se refere à massa de resíduos sólidos coletada por habitante do município, foi dividido em três grupos. Deste modo, o grupo 1 é identificado pela sigla $in036\_rs_1$; o grupo 2 pela sigla $in036\_rs_2$ e o grupo 3 por $in036\_rs_3$. A tabela 4.1.3 resume o resultado obtido com o agrupamento. Informações detalhadas para cada indicador estão disponíveis no apêndice A.

Tabela 2 – Resultado da clusterização

k	**BDMEP**	**SNIS**	**CNES**	**Total**
2	45	102	10	**159**
3	0	10	5	**18**
4	0	6	5	**15**
5	0	3	1	**9**

Fonte: Dados da pesquisa

4.2 Mineração de itens frequentes

A procura por itens frequentes é uma das tarefas mais comuns e importantes da mineração de dados. Para minerar itens frequentes utilizou-se o algoritmo Apriori, que é um dos mais simples e populares. A atividade de minerar itens frequentes possui uma complexidade computacional significativa e essa complexidade é diretamente proporcional à quantidade de itens analisados.

No caso do Apriori, essa complexidade é, no pior cenário, igual a $O(|\Gamma| \times |\mathbf{D}| \times 2^{|\Gamma|})$, onde $|\mathbf{D}|$ corresponde ao tamanho do conjunto de dados e $|\Gamma|$ à quantidade de itens analisados. No presente trabalho, o conjunto de dados corresponde aos dados semanais de incidência de dengue em 79 municípios do Brasil para os anos de 2011, 2012 e 2013. Sendo assim, o tamanho do conjunto de dados é $|\mathbf{D}| = 3 \times 52 \times 79 = 12.324$.

Os itens analisados, por sua vez, correspondem aos indicadores climáticos e sociais. Conforme detalhado na seção 4.1, foram coletados 9 diferentes indicadores climáticos. Esses indicadores climáticos passaram por um processo de deslocamento temporal (vide seção 3.3) que teve o objetivo de simular o tempo necessário para que variações climáticas se reflitam na taxa de incidência de dengue. Sendo assim, foi realizado um deslocamento temporal de 0, 1, 2, 3 e de 4 semanas. Os indicadores climáticos foram, portanto, analisados segundo esse deslocamento no tempo. Deste modo, cada um dos 9 indicadores relacionados a condições climáticas foi deslocado no tempo por 0, 1, 2, 3 e por 4 semanas, resultando em um total de 45 indicadores climáticos analisados. Além disso, conforme detalhado na seção 4.1, outros 156 indicadores relacionados a condições sociais foram acrescentados à análise. Por fim, dois indicadores relacionados a população dos municípios foram acrescentados à análise: um indicando a população total do município e outro referente ao percentual da população que vive em área urbana.

Considerando, então, a complexidade computacional do Apriori, o tamanho do conjunto de dados e a quantidade de indicadores analisados, fez-se necessário realizar uma análise prévia para selecionar os indicadores, otimizando a execução do algoritmo de mineração de itens frequentes. Essa análise foi realizada através do cálculo da correlação de *Pearson*, conforme apresentado nas seções 3.5 e 4.1.2.

Após selecionar os principais indicadores através da análise de correlação, teve início o processo de encontrar regras de associação entre os indicadores e a incidência de dengue. Para isso, a atividade de minerar itens frequentes foi realizada utilizando os indicadores discretizados de acordo com os procedimentos detalhados nas seções 3.4 e 4.1.3.

Na mineração dos itens frequentes foi estabelecido o nível de suporte mínimo de 0.04. Foram encontrados, ao todo, 8.612.163 itens frequentes. A tabela 4.2 apresenta um resumo dos itens frequentes, separando-os em itens relacionados a alta incidência de dengue, itens relacionados a baixa incidência de dengue e itens que não possuem relação com o indicador de dengue.

Tabela 3 – Itens frequentes agrupados segundo indicador de incidência de dengue

Incidência de Dengue	**Itens Frequentes**
Alta	58.688
Baixa	2.212.514
Sem relação	6.340.961
Total	8.612.163

Fonte: Dados da pesquisa

Considerando a quantidade de itens frequentes encontrados, o presente trabalho focou na análise dos itens frequentes relacionados ao indicador de alta incidência de dengue. Conforme detalhado na seção 4.3, essa segmentação do trabalho otimizou significativamente a busca por regras de associação, pois apenas os itens frequentes relacionados a alta incidência de dengue foram considerados para derivar as regras de associação.

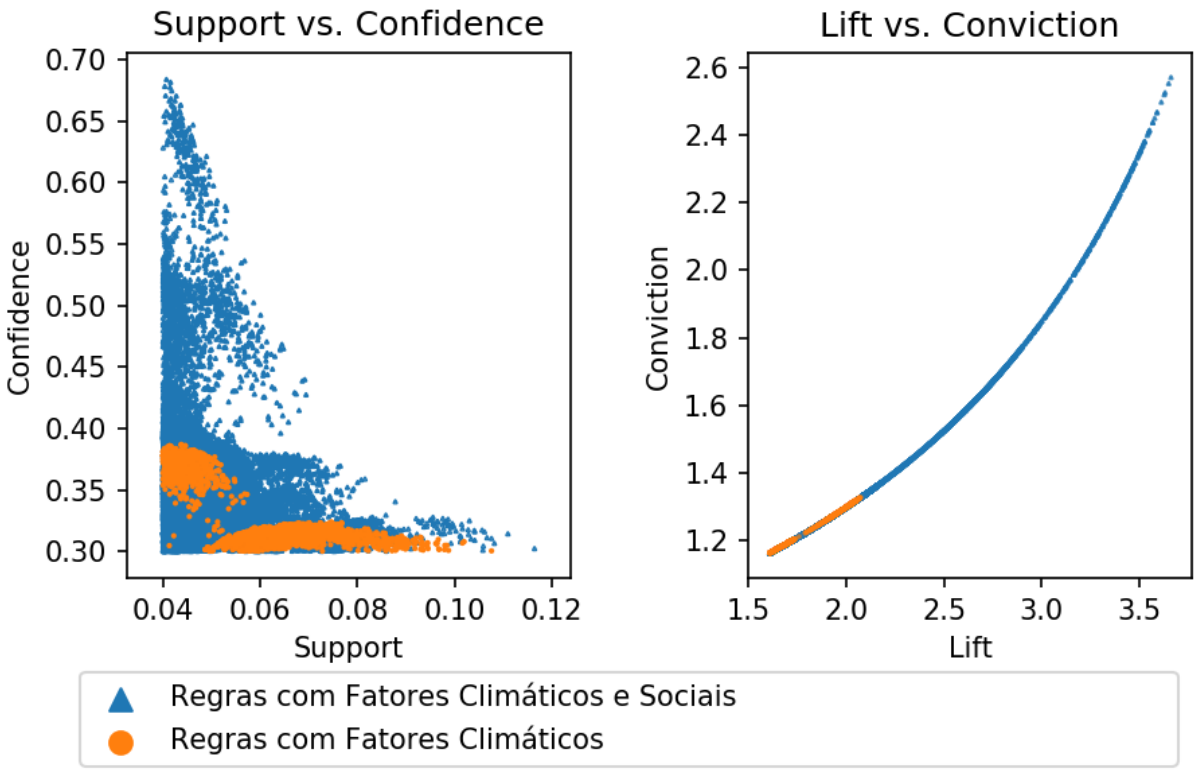
4.3 Regras de associação

Para derivar regras de associação é necessário, antes, minerar todos os itens frequentes do conjunto de dados, calculando o valor do suporte para cada item. Após enumerar todos os itens frequentes, a derivação de regras de associação é realizada seguindo o algoritmo 2 e o nível de confiança para a regra é calculado de acordo com a equação 2.2. Conforme detalhado na seção 3.6, a derivação de regras de associação foi simplificada neste trabalho com o objetivo de encontrar apenas regras que estivessem associadas a alta taxa de incidência de dengue, isto é, regras que tivessem, no consequente, o item frequente de incidência "Alta".

Considerando o exposto, na derivação de regras de associação foram analisados apenas os itens frequentes que possuíam o indicador de alta taxa de dengue. A estratégia em questão permitiu significativo ganho de desempenho, já que, conforme detalhado na tabela 4.2, a atividade de derivar regras de associação analisou apenas 58.688 de um total de 8.612.163 itens frequentes. O nível mínimo de confiança estabelecido foi de 0.3. Foram, com esses critérios, derivadas 43.003 regras de associação. A figura 4.3 apresenta as principais métricas dessas regras, dividindo-as em regras que apresentam, no antecedente, apenas fatores climáticos e regras que apresentam o antecedente com fatores climáticos e sociais.

Conforme pode ser observado na figura 4.3, regras que envolviam fatores climáticos e sociais apresentaram níveis de confiança, *lift* e *conviction* significativamente superiores aos de regras envolvendo apenas fatores climáticos. As tabelas 4.4 e 4.4 apresentam, resumidamente, algumas dessas regras. A seção 4.4, por sua vez, analisa detalhadamente esse resultado.

Figura 5 – Regras relacionadas a alta taxa de incidência de dengue



Fonte: Dados da pesquisa

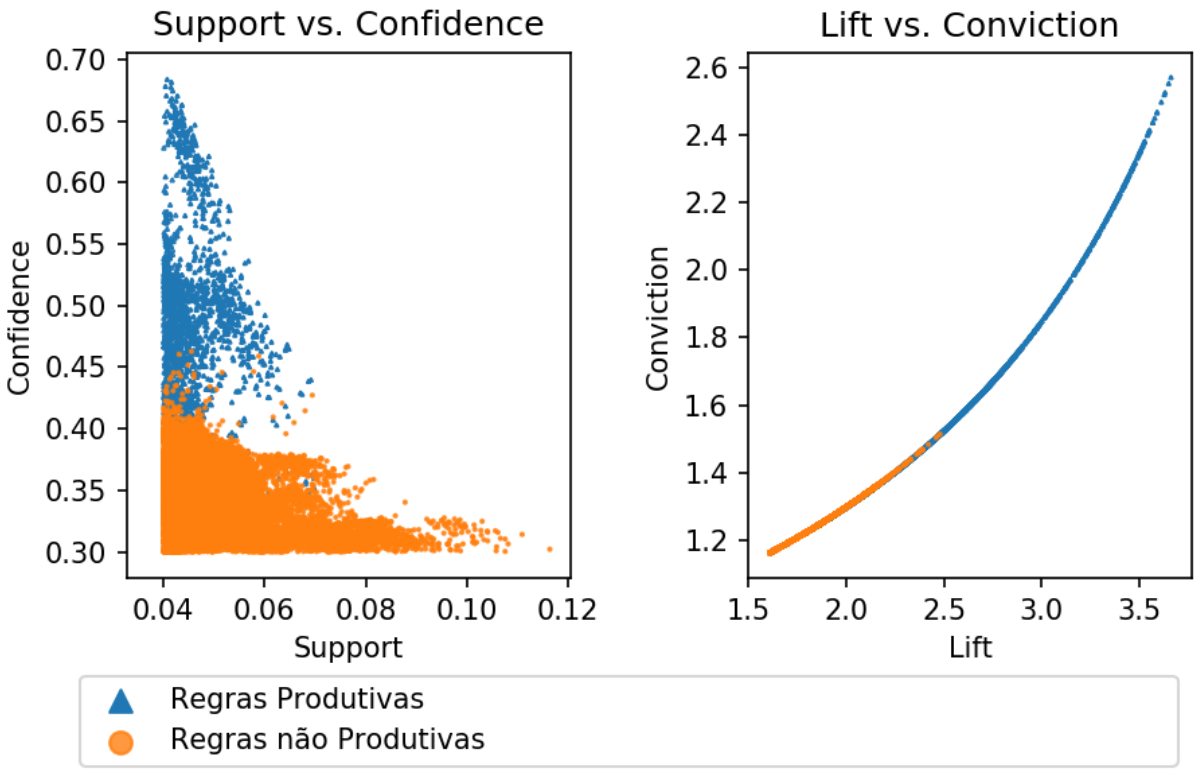
4.4 Análise de Resultados

A significância estatística das regras encontradas foi verificada realizando o *Fisher Exact Test*. O teste em questão tem o objetivo de afastar a hipótese nula calculando o *p-value* das tabelas de contingência de todas as generalizações da regra que se deseja testar. Caso o *p-value* seja menor que um dado valor α para todas as generalizações, então a hipótese nula é afastada e a regra é considerada produtiva.

Nos testes realizados foi adotado o valor limite de $\alpha \leq 0.01$. Além disso, conforme sugerido por Webb (2007), foi adotado um critério adicional para verificar se todos os fatores do antecedente da regra contribuíam, de fato, para seu nível de confiança. Em outras palavras, desejava-se verificar se o nível de confiança da regra é maior que o de todas as suas especializações. Com essa restrição, o *p-value* da regra que está sendo testada deveria ser menor que o de todas as suas especializações. Com esse critério adicional, apenas regras envolvendo fatores climáticos e sociais foram consideradas produtivas.

A figura 4.3 apresentou as principais métricas de todas as regras encontradas, separando-as em regras envolvendo apenas fatores climáticos e regras envolvendo fatores climáticos e sociais. A figura 4.4, por sua vez, apresenta as mesmas métricas separando as regras em produtivas e não produtivas. Todas as regras consideradas produtivas envolvem fatores climáticos e sociais.

Figura 6 – Regras Produtivas vs. não Produtivas



Fonte: Dados da pesquisa

Após a verificação da produtividade das regras, passou-se a análise das principais regras encontradas. Para isso, entre as regras consideradas produtivas, as principais foram selecionadas seguindo dois critérios: níveis de suporte/confiança e níveis de *conviction*/*lift*. Todas essas regras possuem, no consequente, o indicador de Alta incidência de dengue.

Seguindo o primeiro desses critérios, foram selecionadas as 5 regras com maiores níveis de suporte e confiança. Para realizar essa seleção, estabeleceu-se uma margem de 10% de variação no suporte e, dentro dessa margem, as regras com maior nível de confiança foram selecionadas. Considerando, então, que o maior nível de suporte entre as regras produtivas foi de 0.0699436, essa margem de 10% permitiu selecionar as regras com maiores níveis de confiança desde que o nível de suporte fosse maior que 0.06294924. Essas regras estão elencadas na tabela 4.4.

Por fim, de acordo com o segundo critério, foram selecionadas as 5 regras com maiores níveis de convicção e *lift*. De modo semelhante ao mecanismo de seleção anteriormente explicado, estabeleceu-se uma margem de 10% de variação na convicção e, dentro dessa margem, as regras com maiores *lift* foram selecionadas. Deste modo, considerando que o maior nível de convicção foi de 2.57092, foram selecionadas as regras com maiores níveis de *lift* desde que o nível de convicção fosse maior que 2.313828. Essas regras estão elencadas na tabela 4.4.

Tabela 4 – Principais regras segundo critério de Confiança e Suporte

Antecedente	***Support***	***Confidence***	***Lift***	***Conviction***
$in052\_ae_1$, $sem2\_vento_1$, $sem4\_temp\_minima_1$, $sem4\_vento_1$	0.0645	0.4682	2.5092	1.5296
pop_urb_0, $in052\_ae_1$, $sem3\_temp\_minima_1$, $sem4\_temp\_minima_1$	0.0648	0.4647	2.4903	1.5195
$in052\_ae_1$, $sem2\_vento_1$, $sem3\_temp\_minima_1$, $sem4\_temp\_minima_1$	0.0641	0.4646	2.4900	1.5193
$in052\_ae_1$, $sem3\_temp\_minima_1$, $sem4\_temp\_minima_1$, $sem4\_vento_1$	0.0641	0.4638	2.4854	1.5169
$in052\_ae_1$, $sem2\_vento_1$, $sem3\_temp\_minima_1$, $sem4\_vento_1$	0.0630	0.4584	2.4565	1.5018

Fonte: Dados da pesquisa

A análise das tabelas 4.4 e 4.4 nos mostra que as regras selecionadas pelo critério dos níveis de suporte e confiança são diferentes das selecionadas pelo critério de *lift* e *conviction*, sugerindo que deve haver um equilíbrio entre as métricas para selecionar as melhores regras.

Entretanto, nota-se, também, que todas essas regras possuem, no antecedente, o indicador *in052_ae*. Esse indicador corresponde ao índice de volume de água consumido por todos os moradores do município[1], sugerindo a existência de uma relação entre o consumo de água e a incidência do vírus da dengue.

Para analisar mais detidamente a hipótese apresentada, segmentamos as regras de associação encontradas em duas classes: a primeira classe com regras que possuíam o indicador $in052\_ae_1$ no antecedente e segunda classe com regras análogas a primeira, porem sem o indicador $in052\_ae_1$ no antecedente. Em outras palavras, a primeira classe era formada por regras da forma $X \rightarrow Y \mid in052\_ae_1 \subset X$ enquanto a segunda classe era constituída por regras tais que $X' \rightarrow Y \mid X' = X \setminus in052\_ae_1$. A figura 4.4 apresenta o resultado dessa análise.

Além do indicador relativo ao consumo de água, merece, também, destaque o indicador $in021\_rs$. Esse indicador corresponde à massa coletada de lixo (domiciliar e público) *per capita*. A análise da figura 4.4 permite observar a influência desse indicador nas métricas das regras, mostrando que existe uma relação entre o serviço de coleta de lixo urbano e a incidência de dengue nos municípios.

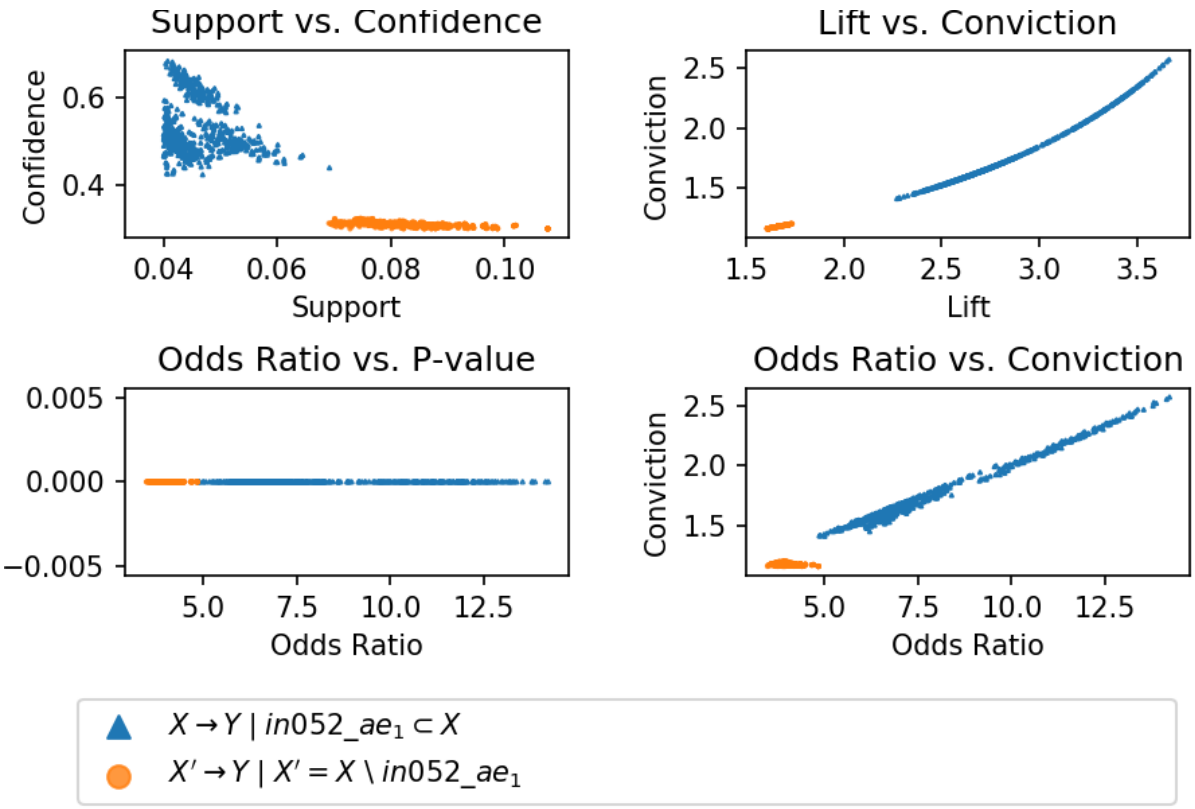
[1] Vide glossário de indicadores do SNIS disponível em: <http://snis.gov.br/downloads/manuais-atualizados/Glossario_Indicadores_RS2017.zip>

Tabela 5 – Principais regras segundo critério de *Conviction* e *Lift*

Antecedente	***Support***	***Confidence***	***Lift***	***Conviction***
pop_urb_0, $sem3\_temp\_media_0$, $sem4\_temp\_media_0$, $in052\_ae_1$, $sem1\_temp\_minima_1$, $sem2\_vento_1$, $sem3\_temp\_minima_1$, $sem4\_temp\_minima_1$, $sem4\_vento_1$	0.0407	0.6836	3.6635	2.5709
pop_urb_0, $sem2\_temp\_minima_0$, $sem4\_temp\_media_0$, $in052\_ae_1$, $sem1\_temp\_minima_1$, $sem2\_vento_1$, $sem3\_temp\_minima_1$, $sem4\_temp\_minima_1$, $sem4\_vento_1$	0.0416	0.6814	3.6515	2.5529
pop_urb_0, $sem2\_temp\_minima_0$, $sem3\_temp\_media_0$, $sem4\_temp\_media_0$, $in052\_ae_1$, $sem1\_temp\_minima_1$, $sem2\_vento_1$, $sem4\_temp\_minima_1$, $sem4\_vento_1$	0.0404	0.6780	3.6332	2.5258
pop_urb_0, $sem2\_temp\_minima_0$, $sem3\_temp\_media_0$, $in052\_ae_1$, $sem1\_temp\_minima_1$, $sem2\_vento_1$, $sem3\_temp\_minima_1$, $sem4\_temp\_minima_1$, $sem4\_vento_1$	0.0415	0.6772	3.6291	2.5198
pop_urb_0, $sem2\_temp\_minima_0$, $sem3\_temp\_media_0$, $sem4\_temp\_media_0$, $in052\_ae_1$, $sem2\_vento_1$, $sem3\_temp\_minima_1$, $sem4\_temp\_minima_1$, $sem4\_vento_1$	0.0418	0.6744	3.6139	2.4978

Fonte: Dados da pesquisa

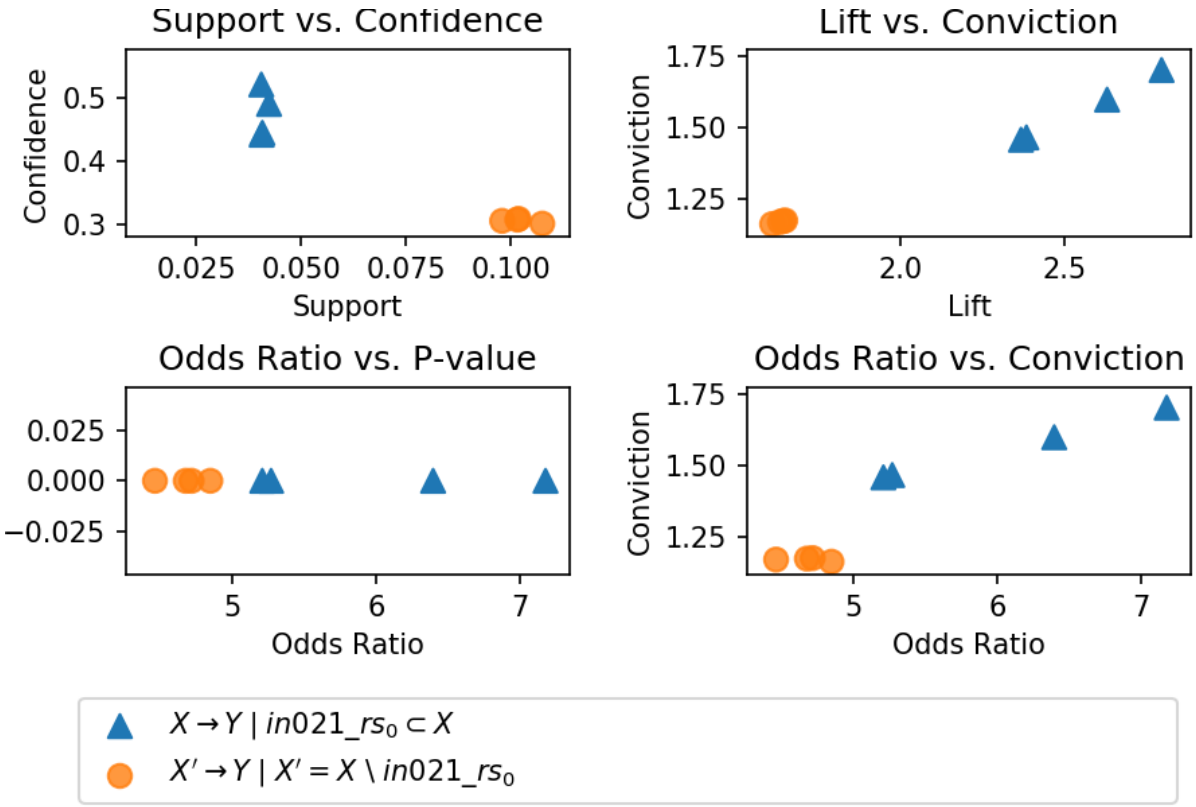
Figura 7 – Influência do indicador de consumo de água



Fonte: Dados da pesquisa

Considerando o exposto, a análise das regras de associação encontradas sugerem a existência de uma relação entre condições sociais e a taxa de incidência de dengue. Mereceu destaque, na análise realizada, a influência do indicador relativo ao consumo de água *per capita* e o indicador de coleta de lixo urbano.

Figura 8 – Influência do indicador de coleta de lixo urbano



Fonte: Dados da pesquisa

5 Conclusão e Trabalhos Futuros

A dengue é uma doença viral que apresenta a situação epidemiológica mais alarmante do mundo(WHO, 2012). No continente americano, a maior parte dos casos de dengue está concentrada no Brasil(WHO, 2018).

A compreensão dos fatores que influenciam a transmissão do vírus pode contribuir para diminuir a incidência dessa doença. Diversos estudos já evidenciaram a influência que condições climáticas possuem na transmissão do vírus da dengue. Essa influência se deve ao fato de os mosquitos que são os vetores de transmissão precisarem de água parada para procriar e, também, de um ambiente quente para favorecer o desenvolvimento da larva e aumentar a velocidade de replicação do vírus(HALES et al., 2002).

A fim de controlar a proliferação da dengue, a Organização Mundial de Saúde recomenda a implantação de mecanismos que visem controlar a proliferação do vetor de transmissão da doença. Essas medidas incluem a monitoração de fatores de risco relacionados a condições sociais e econômicas(WHO, 2012).

Considerando o exposto, o presente trabalho analisou a relação entre a taxa de incidência de dengue e condições sociais e climáticas de municípios do Brasil. Foram obtidos com o Ministério da Saúde dados relativos a taxa de incidência de dengue para os anos de 2011, 2012 e 2013. Informações referentes às condições climáticas desse mesmo período foram obtidas através do Banco de Dados Meteorológicos para Ensino e Pesquisa (BDMEP/INMET). Por sua vez, dados relativos a condições sociais foram obtidos através do Sistema Nacional de Informações sobre Saneamento (SNIS/Ministério das Cidades) e através do Cadastro Nacional de Estabelecimentos de Saúde (CNES/Ministério da Saúde).

Através do BDMEP/INMET foi possível obter dados meteorológicos de 240 municípios do Brasil para o período analisado. Entretanto, com o intuito de reduzir o esforço de avaliação do modelo, foram analisadas apenas cidades com mais de 100 mil habitantes. Essa estratégia é semelhante à adotada por Albinati et al. (2017). Considerando, então, essa restrição adicional, foram analisados dados de 79 cidades do Brasil.

A metodologia proposta, apresentada no capítulo 3, buscou encontrar regras de inferência entre os indicadores analisados e a alta taxa de incidência de dengue. Para isso, os indicadores climáticos passaram, antes, por um processo de deslocamento ao longo do tempo. Conforme detalhado na seção 3.3, esse deslocamento teve o objetivo de simular o tempo necessário para que mudanças climáticas se refletissem na atividade do mosquito da dengue.

Na mineração de itens frequentes foi estabelecido o nível de suporte mínimo de 0.04, o que resultou em 8.612.163 itens frequentes. Destes, apenas 58.688 estavam relacionados a alta taxa de incidência de dengue. Por sua vez, na derivação de regras de associação foi estabelecido o nível mínimo de confiança de 0.3, o que resultou em 43.003 regras de

inferência associadas a alta taxa de incidência de dengue, isto é, regras que possuíam, no consequente, o indicador de epidemia de dengue.

A análise das regras encontradas, realizada na seção 4.4, permitiu realizar uma comparação entre regras envolvendo, no antecedente, apenas fatores climáticos e regras envolvendo fatores climáticos e sociais. Essa análise sugere que a taxa de incidência de dengue é influenciada pelo indicador de consumo de água *per capita* e pelo indicador relativo a quantidade de lixo urbano coletado por habitante do município.

O resultado encontrado está alinhado a recomendações da Organização Mundial de Saúde para prevenir a dengue. A OMS sugere, em WHO (2012), que a prevenção à dengue deve envolver a disposição regular do lixo e o controle sobre o consumo e armazenamento da água.

Uma outra contribuição desse trabalho é na adaptação da metodologia típica de mineração de dados para as peculiaridades do problema em questão. Em particular foram apresentados critérios e estratégias utilizados tanto na preparação e engenharia de dados, quanto na calibração dos modelos utilizados, sempre buscando a melhoria da qualidade desses modelos e a assertividade dos resultados.

Uma primeira adaptação foi a de realizar um procedimento de eliminação de *outliers* antes da etapa de clusterização dos indicadores. A eliminação de *outliers* foi necessária pois o algoritmo KMeans, utilizado no procedimento de clusterização é do tipo *hard clustering*, o que significa que ele atribui um grupo (e apenas um) a cada um dos pontos. Com isso, a presença de *outliers* pode afetar significativamente a formação geral dos grupos.

Por sua vez, como este trabalho envolveu a análise de uma quantidade significativa de indicadores climáticos e sociais, a execução do algoritmo de mineração de itens frequentes com todos esses indicadores se mostrou uma tarefa computacionalmente inviável. Sendo assim, a metodologia foi mais uma vez adaptada de modo a selecionar os indicadores através de uma etapa prévia de análise de correlação.

Por fim, uma adaptação final na metodologia foi realizada na etapa de derivação de regras de associação. A derivação de regras para todos os itens frequentes encontrados seria uma atividade muito onerosa, já que foram encontrados mais de 8 milhões de itens frequentes com os parâmetros adotados. Sendo assim, a derivação de regras de associação se limitou a encontrar regras que tivessem, no consequente, o indicador de *Alta* incidência de dengue. Essa estratégia permitiu significativo ganho de performance pois os itens frequentes que possuem o indicador de *Alta* incidência de dengue correspondem apenas a 0.68% do total de itens encontrados. Além disso, os itens frequentes foram armazenados em uma estrutura de dados chamada de dicionário[1]. A construção desse dicionário possui tempo de execução $O(n)$, mas o tempo para consulta é $O(1)$. Essa estratégia permitiu, mais uma vez, significativo ganho de performance na derivação de regras de associação.

[1] <https://docs.python.org/2/library/stdtypes.html#typesmapping>

Considerando o resultado obtido, sugere-se que trabalhos futuros aprimorem o estudo da influência de fatores socioeconômicos na taxa de incidência de dengue. Esses estudos poderão, no futuro, subsidiar a elaboração de políticas públicas voltadas ao controle epidemiológico através do controle de fatores sociais.

Referências

ALBINATI, J. et al. Enhancement of epidemiological models for dengue fever based on twitter data. In: *Proceedings of the 2017 International Conference on Digital Health*. New York, NY, USA: ACM, 2017. (DH '17), p. 109–118. ISBN 978-1-4503-5249-9. Disponível em: <http://doi.acm.org/10.1145/3079452.3079464>.

BANU, S. et al. Projecting the impact of climate change on dengue transmission in dhaka, bangladesh. *Environment International*, v. 63, p. 137 – 142, 2014. ISSN 0160-4120. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0160412013002626>.

CHOWELL, G. et al. The influence of geographic and climate factors on the timing of dengue epidemics in perú, 1994-2008. *BMC Infectious Diseases*, v. 11, n. 1, p. 164, 2011. Disponível em: <https://doi.org/10.1186/1471-2334-11-164>.

CIDADES, M. das. *Sistema Nacional de Informações sobre Saneamento*. 2018. Disponível em: <http://www.snis.gov.br/institucional-snis>.

DATASUS. *Cadastro Nacional de Estabelecimentos de Saúde*. 2018. Disponível em: <http://cnes.datasus.gov.br/pages/sobre/institucional.jsp>.

GUBLER, D. J.; CLARK, G. G. Community involvement in the control of aedes aegypti. *Acta Tropica*, v. 61, n. 2, p. 169 – 179, 1996. ISSN 0001-706X. Community participation in the control of tropical diseases. Disponível em: <http://www.sciencedirect.com/science/article/pii/0001706X9500103L>.

HALES, S. et al. Potential effect of population and climate changes on global distribution of dengue fever: an empirical model. *The Lancet*, v. 360, n. 9336, p. 830 – 834, 2002. ISSN 0140-6736. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0140673602099646>.

HAN MICHELINE KAMBER, J. P. J. *Data Mining Concepts ans Techniques*. [S.l.]: Elsevier, 2012.

HU, W. et al. Spatial patterns and socioecological drivers of dengue fever transmission in queensland, australia. *Environmental Health Perspectives*, National Institute of Environmental Health Sciences, v. 120, n. 2, p. 260–266, 02 2012. Disponível em: <http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3279430/>.

INMET. *Banco de Dados Meteorológicos para Ensino e Pesquisa*. 2018. Disponível em: <http://www.inmet.gov.br/projetos/rede/pesquisa/>.

LOWE, R. et al. Dengue outlook for the world cup in brazil: an early warning model framework driven by real-time seasonal climate forecasts. *The Lancet Infectious Diseases*, v. 14, n. 7, p. 619 – 626, 2014. ISSN 1473-3099. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1473309914707819>.

MS. *Descrição da Doença*. 2018. Disponível em: <http://portalms.saude.gov.br/saude-de-a-z/dengue/descricao-da-doenca>.

SOUZA, R. et al. Infection hot spot mining from social media trajectories. v. 9852, p. 739–755, 09 2016.

TAN MICHAEL STEINBACH, V. K. P.-N. *Introduction to Data Mining*. [S.l.]: Pearson, 2014.

WEBB, G. I. Discovering significant patterns. *Machine Learning*, v. 68, n. 1, p. 1–33, Jul 2007. ISSN 1573-0565. Disponível em: <https://doi.org/10.1007/s10994-007-5006-x>.

WHO. *Global strategy for dengue prevention and control 2012-2020*. [S.l.], 2012.

WHO. *Dengue and severe dengue*. 2018. Disponível em: <http://www.who.int/news-room/fact-sheets/detail/dengue-and-severe-dengue>.

ZAKI, M. J.; MEIRA, J. W. *Data Mining and Analysis: Fundamental Concepts and Algorithms*. [S.l.]: Cambridge University Press, 2014. ISBN 9780521766333.

ZHOU, G. et al. Association between climate variability and malaria epidemics in the east african highlands. *Proceedings of the National Academy of Sciences*, National Academy of Sciences, v. 101, n. 8, p. 2375–2380, 2004. ISSN 0027-8424. Disponível em: <http://www.pnas.org/content/101/8/2375>.

APÊNDICE A – Dicionário de Dados

A.1 SNIS - Água e Esgoto

Tabela 6 – Clusterização dos dados do SNIS - Água e Esgoto

Indicador	Clusters	Silhouette	Descrição
in001_ae	2	0.635189	Densidade de economias de água por ligação
in002_ae	3	0.589411	Índice de produtividade: economias ativas por pessoal próprio
in003_ae	2	0.583491	Despesa total com os serviços por m3 faturado
in004_ae	2	0.589935	Tarifa média praticada
in005_ae	2	0.594003	Tarifa média de água
in006_ae	5	0.568316	Tarifa média de esgoto
in007_ae	2	0.559527	Incidência da desp. de pessoal e de serv. de terc. nas despesas totais com os serviços
in008_ae	2	0.615151	Despesa média anual por empregado
in009_ae	2	0.811832	Índice de hidrometração
in010_ae	2	0.567500	Índice de micromedição relativo ao volume disponibilizado
in011_ae	2	0.883624	Índice de macromedição
in012_ae	2	0.541351	Indicador de desempenho financeiro
in013_ae	2	0.562648	Índice de perdas faturamento
in014_ae	2	0.584249	Consumo micromedido por economia
in015_ae	2	0.630395	Índice de coleta de esgoto

Continua na próxima página

Tabela6 – continuando da página anterior

Indicador	Clusters	Silhouette	Descrição
in016_ae	2	0.812945	Índice de tratamento de esgoto
in017_ae	2	0.570929	Consumo de água faturado por economia
in018_ae	2	0.684885	Quantidade equivalente de pessoal total
in019_ae	2	0.602272	Índice de produtividade: economias ativas por pessoal total
in020_ae	2	0.572882	Extensão da rede de água por ligação
in021_ae	2	0.615036	Extensão da rede de esgoto por ligação
in022_ae	2	0.600865	Consumo médio percapita de água
in023_ae	2	0.787235	Índice de atendimento urbano de água
in024_ae	2	0.700665	Índice de atendimento urbano de esgoto referido aos municípios atendidos com água
in025_ae	2	0.574638	Volume de água disponibilizado por economia
in026_ae	5	0.565256	Despesa de exploração por m3 faturado
in027_ae	2	0.582870	Despesa de exploração por economia
in028_ae	2	0.562648	Índice de faturamento de água
in029_ae	2	0.661398	Índice de evasão de receitas
in030_ae	2	0.566147	Margem da despesa de exploração
in031_ae	3	0.549557	Margem da despesa com pessoal próprio
in032_ae	2	0.564277	Margem da despesa com pessoal total

Continua na próxima página

Tabela6 – continuando da página anterior

Indicador	**Clusters**	**Silhouette**	**Descrição**
in033_ae	2	0.686926	Margem do serviço da divida
in034_ae	2	0.632866	Margem das outras despesas de exploração
in035_ae	3	0.559499	Participação da despesa com pessoal próprio nas despesas de exploração
in036_ae	4	0.559531	Participação da despesa com pessoal total
in037_ae	2	0.592887	Participação da despesa com energia elétrica nas despesas de exploração
in038_ae	2	0.615512	Participação da despesa com produtos químicos nas despesas de exploração
in039_ae	2	0.609025	Participação das outras despesas nas despesas de exploração
in040_ae	2	0.643883	Participação da receita operacional direta de água na receita operacional total
in041_ae	2	0.662985	Participação da receita operacional direta de esgoto na receita operacional total
in042_ae	2	0.614676	Participação da receita operacional indireta na receita operacional total
in043_ae	2	0.590453	Participação das economias residenciais de água no total das economias de água
in044_ae	2	0.798950	Índice de micromedição relativo ao consumo
in045_ae	2	0.625959	Índice de produtividade: empregados próprios por 1000 ligações de água

Continua na próxima página

Tabela6 – continuando da página anterior

Indicador	Clusters	Silhouette	Descrição
in046_ae	2	0.642471	Índice de esgoto tratado referido à água consumida
in047_ae	2	0.700665	Índice de atendimento urbano de esgoto referido aos municípios atendidos com esgoto
in048_ae	2	0.640062	Índice de produtividade: empregados próprios por 1000 ligações de água + esgoto
in049_ae	2	0.554216	Índice de perdas na distribuição
in050_ae	2	0.624359	Índice bruto de perdas lineares
in051_ae	3	0.572396	Índice de perdas por ligação
in052_ae	2	0.554216	Índice de consumo de água
in053_ae	2	0.600937	Consumo médio de água por economia
in054_ae	2	0.640346	Dias de faturamento comprometidos com contas a receber
in055_ae	2	0.687356	Índice de atendimento total de água
in056_ae	2	0.689924	Índice de atendimento total de esgoto referido aos municípios atendidos com água
in057_ae	2	0.919907	Índice de fluoretação de água
in058_ae	2	0.580441	Índice de consumo de energia elétrica em sistemas de abastecimento de água
in059_ae	2	0.657444	Índice de consumo de energia elétrica em sistemas de esgotamento sanitário

Continua na próxima página

Tabela6 – continuando da página anterior

Indicador	**Clusters**	**Silhouette**	**Descrição**
in060_ae	3	0.589004	Índice de despesas por consumo de energia elétrica nos sistemas de água e esgotos
in071_ae	2	0.747546	Economias atingidas por paralisações
in072_ae	2	0.552132	Duração média das paralisações
in073_ae	3	0.773267	Economias atingidas por intermitências
in074_ae	4	0.642200	Duração média das intermitências
in075_ae	2	0.781253	Incidência das análises de cloro residual fora do padrão
in076_ae	2	0.782463	Incidência das análises de turbidez fora do padrão
in077_ae	2	0.726692	Duração média dos reparos de extravasamentos de esgotos
in079_ae	2	0.644282	Índice de conformidade da quantidade de amostras - cloro residual
in080_ae	2	0.654925	Índice de conformidade da quantidade de amostras - turbidez
in082_ae	2	0.680667	Extravasamentos de esgotos por extensão de rede
in083_ae	2	0.786677	Duração média dos serviços executados
in084_ae	2	0.713660	Incidência das análises de coliformes totais fora do padrão
in085_ae	4	0.652476	Índice de conformidade da quantidade de amostras - coliformes totais

Continua na próxima página

Tabela6 – continuando da página anterior

Indicador	Clusters	Silhouette	Descrição
in101_ae	2	0.555782	Índice de suficiência de caixa
in102_ae	2	0.609876	Índice de produtividade de pessoal total

A.2 SNIS - Resíduos Sólidos

Tabela 7 – Clusterização dos dados do SNIS - Resíduos Sólidos

Indicador	Clusters	Silhouette	Descrição
in001_rs	3	0.581605	Taxa de empregados em relação à população urbana
in002_rs	2	0.587969	Despesa média por empregado alocado nos serviços do manejo de rsu
in003_rs	2	0.581058	Incidência das despesas com o manejo de rsu nas despesas correntes da prefeitura
in004_rs	2	0.795594	Incidência das despesas com empresas contratadas para execução de serviços de manejo rsu nas despesas com manejo de rsu
in005_rs	2	0.643783	Auto-suficiência financeira da prefeitura com o manejo de rsu
in006_rs	2	0.587495	Despesa per capita com manejo de rsu em relação à população urbana
in007_rs	2	0.782653	Incidência de empregados próprios no total de empregados no manejo de rsu

Continua na próxima página

Tabela7 – continuando da página anterior

Indicador	**Clusters**	**Silhouette**	**Descrição**
in008_rs	2	0.782654	Incidência de empregados de empresas contratadas no total de empregados no manejo de rsu
in010_rs	2	0.619405	Incidência de empregados gerenciais e administrativos no total de empregados no manejo de rsu
in011_rs	2	0.625312	Receita arrecadada per capita com taxas ou outras formas de cobrança pela prestação de serviços de manejo rsu
in014_rs	2	0.856425	Taxa de cobertura do serviço de coleta domiciliar direta
in015_rs	4	0.789094	Taxa de cobertura do serviço de coleta de rdo em relação à população total do município
in016_rs	2	0.998208	Taxa de cobertura do serviço de coleta de rdo em relação à população urbana
in017_rs	2	0.885937	Taxa de terceirização do serviço de coleta de
in018_rs	2	0.581662	Produtividade média dos empregados na coleta
in019_rs	2	0.565214	Taxa de empregados
in021_rs	2	0.563301	Massa coletada
in022_rs	5	0.559143	Massa
in023_rs	2	0.633948	Custo unitário médio do serviço de coleta
in024_rs	2	0.599200	Incidência do custo do serviço de coleta

Continua na próxima página

Tabela7 – continuando da página anterior

Indicador	Clusters	Silhouette	Descrição
in025_rs	2	0.586104	Incidência de coletadores + motoristas na quantidade total de empregados no manejo de rsu
in026_rs	2	0.646344	Taxa de resíduos sólidos da construção civil
in027_rs	2	0.716145	Taxa da quantidade total coletada de resíduos públicos
in028_rs	2	0.577422	Massa de resíduos domiciliares e públicos
in029_rs	2	0.691795	Massa de rcc per capita em relação à população urbana
in030_rs	2	0.743222	Taxa de cobertura do serviço de coleta seletiva porta-a-porta em relação à população urbana do município.
in031_rs	2	0.707030	Taxa de recuperação de materiais recicláveis
in032_rs	2	0.707980	Massa recuperada per capita de materiais recicláveis
in034_rs	2	0.578343	Incidência de papel e papelão no total de material recuperado
in035_rs	2	0.590837	Incidência de plásticos no total de material recuperado
in036_rs	3	0.648545	Massa de rss coletada per capita em relação à população urbana
in037_rs	3	0.619241	Taxa de rss coletada em relação à quantidade total coletada
in038_rs	2	0.579948	Incidência de metais no total de material recuperado

Continua na próxima página

Tabela7 – continuando da página anterior

Indicador	**Clusters**	**Silhouette**	**Descrição**
in039_rs	4	0.594521	Incidência de vidros no total de material recuperado
in040_rs	2	0.724900	Incidência de outros materiais
in041_rs	4	0.889643	Taxa de terceirização dos varredores
in043_rs	2	0.580968	Custo unitário médio do serviço de varrição
in044_rs	3	0.591405	Produtividade média dos varredores
in045_rs	2	0.667982	Taxa de varredores em relação à população urbana
in046_rs	2	0.620616	Incidência do custo do serviço de varrição no custo total com manejo de rsu
in047_rs	2	0.573374	Incidência de varredores no total de empregados no manejo de rsu
in048_rs	2	0.646008	Extensão total anual varrida per capita
in051_rs	2	0.619176	Taxa de capinadores em relação à população urbana
in052_rs	2	0.582865	Incidência de capinadores no total empregados no manejo de rsu
in053_rs	2	0.724647	Taxa de material recolhido pela coleta seletiva
in054_rs	2	0.729056	Massa per capita de materiais recicláveis recolhidos via coleta seletiva

A.3 BDMEP - Dados Meteorológicos

Tabela 8 – Clusterização dos dados do BDMEP

Indicador	Clusters	Silhouette	Descrição
sem0_precipitacao	2	0.711581	Precipitação(mm), sem deslocamento temporal
sem0_temp_maxima	2	0.568739	Temperatura máxima, sem deslocamento temporal
sem0_temp_minima	2	0.619666	Temperatura mínima, sem deslocamento temporal
sem0_temp_media	2	0.610054	Temperatura compensada média, sem deslocamento temporal
sem0_temp_amplitude	2	0.578369	Amplitude térmica, sem deslocamento temporal
sem0_insolacao	2	0.591479	Insolação (horas), sem deslocamento temporal
sem0_evaporacao	2	0.607561	Evaporação do piche (mm), sem deslocamento temporal
sem0_umidade	2	0.598457	Umidade relativa média, sem deslocamento temporal
sem0_vento	2	0.597475	Velocidade média do vento, sem deslocamento temporal
sem1_precipitacao	2	0.711619	Precipitação(mm), deslocamento temporal de 1 semana
sem1_temp_maxima	2	0.568885	Temperatura máxima, deslocamento temporal de 1 semana
sem1_temp_minima	2	0.619085	Temperatura mínima, deslocamento temporal de 1 semana
sem1_temp_media	2	0.609718	Temperatura compensada média, deslocamento temporal de 1 semana
sem1_temp_amplitude	2	0.578334	Amplitude térmica, deslocamento temporal de 1 semana

Continua na próxima página

Tabela8 – continuando da página anterior

Indicador	**Clusters**	**Silhouette**	**Descrição**
sem1_insolacao	2	0.591663	Insolação (horas), deslocamento temporal de 1 semana
sem1_evaporacao	2	0.607345	Evaporação do piche (mm), deslocamento temporal de 1 semana
sem1_umidade	2	0.597926	Umidade relativa média, deslocamento temporal de 1 semana
sem1_vento	2	0.597350	Velocidade média do vento, deslocamento temporal de 1 semana
sem2_precipitacao	2	0.710936	Precipitação(mm), deslocamento temporal de 2 semanas
sem2_temp_maxima	2	0.568987	Temperatura máxima, deslocamento temporal de 2 semanas
sem2_temp_minima	2	0.619433	Temperatura mínima, deslocamento temporal de 2 semanas
sem2_temp_media	2	0.610234	Temperatura compensada média, deslocamento temporal de 2 semanas
sem2_temp_amplitude	2	0.577818	Amplitude térmica, deslocamento temporal de 2 semanas
sem2_insolacao	2	0.592086	Insolação (horas), deslocamento temporal de 2 semanas
sem2_evaporacao	2	0.607144	Evaporação do piche (mm), deslocamento temporal de 2 semanas
sem2_umidade	2	0.598119	Umidade relativa média, deslocamento temporal de 2 semanas

Continua na próxima página

Tabela8 – continuando da página anterior

Indicador	Clusters	Silhouette	Descrição
sem2_vento	2	0.596928	Velocidade média do vento, deslocamento temporal de 2 semanas
sem3_precipitacao	2	0.710651	Precipitação(mm), deslocamento temporal de 3 semanas
sem3_temp_maxima	2	0.569011	Temperatura máxima, deslocamento temporal de 3 semanas
sem3_temp_minima	2	0.620111	Temperatura mínima, deslocamento temporal de 3 semanas
sem3_temp_media	2	0.610671	Temperatura compensada média, deslocamento temporal de 3 semanas
sem3_temp_amplitude	2	0.577347	Amplitude térmica, deslocamento temporal de 3 semanas
sem3_insolacao	2	0.591790	Insolação (horas), deslocamento temporal de 3 semanas
sem3_evaporacao	2	0.607357	Evaporação do piche (mm), deslocamento temporal de 3 semanas
sem3_umidade	2	0.598907	Umidade relativa média, deslocamento temporal de 3 semanas
sem3_vento	2	0.596845	Velocidade média do vento, deslocamento temporal de 3 semanas
sem4_precipitacao	2	0.710584	Precipitação(mm), deslocamento temporal de 4 semanas
sem4_temp_maxima	2	0.570064	Temperatura máxima, deslocamento temporal de 4 semanas

Continua na próxima página

Tabela8 – continuando da página anterior

Indicador	Clusters	Silhouette	Descrição
sem4_temp_minima	2	0.619869	Temperatura mínima, deslocamento temporal de 4 semanas
sem4_temp_media	2	0.610935	Temperatura compensada média, deslocamento temporal de 4 semanas
sem4_temp_amplitude	2	0.577612	Amplitude térmica, deslocamento temporal de 4 semanas
sem4_insolacao	2	0.591729	Insolação (horas), deslocamento temporal de 4 semanas
sem4_evaporacao	2	0.607750	Evaporação do piche (mm), deslocamento temporal de 4 semanas
sem4_umidade	2	0.599190	Umidade relativa média, deslocamento temporal de 4 semanas
sem4_vento	2	0.596483	Velocidade média do vento, deslocamento temporal de 4 semanas

A.4 CNES - Estabelecimentos de Saúde

Tabela 9 – Clusterização dos dados do CNES

Indicador	Clusters	Silhouette	Descrição
central_regualacao	4	0.904565	Central de Regualacao
central_regulacao_urgencias	4	0.795013	Central de Regulacao Medica das Urgencias
centro_hemoterapico	3	0.971466	Centro de Atencao Hemoterapica ou Hematologica
centro_psicossocial	2	0.640477	Centro de Atencao Psicossocial-CAPS

Continua na próxima página

Tabela9 – continuando da página anterior

Indicador	Clusters	Silhouette	Descrição
centro_saude	2	0.599604	Centro de Saude/Unidade Basica de Saude
central_regulacao	2	0.859137	Central de Regulacao de Servicos de Saude
clinica_especializada	2	0.638612	Clinica Especializada/Ambulatorio Especializaco
consultorio	2	0.664597	Consultorio
farmacia	3	0.706840	Farmacia
hospital_especializado	5	0.747999	Hospital Especializado
hospital_geral	2	0.587886	Hospital Geral
hospital_dia	3	0.855530	Hospital Dia
policlinica	2	0.634849	Policlinica
posto_saude	2	0.807965	Posto de Saude
pronto_atendimento	4	0.782557	Pronto Atendimento
ups_geral	3	0.837373	Pronto Socorro Geral
secretaria_saude	3	0.589634	Secretaria de Saude
unidade_diagnose	2	0.612361	Unidade de Servico de Apoio de Diagnose e Terapia
unidade_vigilancia	4	0.713626	Unidade de Vigilancia em Saude
unidade_movel_urgencia	2	0.669382	Unidade movel de nivel pre-hosp-Urgencia/Emergencia
unidade_movel_terrestre	4	0.748940	Unidade Movel Terrestre

APÊNDICE B – Códigos-Fonte

B.1 Cálculo de Correlação

```
1  # -*- coding: utf-8 -*-
2
3  import numpy
4  import time
5  import sys
6  import pandas as pd
7  import scipy.stats as stats
8
9  from mlxtend.preprocessing import minmax_scaling
10
11
12 tabela = "indicadores_sociais"
13
14 #tab_clima = "bdmed_completo_acima-de-100mil"
15 tab_clima = "bdmed-sem-0-20_acima-de-100mil"
16
17 arquivo_entrada = "./dados/" + tabela + ".csv"
18 arq_entrada_clima = "./dados/bdmed/" + tab_clima + ".csv"
19 arq_saida_kmeans = "./saida/saida_kmeans_" + tabela + "_" + time
     .strftime("%Y%m%d-%Hh%Mm") + ".csv"
20 arq_saida_apriori = "./saida/saida_apriori_" + tabela + "_" +
     time.strftime("%Y%m%d-%Hh%Mm") + ".csv"
21 arquivo_log = "./log/log_apriori_" + tabela + "_" + time.
     strftime("%Y%m%d-%Hh%Mm") + ".txt"
22
23
24 def normaliza_dataset(dataset):
25     qtde = len(dataset[0,:])
26     dataset_normalizado = numpy.full([qtde], numpy.NaN)
27     qtde_cols = len(dataset)
28     var = 0
29     while (var < qtde_cols):
30         sys.stdout.write("\r{0:s}: Normalizando dataset (
```

```
indicador␣{1:d}␣de␣{2:d}␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣".format(time.strftime("%Y-%m-%d␣%H:%M:%S"), var+1, qtde_cols))
        indicador = dataset[var,:]
        inds_nan = numpy.where(numpy.isnan(indicador))[0]

        # cria uma nova lista, sem os valores nulos
        indicador_limpo = numpy.delete(indicador, inds_nan, axis=0)
        if ( len(indicador_limpo) > 0 ):
            indicador_normalizado = minmax_scaling(indicador_limpo, columns=[0])[:,0]
        else:
            indicador_normalizado = indicador

        indicador_saida = numpy.full(qtde, 100.0, dtype=float)
        indicador_saida[inds_nan] = numpy.NaN
        inds_not_null = numpy.where(numpy.isfinite(indicador_saida))[0]
        numpy.put(indicador_saida, inds_not_null, indicador_normalizado)

        dataset_normalizado = numpy.vstack((dataset_normalizado, indicador_saida))

        var += 1
    dataset_normalizado = numpy.delete(dataset_normalizado, 0, 0)

    return dataset_normalizado



def filter_outlier_statistically(dados):
    q1 = numpy.nanpercentile(dados, 25, axis=0)
    q3 = numpy.nanpercentile(dados, 75, axis=0)
    iqr = q3-q1
    limite_inferior = q1 - 1.5*iqr
```

```
60      limite_superior = q3 + 1.5*iqr
61      filtro = numpy.where(numpy.logical_and(dados>=
            limite_inferior, dados<=limite_superior), True, False)
62      return filtro
63
64
65
66  def calcula_correlacao(casos_provaveis, dataset, dataset_header)
        :
67      pearson = numpy.array([], numpy.dtype([('correlacao', numpy.
            float64, (2)), ('pvalue', numpy.float64, (2)), ('header',
             'U50'), ('selecionado', '?')]))
68      for idx, indicador in enumerate(dataset):
69          sys.stdout.write("\r{0:s}: Correlacionando indicador {1:
                d} de {2:d}                                        
                ".format(time.strftime("%Y-%m-%d %H:%M:%S"),
                idx+1, len(dataset)))
70          # Eliminando valores nulos do indicador
71          filtro = ~numpy.isnan(indicador)
72          x = numpy.compress(filtro, indicador)
73          y1 = numpy.compress(filtro, casos_provaveis[0])
74          y2 = numpy.compress(filtro, casos_provaveis[1])
75          # Eliminando outliers por metodo estatistico
76          filtro = filter_outlier_statistically(x)
77          x_limpo = x[filtro]
78          y1_limpo = y1[filtro]
79          y2_limpo = y2[filtro]
80
81          corr1, pvalue1 = stats.pearsonr(x_limpo, y1_limpo)
82          corr2, pvalue2 = stats.pearsonr(x_limpo, y2_limpo)
83
84          aux = numpy.array(((corr1, corr2), (pvalue1, pvalue2))+(
                dataset_header[idx],False), numpy.dtype([('correlacao
                ', numpy.float64, (2)), ('pvalue', numpy.float64, (2)
                ), ('header', 'U50'), ('selecionado', '?')]))
85
86          pearson = numpy.hstack((pearson, aux))
87      return pearson
88
```

```
89
90 #correlacao_pearson['selecionado'][numpy.where(numpy.array(list(
       map(lambda x: max(abs(x)), correlacao_pearson['correlacao']))
       )>=0.1)] = True
91
92 #teste = numpy.where(numpy.array(list(map(lambda x: max(abs(x)),
       correlacao_pearson['correlacao']))))>=0.1, True, False)
93
94 with open(arquivo_log, 'w', buffering=1) as arq_log:
95     j = 0
96     tempo = time.time()
97
98     sys.stdout.write("\n{0:s}: Iniciando Execucao...".format(
           time.strftime("%Y-%m-%d %H:%M:%S")))
99 #     print (str_cabecalho)
100
101     # carrega dicionario de dados
102 #     dicionario = numpy.genfromtxt(arquivo_dicionario, delimiter
       =",", dtype=(int, "|U10", "|U30", "|U50", "|U30", int, int,
       int, bool, "|S110",), names=True, unpack=True)
103
104     sys.stdout.write("\n{0:s}: Carregando dados de entrada...".
           format(time.strftime("%Y-%m-%d %H:%M:%S")))
105
106     # carrega o dataset
107     dataset = numpy.genfromtxt(arquivo_entrada, dtype='str',
           delimiter=",", skip_header=1, unpack=True)
108     dataset_clima = numpy.genfromtxt(arq_entrada_clima, dtype='
           str', delimiter=",", skip_header=1, unpack=True)
109
110     # carrega o cabecalho do arquivo
111     header_entrada = numpy.genfromtxt(arquivo_entrada, dtype='
           str', delimiter=",", skip_header=0, deletechars='\"',
           unpack=True, max_rows=1)
112     dataset_header = header_entrada.tolist()
113     header_entrada_clima = numpy.genfromtxt(arq_entrada_clima,
           dtype='str', delimiter=",", skip_header=0, deletechars='
           \"', unpack=True, max_rows=1)
114     dataset_clima_header = header_entrada_clima.tolist()
```

```
115
116
117         dataset_float = numpy.apply_along_axis(lambda linha: pd.
                to_numeric(linha, errors='coerce'), 1, dataset[2:,:])
118         dataset_float_header = dataset_header[2:]
119
120         dataset_clima_float = numpy.apply_along_axis(lambda linha:
                pd.to_numeric(linha, errors='coerce'), 1, dataset_clima
                [8:,:])
121         dataset_clima_float_header = dataset_clima_header[8:]
122
123         sys.stdout.write("\n{0:s}: Iniciando calculo de correlacoes
                sem normalizacao...".format(time.strftime("%Y-%m-%d %H:%M
                :%S")))
124
125 #       dataset_corr_casos_provaveis = pearson_correlation(
        dataset_float[2], dataset_float, dataset_float_header)
126         correlacao_pearson = calcula_correlacao(dataset_float[3:5],
                dataset_float, dataset_float_header)
127
128         taxa_incidencia_semanal = numpy.vstack((dataset_clima[5].
                astype(float), dataset_clima[5].astype(float)))
129         correlacao_clima_pearson = calcula_correlacao(
                taxa_incidencia_semanal, dataset_clima_float,
                dataset_clima_float_header)
130 #       semanas_epidemia_pearson = calcula_correlacao(dataset_float
        [4], dataset_float, dataset_float_header)
131
132
133         sys.stdout.write("\n{0:s}: Execucao Finalizada ...
                ------------------------\n".format(time.strftime("%Y-%m-%
                d %H:%M:%S")))
```

B.2 *Clusterização*

```
1 # -*- coding: utf-8 -*-
2
3 import numpy
4 import time
5 import sys
```

```
6  import pandas as pd
7
8  from sklearn.cluster import KMeans
9  from sklearn.metrics import silhouette_score
10
11 tabela = "indicadores_sociais"
12 tab_clima = "bdmed_completo_acima-de-100mil"
13
14 arquivo_entrada = "./dados/" + tabela + ".csv"
15 arq_entrada_clima = "./dados/" + tab_clima + ".csv"
16 arquivo_log = "./log/log_kmeans_" + tabela + "_" + time.strftime
       ("%Y%m%d-%Hh%Mm") + ".txt"
17
18
19 def filter_outlier_statistically(dados):
20     q1 = numpy.nanpercentile(dados, 25, axis=0)
21     q3 = numpy.nanpercentile(dados, 75, axis=0)
22     iqr = q3-q1
23     limite_inferior = q1 - 1.5*iqr
24     limite_superior = q3 + 1.5*iqr
25
26     filtro_nan = numpy.isnan(dados)
27
28     filtro_lim_inicial = numpy.greater_equal(dados,
           limite_inferior, where=~filtro_nan)
29     filtro_lim_final = numpy.less_equal(dados, limite_superior,
           where=~filtro_nan)
30     filtro = filtro_lim_inicial & filtro_lim_final
31
32     return filtro
33
34
35 def discretiza_kmeans(dataset, dataset_header, arq_log):
36
37     dataset_float = numpy.apply_along_axis(lambda linha: pd.
           to_numeric(linha, errors='coerce'), 0, dataset)
38
39     str_cabecalho = ("Hora, Etapa, Variavel, k, Silhouette,
           Grupo, Min, Max, Qtde\n")
```

```
40      arq_log.write(str_cabecalho)
41
42      # Vamos percorrer todas as colunas do arquivo de entrada
43      str_cabecalho = None
44      inds_nan = numpy.full((len(dataset_float.T)), False, dtype='
            bool')
45      for idx, header in enumerate(dataset_header):
46
47          indicador = None
48          qtde_total = None
49          if ( str_cabecalho == None ):
50              str_cabecalho = header
51          else:
52              str_cabecalho += '-' + header
53
54          # Leitura do indicador a ser discretizado
55          if ( len(dataset_header) > 1 ):
56              indicador = dataset_float[idx,:]
57          else:
58              indicador = dataset_float
59
60  #          sys.stdout.write("\r{0:s}: Iniciando analise da coluna
        {1:d} de {2:d} ({3:s})
        ".format(time.strftime("%Y-%m-%d %H:%M:%S"), idx+1, len(
        dataset_header), dataset_header[idx]))
61
62          qtde_total = len(indicador)
63
64          # Eliminando outliers por metodo estatistico
65          filtro = filter_outlier_statistically(indicador)
66          qtde_outliers = len(numpy.where(~filtro)[0])
67          indicador[~filtro] = numpy.NaN
68          linha_log = ("{0:s}, KMeans-Outliers, {1:s}, {2:d}
                Outliers ({3:f}%),,,,,\n".format(time.strftime("%Y-%m
                -%d %H:%M:%S"),dataset_header[idx],qtde_outliers,
                qtde_outliers/qtde_total))
69          arq_log.write(linha_log)
70
71          # pega o indice dos valores nulos do indicador
```

```
72          inds_nan = numpy.logical_or(inds_nan, numpy.where(numpy.
                isnan(indicador), True, False))
73
74      if numpy.all(inds_nan):
75          # Todos os valores da variavel sao nulos
76          linha_log = ("Variavel: {0:s} contem apenas valores
                nulos.\n".format(str_cabecalho))
77          print (linha_log)
78          arq_log.write(linha_log)
79      else:
80          # cria uma nova lista, sem os valores nulos
81          if ( len(dataset_header) > 1 ):
82              indicador_limpo = numpy.compress(~inds_nan,
                    dataset_float, axis=1)
83          else:
84              indicador_limpo = numpy.compress(~inds_nan,
                    dataset_float, axis=0)
85
86          # Loop atraves das variaveis do arquivo csv,
                discretizando cada uma com o k-means
87          k = 2
88          silhouette_k = numpy.empty((0,2))
89          while k < 6:
90
91  #              sys.stdout.write("\r{0:s}: Coluna {1:s} - k={2:d}
                        ".format(time.strftime("%Y-%m-%d %H:%M:%S
    "), str_cabecalho, k))
92
93              grupos_limpo=kmeans_limpo=silhouette_media_limpo=
                    None
94              linha_log = None
95
96              # Execucao do kmeans
97              if ( len(dataset_header) > 1 ):
98                  kmeans_limpo = KMeans(n_clusters=k, random_state
                        =0).fit(indicador_limpo.T)
99              else:
100                 kmeans_limpo = KMeans(n_clusters=k, random_state
                        =0).fit(indicador_limpo.reshape(-1,1))
```

```
101
102             # Pegando o label dos grupos criados e os dados dos
                    Centroids
103             grupos_limpo = kmeans_limpo.labels_
104 #           centroids_limpo = kmeans_limpo.cluster_centers_
105
106             # Calculando o coeficiente de Silhouette
107 #             sys.stdout.write("\r{0:s}: Coluna {1:s} - k={2:d}
    Calculando Silhouette".format(time.strftime("%Y-%m-%d %H:%M:%
    S"), str_cabecalho, k))
108             if ( len(dataset_header) > 1 ):
109                 silhouette_media_limpo = silhouette_score(
                        indicador_limpo.T, grupos_limpo)
110             else:
111                 silhouette_media_limpo = silhouette_score(
                        indicador_limpo.reshape(-1,1), grupos_limpo)
112
113                 # Lista que armazena os coeficientes de
                        Silhouette para diferentes valores de k
114 #             sys.stdout.write("\r{0:s}: Coluna {1:s} - k={2:d}
    Atualizando lista com coeficientes".format(time.strftime("%Y
    -%m-%d %H:%M:%S"), str_cabecalho, k))
115             # Pegando o indice para inserir o valor de
                    silhouette em ordem no array, ja contando com uma
                     margem de 1%. Isto e, para escolher o melhor
                    valor de silhouette, aplicaremos um desconto de
                    1%. Caso um agrupamento x possua um silhouette
                    menor que um agrupamento y mas dentro de uma
                    margem de 1%, o que sera escolhido sera o que
                    possuir o menor valor de k.
116             if ( len(silhouette_k) == 0 ):
117                 k_idx = 0
118             else:
119                 idx_menores = numpy.where(silhouette_k[:,1] <
                        silhouette_media_limpo*0.99)[0]
120                 if ( len(idx_menores) == 0 ):
121                     k_idx = len(silhouette_k)
122                 else:
123                     k_idx = min(idx_menores)
```

```
124             silhouette_k = numpy.insert(silhouette_k, k_idx, [[k
                    , silhouette_media_limpo]], axis=0)
125
126             i = 0
127             while i <= max(grupos_limpo):
128                 inds_grupo = numpy.where(grupos_limpo == i)[0]
129                 #inds_grupo_min = min(indicador_limpo[inds_grupo
                        ])
130                 #inds_grupo_max = max(indicador_limpo[inds_grupo
                        ])
131                 inds_grupo_qtde = len(inds_grupo)
132                 linha_log = ("{0:s}, KMeans, {1:s}, {2:d}, {3:f
                        }, {4:d}, {5:f}\n".format(time.strftime("%Y-%
                        m-%d %H:%M:%S"),str_cabecalho,k,
                        silhouette_media_limpo, i, inds_grupo_qtde))
133                 arq_log.write(linha_log)
134                 i += 1
135
136             k += 1
137
138         # Recalculando os clusters com o melhor valor para k
139         melhor_k = silhouette_k[0]
140         if ( len(dataset_header) > 1 ):
141             kmeans_limpo = KMeans(n_clusters=melhor_k[0].astype(
                    int), random_state=0).fit(indicador_limpo.T)
142         else:
143             kmeans_limpo = KMeans(n_clusters=melhor_k[0].astype(
                    int), random_state=0).fit(indicador_limpo.reshape
                    (-1,1))
144         grupos_limpo = kmeans_limpo.labels_
145
146         linha_log = ("{0:s}, KMeans-Final, {1:s}, {2:d}, {3:f
                }, , , , \n".format(time.strftime("%Y-%m-%d %H:%M:%S"),
                str_cabecalho,melhor_k[0].astype(int), melhor_k[1].
                astype(float)))
147         arq_log.write(linha_log)
148
149         # Criando o array final com dados de saida dos grupos
                criados pelo KMeans
```

```
150         grupos_final = numpy.full(qtde_total, 100.0, dtype=float
                )
151         # Valores nulos serao inseridos nas posicoes aonde,
                originalmente, haviam nulos
152         grupos_final[inds_nan] = numpy.NaN
153         # Pegando, agora, o indice dos valores nao nulos, isto e
                , dos valores para os quais o KMeans atribuiu grupo
154         inds_not_null = numpy.where(numpy.isfinite(grupos_final)
                )[0]
155         # Substituindo no array os valores dos indices acima
                pelos valores dos grupos criados pelo KMeans
156         numpy.put(grupos_final, inds_not_null, grupos_limpo)
157
158         grupos_final_cluster = list(map(lambda x: (('NaN') if (
                numpy.isnan(x)) else (numpy.around(x, decimals=0).
                astype(str) + '-' + str_cabecalho)), grupos_final))
159
160     return grupos_final_cluster
161
162
163 def discretiza_matriz_individual(dataset, dataset_header,
        arq_log):
164     cluster_final = numpy.empty((0,len(dataset.T)), dtype='float
            ')
165     for idx, header in enumerate(dataset_header):
166         sys.stdout.write("\r{0:s}: Iniciando analise da coluna
                {1:d} de {2:d} ({3:s})                                
                        ".format(time.strftime("%Y-%m-%d %H:%M:%S"),
                idx+1, len(dataset_header), dataset_header[idx]))
167
168         cluster_indicador = discretiza_kmeans(dataset[idx,:], [
                header], arq_log)
169
170         cluster_final = numpy.vstack((cluster_final,
                cluster_indicador))
171     return cluster_final
172
173
174 def discretiza_populacao(dados):
```

```
175     pop_disc = numpy.empty((0), dtype='str')
176     for pop in dados:
177         if ( pop.astype(float) < 100000 ):
178             pop_disc = numpy.append(pop_disc, 'ate-100mil')
179             continue
180         if ( pop.astype(float) < 500000 ):
181             pop_disc = numpy.append(pop_disc, 'de-100-a-499mil')
182             continue
183         if ( pop.astype(float) < 1000000 ):
184             pop_disc = numpy.append(pop_disc, 'de-500-a-999mil')
185             continue
186         pop_disc = numpy.append(pop_disc, 'acima-de-1milhao')
187     return pop_disc
188
189
190 def analisa_percentual(dados):
191     analise = numpy.zeros((60), dtype='float')
192     filtro_nan = numpy.isnan(dados)
193     i = 0
194     while ( i <= 50 ):
195         limite_final = i
196         if ( i == 0 ):
197             limite_inicial = -50
198         else:
199             limite_inicial = i-1
200
201         filtro_lim_inicial = numpy.greater(dados, limite_inicial
                , where=~filtro_nan)
202         filtro_lim_final = numpy.less_equal(dados, limite_final,
                where=~filtro_nan)
203         filtro = filtro_lim_inicial & filtro_lim_final
204
205         analise[i] = len(numpy.where(filtro)[0])/len(dados)
206         i += 1
207
208     return analise
209
210
211 def analisa_quantis(dados, k):
```

```
212     analise = numpy.empty((k*4), dtype='float')
213     filtro_nan = numpy.isnan(dados)
214     i = 1
215     while ( i <= k):
216         limite_inicial = numpy.nanpercentile(dados, (100/k)*(i
                -1), axis=0)
217         limite_final = numpy.nanpercentile(dados, (100/k)*i,
                axis=0)
218
219         filtro_lim_inicial = numpy.greater_equal(dados,
                limite_inicial, where=~filtro_nan)
220         filtro_lim_final = numpy.less_equal(dados, limite_final,
                 where=~filtro_nan)
221         filtro = filtro_lim_inicial & filtro_lim_final
222
223         analise[(i-1)*4] = limite_inicial
224         analise[(i-1)*4+1] = limite_final
225         analise[(i-1)*4+2] = len(numpy.where(filtro)[0])
226         analise[(i-1)*4+3] = len(numpy.where(filtro)[0])/len(
                dados)
227
228         i += 1
229     return analise
230
231
232 def discretiza_quantis(dados, dados_header, k):
233     dados_cluster = numpy.empty((len(dados),len(dados.T)), dtype
            ='U50')
234     for idx, indicador in enumerate(dados_header):
235         filtro_nan = numpy.isnan(dados[idx])
236         dados_cluster[idx][filtro_nan] = 'NaN'
237         i = 1
238         while ( i <= k ):
239             limite_inicial = numpy.nanpercentile(dados[idx],
                    (100/k)*(i-1), axis=0)
240             limite_final = numpy.nanpercentile(dados[idx], (100/
                    k)*i, axis=0)
241
242
```

```
243             filtro_lim_inicial = numpy.greater_equal(dados[idx],
                    limite_inicial, where=~filtro_nan)
244             filtro_lim_final = numpy.less_equal(dados[idx],
                   limite_final, where=~filtro_nan)
245
246             filtro = filtro_lim_inicial & filtro_lim_final
247             dados_cluster[idx][filtro] = dados_header[idx] + '_q
                    ' + str(i)
248             i += 1
249     return dados_cluster
250
251
252
253 def discretiza_temp_minima_media(dataset_clima_float,
        dataset_clima_float_header, arq_log):
254
255     sys.stdout.write("\r{0:s}: Iniciando analise temp_min_media
            da Semana 0 de 4 ".
            format(time.strftime("%Y-%m-%d %H:%M:%S")))
256     cluster_sem0 = discretiza_kmeans(dataset_clima_float[2:4],
            dataset_clima_float_header[2:4], arq_log)
257     sys.stdout.write("\r{0:s}: Iniciando analise temp_min_media
            da Semana 1 de 4 ".
            format(time.strftime("%Y-%m-%d %H:%M:%S")))
258     cluster_sem1 = discretiza_kmeans(dataset_clima_float[11:13],
             dataset_clima_float_header[11:13], arq_log)
259     sys.stdout.write("\r{0:s}: Iniciando analise temp_min_media
            da Semana 2 de 4 ".
            format(time.strftime("%Y-%m-%d %H:%M:%S")))
260     cluster_sem2 = discretiza_kmeans(dataset_clima_float[20:22],
             dataset_clima_float_header[20:22], arq_log)
261     sys.stdout.write("\r{0:s}: Iniciando analise temp_min_media
            da Semana 3 de 4 ".
            format(time.strftime("%Y-%m-%d %H:%M:%S")))
262     cluster_sem3 = discretiza_kmeans(dataset_clima_float[29:31],
             dataset_clima_float_header[29:31], arq_log)
263     sys.stdout.write("\r{0:s}: Iniciando analise temp_min_media
            da Semana 4 de 4 ".
            format(time.strftime("%Y-%m-%d %H:%M:%S")))
```

```
264     cluster_sem4 = discretiza_kmeans(dataset_clima_float[38:40],
            dataset_clima_float_header[38:40], arq_log)
265
266     cluster_final_dados = numpy.vstack((cluster_sem0,
            cluster_sem1))
267     cluster_final_dados = numpy.vstack((cluster_final_dados,
            cluster_sem2))
268     cluster_final_dados = numpy.vstack((cluster_final_dados,
            cluster_sem3))
269     cluster_final_dados = numpy.vstack((cluster_final_dados,
            cluster_sem4))
270
271     cluster_final_header = list(['sem0-temp_min_media', 'sem1-
            temp_min_media', 'sem2-temp_min_media', 'sem3-
            temp_min_media', 'sem4-temp_min_media'])
272
273     cluster_final = numpy.vstack((cluster_final_header,
            cluster_final_dados.T))
274
275     return cluster_final
276
277
278
279 def discretiza_temp_minima_amp(dataset_clima_float,
        dataset_clima_float_header, arq_log):
280
281     sys.stdout.write("\r{0:s}: Iniciando analise temp_min_amp da
            Semana 0 de 4                                        ".
            format(time.strftime("%Y-%m-%d %H:%M:%S")))
282     indicador = numpy.vstack((dataset_clima_float[2],
            dataset_clima_float[4]))
283     indicador_header = [dataset_clima_float_header[2],
            dataset_clima_float_header[4]]
284     cluster_sem0 = discretiza_kmeans(indicador, indicador_header
            , arq_log)
285
286     sys.stdout.write("\r{0:s}: Iniciando analise temp_min_amp da
            Semana 1 de 4                                        ".
            format(time.strftime("%Y-%m-%d %H:%M:%S")))
```

```
287     indicador = numpy.vstack((dataset_clima_float[11],
            dataset_clima_float[13]))
288     indicador_header = [dataset_clima_float_header[11],
            dataset_clima_float_header[13]]
289     cluster_sem1 = discretiza_kmeans(indicador, indicador_header
            , arq_log)
290
291     sys.stdout.write("\r{0:s}: Iniciando analise temp_min_amp da
            Semana 2 de 4                                        ".
            format(time.strftime("%Y-%m-%d %H:%M:%S")))
292     indicador = numpy.vstack((dataset_clima_float[20],
            dataset_clima_float[22]))
293     indicador_header = [dataset_clima_float_header[20],
            dataset_clima_float_header[22]]
294     cluster_sem2 = discretiza_kmeans(indicador, indicador_header
            , arq_log)
295
296     sys.stdout.write("\r{0:s}: Iniciando analise temp_min_amp da
            Semana 3 de 4                                        ".
            format(time.strftime("%Y-%m-%d %H:%M:%S")))
297     indicador = numpy.vstack((dataset_clima_float[29],
            dataset_clima_float[31]))
298     indicador_header = [dataset_clima_float_header[29],
            dataset_clima_float_header[31]]
299     cluster_sem3 = discretiza_kmeans(indicador, indicador_header
            , arq_log)
300
301     sys.stdout.write("\r{0:s}: Iniciando analise temp_min_amp da
            Semana 4 de 4                                        ".
            format(time.strftime("%Y-%m-%d %H:%M:%S")))
302     indicador = numpy.vstack((dataset_clima_float[38],
            dataset_clima_float[40]))
303     indicador_header = [dataset_clima_float_header[38],
            dataset_clima_float_header[40]]
304     cluster_sem4 = discretiza_kmeans(indicador, indicador_header
            , arq_log)
305
306     cluster_final_dados = numpy.vstack((cluster_sem0,
            cluster_sem1))
```

```
307     cluster_final_dados = numpy.vstack((cluster_final_dados,
            cluster_sem2))
308     cluster_final_dados = numpy.vstack((cluster_final_dados,
            cluster_sem3))
309     cluster_final_dados = numpy.vstack((cluster_final_dados,
            cluster_sem4))
310
311     cluster_final_header = list(['sem0-temp_min_amp', 'sem1-
            temp_min_amp', 'sem2-temp_min_amp', 'sem3-temp_min_amp',
            'sem4-temp_min_amp'])
312
313     cluster_final = numpy.vstack((cluster_final_header,
            cluster_final_dados.T))
314
315     return cluster_final
316
317
318
319 def discretiza_temp_minima_media_amp(dataset_clima_float,
        dataset_clima_float_header, arq_log):
320
321     sys.stdout.write("\r{0:s}: Iniciando analise
            temp_min_media_amp da Semana 0 de 4
            ".format(time.strftime("%Y-%m-%d %H:%M:%S
            ")))
322     cluster_sem0 = discretiza_kmeans(dataset_clima_float[2:5],
            dataset_clima_float_header[2:5], arq_log)
323     sys.stdout.write("\r{0:s}: Iniciando analise
            temp_min_media_amp da Semana 1 de 4
            ".format(time.strftime("%Y-%m-%d %H:%M:%S
            ")))
324     cluster_sem1 = discretiza_kmeans(dataset_clima_float[11:14],
             dataset_clima_float_header[11:14], arq_log)
325     sys.stdout.write("\r{0:s}: Iniciando analise
            temp_min_media_amp da Semana 2 de 4
            ".format(time.strftime("%Y-%m-%d %H:%M:%S
            ")))
326     cluster_sem2 = discretiza_kmeans(dataset_clima_float[20:23],
             dataset_clima_float_header[20:23], arq_log)
```

```
327     sys.stdout.write("\r{0:s}: Iniciando analise
            temp_min_media_amp da Semana 3 de 4
            ".format(time.strftime("%Y-%m-%d %H:%M:%S
            ")))
328     cluster_sem3 = discretiza_kmeans(dataset_clima_float[29:32],
             dataset_clima_float_header[29:32], arq_log)
329     sys.stdout.write("\r{0:s}: Iniciando analise
            temp_min_media_amp da Semana 4 de 4
            ".format(time.strftime("%Y-%m-%d %H:%M:%S
            ")))
330     cluster_sem4 = discretiza_kmeans(dataset_clima_float[38:41],
             dataset_clima_float_header[38:41], arq_log)
331
332     cluster_final_dados = numpy.vstack((cluster_sem0,
            cluster_sem1))
333     cluster_final_dados = numpy.vstack((cluster_final_dados,
            cluster_sem2))
334     cluster_final_dados = numpy.vstack((cluster_final_dados,
            cluster_sem3))
335     cluster_final_dados = numpy.vstack((cluster_final_dados,
            cluster_sem4))
336
337     cluster_final_header = list(['sem0-temp_min_media_amp', '
            sem1-temp_min_media_amp', 'sem2-temp_min_media_amp', '
            sem3-temp_min_media_amp', 'sem4-temp_min_media_amp'])
338
339     cluster_final = numpy.vstack((cluster_final_header,
            cluster_final_dados.T))
340
341     return cluster_final
342
343
344
345
346 with open(arquivo_log, 'w', buffering=1) as arq_log:
347     j = 0
348     tempo = time.time()
349
350     sys.stdout.write("\n{0:s}: Iniciando Execucao ...".format(
```

```
        time.strftime("%Y-%m-%d %H:%M:%S")))
351
352     sys.stdout.write("\n{0:s}: Carregando dados de entrada...".
            format(time.strftime("%Y-%m-%d %H:%M:%S")))
353
354     # carrega o dataset
355     dataset = numpy.genfromtxt(arquivo_entrada, dtype='str',
            delimiter=",", skip_header=1, unpack=True)
356     dataset_clima = numpy.genfromtxt(arq_entrada_clima, dtype='
            str', delimiter=",", skip_header=1, unpack=True)
357
358     # carrega o cabecalho do arquivo
359     header_entrada = numpy.genfromtxt(arquivo_entrada, dtype='
            str', delimiter=",", skip_header=0, deletechars='\"',
            unpack=True, max_rows=1)
360     dataset_header = header_entrada.tolist()
361     header_entrada_clima = numpy.genfromtxt(arq_entrada_clima,
            dtype='str', delimiter=",", skip_header=0, deletechars='
            \"', unpack=True, max_rows=1)
362     dataset_clima_header = header_entrada_clima.tolist()
363
364
365
366     dataset_social_float = numpy.apply_along_axis(lambda linha:
            pd.to_numeric(linha, errors='coerce'), 1, dataset[7:,:])
367     dataset_social_float_header = dataset_header[7:]
368
369     dataset_clima_float = numpy.apply_along_axis(lambda linha:
            pd.to_numeric(linha, errors='coerce'), 1, dataset_clima
            [8:,:])
370     dataset_clima_float_header = dataset_clima_header[8:]
371
372
373
374     sys.stdout.write("\n{0:s}: Discretizando dados sociais...\n"
            .format(time.strftime("%Y-%m-%d %H:%M:%S")))
375     dataset_social_cluster = discretiza_matriz_individual(
            dataset_social_float, dataset_social_float_header,arq_log
            )
```

```
376
377     pop_disc = discretiza_populacao(dataset[2].astype(float))
378
379     cluster_social_dados = dataset[0:2,:].copy()
380     cluster_social_dados = numpy.vstack((cluster_social_dados,
            pop_disc))
381     cluster_social_dados = numpy.vstack((cluster_social_dados,
            dataset_social_cluster))
382
383     cluster_social_header = dataset_header[0:2].copy()
384     cluster_social_header += [dataset_header[2]]
385     cluster_social_header += dataset_social_float_header
386
387     cluster_social = numpy.vstack((cluster_social_header,
            cluster_social_dados.T))
388
389     sys.stdout.write("\r{0:s}: Discretizando dados climaticos ...
            ----------------------------------------------------------
            --------------------\n".format(time.strftime("%Y-%m-%d %H
            :%M:%S")))
390
391     #dataset_clima_cluster = discretiza_quantis(
            dataset_clima_float, dataset_clima_float_header, 3)
392     dataset_clima_cluster = discretiza_matriz_individual(
            dataset_clima_float, dataset_clima_float_header, arq_log)
393
394     cluster_clima_dados = dataset_clima[0:4,:].copy()
395     # Criando um indice de incidencia unindo os indices Alto e
            Medio, chamado de 'incidencia_grp'
396     cluster_clima_dados = numpy.vstack((cluster_clima_dados,
            dataset_clima[3].copy()))
397     cluster_clima_dados[4,numpy.where('Medio' ==
            cluster_clima_dados[4,:])[0]] = 'Alto'
398     # Acrescentando ao cluster o restante dos dados
            discretizados
399     cluster_clima_dados = numpy.vstack((cluster_clima_dados,
            dataset_clima_cluster))
400
401     cluster_clima_header = dataset_clima_header[0:4].copy()
```

```
402     cluster_clima_header += ['incidencia_grp']
403     cluster_clima_header += dataset_clima_float_header
404
405     cluster_clima_temp_min_media = discretiza_temp_minima_media(
            dataset_clima_float, dataset_clima_float_header, arq_log)
406
407     cluster_clima_temp_min_media_amp =
            discretiza_temp_minima_media_amp(dataset_clima_float,
            dataset_clima_float_header, arq_log)
408
409     cluster_clima_temp_min_amp = discretiza_temp_minima_amp(
            dataset_clima_float, dataset_clima_float_header, arq_log)
410
411     cluster_clima = numpy.vstack((cluster_clima_header,
            cluster_clima_dados.T))
412     cluster_clima = numpy.hstack((cluster_clima,
            cluster_clima_temp_min_media))
413     cluster_clima = numpy.hstack((cluster_clima,
            cluster_clima_temp_min_media_amp))
414     cluster_clima = numpy.hstack((cluster_clima,
            cluster_clima_temp_min_amp))
415
416     # Montando o cluster final, como uma juncao dos dados
            climaticos e dos sociais
417     cluster_final = cluster_clima.copy()
418     cluster_final = numpy.hstack((cluster_final, numpy.empty((
            len(cluster_final), len(cluster_social.T[2:])), dtype='
            U50')))
419
420     cluster_final[0,len(cluster_clima.T):] = cluster_social
            [0,2:]
421
422     for idx, indicador in enumerate(cluster_social[1:]):
423         linhas = numpy.where(numpy.logical_and(cluster_final
                [:,0]==indicador[0], cluster_final[:,1]==indicador
                [1]))[0]
424         cluster_final[linhas,len(cluster_clima.T):] = indicador
                [2:]
425
```

```
426
427         sys.stdout.write("\r{0:s}: Clusterizacao Finalizada ...       
                ..............................\n".format(time.strftime("%Y
                -%m-%d %H:%M:%S")))
```

B.3 Mineração de Itens Frequentes e Regras de Associação

```
1  # -*- coding: utf-8 -*-
2
3  import numpy
4  import time
5  import sys
6  import pandas as pd
7  import scipy.stats as stats
8  #from functools import reduce
9  #import functools
10
11 #from sklearn.cluster import KMeans
12 from mlxtend.preprocessing import TransactionEncoder
13 from mlxtend.frequent_patterns import apriori
14 from mlxtend.frequent_patterns import association_rules
15 #from mlxtend.preprocessing import minmax_scaling
16 #from mlxtend.feature_extraction import RBFKernelPCA as KPCA
17
18 #from sklearn.metrics import silhouette_samples,
       silhouette_score
19
20
21 #tabela = "bdmed_chuva4"
22 #tabela = "bdmed_completo"
23 #tabela = "bdmed_chuva3-4"
24 #tabela = "bdmed_chuva2-4_acima-de-100hab"
25 #tabela = "bdmed_chuva2-4_acima-de-100hab_selecionadas"
26 #tabela = "bdmed_social_sem4_acima-de-100hab"
27 #tabela = "bdmed_chuva4_acima-de-100hab"
28 #tabela = "bdmed_chuva3-4_acima-de-100hab"
29 #tabela = "bdmed_chuva2-4_de-500-a-999hab"
30 #tabela = "bdmed_chuva2-4_acima-de-1milhao_hab"
31 #tabela = "bdmed_chuva3-4_acima-de-1milhao_hab"
32 #arquivo_entrada = "./dados/" + tabela + ".csv"
```

```
#arq_saida_kmeans = "./saida/saida_kmeans_" + tabela + "_" + time.strftime("%Y%m%d-%Hh%Mm") + ".csv"
#arq_saida_apriori = "./saida/saida_apriori_" + tabela + "_" + time.strftime("%Y%m%d-%Hh%Mm") + ".csv"
arquivo_log = "./log/log_apriori_analise_" + time.strftime("%Y%m%d-%Hh%Mm") + ".txt"
#arquivo_dicionario = "../dados/dicionario-" + tabela + ".csv"
#arq_imagem = "imagens/img_" + tabela + "_" + time.strftime("%Y%m%d-%Hh%Mm")




# Funcao que retorna uma lista com todas as combinacoes possiveis dos itens da lista passada como parametro.
# Ex.: se a lista de entrada for [item1, item2, item3] a lista de saida sera:
# [[item1], [item2], [item3], [item1, item2], [item1, item3], [item2, item3], [item1, item2, item3]]
def lista_generalizacoes(lista_completa, idx=None, idx_total=None, hora=None):

    if ( idx != None ):
        sys.stdout.write("\r{0:s}: Criando generalizacao {1:d} de {2:d}.                                                                                                                                  ".format(hora, idx+1, idx_total))

    lista = list(map(lambda x: [x], lista_completa))
    lista.sort()
    lista_aux = lista.copy()
    qtde = len(lista_completa)
    i = 1
    while ( i < qtde ):
        # Realizando o produto cartesiano da lista com a lista original, concatenando os itens
        lista = list(list(set(sorted(item_lista+item_aux))) for item_lista in lista for item_aux in lista_aux)
```

```
58          lista.sort()
59
60          # Eliminando itens duplicados da lista
61          lista_tuple = list(map(lambda x: tuple(x), lista))
62          lista_limpa = list(set(lista_tuple))
63          lista = list(map(lambda x: list(x), lista_limpa))
64
65          i += 1
66
67      return lista
68
69
70
71  def adiciona_generalizacoes(itens_freq_dict, itens_freq_final,
        item, itens_freq_array):
72      item_tuple = tuple(sorted(item))
73      if ( item_tuple in itens_freq_final ): return
            itens_freq_final, itens_freq_array
74
75      itens_freq_final.add(item_tuple)
76      itens_freq_array[itens_freq_dict[item_tuple]] = True
77
78      if ( len(item_tuple) <= 1 ): return itens_freq_final,
            itens_freq_array
79      for idx, subitem in enumerate(item_tuple):
80          subset = set(item_tuple).difference([subitem])
81          itens_freq_final, itens_freq_array =
                adiciona_generalizacoes(itens_freq_dict,
                itens_freq_final, subset, itens_freq_array)
82      return itens_freq_final, itens_freq_array
83
84
85
86  def seleciona_itens_freq(itens_freq, filtro):
87      hora_inicio = time.strftime("%Y-%m-%d %H:%M:%S")
88      sys.stdout.write("\r{0:s}: {1:d} itens frequentes. Filtrando
            itens \"{2:s}\".                                    
            ____________________________________________________
            ____________________________________________________
            ____________________".
```

```
            format(hora_inicio, len(itens_freq), filtro))
89      # Pegando, da relacao completa de itens frequentes, aqueles
            que correspondem ao filtro
90      itens_freq_filtrado = itens_freq[itens_freq['itemsets'].
            apply(lambda x: (filtro in str(x)) )]
91  #     itens_freq_filtrado = itens_freq_filtrado[
        itens_freq_filtrado['length']<=10]
92
93      sys.stdout.write("\r{0:s}: {1:d} itens frequentes \"{2:s}\".
            ____________________________________________________
            ____________________________________________________
            ____________________________________".format(hora_inicio,
            len(itens_freq_filtrado), filtro))
94
95      if ( len(itens_freq_filtrado) == 0 ): return
            itens_freq_filtrado
96
97      sys.stdout.write("\r{0:s}: Construindo dicionario dos itens
            frequentes.________________________________________
            ____________________________________________________
            ____________________________________________".format(
            hora_inicio))
98      itens_freq_dict = dict(list(map(lambda x: (tuple(sorted(x.
            itemsets)), x.Index), itens_freq.itertuples())))
99
100     itens_freq_final = set()
101     itens_freq_array = numpy.full(len(itens_freq), False)
102     i = 1
103     for item in itens_freq_filtrado.itertuples():
104         sys.stdout.write("\r{0:s}: Criando lista de
                generalizacoes ({1:d} de {2:d}) - tamanho: {3:d}.____
                ____________________________________________________
                ____________________________________________________
                ____________________________________".format(
                hora_inicio, i, len(itens_freq_filtrado), item.length
                ))
105         itens_freq_final, itens_freq_array =
                adiciona_generalizacoes(itens_freq_dict,
                itens_freq_final, item.itemsets, itens_freq_array)
```

```
106             i += 1
107
108         sys.stdout.write("\r{0:s}: Criando dataframe de itens
                frequentes final ({1:d} itens).                    
                                                                   
                                                                   
                                                                   
                ".format(hora_inicio, len(itens_freq_final)))
109
110         itens_freq_filtrado = itens_freq[itens_freq_array]
111
112         return itens_freq_filtrado
113
114
115
116
117 def fisher_test(itens_freq_support_dict, X_tuple, Y_tuple, qtde,
        alfa=0.01):
118         X_list = list(map(lambda x: x, X_tuple))
119         X_list.sort()
120         X_tuple = tuple(X_list)
121         Y_list = list(map(lambda x: x, Y_tuple))
122         Y_list.sort()
123         Y_tuple = tuple(Y_list)
124         XY_list = X_list+Y_list
125         XY_list.sort()
126         XY_tuple = tuple(XY_list)
127
128         a = qtde*(itens_freq_support_dict[XY_tuple])
129         b = qtde*(itens_freq_support_dict[X_tuple] -
                itens_freq_support_dict[XY_tuple])
130
131         c = qtde*(itens_freq_support_dict[Y_tuple] -
                itens_freq_support_dict[XY_tuple])
132         d = qtde*(1 - itens_freq_support_dict[XY_tuple])
133
134         oddsratio_regra, pvalue_regra = stats.fisher_exact([[a, b],
                [c, d]])
135
136         if ( pvalue_regra <= alfa ):
```

```
137             productive = True
138         else:
139             productive = False
140             return productive, pvalue_regra, oddsratio_regra
141 
142 #      X_gen = lista_generalizacoes(X_list)
143 
144         for Z_list in X_list:
145 
146             Z_list = sorted([Z_list])
147             if (Z_list == X_list): continue
148 
149             W_list = list(set(X_list).difference(Z_list))
150             W_list.sort()
151             W_tuple = tuple(W_list)
152             WY_list = W_list+Y_list
153             WY_list.sort()
154             WY_tuple = tuple(WY_list)
155 
156             c = qtde*(itens_freq_support_dict[WY_tuple] -
                    itens_freq_support_dict[XY_tuple])
157             d = qtde*(itens_freq_support_dict[W_tuple] -
                    itens_freq_support_dict[XY_tuple])
158 
159             oddsratio, pvalue = stats.fisher_exact([[a, b], [c, d]])
160 
161             if ( (pvalue > alfa) | (pvalue < pvalue_regra) ):
162                 productive = False
163                 return productive, pvalue_regra, oddsratio_regra
164 
165         return productive, pvalue_regra, oddsratio_regra
166 
167 
168 def itens_frequentes(dataset, itens_freq=None, min_support=0.04,
        max_length=None):
169         df=te=te_ary = None
170 
171 #       sys.stdout.write("\r{0:s}: Criando o Dataframe Pandas...".
        format(time.strftime("%Y-%m-%d %H:%M:%S")))
```

```
172     te = TransactionEncoder()
173     dataset_full = numpy.transpose(dataset)
174     te_ary = te.fit(dataset_full).transform(dataset_full, sparse
            =True)
175     df = pd.SparseDataFrame(te_ary, columns=te.columns_,
            default_fill_value=False)
176     df.drop(columns=['NaN'], inplace=True)
177
178     if ( itens_freq is None ):
179         sys.stdout.write("\r{0:s}: Detectando itens frequentes (
                dimensoes do df: {1:d})                                    
                                                                           \n".format(
                time.strftime("%Y-%m-%d %H:%M:%S"), numpy.size(df,
                axis=1)))
180
181         itens_freq = apriori(df, min_support=0.04, use_colnames=
                True, max_len=max_length)
182
183         sys.stdout.write("\r{0:s}: Acrescentando coluna com
                quantidade de itens da tupla (itens frequentes: {1:d
                })                                                         
                                                                           \n".format(time.strftime("%Y-%m-%d
                %H:%M:%S"), len(itens_freq)))
184
185         itens_freq['length'] = itens_freq['itemsets'].apply(
                lambda x: len(x))
186
187     sys.stdout.write("\r{0:s}: Criando dicionario dos itens
            frequentes (itens frequentes: {1:d})                           
                                                                           \n".
            format(time.strftime("%Y-%m-%d %H:%M:%S"), len(itens_freq
            )))
188
189     itens_freq_support_dict = dict(list(map(lambda x: (tuple(
            sorted(x.itemsets)), x.support), itens_freq.itertuples())
            ))
190
191     return itens_freq, itens_freq_support_dict
192
```

```
193
194
195
196
197 def filtra_regras(regras_alto):
198     aux = [
199             [reg_princ.Index, reg_princ.antecedants_clima,
                reg_princ.antecedants_social, reg_princ.
                antecedants, reg_princ.consequents, reg_princ.
                antecedent_support, reg_princ.consequent_support,
                 reg_princ.support, reg_princ.confidence,
                reg_princ.lift, reg_princ.leverage, reg_princ.
                conviction, reg_princ.pvalue, reg_princ.oddsratio
                , reg_princ.productive, reg_sec.Index, reg_sec.
                antecedants_clima, reg_sec.antecedants_social,
                reg_sec.antecedants, reg_sec.consequents, reg_sec
                .antecedent_support, reg_sec.consequent_support,
                reg_sec.support, reg_sec.confidence, reg_sec.lift
                , reg_sec.leverage, reg_sec.conviction, reg_sec.
                pvalue, reg_sec.oddsratio, reg_sec.productive]
200             for reg_princ in regras_alto.itertuples() for
                reg_sec in regras_alto.itertuples() if ((
                reg_princ.antecedants_clima == reg_sec.
                antecedants) & (len(reg_princ.antecedants_social)
                 > 0) & (reg_princ.Index != reg_sec.Index) & (
                reg_princ.productive == True))
201         ]
202
203     regras = pd.DataFrame(aux, columns=['reg_princ_Index', '
            reg_princ_antecedants_clima', '
            reg_princ_antecedants_social', 'reg_princ_antecedants', '
            reg_princ_consequents', 'reg_princ_antecedent_support', '
            reg_princ_consequent_support', 'reg_princ_support', '
            reg_princ_confidence', 'reg_princ_lift', '
            reg_princ_leverage', 'reg_princ_conviction', '
            reg_princ_pvalue', 'reg_princ_oddsratio', '
            reg_princ_productive', 'reg_sec_Index', '
            reg_sec_antecedants_clima', 'reg_sec_antecedants_social',
             'reg_sec_antecedants', 'reg_sec_consequents', '
```

```
        reg_sec_antecedent_support', 'reg_sec_consequent_support'
        , 'reg_sec_support', 'reg_sec_confidence', 'reg_sec_lift'
        , 'reg_sec_leverage', 'reg_sec_conviction', '
        reg_sec_pvalue', 'reg_sec_oddsratio', 'reg_sec_productive
        '])
204
205     return regras
206
207 def regras_associacao(itens_freq, itens_freq_support_dict,
    dataset_base, consequent, min_threshold=0.3):
208     Y_tuple = tuple(sorted([consequent]))
209     itens_freq_filtrado = itens_freq[itens_freq['itemsets'].
        apply(lambda x: (consequent in str(x)) )]
210
211     regras_np = numpy.empty((0,14))
212     i = 1
213     itens_freq_filtrado_qtde = len(itens_freq_filtrado)
214     for item in itens_freq_filtrado.itertuples():
215         sys.stdout.write("\r{0:s}: Criando regras de associacao
            (item frequente {1:d} de {2:d})
            ".
            format(time.strftime("%Y-%m-%d %H:%M:%S"), i,
            itens_freq_filtrado_qtde))
216         if ( len(item.itemsets) < 2 ): continue
217         XY_tuple = tuple(sorted(item.itemsets))
218         X_tuple = tuple(sorted(set(XY_tuple).difference(Y_tuple)
            ))
219
220         # Calculando metricas associadas a regra
221         confidence = itens_freq_support_dict[XY_tuple]/
            itens_freq_support_dict[X_tuple]
222         lift = confidence/itens_freq_support_dict[Y_tuple]
223         leverage = itens_freq_support_dict[XY_tuple] - (
            itens_freq_support_dict[X_tuple]*
            itens_freq_support_dict[Y_tuple])
224         conviction = (1-itens_freq_support_dict[Y_tuple])/(1-
            confidence) if confidence < 1 else numpy.inf
225
226         oddsratio = numpy.NaN
```

```
227             pvalue = numpy.NaN
228             productive = numpy.NaN
229 
230             if ( confidence >= min_threshold ):
231                 # Executando Fisher Test para verificar a
                        significancia estatistica da regra
232                 sys.stdout.write("\r{0:s}: Criando regras de
                        associacao (item frequente {1:d} de {2:d}) -
                        Fisher Test                                  
                                                                     
                        ".format(time.
                        strftime("%Y-%m-%d %H:%M:%S"), i,
                        itens_freq_filtrado_qtde))
233 #                 productive, pvalue, oddsratio = fisher_test(
    itens_freq_support_dict, X_tuple, Y_tuple, len(dataset_base.T
    ))
234                 # Separando os itens do antecedente relacionados a
                        fatores climaticos dos sociais
235                 X_clima = tuple(sorted([item for item in X_tuple if
                        ('sem' in item[4:7])]))
236                 X_social = tuple(sorted(set(X_tuple).difference(
                        X_clima)))
237                 # Construindo o array com as regras de associacao
238                 regras_np = numpy.vstack((regras_np, [X_clima,
                        X_social, X_tuple, Y_tuple,
                        itens_freq_support_dict[X_tuple],
                        itens_freq_support_dict[Y_tuple],
                        itens_freq_support_dict[XY_tuple], confidence,
                        lift, leverage, conviction, pvalue, oddsratio,
                        productive]))
239             i += 1
240 
241         regras = pd.DataFrame(regras_np, columns=['antecedants_clima
                ', 'antecedants_social', 'antecedants', 'consequents', '
                antecedent_support', 'consequent_support', 'support', '
                confidence', 'lift', 'leverage', 'conviction', 'pvalue',
                'oddsratio', 'productive'])
242 
243         return regras
244 
```

```
245
246
247
248
249
250 with open(arquivo_log, 'w', buffering=1) as arq_log:
251     j = 0
252     tempo = time.time()
253
254     sys.stdout.write("\n{0:s}: Iniciando Execucao...\n".format(
            time.strftime("%Y-%m-%d %H:%M:%S")))
255
256     sys.stdout.write("\r{0:s}: Carregando dados de entrada...".
            format(time.strftime("%Y-%m-%d %H:%M:%S")))
257
258     cluster_final = numpy.load('./dados/spider/cluster_final.npy
            ')
259     cluster_clima = numpy.load('./dados/spider/cluster_clima.npy
            ')
260     cluster_social = numpy.load('./dados/spider/cluster_social.
            npy')
261
262     correlacao_clima = numpy.load('./dados/spider/
            correlacao_clima-sem1-a-4_selecionados.npy')
263 #     correlacao_social = numpy.load('./dados/spider/
    correlacao_social_teste1.npy')
264     correlacao_social = numpy.load('./dados/spider/
            correlacao_social_selecionados.npy')
265
266     itens_freq = None
267     itens_freq = pd.read_pickle('./dados/spider/
            clima_social_final-itens_freq.pkl')
268 #     itens_freq = pd.read_pickle('./dados/spider/clima_grp-
    incidencia_grp-itens-freq.pkl')
269
270     regras_alto = None
271     regras_alto = pd.read_pickle('./dados/spider/
            clima_social_final-regras.pkl')
272
```

```
273     regras_alto_analise = pd.read_pickle('./dados/spider/
            clima_social_final-regras-analise.pkl')
274
275     regras_baixo = None
276
277
278 #     str_rodape = ("\nHora termino: {0:s}\n".format(time.
        strftime("%Y-%m-%d %H:%M:%S")))
279 #     arq_log.write(str_rodape)
280 #     print (str_rodape)
281
282     # APRIORI #
283     selecionados_clima = correlacao_clima['header'][numpy.where(
            correlacao_clima['selecionado']==True)[0]]
284 #     selecionados = numpy.append(selecionados, ['sem1-
        temp_min_amp', 'sem2-temp_min_amp', 'sem3-temp_min_amp', '
        sem4-temp_min_amp'])
285
286 #     selecionados = numpy.append(selecionados, ['sem0-
        temp_min_media', 'sem1-temp_min_media', 'sem2-temp_min_media
        ', 'sem3-temp_min_media', 'sem4-temp_min_media', 'sem0-
        temp_min_media_amp', 'sem1-temp_min_media_amp', 'sem2-
        temp_min_media_amp', 'sem3-temp_min_media_amp', 'sem4-
        temp_min_media_amp', 'sem0-temp_min_amp', 'sem1-temp_min_amp
        ', 'sem2-temp_min_amp', 'sem3-temp_min_amp', 'sem4-
        temp_min_amp'])
287
288     selecionados_clima = numpy.append(selecionados_clima, ['
            incidencia_grp'])
289
290 #     selecionados = list([['sem3_precipitacao'], ['
        sem3_temp_media'], ['sem4_temp_amplitude'], ['
        sem2_temp_minima'], ['sem4_temp_media'], ['sem4_temp_minima
        '], ['sem1_temp_minima'], ['sem3_temp_amplitude'], ['
        sem3_temp_minima'], ['sem2_vento'], ['sem1_vento']])
291
292 #     selecionados = numpy.append(selecionados, ['sem3-
        temp_min_media', 'sem4-temp_min_media', 'sem3-
        temp_min_media_amp', 'sem4-temp_min_media_amp', 'sem1-
```

```
    temp_min_amp', 'sem2-temp_min_amp', 'sem3-temp_min_amp', '
    sem4-temp_min_amp'])
293
294
295 #     selecionados_teste_social = list(['sem4-temp_min_media_amp
    ', 'sem4-temp_min_media', 'sem4-temp_min_amp', '
    sem4_temp_minima', 'sem3-temp_min_media_amp', 'sem3-
    temp_min_amp', 'sem3_temp_minima', 'sem3_precipitacao', '
    sem2_vento', 'sem1_vento'])
296
297 #     selecionados_teste_social = numpy.append(
    selecionados_teste_social, ['incidencia_grp'])
298
299      selecionados_social = correlacao_social['header'][numpy.
         where(correlacao_social['selecionado']==True)[0]]
300
301      selecionados = numpy.append(selecionados_clima,
         selecionados_social)
302
303 #     selecionados_teste_social = numpy.append(
    selecionados_teste_social, selecionados_social)
304
305      sys.stdout.write("\r{0:s}: Detectando itens frequentes ...
         ____________________________________________________________
         _________\n".format(time.strftime("%Y-%m-%d %H:%M:%S")))
306
307      dataset_base = cluster_final.T[numpy.isin(cluster_final.T
         [:,0], selecionados), 1:]
308 #     dataset_base = cluster_clima[1:,3:].T.copy()
309
310      itens_freq, itens_freq_support_dict = itens_frequentes(
         dataset_base, itens_freq)
311 #     itens_freq, itens_freq_support_dict =
    itens_frequentes_manual(dataset_base, itens_freq)
312
313      sys.stdout.write("\r{0:s}: Encontrando regras de associacao
         ...________________________________________________________
         _____________\n".format(time.strftime("%Y-%m-%d %H:%M:%
         S")))
```

```
314
315     i = 0
316
317
318     if ( regras_alto is None ):
319         regras_alto = regras_associacao(itens_freq,
                itens_freq_support_dict, dataset_base, 'Alto')
320
321     regras_baixo = regras_associacao(itens_freq,
            itens_freq_support_dict, dataset_base, 'Baixo')
322
323
324 #     sys.stdout.write("\r{0:s}: Separando regras de associacao
    ({1:d} regras)

    ".format(time.strftime("%Y-%m-%d %H:%M:%S"), numpy.size(
    regras_alto, axis=0)))
325 #     regras_consequent_alto = regras_alto[regras_alto["
    consequents"].apply(lambda x: ( "Alto" in str(x) ))]
326 #     regras_antecedants_alto = regras_alto[regras_alto["
    antecedants"].apply(lambda x: ( "Alto" in str(x) ))]
327
328     sys.stdout.write("\r{0:s}: Execucao Finalizada ...
            ------------------------\n".format(time.strftime("%Y-%m-%
            d %H:%M:%S")))
```