Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Técnicas de Machine Learning aplicada a Threat Intelligence sobre fontes abertas

Álvaro Torres Vieira

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Orientador
Prof. Dr. João José Costa Gondim

Brasília
2019

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Técnicas de Machine Learning aplicada a Threat Intelligence sobre fontes abertas

Álvaro Torres Vieira

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Prof. Dr. João José Costa Gondim (Orientador)
CIC/UnB

Dr. Luis Pacheco
CIC/UnB

Dr. Paulo Angelo Alves Resende
PCTEC/UnB

Prof. Dr. José Edil Guimarães de Medeiros
Coordenadora do Curso de Engenharia da Computação

Brasília, 16 de Julho de 2019

Dedicatória

Dedico este trabalho aos meus pais Carlos Henrique e Moema, que sempre com muita dedicação, me apoiaram durante toda trajetória de vida. Ao meu falecido avô Jorge, que sempre acreditou no papel fundamental que a universidade tem na formação de um cidadão do bem.

Agradecimentos

Agradeço aos colegas de curso da UnB que sempre ajudaram nos momentos difíceis da universidade com muita perseverança. Em especial, aos meus pais, por sempre acreditarem em mim, e me darem todo suporte necessário. Ao meu irmão Arthur, que sempre me ajudou a escolher o caminho certo. As minhas avós Inah e Maria José pelo carinho e apoio durante toda a jornada. Aos meus avôs falecidos Jorge e Cláudio, me mostraram com exemplo o melhor caminho a se seguir. Agradeço também ao meu orientador João Gondim pelo apoio e conhecimento compartilhado nesse trabalho de conclusão de curso.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES), por meio do Acesso ao Portal de Periódicos.

Resumo

As ameaças cibernéticas evoluem em uma taxa de velocidade muito maior que o cérebro humano pode acompanhar. A criação de novos ataques, assim como o volume de ameaças que acontecem a cada segundo possuem tamanho estratosféricos, necessitando de ferramentas que comportam o processamento de *bigdata*. Muitas das ameaças cibernéticas provém de máquinas que as criam de forma massiva e automatizada. O trabalho tem como objetivo aplicar algoritmos de aprendizado de máquina em busca de uma visualização melhorada dos grupos de ataques, com objetivo de identificar se existem padões de características dos ataques. Para isso, foi aplicado o algoritmo K-means com o valor de K igual a dez em diferentes *subsets* de dados contendo informações das ameaças cibernéticas. O objetivo é entender se existem padrões nos ataques, visto que o algoritmo agrupa os dados em grupos com características semelhantes. A partir dos resultados encontrados, pode-se observar em uma das tentativas de agrupamento, uma distribuição homogênea de dados em dez grupos diferentes, confirmado o pressuposto de que as ameaças cibernéticas são geradas de forma massiva e com um certo padrão a partir de sistemas maliciosos. Entendendo esse padrão, se torna mais fácil criar estratégias de segurança cibernética, algo em grande discursão nos dias atuais no mundo da tecnologia.

Palavras-chave: Aprendizado de Máquina, Spark, Ataques Cibernéticos

Abstract

This work presents a study of machine learning techniques for clustering applied to data from cyber attacks taken from open sources. The study of data clustering algorithms aims to aid in the analysis and prevention of cyber attacks. The results were obtained from the analysis from the application of the k-means algorithm in different subsets of data, in order to extract the best practice. Future studies can be applied to deepen the analyzes of each of the groups obtained, using other techniques of machine learning, in order to obtain a micro analysis and find trends on cyber attacks.

Keywords: Machine Learning, Spark, Threat Intelligence

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Introdução

Existe muito espaço para desenvolver conhecimento a respeito do estudo de grandes quantidades de dados, devido ao tamanho do universo que essa área de conhecimento pode ter. Até alguns anos atrás, não se considerava *big data* como um objeto de estudo. Foi em 2005, em que O'Reilley definiu pela primeira vez como uma "grande quantidade de dados que quase impossível gerenciar e processar usando ferramentas de inteligencia tradicionais".

Como podemos ver na Figura **??**, é possível separar em três áreas as definições principais que, com suas intersecções, formam diferentes áreas de estudo. As definições são: matemática e estatística; métodos e algoritmos; análise de *big data*. A área de estudo explorada nesse trabalho é sobre aprendizado de máquina, que exige conhecimentos estatísticos e computacionais, para aplicação de algoritmos e análises dos resultados.

O aprendizado de máquina é uma das áreas de estudo que depende de *big data* para acontecer com um certo grau de confiabilidade. De forma resumida, aprendizado de máquina é ensinar da maneira mais eficiente e com menor erro possível, uma máquina a aprender [2]. Isso é possível a partir da criação de modelos resultados do treino do algoritmo com entradas tratadas, que são utilizadas como parâmetros para os algoritmos, sendo essas informações necessárias para que uma máquina aprenda algo que julgue certo ou errado e possa tomar decições futuras com base nesse aprendizado. O conceito pode ser traduzido de maneiras diferentes pelo fato de possuir algoritmos diferentes com distintos objetivos e aplicações, como abordado com mais profundidade no Capítulo 2. Ensinar uma máquina a aprender é uma das pilastras da inteligência artificial que, por sua vez, possui outras áreas de estudos também, como o uso e análise de de sensores para tomadas de decisões.

1.2 Objetivo Geral

O objetivo do trabalho é avaliar características de agrupamento em cima de ameaças cibernéticas a partir da aplicação do algoritmo de clusterização k-means. A aplicação do algoritmo foi feita em quatro *subsets* de dados diferentes, separando-os em 10 grupos que com características semelhantes. A procura de semelhanças entre as ameaças serve como base para entender melhor se existe uma padronização ou aleatoriedade em sua criação. O trabalho também tem como objetivo explicitar o processo de produção de conhecimento sobre ameaças a partir da informação de fontes abertas, utilizando ferramentas de manipulação de grandes quantidades de dados.

1.3 Organização do Trabalho

O trabalho está estruturado em capítulos. No Capítulo 2 será apresentado a definição de conceitos básicos e ferramentas utilizadas, conhecimentos necessários para entender o desenvolvimento desse trabalho em sua completude. No Capítulo 3, está a metodologia utilizada e executado, com explicação do algortimo desenvolvido e das ferramentas utilizadas. O Capítulo 4 expõe os resultados encontrados a partir de visualizações criadas para melhor análise humana, criadas a partir de busca e filtro de tabelas no banco de dados. Por fim, o capítulo Capítulo 5 possui uma breve conclusão sobre os principais aspectos observados no experimento, com base nos dados e gráficos obtidos nas diferentes análises.

Capítulo 2

Conceitos Básicos e Ferramentas Utilizadas

2.1 Big Data

Apesar do entendimento de *big data* a ser onipresente para todas as nações, a origem da definição do conceito é pouco conhecida. O crescimento de estudos relacionados ao tema nos ultimos anos tem sido algo exponencial, como podemos ver na Figura 2.1 retirada da referência [1]. Pelo fato do seu crescimento acelerado, sua definição ainda é, nos dias de hoje, confusa para muitos pesquisadores. A falta da definição concreta dilui os campos de estudos e dificulta a definição e objeto de estudo permanente. O artigo [3] tenta criar uma definição do significado por trás da fama e da popularização do termo nos ultimos anos.

Apesar de ser um termo relativamente recente no mundo dos negócios e estudos, nos dias de hoje, é um dos termos mais comuns em teses, dissertações e discussões sobre negócios. Muitas empresas procuram colaboradores capacitados em manipular e analisar *big data*, que traz inteligência de mercado a partir da análise e manipulação de dados que trazem informações importantes sobre o comportamento do cliente público alvo da empresa.

2.1.1 Definição de *Big Data*

Com objetivo de entender melhor o campo de estudo, Laney [4] criou a definição dos Três V's, três desafios em gerenciamento de dados: volume, variedade e velocidade. Com objetivo de se usar *big data* como objeto de estudo ou trabalho, deve-se levar em consideração essas três características importantes descritas nos próximos parágrafos.

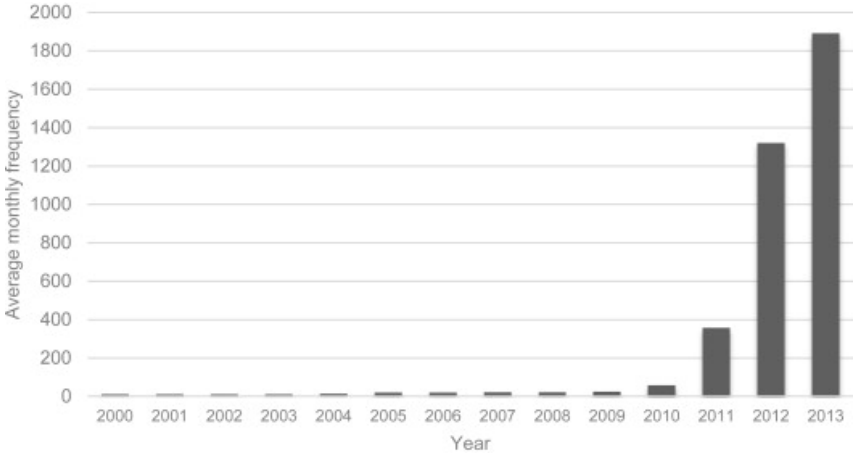


Figura 2.1: Crescimento das buscas pela palavra Big Data. Referência [1].

Volume se refere a magnitude do dado, muita das vezes multiplos de terabytes e petabytes. É importante salientar que o tamanho de um dado que hoje é considerado *big data*, era visto de outra forma nos anos passados e pode tomar proporções muito maiores no futuro, visto que a capacidade de armazenamento vem crescendo ao longo dos anos, além da melhoria de algoritmos de compressão.

Variedade se refere a estrutura heterogênea do dado, que pode estar estruturado, semi-estruturado ou desestruturado. O dado estruturado corresponde a 5% de todo os dados existentes segundo a fonte [5], que consistem em planilhas ou bancos de dados relacionais. Textos, imagens, áudio e vídeo são exemplos de dados não estuturados, que necessitam de uma máquina para analisar e se tornar compreensivel ao ser humano. Os dados semi estruturados são aqueles não possuem conformidade com os padrões rígidos de estrutura, como os arquivos de *Extensibele Markup Lenguage (XML)* ou *HyperText Markup Language (HTML)*.

Velocidade se refere a taxa em que os dados são gerados e a velocidade em que ela deve ser analizada para se obter resultados relevantes. A enorme quantidade de sensores, smartfones, casas e carros inteligentes e redes sociais geram um volume imenso de dados todos os dias que precisam ser analisados por sistemas robustos e capazes de processar enormes quantidades de dados desestruturados em curta quantidade de tempo. Várias estratégias de implementação são utilizadas para que isso aconteça, como a paralelização de processos em clusters diferentes.

Existem ainda outros três V's, que foram definidos ao longo do tempo como características complementares e inerentes ao estudo de *big data*. Entre eles estão veracidade, variabilidade e valor.

A *veracidade* de um dado traz a reflexão a respeito do quão verdadeiro é a informação

de um dado. Um exemplo a respeito desse tema, são os dados que trazem sentimentos de usuários de uma rede social a partir de suas ações, análise que traz uma incerteza na definição pois pode ser contratriada a partir de um julgamento humano. Dessa forma, é necessário colocar em pauta se aquela informação é uma verdade absoluta, ou se sua fonte necessita de ajustes de parâmetros, afim de trazer uma confiabilidade maior a respeito da informação.

A *variabilidade* e complexidade se refere a variação das taxas de fluxo de dados, que podem ter períodos de picos e vales. Além disso, o big data vem, em sua maioria, de uma grande variedade de fontes, o que acrescenta um desafio em termos de complexidade. Portanto, uma dimensão que precisa ser discutida a respeito do tema, é sobre a combinação e limpeza da grande quantidade de massa de dados provindos de diferentes fontes e tabelas.

O *valor* de um dado representa a sua densidade em relação ao tamanho da base de dados analisada. Uma linha de uma tabela, dentro de uma massa de terabyte de dados, possui pouco valor em relação ao big data como um todo. Porém, ao criar analisar esse dado e gerar uma informação a partir dele, ele se tranforma em uma informação com alto valor.

2.2 Ameaças Cibernéticas

A definição de um ataque cibernético ainda é, nos dias atuais, algo em construção. Os cenários que podem ser considerados um ataque são dos mais diversos, desde mensagens maliciosas de email, vírus em computadores domésticos com objetivo de reter senhas bancárias até ataques que geram *blackout* em torres de controle aéreo. A falta de uma definição exata e universal agrava a dificuldade de se ter ataques cibernéticos como objeto de estudo comum e claro entre todos os países do mundo.

As definições de ataque cibernético são de uma variedade imensa ao redor do mundo. A definição mais abrangente vem do Richard A. Clarke [3], que define guerra cibernética como "ações de um estado-nação que penetra computadores ou redes de outra nação com objetivo de causar dano ou disrupção". Outra definição criada por um antigo agente do NSA e CIA, o diretor Michael Hayden [3] é de que ataques cibernéticos são "uma tentativa deliberade de desativar ou destruir as redes de computador de outro país". Essa definições, por serem abrangentes, desconsideram os diferentes tipos de tentativas maliciosas, que inclui a definição entre crime cibernético, ataque cibernético e guerra cibernética.

Apesar dos três tipos de ameaças serem parecidas entre elas, existem diferenças claras entre uma e outra, que permite a classificação em crime, ataque ou de uma guerra cibernética. O crime cibernético envolve apenas autores não governamentais e fazem a violação do direito penal, cometido por meio de um sistema de computador. O ataque

cibernético e a guerra cibernética possuem características semelhantes, como o objetivo de minar a função de uma rede de computadores e possuir propósito político ou de segurança nacional. A diferença entre as duas se dá pelo fato das ações feitas durante uma guerra cibernética serem executadas em contexto de conflito armado entre nações. Como podemos ver na Figura 2.2, os três tipos de ataque possuem intersecções, ou seja, uma ação cibernética pode ser considerada um crime, ataque e guerra ao mesmo tempo.

TABLE 1: Essential characteristics of different cyber-actions

	Type of cyber-action		
	Cyber-attack	*Cyber-crime*	*Cyber-warfare*
Involves only non–state actors		√	
Must be violation of criminal law, committed by means of a computer system		√	
Objective must be to undermine the function of a computer network	√		√
Must have a political or national security purpose	√		√
Effects must be equivalent to an "armed attack," or activity must occur in the context of armed conflict			√

Figura 2.2: Definição entre Crime, Ataque e Guerra Cibernética.

FIGURE 1: Relationship between cyber-actions

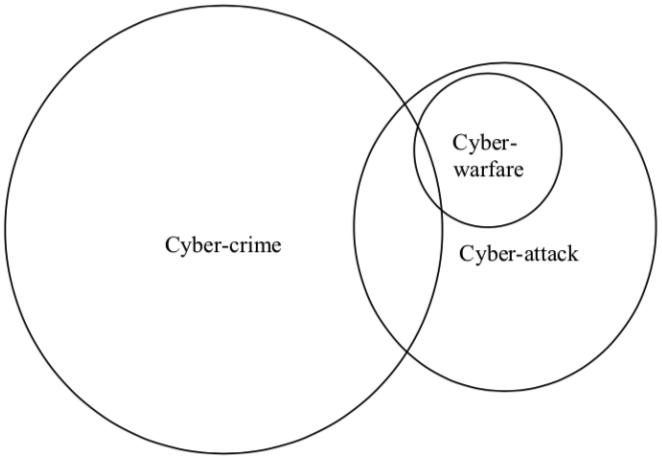


Figura 2.3: Relação entre Crime, Ataque e Guerra Cibernética.

Um exemplo de ataque que é considerado um crime cibernético, mas não é considerado um ataque cibernético, é um ator não-estatal está envolvido em atividade ilícita usando um computador ou rede, mas não prejudica a função de uma rede de computadores e não opera com um propósito político ou de segurança nacional. Uma pessoa que transfira pornografia infantil, por exemplo, cometeria um crime cibernético, mas não um ataque cibernético, porque as ações não prejudicam a função de uma rede de computadores e porque ele não é motivado por uma política ou segurança nacional.

2.3 Aprendizado de Máquina

As máquinas que foram programadas para aprender sempre foram um desafio na área de estudo da computação. Existem vários meios de se obter o aprendizado e também vários tipos de propósitos para se criar um modelo que, a partir do treinamento e estudo de uma grande quantidade de dados, consegue entregar resultados que o ser humano seria incapaz de resolver. As referências de definição encontram-se em [6].

Diz-se que um programa de computador aprende a partir da experiência "E"com respeito a um grande número de tarefas "T" e performance medida por "P", se a performance das tarefas em "T", como medidas em "P", melhoram a experiência de "E". Essa definição vem do autor Tom Mitchell no livro Machine Learning and Data Mining [7]. Em outra linguagem, um algoritmo pode aprender a atingir uma finalidade a partir de um grande volume de dados, que são suas experiências anteriores.

A partir dessa definição, pode-se citar vários exemplos de tarefas que envolvem algoritmos de aprendizado de máquina, como aprender a reconhecer palavras faladas, aprender a dirigir um veículo autônomo, e aprender a jogar xadrez. Dependendo do objeto de estudo e do objetivo, é utilizado um tipo de algoritmo diferente, que pode ter como objetivo predição, clusterização, regressão ou outros. Também deve-se analisar qual tipo de aprendizado de máquina será feito, se ele será supervisionado, não-supervisionado e semi-supervisionado.

2.3.1 Aprendizado Supervisionado

O aprendizado de máquinas supervisionado é aquele em que a máquina é treinada com um tabelas que possuem a resposta de desejada. Nesse caso, existe um pacote de dados que é utilizado para treino do algoritmo. Nesse pacote existe uma ou mais variáveis dependentes com valores que se dão a partir de uma ou mais variáveis independentes. Dessa forma, o algoritmo será treinado a partir de dados (variáveis independentes) com respostas concretas (variáveis dependentes) para se criar um modelo e, utilizando esse

modelo, tentará prever as respostas (variáveis dependentes) a partir das entradas futuras (variáveis independentes).

Um dos desafios desse tipo de aprendizado é trazer um treino com um equilíbrio correto entre viéis e variância dentro do conjunto de dados escolhido para treino, para que, ao aplicar o modelo treinado, o algoritmo consiga prever corretamente independente dos valores de entrada. Um exemplo desse tipo de aprendizado, é treinar um algoritmo com dados a respeito do histórico escolar de vários alunos e sua nota no exame ENEM, Exame Nacional do Ensino Médio. Com esse treino, a máquina seria capaz de prever com um certo nível de confiabilidade a futura nota de um aluno a partir de seu histórico escolar.

2.3.2 Aprendizado Não Supervisionado

O aprendizado de máquina não supervisionado é aquele que não se tem respostas anotadas para o treino do algoritmo. De forma geral, queremos achar uma representação mais informativa dos dados que temos que, em sua maioria, consiste em clusterizar os dados em grupos diferentes. A complexidade é consideravelmente maior do que em algoritmos supervisionados, pois deve-se ajustar os parâmetros do algoritmo, em sua maioria das vezes a partir da tentativa e erro, para se obeter uma precisão e menor erro. Isso se deve pela natureza dos algoritmos, que buscam em sua maioria apenas separar os dados em grupos diferentes e por isso, não existir respostas (variável dependente) para se treinar o programa.

Um exemplo desse tipo de aprendizado, é de classificar consumidores de uma loja a partir de suas compras anteriores em grupos de consumo diferente, com objetivo de gerar interligência de mercado para criar estratégias de aumento de faturamento.

2.3.3 Aprendizado Por Esforço

Esse tipo de aprendizado é feito a partir da análise das circunstâncias na qual a ação será executada. Nesse caso, é dado uma punição ou recompensa ao algoritmo, dependendo da decisão tomada. Com o tempo e a repetição de vários experimentos, espera-se que o agente consiga associar as ações para cada situação que o ambiente apresenta, afim de tomar a melhor decisão. A melhor decisão será feita a partir do histórico de experimentos e do resultado que obteve em cada um deles, se foi punição ou recompensa, em que o algoritmo irá priorizar sempre a decisão que trará mais recompensas. Em outras palavras, o aprendizado por esforço é aquele em que o algoritmo cria seu próprio data set de treino, com a variável dependente de punição ou recompensa, que traz o resultado de cada ação.

Um exemplo do uso desse algoritmo são os carros autônomos, que toma ações a partir da melhor ação (que possui maior recompensa) a partir análise do cenário e dos diversos inputs em seus sensores.

2.3.4 Clusterização

A clusterização é um problema de aprendizado de máquina não supervisionado pelo qual pretende-se agrupar subconjuntos de entidades uns com os outros com base em alguma noção de similiaridade. Dessa forma é possível destrinchar o *big data* em tabelas menores de dados para que seja feito uma análise mais aprofundada a respeito do tema. Em sua maioria, a clusterização faz parte de um dos passos de um *pipeline* em que são aplicados mais de um tipo de algoritmo de aprendizado de máquina para se obter o resultado esperado. Essa definição encontra-se na referência [8] e podemos ver um exemplo desse tipo de algoritmo na Figura 2.4.

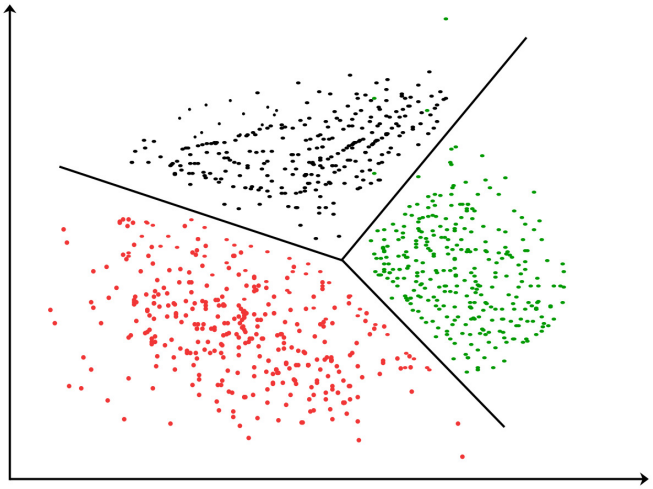


Figura 2.4: Exemplo de clusterização utilizando aprendizado de máquina.

2.3.5 Regressão

A regressão é uma medida que busca traduzir a relação entre duas variáveis, sendo uma dependente e outra não. Existem vários tipos de regressão e cada um possui um resultado distinto, como por exemplo, a regressão linear usa apenas uma variável independente prever o resultado da variável dependente, enquanto a regressão múltipla usa duas ou mais variáveis independentes para essa predição. Existe também a regressão logística, que a partir da análise da relação entre as variáveis, traz um resultado binário se um cojunto de dados faz parte ou não de um grupo específico.

2.3.6 Algoritmo de Clusterização: K-Means

O método de clusterização k-means é utilizado para particionar automaticamente conjunto de dados em k grupos. O algoritmo iterativo pode ser dividido em duas etapas: atribuição de cluster e movimento do centroide. A primeira etapa consiste em escolher uma posição inicial para os k centros de grupos. A segunda etapa consiste na movimentação dos centroides de grupos, feita a partir do cálculo da média entre a distância dos pontos pertencentes aquele grupo. Após calcular essa distância média, o centroide se movimenta para aquela localização, considerada como centro do cluster. Essa segunda etapa é iterada repetitivamente até que a mudança de localização do centroide seja muito pequena, definida pelo epsilon (parâmetro do algoritmo), ou até que a quantidade máxima de iterações seja atingida (parâmetro do algoritmo). Mais informações sobre a definição e variações do algoritmo podem ser encontradas no livro [9].

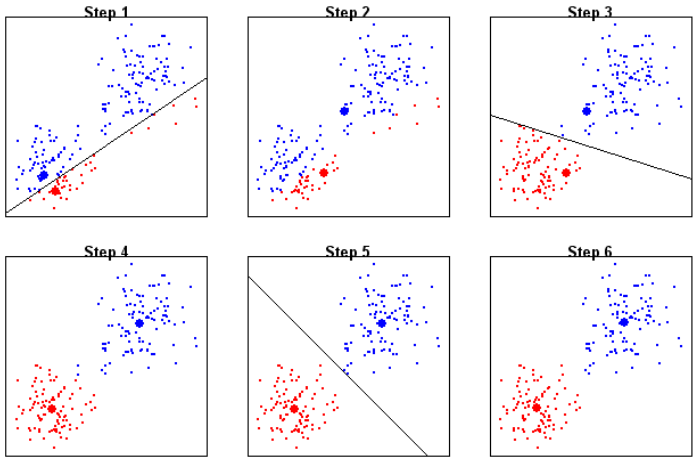


Figura 2.5: Recursão do Algoritmo K-means para encontrar o melhor centro.

Os parâmetros de entrada que devem ser modificados para melhor adaptação do algoritmo ao tipo de dado são: K (quantidade de clusters desejado), MaxInterations (máximo de interações que o algoritmo deve fazer), initializationMode (randomico ou com a posição de cada um dos k centros), epsilon (limiar da distância do elemento mais longe do centro). Existe uma grande dificuldade em encontrar uma convergência nesse modelo de algoritmo pois ele é baseado em aprendizado de máquina não supervisionado. Dessa forma, ao desconhecer o tamanho de possíveis clusters presentes do conjunto de dados, deve-se medir o erro quadrático máximo a partir de tentativa e erro para se obter um número que traz uma confiabilidade para o algoritmo, conforme citado na referência [10].

2.4 Ferramentas Utilizadas

As ferramentas utilizadas para esse trabalho baseam-se na arquitetura do Hadoop. Cada uma dessas ferramentas possui uma arquitetura diferente e soluções para manipulação de grandes quantidades de dados de forma confiável e persistente, fazendo a leitura, escrita e manipulação de forma paralela com o objetivo de aumentar a eficiência do algoritmo. Foi utilizando a solução web do *Cloudera Manager* como ferramenta para configuração e integração entre as diferentes ferramentas utilizadas. Nas sessões a seguir são apresentadas breve explicações sobre a arquitetura de cada uma das ferramentas utilizadas nesse trabalho.

2.4.1 CDH

O CDH é a distribuição do Apache Hadoop mais completa e popular. Ele possui os elementos centrais do Hadoop - um sistema de armazenamento distribuído e escalável, criado para utilização de *big data*. O CDH provê de características que permitem a flexibilidade, integração, segurança, escalabilidade, alta disponibilidade e compatibilidade - elementos de extrema importância para desenvolvimentos de projetos com grandes quantidades de dados. É licenciado pela Apache, é *open source* e possui uma versatilidade de aplicativos opcionais para trabalhar em diversas vertentes do *big data*, como banco de dados não relacionais (HBase), *streaming* e aprendizado de máquina (Spark), SQL (Impala) entre outros, como mostra figura Figura 2.6 [11]

2.4.2 HDFS

O *Hadoop Distributed File System* é um sistema de distribuição padrão de dados padrão do Hadoop, com foco em ser altamente tolerante a falhas e desenhado para ser implantado em *hardware* de baixo custo. Sua arquitetura padrão possui clusters paralelos com blocos de dados de tamanho padrão e, para evitar a perda de informação, o HDFS replica cada bloco três vezes. Dessa forma, cada arquivo possui três cópias distribuídas ao longo do cluster que trabalham de forma paralela garantindo que a falha de *hardware*, algo comum ao se trabalhar com grandes quantidades de dados, seja tratada. Mais informações sobre o Hadoop podem ser encontradas na referência [12].

2.4.3 Cloudera Manager

O *Cloudera Manager* é um dos produtos desenvolvidos pela Cloudera Inc. [13], uma empresa americana com sede na Califórnia. O *Cloudera Manager* é uma aplicação *end-to-end* criada com objetivo de facilitar a gerência e configuração de sistemas de processamento

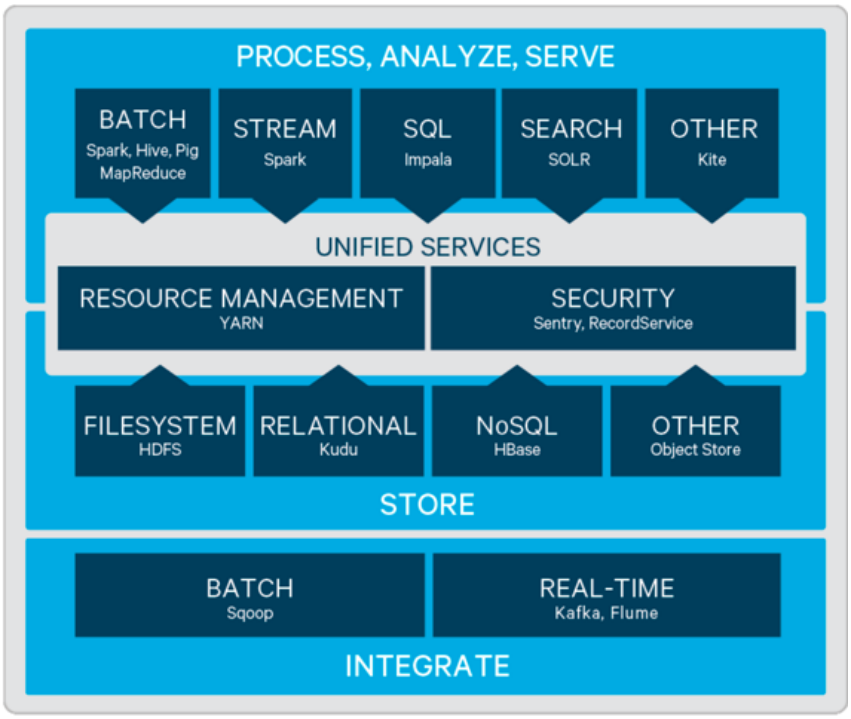


Figura 2.6: Exemplo de Arquitetura CDH.

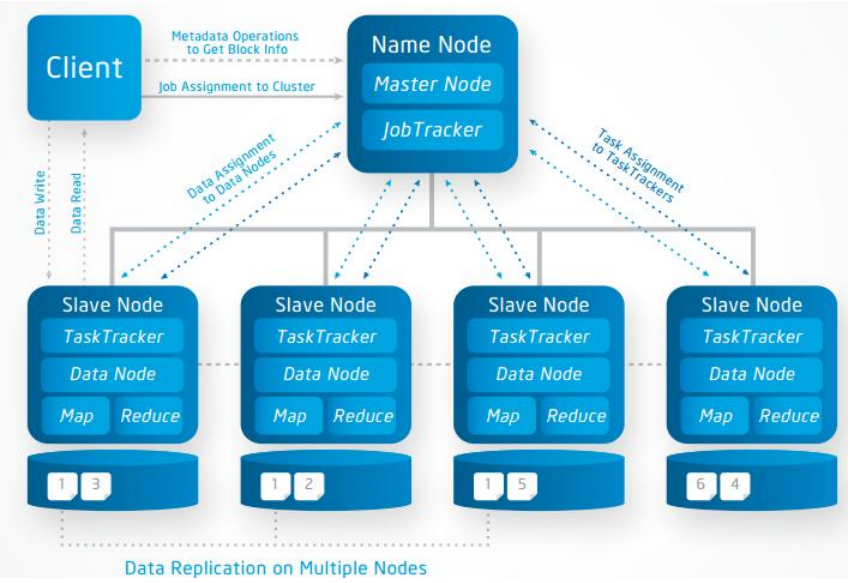


Figura 2.7: Arquitetura de Arquivos HDFS.

e clusters do CDH. Ele possui uma interface web amigável que permite o usuário a criar visualizações real time a respeito da performance de cada cluster, assim como as instâncias de cada um das aplicações utilizadas em cada um dos nós. Entre outras funções, o *Cloudera Manager* permite ver o status da saúde de cada um dos aplicativos, fazer a

auditoria de governança e criar políticas para proteção de acesso aos dados.

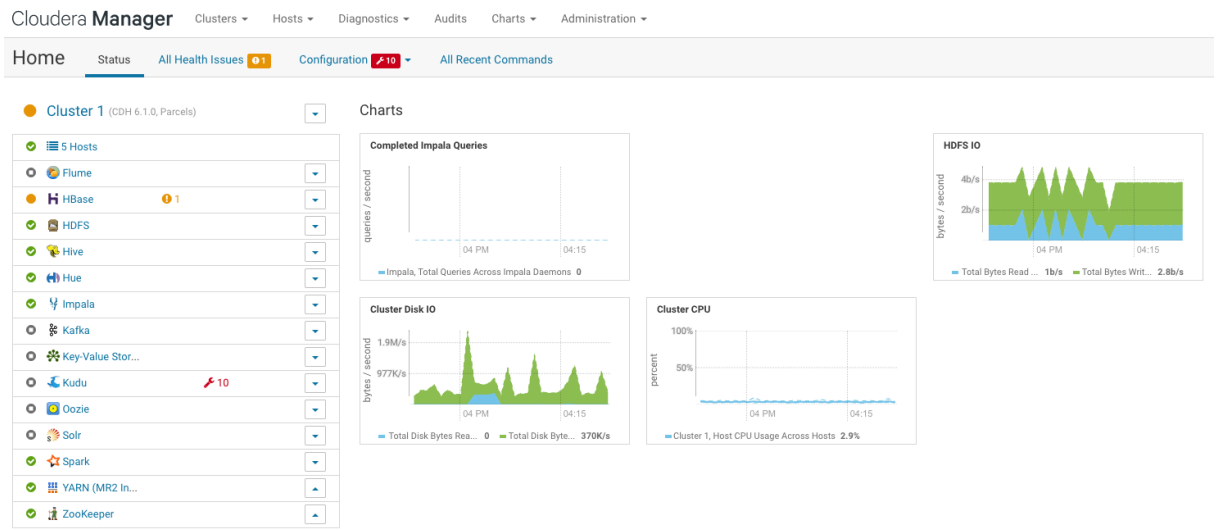


Figura 2.8: Cloudera Manager - Página Inicial.

2.4.4 *YARN - Yet Another Resource Negociator*

O *Yet Another Resource Negotiator (YARN)* é uma peça fundamental para gerenciamento de recursos do cluster. Ela permite que as tarefas sejam executadas continuamente, permitindo que sejam feitas análises de dados em *streaming* e em tempo real, por exemplo. O YARN é uma camada intermediária entre as aplicações do cluster, como por exemplo o canal de comunicação que gera recursos entre o Spark e os nós do clusters, podendo alocar mais ou menos executores, de acordo com a tarefa. Ele tem como objetivo também gerenciar os recursos para evitar desbalanceamento de carga entre os nós do cluster, além da replicação de código de controle.

Dentre diversas vantagens de se utilizar essa ferramenta, estão a alta escalabilidade podendo chegar até 10.00 nós executando 100.000 tarefas ao mesmo tempo e a alta disponibilidade por trazer uma grande vazão, com balanceamento de recurso do cluster. Ela é extremamente recomendada para utilização de projetos que utilizam processamento paralelo e é instalada de como gerenciador padrão em todas as ferramentas utilizadas pela ferramenta *Cloudera Manager*.

2.4.5 Spark

Apache Spark é um sistema de computação em cluster que provê de APIs em Java, Scala, Python e R. Ele possui vastas ferramentas e biblioteca facilitam o uso de SQL (Spark

SQL), *machine learning* (MLlib), gráficos (GraphX) e *streaming* (Spark Streaming) nos dados. Ele utiliza a infraestrutura fornecida pelo HDFS para criar soluções em paralelo e confiáveis para utilização de enormes quantidades de dados. Dessa forma, o Spark possui o componente de armazenamento de dadods com base no Hadoop (HDFS), a API que é o núcleo que permite que as instruções sejam executadas (Python, Java, Scala ou R) e a ferramenta de gerenciamento que pertmite que as tarefas sejam executadas de forma local (indicado para testes) ou no cluster, que pode ser feita de forma paralela (indicado para alta performance).

O Spark possui como estrutura base para as operações, que permite sua alta performance e confiabilidade, o *RDD - Resilient Distributed Dataset*. As bibliotecas utilizadas para tratamento de dados em grande escalas possuem como argumento de entrada de dados no formado RDD, que trazem uma abstração que permite que a coleção de dados sejam operadas paralelamente nos nós do *cluster*. Existem duas maneiras de se obter dados no formato RDD: a primeira delas é criando os dados no programa e depois chamando uma função de paralização; a segunda delas é importar um arquivo externo localizado no HDFS uma tabela em csv, json ou outros modelos de arquivos. É possível também salvar uma variável no formato RDD para ser reutilizada em outros programas. Outros pontos importantes do RDD é de ser resiliente a falhas entre nós e partilhar variáveis entre nós, características de extrema importância quando para o processamento em paralelo.

Ao inicializar um programa utilizando Spark, é necessário criar um objeto SparkContext, responsável por ser um driver da conexão entre a aplicação spark e o cluster, um objeto que deve ser único dentro da aplicação. Além disso, ele pode exercer tarefas como inserir configurações, cancelar um processo, acessar um RDD persistido, entre outras funções, como podemos ver na referência [14].

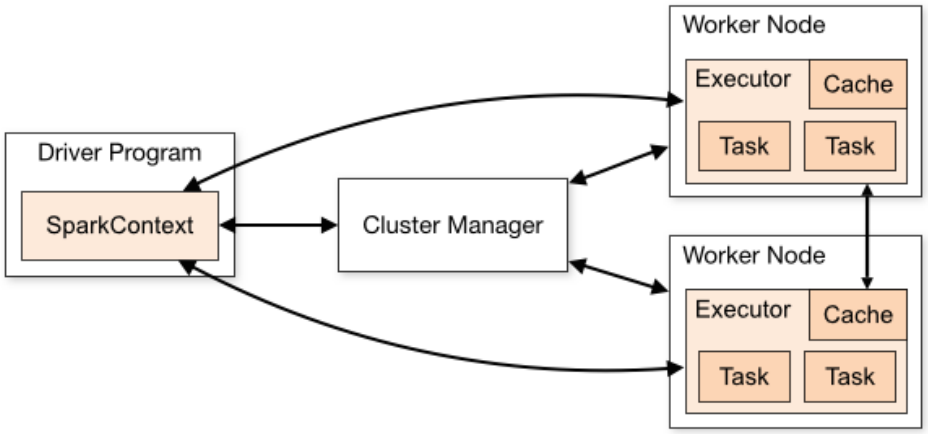


Figura 2.9: Arquitetura Spark.

2.4.6 Spark MLlib - *Machine Learning Library*

A biblioteca MLlib é uma poderosa ferramenta com objetivo de fornecer um conjunto de APIs que ajudam a implementação *pipelines* de aprendizado de máquina utilizando o Spark. Ela possui algoritmos que permitem a classificação, regressão, clusterização, filtro colaborativo entre outras ferramentas com foco em estatística.

Existem quatro componentes importantes para implementação de algoritmos utilizando a bibliotéca MLlib. O primeiro deles é o conjunto de dados que é baseado no formato de arquivo RDD e suporta uma grande variedade de dados, como texto, vetores e números. O segundo são as funções de transformação, que transformam uma tabela em outra, com objetivo de organizar os dados ou reduzi-los para melhor processamento. Já o terceiro, são as funções estimadoras, que criam uma informação nova no conjunto de dados a partir de uma operação nos dados, podendo ser uma estimação. O resultado dessa função, no caso do treinamento de um algoritmo, é a criação de um modelo de aprendizado de máquina, que por sua vez, é uma função transformadora. O ultimo componente é o *pipeline* que encadeia diversos transformadores e estimadores em uma ordem específica para criar um fluxo de trabalho de aprendizado de máquina. Dessa forma, um pipeline pode possuir diversas funções transformadoras e estimadores, dependendo do tipo de dado que será tratado e do objetivo do algoritmo.

2.4.7 HUE

A ferramenta *Hadoop User Experience* é um *software* livre para operação de aplicações em SQL que faz interface com os componentes do Hadoop. Ela possui uma arquitetura capaz de fornecer eficiência no armazenamento de tabelas e nas buscas em SQL. Ela é uma aplicação web e pode ser acessada pela porta 8889, tornando-se de fácil acesso e entendimento do usuário. Com o HUE é possível salvar buscas, criar *dashboards* em cima de fluxo de trabalho para visualizar dados em transmissão e acompanhar o comportamento de grandes quantidades de dados em tempo real com uma boa eficiência e capacidade de processamento.

Esse capítulo mostrou, de forma teórica, uma breve história do surgimento de *big data*, assim como sua definição mais aceitada nos dias de hoje. O capítulo também explicou brevemente as arquiteturas das ferramentas utilizadas para a confecção desse trabalho, assim como qual uso cada uma das ferramentas teve durante o desenvolvimento. Além disso, propôs uma base teórica a respeito de aprendizado de máquina, necessário para o entendimento do desenvolvimento do projeto, como mostrado no Capítulo 3.

Capítulo 3

Projeto e Implementação

3.1 Implementação

Para esse projeto, foi utilizado a ferramenta Spark com a linguagem Python, desenvolvido utilizando HDFS, PySpark, YARN, HIVE e HUE. O HDFS é o banco de dados baseado no sistema Hadoop que serve como base para todas persistências do projeto relacionadas a *big data*. O PySpark é um *shell* interativo para programação em Python que utiliza de bibliotecas de aprendizado de máquina como a biblioteca MLlib. Além disso, foram utilizadas as bibliotecas MLlib e Numpy em Python. Os códigos desse projeto encontram-se para consulta nos Apêndices A, B e C.

3.2 Ferramentas Utilizadas

As aplicações utilizadas para o desenvolvimento do trabalho foram o HDFS, YARN, Spark, HIVE e HUE. Para configuração, integração e definição dos parâmetros, foi utilizado o *Cloudera Manager* a fim de conectar e sincronizar as ferramentas. O *Cloudera Manager* permite configurar utilizando uma aplicação web com interface amigável. O desenvolvimento do trabalho e processamento do algoritmo foi feito utilizando o terminal com uma conexão SSH com o cluster. Para desenvolvimento, testes e depuração, foi utilizado o *shell* interativo PySpark, com objetivo de visualizar de forma mais ágil os dados, pois o ambiente de trabalho ficam com as variáveis salvas na mesma sessão.

Além disso, o PySpark cria uma abstração referente a inicialização da SparkSession e o SparkContext, que contém ferramentas necessárias para leitura e escrita de dados no administrador de arquivos HDFS. O PySpark é inicializado utilizando o YARN como ferramenta master responsável pela gestão de recurso que, por sua vez, faz a gestão de executores em cada um dos nós do cluster para garantir uma eficiência no algoritmo. O YARN faz também a gestão de memória e garante que a paralelilzação ocorra de

maneira confiável e escalável. O editor de texto utilizado foi o VIM, editor disponível no terminal de sistemas Unix. A linguagem desenvolvida no trabalho foi Python, utilizando as bibliotecas Numpy, MLlib e Spark. Para criar tabelas inseridas no HDFS, foi utilizado o *shell* interativo do HIVE, como exemplificado no Apêndice 2, que possibilita a criação de tabelas que ficam visíveis na ferramenta HUE. A ferramenta HUE foi acessada através de uma aplicação web utilizando a porta 8889. Essa aplicação permite a criação de buscas em SQL, disponíveis no Apêndice C. Além disso, o HUE permite a criação de gráficos como o de linhas, barras e *scatter plot*.

3.3 Coletando o Dado

O conjunto de dados utilizado para o estudo foi retirado do site Norse Corp que disponibiliza de forma gratuita informações sobre ataques cibernéticos em tempo real. A empresa Norse Corp tem como principal solucao de seguranca chamada de IPViking, um programa responsavel por monitorar ativamente as aplicacoes e maquinas de seus clientes listando os ataques que estao ocorrendo em tempo real. Dessa forma, e possivel identificar a fonte e tipo da ameaca permitindo sanar as vulnerabilidades e reagir de forma proativa contra elas. Os dados foram coletados do site entre os dias 26 de março de 2017 e o dia 16 de fevereiro de 2018 e salvos em um arquivo formato JSON. O domínio da Norse Corp, encontra-se atualmente fora do ar.

A tabela retirada do site possui 19 colunas diferentes, com informações distintas do ataque como país, cidade, latitude e longitude do país de origem do ataque e do receptor do ataque, a porta afetada, o IP do atacante, o time stamp, entre outros, como podemos ver na Figura 3.1. O arquivo utilizado para o estudo possui 72.163.089 linhas, contendo 32 *GigaByte* de dados. Após inserida no HDFS através de um arquivo JSON, a tabela pôde ser lida pelo PySpark.

```
>>> parquet.show()
+--------------------+--------------------+--------------------+-------+--------+-----------+-----------+-----+--------+---------+---------+---------+--------------------+--------------------+-----+--------------------+--------------------+--------------------+----+
|                 _id|                city|               city2|country|country2|countrycode|countrycode2|dport|latitude|latitude2|longitude|longitude2|                md5|                 org|  svc|           timestamp|                type|           vector_id|zerg|
+--------------------+--------------------+--------------------+-------+--------+-----------+-----------+-----+--------+---------+---------+---------+--------------------+--------------------+-----+--------------------+--------------------+--------------------+----+
|[592f7a6be2e55802...|League City|De Kalb Junction|     US|      US|         US|         US|  443|   29.5|    44.48|   -95.09|    -75.3|  50.200.102.48|American National...|  443|2017-06-01 02:22:...|ipviking.honey|[303241851924|null|
|[592f7a6be2e55802...|      Geneve|              Dubai|     CH|      AE|         CH|         AE|  123|   46.2|    25.27|     6.14|    55.33|  185.35.62.205|This Ip Network I...|  123|2017-06-01 02:22:...|ipviking.honey|[303241851914|null|
|[592f7a6be2e55802...|      Ningbo|           Lynnwood|     CN|      US|         CN|         US|50864|   29.86|     47.8|   121.55|  -122.29| 60.178.154.200|Chinanet-Zj Ningb...|50864|2017-06-01 02:22:...|ipviking.honey|[303241851904|null|
|[592f7a6ce2e55802...| Washington|De Kalb Junction|     US|      US|         US|         US|   25|   38.95|    44.48|   -77.02|    -75.3|  65.55.169.252|Microsoft Corpora...|   25|2017-06-01 02:22:...|ipviking.honey|[303241851894|null|
|[592f7a6ce2e55802...|    Redmond|De Kalb Junction|     US|      US|         US|         US|   25|   47.68|    44.48|   -122.12|    -75.3|  157.56.111.246|Microsoft Corpora...|   25|2017-06-01 02:22:...|ipviking.honey|[303241851884|null|
|[592f7a6ce2e55802...|    Redmond|De Kalb Junction|     US|      US|         US|         US|   25|   47.68|    44.48|   -122.12|    -75.3|  207.46.100.253|Microsoft Corpora...|   25|2017-06-01 02:22:...|ipviking.honey|[303241851874|null|
|[592f7a6de2e55802...|   Las Vegas|           Las Vegas|     US|      US|         US|         US|  138|   36.18|    36.18|   -115.12|   -115.12|   70.96.87.192|    Syptec Las Vegas|  138|2017-06-01 02:22:...|ipviking.honey|[303241851864|null|
|[592f7a6de2e55802...|      Rosario|   San Francisco|     AR|      US|         AR|         US|   23|  -32.95|    37.79|    -68.67|   -122.4|  200.69.233.229|             Nss S.A.|   23|2017-06-01 02:22:...|ipviking.honey|[303241851854|null|
|[592f7a6ee2e55802...|        Nola|De Kalb Junction|     IT|      US|         IT|         US|   23|   40.95|    44.48|    14.53|    -75.3|  95.241.212.48|Telecom Italia S....|   23|2017-06-01 02:22:...|ipviking.honey|[303241851844|null|
|[592f7a6ee2e55802...|Kansas City|              Dubai|     US|      AE|         US|         AE|   22|   39.15|    25.27|    -94.57|    55.33|    208.67.1.59|Wholesale Data Ce...|   22|2017-06-01 02:22:...|ipviking.honey|[303241851834|null|
|[592f7a6ee2e55802...|    New York|                Oslo|     US|      NO|         US|         NO| 1900|    40.8|    59.93|    -73.97|    10.79|184.105.139.109|Hurricane Electri...| 1900|2017-06-01 02:22:...|ipviking.honey|[303241851824|null|
|[592f7a6ee2e55802...| Washington|De Kalb Junction|     US|      US|         US|         US|   25|   38.95|    44.48|   -77.02|    -75.3|  65.55.169.247|Microsoft Corpora...|   25|2017-06-01 02:22:...|ipviking.honey|[303241851814|null|
|[592f7a6fe2e55802...|    Redmond|De Kalb Junction|     US|      US|         US|         US|   25|   47.68|    44.48|   -122.12|    -75.3|  207.46.100.252|Microsoft Corpora...|   25|2017-06-01 02:22:...|ipviking.honey|[303241851804|null|
|[592f7a6fe2e55802...|     Tianjin|   San Francisco|     CN|      US|         CN|         US|53413|   39.09|    37.79|    117.22|   -122.4|   42.122.47.44|Chinanet Tianjin ...|53413|2017-06-01 02:22:...|ipviking.honey|[303241851794|null|
|[592f7a70e2e55802...|     Chungju|   San Francisco|     KR|      US|         KR|         US|53413|   36.97|    37.77|    127.94|  -122.41|175.202.157.144|     Korea Telecom|53413|2017-06-01 02:22:...|ipviking.honey|[303241851784|null|
|[592f7a70e2e55802...|      Fresno|   San Francisco|     US|      US|         US|         US| 9755|   36.86|    37.79|   -119.76|   -122.4|  206.78.213.30|Kings County Offi...| 9755|2017-06-01 02:22:...|ipviking.honey|[303241851774|null|
|[592f7a71e2e55802...| Washington|De Kalb Junction|     US|      US|         US|         US|   25|   38.95|    44.48|    -77.02|    -75.3|  65.55.169.251|Microsoft Corpora...|   25|2017-06-01 02:22:...|ipviking.honey|[303241851764|null|
|[592f7a71e2e55802...|    Redmond|De Kalb Junction|     US|      US|         US|         US|   25|   47.68|    44.48|   -122.12|    -75.3|  207.46.100.246|Microsoft Corpora...|   25|2017-06-01 02:22:...|ipviking.honey|[303241851754|null|
|[592f7a71e2e55802...| Washington|De Kalb Junction|     US|      US|         US|         US|   25|   38.95|    44.48|    -77.02|    -75.3|  65.55.169.248|Microsoft Corpora...|   25|2017-06-01 02:22:...|ipviking.honey|[303241851744|null|
|[592f7a72e2e55802...|    Redmond|De Kalb Junction|     US|      US|         US|         US|   25|   47.68|    44.48|   -122.12|    -75.3|  157.56.111.251|Microsoft Corpora...|   25|2017-06-01 02:22:...|ipviking.honey|[303241851734|null|
+--------------------+--------------------+--------------------+-------+--------+-----------+-----------+-----+--------+---------+---------+---------+--------------------+--------------------+-----+--------------------+--------------------+--------------------+----+
only showing top 20 rows
```

Figura 3.1: Tabela dos Ataques Retirada da Norse Corp.

3.4 Preparação do Dado

Antes de trabalhar com dados em larga escala, é necessário conhecer as características da tabela será utilizada como objeto de estudo. Para isso, foram utilizadas algumas funções nativas do spark para manipulação de tabelas, como a *printSchema, describe e summary*, que trazem informações que permitem conhecer melhor a estrutura da tabela. Com objetivo de facilitar a manipulação da tabela contendo os dados dos ataques em dados de leitura e processamento mais ágil, o JSON foi persistido no HDFS no formato CSV. Com isso, é possível fazer uma separação de colunas de maneira mais simples, pois as colunas ficam separadas por vírgulas, enquanto as linhas por quebra de linha.

O resultado da aplicação da função *printSchema* pode ser vista na Figura 3.2, contendo características de cada coluna, como se a tabela pode conter dados nulos ou não e o tipo de dado de cada uma das colunas.

```
>>> parquet.printSchema()
root
 |-- _id: struct (nullable = true)
 |    |-- $oid: string (nullable = true)
 |-- city: string (nullable = true)
 |-- city2: string (nullable = true)
 |-- country: string (nullable = true)
 |-- country2: string (nullable = true)
 |-- countrycode: string (nullable = true)
 |-- countrycode2: string (nullable = true)
 |-- dport: long (nullable = true)
 |-- latitude: double (nullable = true)
 |-- latitude2: double (nullable = true)
 |-- longitude: double (nullable = true)
 |-- longitude2: double (nullable = true)
 |-- md5: string (nullable = true)
 |-- org: string (nullable = true)
 |-- svc: long (nullable = true)
 |-- timestamp: string (nullable = true)
 |-- type: string (nullable = true)
 |-- vector_id: struct (nullable = true)
 |    |-- $numberLong: string (nullable = true)
 |-- zerg: string (nullable = true)
```

Figura 3.2: Esquema da Tabela Contendo os Ataques.

3.5 Transformação do Dado

Pelo fato de a tabela ter sido importada de uma fonte que já organizava os dados, não foi necessária nenhuma arrumação da tabela. Os dados da tabela não possuíam valores nulos nem *outliers*, algo que facilitou a parte de transformação. O único trabalho feito nessa sessão, foi de selecionar as colunas importantes para o estudo. Foram separados em quatro tabelas menores, as colunas importantes para aplicação de cada estudo feito nesse trabalho. A primeira tabela contém a seleção das colunas "dport"e "org". A segunda, as

colunas "dport", "latitude2", "longitude2". A terceira tabela, "dport"e "md5". A quarta e última tabela, apenas a coluna "dport". O código desenvolvido para realizar essa seleção está disponível no Apêndice A.

Esses conjunto de dados foram selecionados a partir do quadro de dados contendo todas as informações do ataque e salvos em CSV para manipulação posterior. Para isso, foi usado o PySpark, aplicando funções já implementadas de bibliotecas nativas para manipulação de dados. Essas funções pertencem a biblioteca da Spark Session, criada na inicialização do PySpark de forma automatizada. Dessa forma, foram criados quatro arquivos novos no HDFS, cada um deles contendo subconjuntos menores que serão utilizados para desenvolvimento do projeto.

3.6 Aplicação do K-Means

Para aplicação do algoritmo k-means da biblioteca MLlib, nativa do Spark, foi criado o programa k-means.py. Ele é divido em quatro partes: leitura no banco de dados, indexação de string, criação de vetor, aplicação do k-means e escrita no banco de dados. O algoritmo desenvolvido em Python está disponível no apêndice A.

3.6.1 Parte 1: Lendo o dado

A leitura é feita com as funções nativas da Spark Session que leem um arquivo CSV e o transformam em um quadro de dados.

```
>>> df.show()
+-----+--------------------+
|dport|                 org|
+-----+--------------------+
|  443|American National...|
|  123|This Ip Network I...|
|50864|Chinanet-Zj Ningb...|
|   25|Microsoft Corpora...|
|   25|Microsoft Corpora...|
|   25|Microsoft Corpora...|
|  138|    Syptec Las Vegas|
|   23|           Nss S.A.|
|   23|Telecom Italia S....|
|   22|Wholesale Data Ce...|
| 1900|Hurricane Electri...|
|   25|Microsoft Corpora...|
|   25|Microsoft Corpora...|
|53413|Chinanet Tianjin ...|
|53413|       Korea Telecom|
| 9755|Kings County Offi...|
|   25|Microsoft Corpora...|
|   25|Microsoft Corpora...|
|   25|Microsoft Corpora...|
|   25|Microsoft Corpora...|
+-----+--------------------+
only showing top 20 rows
```

Figura 3.3: Tabela dos Ataques Inicial.

3.6.2 Parte 2: String Indexer

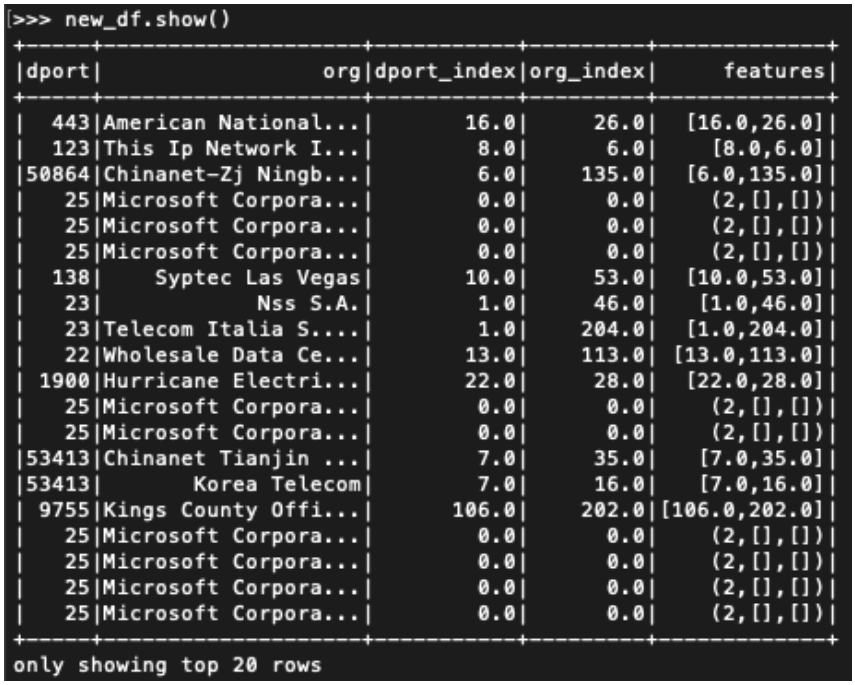
A indexação de *string* é necessária para transformar as colunas que estão em *string* em formato numérico, criando um index para cada *string* diferente, para que possa ser aplicado o algoritmo de aprendizado de máquina, que não aceita dados em *string*. Para essa tarefa, foi utilizada a função *StringIndexer* da biblioteca nativa de aprendizado de máquina. Como a função deve ser aplicada em duas colunas diferentes, foi criado um *pipeline* contendo uma função para cada objeto da lista de colunas. Depois da criação do *pipeline*, foi feito a transformação do dado, criando uma coluna nova com o nome *pipeline*, como mostra a Figura 3.4.

```
>>> df_r.show()
+-----+--------------------+-----------+---------+
|dport|                 org|dport_index|org_index|
+-----+--------------------+-----------+---------+
|  443|American National...|       16.0|     26.0|
|  123|This Ip Network I...|        8.0|      6.0|
|50864|Chinanet-Zj Ningb...|        6.0|    135.0|
|   25|Microsoft Corpora...|        0.0|      0.0|
|   25|Microsoft Corpora...|        0.0|      0.0|
|   25|Microsoft Corpora...|        0.0|      0.0|
|  138|    Syptec Las Vegas|       10.0|     53.0|
|   23|           Nss S.A.|        1.0|     46.0|
|   23|Telecom Italia S....|        1.0|    204.0|
|   22|Wholesale Data Ce...|       13.0|    113.0|
| 1900|Hurricane Electri...|       22.0|     28.0|
|   25|Microsoft Corpora...|        0.0|      0.0|
|   25|Microsoft Corpora...|        0.0|      0.0|
|53413|Chinanet Tianjin ...|        7.0|     35.0|
|53413|       Korea Telecom|        7.0|     16.0|
| 9755|Kings County Offi...|      106.0|    202.0|
|   25|Microsoft Corpora...|        0.0|      0.0|
|   25|Microsoft Corpora...|        0.0|      0.0|
|   25|Microsoft Corpora...|        0.0|      0.0|
|   25|Microsoft Corpora...|        0.0|      0.0|
+-----+--------------------+-----------+---------+
only showing top 20 rows
```

Figura 3.4: Tabela com Index.

3.6.3 Parte 3: Vector Assembler

Para que seja utilizado mais de uma variável na clusterização, faz-se necessário a junção de todas as colunas de variáveis em uma só, no formato de vetor. Isso foi feito utilizando a função *VectorAssembler*, nativo da própria biblioteca de aprendizado de máquina do Spark. A função tem como entrada um quadro de dados, com a indicação do nome das colunas de entrada e o nome da coluna que deverá ser criada como saída. As colunas utilizadas como entrada, foram aquelas indexadas pela função *String Indexer*, descrita na subseção anterior. Dessa forma, é possível utilizar unir todos os dados necessários para o estudo da clusterização utilizando o algoritmo k-means.



```
>>> new_df.show()
+-----+--------------------+----------+---------+-------------+
|dport|                 org|dport_index|org_index|     features|
+-----+--------------------+----------+---------+-------------+
|  443|American National...|      16.0|     26.0|  [16.0,26.0]|
|  123|This Ip Network I...|       8.0|      6.0|    [8.0,6.0]|
|50864|Chinanet-Zj Ningb...|       6.0|    135.0|  [6.0,135.0]|
|   25|Microsoft Corpora...|       0.0|      0.0|    (2,[],[])|
|   25|Microsoft Corpora...|       0.0|      0.0|    (2,[],[])|
|   25|Microsoft Corpora...|       0.0|      0.0|    (2,[],[])|
|  138|    Syptec Las Vegas|      10.0|     53.0|  [10.0,53.0]|
|   23|            Nss S.A.|       1.0|     46.0|   [1.0,46.0]|
|   23|Telecom Italia S....|       1.0|    204.0|  [1.0,204.0]|
|   22|Wholesale Data Ce...|      13.0|    113.0| [13.0,113.0]|
| 1900|Hurricane Electri...|      22.0|     28.0|  [22.0,28.0]|
|   25|Microsoft Corpora...|       0.0|      0.0|    (2,[],[])|
|   25|Microsoft Corpora...|       0.0|      0.0|    (2,[],[])|
|53413|Chinanet Tianjin ...|       7.0|     35.0|   [7.0,35.0]|
|53413|       Korea Telecom|       7.0|     16.0|   [7.0,16.0]|
| 9755|Kings County Offi...|     106.0|    202.0|[106.0,202.0]|
|   25|Microsoft Corpora...|       0.0|      0.0|    (2,[],[])|
|   25|Microsoft Corpora...|       0.0|      0.0|    (2,[],[])|
|   25|Microsoft Corpora...|       0.0|      0.0|    (2,[],[])|
|   25|Microsoft Corpora...|       0.0|      0.0|    (2,[],[])|
+-----+--------------------+----------+---------+-------------+
only showing top 20 rows
```

Figura 3.5: Tabela com vetor indexado.

3.6.4 Parte 4: Aplicação do K-means

Após a preparação do quadro de dados, a fim de criar uma clusterização de K partes dos quatro subconjuntos diferentes de dados, foi criado um modelo de algoritmo k-means utilizando o tamanho de k igual a 10.

O primeiro parâmetro é referente ao número de clusters desejados na análise. O número 10 foi escolhido de forma arbritrária com o objetivo de obter resultados com visualizações mais simples. Dessa forma, os subconjuntos analisados foram separados em 10 grupos diferentes, cada um deles contendo características semelhantes entre si.

Primeiro foi feito a adaptação (*FIT*) do modelo k-means na coluna *features*, que representa a coluna com o vetor de todas as variáveis indexadas que foram selecionadas para a clusterização. Após o uso do *FIT*, foi utilizado o *transform* do modelo, tendo como parâmetro de entrada o quadro de dados. Na segunda etapa, é criado uma nova coluna no quadro de dados com o número do grupo ao qual aquela linha pertence, com características similares aquelas linhas que estão no mesmo grupo de cluster, como podemos ver na figura Figura 3.6.

Após a criação da nova coluna chamada *predicton*, que contém o número de cluster K em que a linha pertence, essa tabela foi persistida no HDFS no formato CSV, para que seja utilizada pelo HIVE, para criação de visualizações que permitem a análise da eficácia do modelo criado para essa clusterização, que serão demonstrados no Capítulo 4 com os

resultados encontrados.

```
>>> df.show()
+-----+--------------------+---+
|  _c0|                 _c1|_c2|
+-----+--------------------+---+
|  443|American National...|  5|
|  123|This Ip Network I...|  0|
|50864|Chinanet-Zj Ningb...|  2|
|   25|Microsoft Corpora...|  6|
|   25|Microsoft Corpora...|  6|
|   25|Microsoft Corpora...|  6|
|  138|    Syptec Las Vegas|  5|
|   23|            Nss S.A.|  5|
|   23|Telecom Italia S....|  4|
|   22|Wholesale Data Ce...|  9|
| 1900|Hurricane Electri...|  5|
|   25|Microsoft Corpora...|  6|
|   25|Microsoft Corpora...|  6|
|53413|Chinanet Tianjin ...|  5|
|53413|       Korea Telecom|  0|
| 9755|Kings County Offi...|  1|
|   25|Microsoft Corpora...|  6|
|   25|Microsoft Corpora...|  6|
|   25|Microsoft Corpora...|  6|
|   25|Microsoft Corpora...|  6|
+-----+--------------------+---+
only showing top 20 rows
```

Figura 3.6: Tabela clusterizada.

3.7 HIVE e HUE

A ferramenta HIVE foi utilizada atraés de um *shell* interativo no terminal para criar tabelas no formado ORC persistidas no HDFS. Para isso, foi necessário criar tabelas no formato ORC tilizando os dados salvos em CSV, como apresentado no Apêndice C.

O formato ORC significa *Optimized Row Columnar* ou Linha de Coluna Otimizada, que provê de uma forma altamente eficiente de armazenar dados no HIVE. Ao criar tabelas utilizando esse formato de dado, é possível melhorar o desempenho da manipulação de grandes quantidades de dados, como a leitura, escrita e processamento [15].

Com os dados persistidos em formatos de tabela HIVE, foi possível utilizar o HUE para criar *querys* e visualizações dos resultados. O HUE é acessado através da porta 8889 e possui uma aplicação web que permite a manipulação de dados contidos nas tabelas HIVE. As buscas ao banco de dados foram criadas com objetivo de filtrar os dados para que fossem criados gráficos a partir dos resultados. Os tipos de gráficos criados dependem do resultado das querys e do objetivo da visualização, podendo ser uma tabela, um scatter plot, um gráfico de pizza, um gráfico de barra ou uma visualização em mapa.

A figura Figura 3.7 mostra a página inicial da aplicação web HUE, que possibilita de forma fácil e intuitiva a manipulação de buscas e tabelas e a criação de visualizações.

Esse capítulo mostrou um breve resumo a respeito das ferramentas utilizadas para o desenvolvimento do trabalho e da metodologia utilizada. Além disso, ele mostra os passos

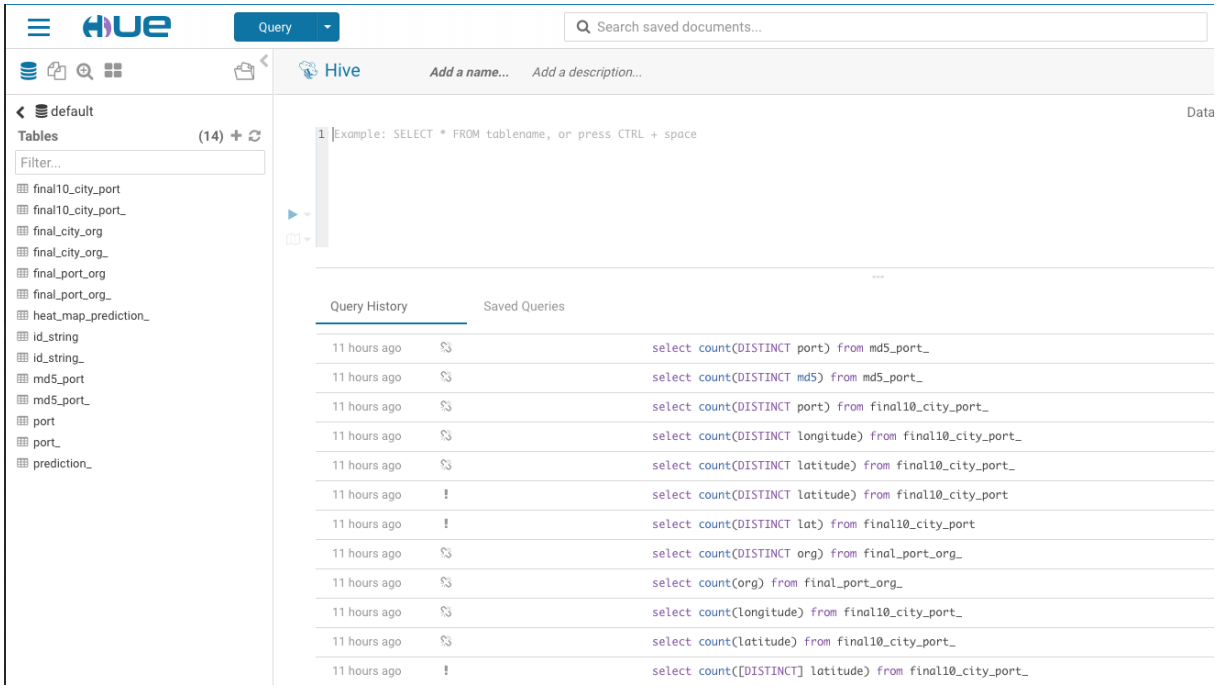


Figura 3.7: Página Inicial HUE.

utilizados para preparação dos dados de coletar e transformar afim de aplicar o algoritmo desejado. O capítulo também explica sobre como foi utilizado o algoritmo de aprendizado de máquina k-means, assim como os seus parâmetros de entrada. Ao final, foi apresentado a ferramenta utilizada para criar visualizações, que serão mostradas no Capítulo 4.

Capítulo 4

Resultado e Análise

Os resultados foram obtidos através da aplicação do algoritmo k-means em quatro tabelas diferentes, utilizando o valor de k igual a 10, ou seja, separando em 10 grupos com características similares. Essas similaridades são encontradas a partir das várias interações recursivas do algoritmo de aprendizado de máquina.

O primeiro conjunto de dados foi criado a partir das colunas contendo a porta atacada e a organização atacante. O segundo conjunto de dados, foi criado a partir da seleção da porta atacada e da latitude e longitude do local que sofreu o ataque. O terceiro conjunto de dados foi criado a partir da porta atacada e do IP da máquina atacante, chamado de MD5. O último conjunto de dados foi criado apenas com os valores das portas atacadas. A única diferença entre as tabelas utilizadas são as colunas selecionadas, sendo as linhas das portas atacadas idênticas em todas elas.

Os resultados obtidos inferem características de cada uma dos conjuntos de dados, assim como visualizações necessárias para entender o comportamento e eficiência do algoritmo nas quatro versões diferentes. Foram criadas *querys*, presentes para consulta no Apêndice C, contendo a extração das informações importantes para análises dos resultados. Dessa forma, o próximo passo após a extração dos dados, foi criar visualizações utilizando a ferramenta HUE para melhorar o entendimento dos dados e poder compará-los.

4.1 Conjunto de dados 1: Porta e Organização

A coluna referente a organização, possui informação da empresa que detem o bloco de endereço de IP ao qual se originou o ataque e está no formado de *string* na tabela, enquanto a coluna porta contém o número de porta atacada e está no formato de inteiro na tabela. O número de organizações diferentes é de 226 organizações diferentes. O dado foi salvo

como uma *string* e foi indexado para que pudesse ser tratado pelo algoritmo k-means da MLlib *Machine Learling Library*.

4.1.1 Números de ataques por porta

Foi observado que duas portas detém de mais de 50% das tentativas de ataque. As portas 25 e 23, referentes aos protocolos SMTP e Telnet respectivamente, duas portas muito utilizadas no dia-a-dia dos usuários da internet. A facilidade e os inúmeros tipos de ataques que podem ser executados nesses serviços são provavelmente motivos que atraem os atacantes. Tanto um usuário comum, quanto as grandes corporações, estão suscetíveis a ataques vindos dessa porta, que por sua vez, precisam enfrentar vários filtros de anti-spam e proxys até completarem seu objetivo. Duas outras portas com grande número de ataques são as portas 8080 e 5900. Enquanto a porta 8080 é utilizada para o protocolo HTTP, a porta 5900 é utilizada para acesso e controle remoto de um computador.

Na Figura 4.1 podemos ver no gráfico de pizza a representatividade no total das quatro portas mais cobiçadas pelos atacantes, que juntas representam aproximadamente 70% dos totais de ataque.

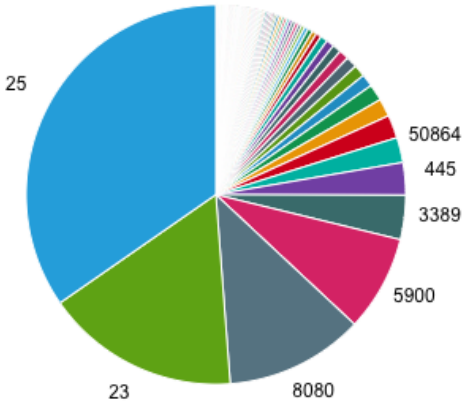


Figura 4.1: Portas mais atacadas.

4.1.2 Número de ataques por organização

Ao agrupar a quantidade de ataques por organização, foi observado que as organizações Microsoft Corporation e Chinanet Hubei Province Networt detem mais de 50% das ten-

tativas de ataque. As duas empresas são as maiores organizações responsáveis pelos IPs dos Estados Unidos e da China, respectivamente.

Observando os países com o maior número de usuários de internet no mundo, os Estados Unidos e a China estão entre os três maiores do mundo. Como era de se esperar, os países com mais usuários da internet também são aqueles que mais criam tentativas de ataques cibernéticos, sendo China o primeiro país com 58,3% da população usando a internet, Índia o segundo país, com 40,9% da população usando a internet e Estados Unidos como terceiro país com 89% da população ativa na internet, como apresentado na referência [16]. Na Figura 4.2 podemos ver as 10 maiores organizações de onde se originam as tentativas de ataque cibernético dentro da tabela de dados utilizada para esse estudo.

	times	org
1	24950568	Microsoft Corporation
2	8705655	Chinanet Hubei Province Network
3	3446776	Shocksrv Internet Services Private Limited
4	1444362	Zhenjiang Sky Netbar
5	1083297	Net For Ankas
6	1083287	Chinanet Guangdong Province Network
7	1083016	This Ip Network Is Used For Internet Security Research. Int
8	1002890	Chinanet Jiangxi Province Network
9	842346	Hostkey B.V.
10	801869	Renome-Service Joint Multimedia Cable Network

Figura 4.2: Organizações mais atacadas.

4.1.3 K-Means: Clusterização por portas

Foi aplicado o algoritmo de k-means nas colunas que possuem os dados com o nome da organização detentora do IP. Isso originou o ataque e a porta em que o ataque foi executado. O algoritmo dividiu as entradas em 10 grupos diferentes, do grupo zero até o nove. Na Figura 4.3 podemos ver o resultado com o grupo encontrado pelo algoritmo em relação ao número da porta atacada. O grupo zero, por exemplo, possui portas na fileira abaixo próximo de 1, próximo de 1500, próximo de 3500 e próximo de 6000.

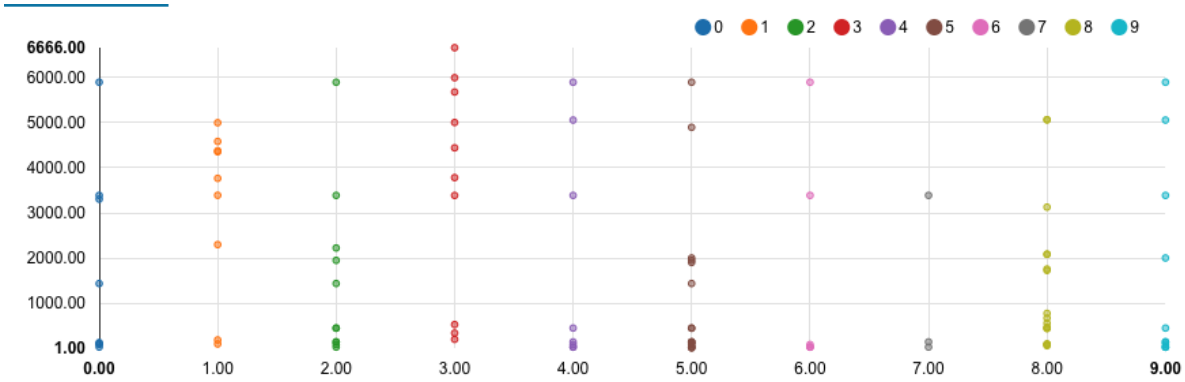


Figura 4.3: Clusterização por porta - Organização Atacada.

4.2 Conjunto de dados 2: Porta, Latitude e Longitude

Na segunda análise, foi selecionado três colunas: a porta que recebeu o ataque, a latitude do local atacado e a longitude do local atacado. O número de latitudes diferentes presentes na tabela é de 55, enquanto o número de longitudes diferentes é de 56. Juntas, as duas informações podem formar até 3080 pares diferentes de latitude e longitude, ou seja, cidades diferentes no mapa. Elas foram salvas na tabela com o formato *double*, por possuirem números decimais enquanto a porta foi salva no formato de inteiro. Isso traz uma complexidade grande ao tentar clusterizar esses dados juntos com as 122 portas diferentes presentes na tabela, pelo fato de existirem inúmeras combinações distintas.

4.2.1 Número de ataques por cidade

Ao agregar as linhas da tabela contendo a mesma latitude e longitude da localização aproximada dos alvos de ataque, pode-se observar que a maior concentração de alvos de ataque estava em 3 cidades: Nova Iorque (Estados Unidos), Dubai (Emirados Árabes) e Seattle (Estados Unidos). Dessa forma foi observado que cada uma dessas cidades possui uma grande empresa de tecnologia, que podem ter sido os alvos desses ataques. Nova Iorque possui uma sede da IBM, enquanto Dubai da Emirates Telecommunications Group e Seattle uma sede da Microsoft. Por se tratar de grandes companhias de valores de mercado alto, as infecções dessas redes se tornam algo valioso para o atacante.

Na Figura 4.4 foi retirado uma pequena amostra da quantidade de ataques e plotado em um mapa dos Estados Unidos. Nela podemos ver que as regiões com maior quantidade de ataques são o norte de Nova Iorque e Seattle. A extração de dados para filtro do país

Tabela 4.1: Localizações mais atacadas.

CIDADE	LATITUDE	LONGITUDE
Nova Yorque	44.48	-75.30
Dubai	25.27	55.33
Seattle	47.80	-122.29

Estados Unidos foi feita a partir da seleção de latitude e longitude contendo os extremos norte, sul, leste e oeste do país.

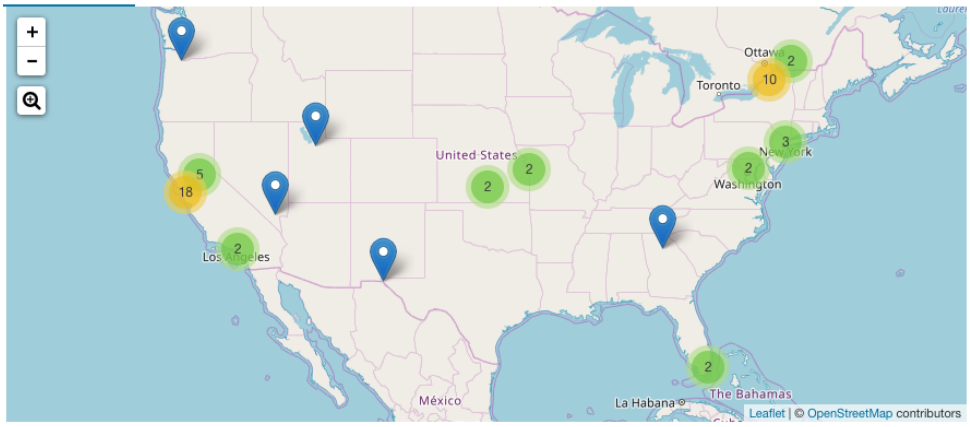


Figura 4.4: Cidades mais atacadas - USA.

4.2.2 K-Means: Clusterização por portas

Para caracterizar os grupos de porta utilizando o algoritmo k-means, foi se utilizado como entrada o número da porta que sofreu o ataque e a latitude e longitude do local atacado. Dessa forma, pode-se clusterizar as linhas da tabela em 10 grupos diferentes e observar como o algoritmo de aprendizado de máquina separou os grupos com características semelhantes. Na Figura 4.5 é possível ver o resultado dessa divisão em grupos a partir da análise do local e porta atacada.

4.3 Conjunto de dados 3: Porta e IP da Máquina Atacante (MD5)

Para o terceiro experimento, foram selecionadas duas colunas que possuem uma quantidade de dados distintos parecido. A coluna MD5 representa o IP da máquina atacante e é representada como um dato em *string*. Ela possui 399 diferentes variações, enquanto as portas, representadas como dados inteiros se dividem em 122 valores diferentes.

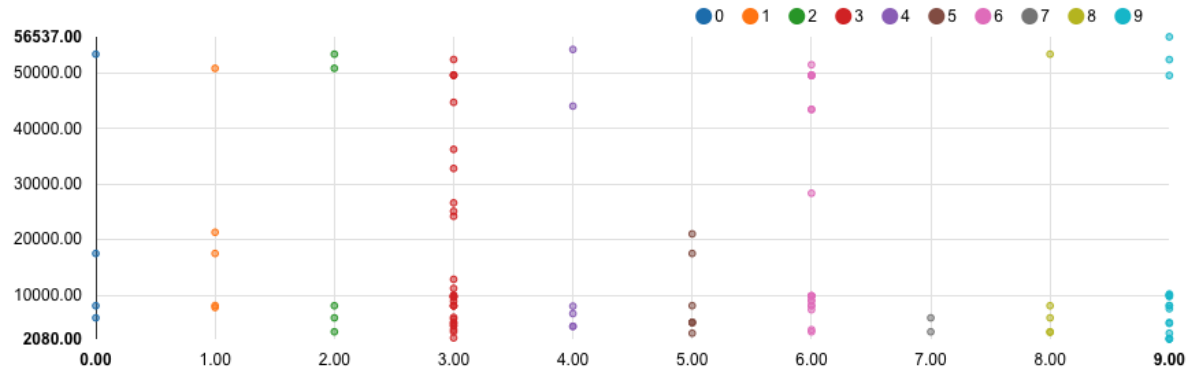


Figura 4.5: Clusterização por porta - Latitude, Longitude.

4.3.1 Quantidade de grupos por número de porta

Foi criado um gráfico de barras, como podemos ver na Figura 4.6 para avaliar quantos tipos de grupo cada porta possui. Pode-se dizer que grande parte das portas atacadas foram classificadas em mais de um grupo, ou seja, foram classificadas em mais de um grupo com conjunto de características semelhantes. Isso pode ter acontecido pelo fato do IP da máquina atacante (MD5) ter sido utilizada como entrada para análise do algoritmo k-means, afim de enriquecer a clusterização, obtendo resultados mais completos.

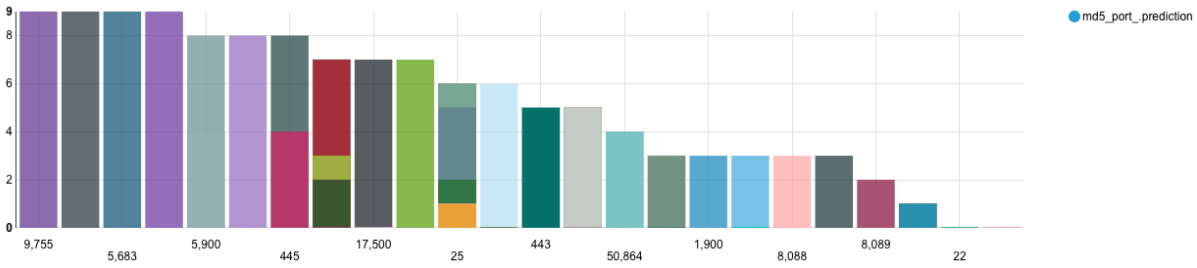


Figura 4.6: Quantidade de Grupos por Porta.

4.3.2 K-Means: Clusterização nas portas

Como podemos ver na Figura 4.7, o resultado encontrado gerou um comportamento levemente diferente daquele encontrado nas análises anteriores de clusterização.

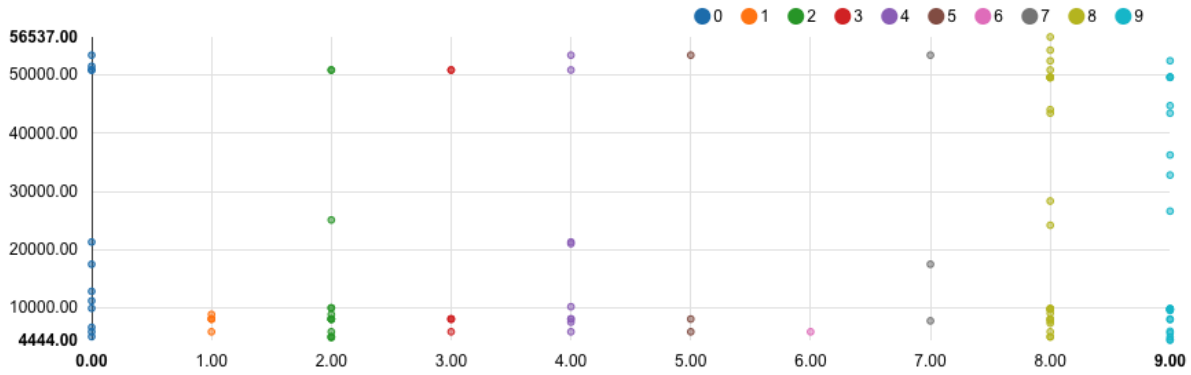


Figura 4.7: Clusterização por porta - MD5.

4.4 Conjunto de dados 4: Porta

Afim de visualizar os dados utilizando apenas a coluna referente as portas atacadas, foi selecionado da tabela de ataques a coluna "dport", contendo 122 portas diferentes. A clusterização com o algoritmo k-means foi feita utilizando apenas 10 grupos diferentes, que devem se agregar a partir de características semelhantes. Nesse caso, a única característica disponível para análise é o número da porta. É esperado uma separação mais homogênea dos dados, por possuir uma complexidade menor de clusterização, devido a menor entropia contida nos dados de entrada disponíveis para análise.

4.4.1 K-Means: Clusterização nas portas

O último experimento foi feito utilizando apenas as portas como entradas para criar a clusterização em 10 grupos diferentes. Como podemos ver na Figura 4.8, a clusterização entre os grupos feita levando em consideração apenas o número de porta se mostrou no gráfico como algo homogêneo, como esperado. Ao analisar apenas essa coluna para gerar o modelo de agrupamento, a clusterização tende a ser feita a partir da diferença entre a gama de números, unindo no mesmo grupos portas com valores parecidos. Dessa forma, valores de portas próximos deveriam estar em grupos semelhantes.

4.5 Análise da Eficiência de Clusterização

Os resultados mais relevantes para esse capítulo, são os gráficos referentes a clusterização das portas nos quatro subconjuntos diferentes. Todos esses resultados seguiram os mesmos passos para serem produzidos, com os mesmos parâmetros de entrada nos algoritmos. Dessa forma é possível fazer uma comparação entre os resultados obtidos afim de entender qual foi a melhor aplicação.

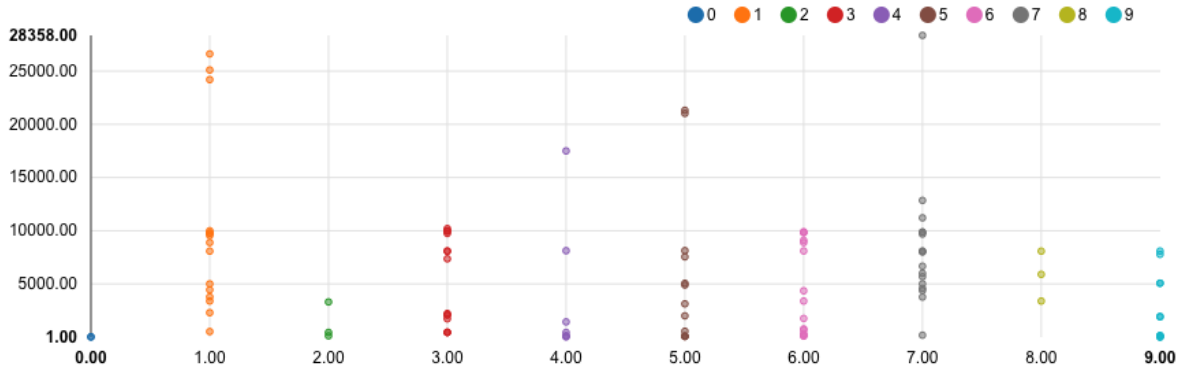


Figura 4.8: Clusterização por porta - Port.

As figuras a seguir possuem um gráfico contendo a quantidade de itens em cada um dos grupos resultados da clusterização aplicada pelo algoritmo K-means. Pode-se observar que em alguns casos, um grupo ficou com mais de 50% de itens, enquanto outros ficaram com quantidades menores que 1%.

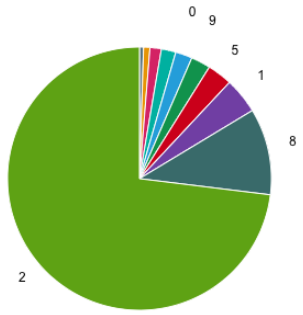


Figura 4.9: Distribuição dos grupos - Latitude, Longitude e Porta.

Na Tabela 4.2, podemos observar a porcentagem de dados em cada um dos 10 grupos clusterizados pelo algoritmo k-means. Nela podemos observar um comparativo entre a eficiência da aplicação do algoritmo para diferentes subconjuntos de dados relacionados ao ataque cibernético. Dessa forma, podemos encontrar uma melhor opção para clusterizar esses tipos de dados.

A clusterização feita apenas com a coluna de porta atacada, trouxe uma separação heterogênea, em que o grupo zero ficou com mais de 50% dos dados, enquanto o grupo oito ficou com aproximadamente 23% e todos os outros com valores menores que 10%. Esse resultado pode ter sido encontrado devido a falta de definição da posição dos centroides iniciais, pois essa análise deveria ser a mais eficiente no quesito clisterização, visto que,

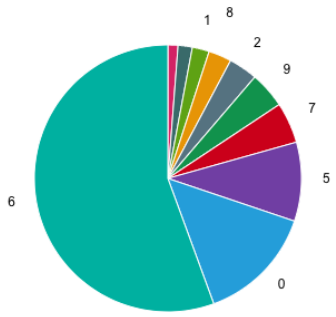


Figura 4.10: Distribuição dos grupos - Organização e Porta.

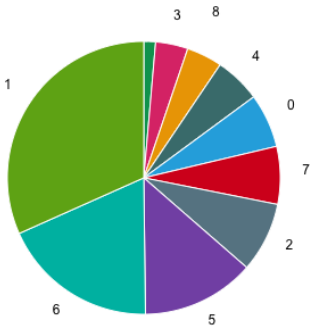


Figura 4.11: Distribuição dos grupos - MD5 e Porta.

analisando apenas as portas, que são dados no formato inteiro e contém apenas 122 tipos de portas diferentes.

A clusterização feita com dados referentes a porta atacada e a latitude e longitude do local que sofreu o ataque, também possui uma distribuição heterogênea, com o grupo 1 contendo 73% dos dados, enquanto o grupo 8 ficou com menos de 1% dos resultados. Devido a grande quantidade de opções de entradas diferentes, contendo até 3080 pares de latitude e longitude diferentes, esse subconjunto não foi eficiente para criar essa separação em grupos. Outra dificuldade apresentada nessa tabela é que os dados de latitude e longitude são do tipo double, algo que aumenta a complexidade e dificuldade do algoritmo que define os clusters.

A clusterização feita pela porta atacada e o IP da máquina atacante (MD5), possui a distribuição mais homogênea entre os experimentos. Cada grupo ficou com uma porcentagem de dados parecida, podendo dizer que a aplicação do algoritmo k-means com k igual a 10 obteve um bom resultado. A quantidade de dados diferentes da coluna MD5 é de 339 tipos diferentes, escritos no formato *string*, enquanto a quantidade de portas diferentes é de 122, escritas no formato inteiro.

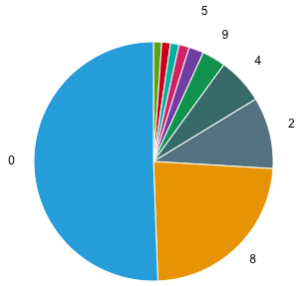


Figura 4.12: Distribuição dos grupos - Porta.

Tabela 4.2: Porcentagem por grupo.

GRUPO	PORT	CITY	MD5	ORG
0	50,59%	2,06%	6,4%	14,24%
1	1,12%	73,04%	31,69%	2,06%
2	9,68%	0,51%	8,29%	3,51%
3	1,39%	1,39%	3,84%	1,23%
4	6,29%	10,62%	5,45%	1,73%
5	2,01%	4,4%	13,4%	9,57%
6	1,17%	1,78%	18,52%	55,59%
7	1,17%	3,06%	6,84%	4,95%
8	23,46%	0,78%	4,23%	2,78%
9	3,17%	2,4%	1,39%	4,4%

A clusterização feita pela porta atacada e o nome da organização atacante, possui a distribuição heterogênea, com o grupo 6 contendo 55% dos dados enquanto outros grupos contêm menos de 5% dos resultados. Os resultados encontrados permitem inferir que essa classificação não foi feita de forma eficiênte, considerando que haviam 226 organizações diferentes, apresentadas no formato *string* e 122 portas diferentes, apresentadas no formato inteiro, não houve uma separação homogênea entre os grupos.

A seguir estão as coordenadas dos centróides obtidos em cada um dos dez centros do modelo criados pelo algoritmo k-means. Na tabela contendo a latitude, longitude e a porta atacada, o espaço vetorial é de três dimensões enquanto que a tabela contendo apenas a porta atacada possui apenas uma dimensão. Neles podemos observar melhor o comportamento da clusterização aplicada, que trouxe resultados heterogêneos para os diferentes subgrupos utilizados para teste. Uma possível melhoria na acurácia da clusterização seria por meio da definição da posição inicial de cada um dos k centróides, para que as iterações do algoritmo criem uma separação melhor no espaço vetorial.

Nesse capítulo podemos observar o comportamento das ameaças cibernéticas, utilizando buscas e visualizações nas tabelas com os resultados do algoritmo k-means. Dessa

Tabela 4.3: Centróide: MD5 e Port.

X	Y
2.57353012	2.81427294
13.96503825	64.48331841
10.01481928	352.29642204
9.51007331	198.96862363
2.35299955	39.49812407
0.51346119	18.7718216
3.26826462	103.19431876
38.37814564	264.84283345
101.87550071	348.46703665

Tabela 4.4: Centróide: Org e Port.

X	Y
5.88317256	11.96526396
90.39795487	181.29064494
7.72997329	140.01483669
94.03825701	27.63990811
7.51687319	199.51252691
10.65031575	40.13368991
1.02091126	0.72497079
2.21341127	74.16880899
40.76073297	23.30103122
8.24151591	104.13989453

Tabela 4.5: Centroide: Latitude, Longitude e Port.

X	Y	Z
6.94612497	29.51311977	29.54015295
16.20279168	1.7089317	1.7089317
1.32411235	0.83107441	0.83107441
105.30716096	5.8747732	5.8747732
78.44317663	36.44444783	36.1110988
23.05470707	19.36333092	19.36333092
70.39928537	4.87992028	4.87992028
13.00041485	45.99931274	47.64194555
5.35098191	9.18324659	9.18324659
41.72077289	4.67443455	4.67443455

Tabela 4.6: Centróide: Port.

X
0.32307952
111.4959724
6.66094035
48.96087785
12.92922942
34.13941129
69.99966293
90.99979464
2.66107089
22.42145158

forma, foi possível entender melhor sobre a melhor aplicação para categorizar em grupos os dados dos ataques retirados de fontes abertas. No Capítulo 5 é apresentado uma breve conclusão a respeito dos resultados encontrados e das visualizações criadas e apresentadas neste capítulo.

Capítulo 5

Conclusão

Analisando os resultados apresentados neste trabalho, pode-se afirmar que os dados referentes a ameaças cibernéticas são passíveis de clusterização utilizando o algoritmo k-means. Ao manipular os dados de fontes abertas, foi possível observar características e tendências referentes à localidade e frequencia dos ataques.

Dessa forma, ao considerar um algoritmo de clusterização eficiente aquele que separa uma tabela em grupos com distribuições parecidas, pode-se afirmar que o subconjunto contendo o MD5 e a porta atacada obteve o melhor resultado. Isso se deve pelo fato do algoritmo ter encontrado grupos com características semelhantes entre si, permitindo que a clusterização traga informações relevantes para análises futuras.

As ferramentas utilizadas para criação deste projeto tiveram o resultado esperado e foram eficazes para tratamento de grandes quantidades de dados. As bibliotécas nativas para manipulação facilitaram o trabalho de leitura, escrita e manipulação dos dados, além da gerência de recurso e a paralelização de processos que reduz o tempo de processamento.

A definição do passo a passo para aplicação de aprendizado de máquina se mostrou eficiênte e pode ser replicado para processamento de outros algoritmos em outros grupos de dados. Os passos de coletar, ler, indexar, vetorizar, aplicar o algoritmo e depois salvar, são passos que podem ser aplicados e, qualquer grupo de de dados, com o objetivo de executar algoritmos de aprendizado de máquina.

5.1 Trabalhos Futuros

Um próximo passo é utilizar a separação em grupos criada pelo subconjunto contendo o MD5 para aprofundar a respeito de características referêntes a cada um dos grupos, afim de criar inteligência e gerar conteúdo importante para a prevenção de ameaças cibernéticas.

Uma outra oportunidade de continuação deste trabalho é de aplicar outras técnicas de aprendizado de máquina com o objetivo de clusterizar os dados e comparar com os resultados obtidos utilizando o algoritmo k-means, afim de obter o algoritmo que traz um resultado mais preciso e assertivo.

Dentro da velocidade da criação de dados e de tipos de ameaças cibernéticas novas, é importante criar inteligências necessárias afim de previnir e prever os tipos de ataques, de forma que o receptor possa se precaver e manter seus dados seguros dentro do mundo atual que é extremamente conectado e vulnerável.

Referências

[1] *Distribuição de frequência do termo big data.* https://www.researchgate.net/figure/Research_fig1_302432602. Acessado: 2019-07-08. ix, 3, 4

[2] *Brief story about big data.* https://www.forbes.com/sites/gilpress/2013/05/09/a-very-short-history-of-big-data/#246dfd2265a1. Acessado: 2019-06-22. 1

[3] Gandomi, Amir e Murtaza Haider: *Beyond the hype: Big data concepts, methods, and analytics.* International journal of information management, 35(2):137–144, 2015. 3, 5

[4] Laney, D: *3-d data management: Controlling data volume, velocity and variety. meta gr. res.* Relatório Técnico, Note 6, 70, 2001. 3

[5] Cukier, K: *The economist, data, data everywhere: A special report on managing information*, 2010. 4

[6] Ayodele, Taiwo Oladipupo: *Types of machine learning algorithms.* Em *New advances in machine learning.* IntechOpen, 2010. 7

[7] Mitchell, Tom M: *Machine learning and data mining.* Communications of the ACM, 42(11), 1999. 7

[8] *Clustering in machine learning.* https://www.geeksforgeeks.org/clustering-in-machine-learning. Acessado: 2019-05-28. 9

[9] Huang, Zhexue: *Extensions to the k-means algorithm for clustering large data sets with categorical values.* Data mining and knowledge discovery, 2(3):283–304, 1998. 10

[10] *Introduction to k-means clustering.* https://medium.com/@dilekamadushan/introduction-to-k-means-clustering-7c0ebc997e00. Acessado: 2019-06-15. 10

[11] *Arquitetura cdh.* https://www.cloudera.com/documentation/enterprise/5-6-x/topics/cdh_intro.html. Acessado: 2019-04-05. 11

[12] *Hadoop architecture and deployment.* http://www.rosebt.com/blog/hadooparchitecture-and-deployment. Acessado: 2019-04-03. 11

[13] *Cloudera manager.* https://www.cloudera.com/products/product-components/cloudera-manager.html. Acessado: 2019-07-03. 11

[14] *Contexto spark e contexto hive.* https://blogs.msdn.microsoft.com/bigdatasupport/2015/09/14/sqlcontext-and-hivecontext/. Acessado: 2019-04-05. 14

[15] *Hive and orc.* https://cwiki.apache.org/confluence/display/Hive/LanguageManual+ORC. Acessado: 2019-06-07. 22

[16] *Top 20 countries with internet users.* https://www.internetworldstats.com/top20.htm. Acessado: 2019-06-28. 26

Apêndice A

Código usado no Pyspark: Aplicação de K-Means

```
1
from pyspark.ml.feature import VectorAssembler
3 from pyspark.ml.clustering import KMeans, KMeansModel
from pyspark import SparkContext, SparkConf
5 from pyspark.sql import SparkSession
from pyspark.ml import Pipeline
7 from pyspark.ml.feature import StringIndexer


9


11 conf = SparkConf().setAppName("Lat Long")
sc = SparkContext(conf=conf)
13 spark = SparkSession(sc)

15 df = spark.read.parquet("home/alvaro/attack.parquet")
df = df.select("dport", "latitude2","longitude2")
17 df.show()

19 #string indexer with pipeline
indexers = [StringIndexer(inputCol=column, outputCol=column+"_index").fit(
    df) for column in list(set(df.columns)) ]
21 pipeline = Pipeline(stages=indexers)
df_r = pipeline.fit(df).transform(df)
23 df_r.show()

25 #vector assembler
vecAssembler = VectorAssembler(inputCols=["dport_index","latitude2_index",
    "longitude2_index"], outputCol="features")
27 new_df = vecAssembler.transform(df_r)
```

```
new_df.show()

#k-means
kmeans_ = KMeans(k=10, seed=420)  # 2 clusters here
model = kmeans_.fit(new_df.select('features'))
transformed = model.transform(new_df)
transformed.show()

transformed = transformed.select("dport", "org", "prediction")

transformed.write.csv("home/alvaro/final_city_port.csv")

aux = df.select("latitude2","longitude2")
indexers = [StringIndexer(inputCol=column, outputCol=column+"_index").fit(
    aux) for column in list(set(aux.columns)) ]


from pyspark.ml.feature import VectorAssembler
from pyspark.ml.clustering import KMeans, KMeansModel
from pyspark import SparkContext, SparkConf
from pyspark.sql import SparkSession
from pyspark.ml import Pipeline
from pyspark.ml.feature import StringIndexer


if __name__ == "__main__":

  conf = SparkConf().setAppName("Kmeans - TCC  Ivaro  Vieira")
  sc = SparkContext(conf=conf)
  spark = SparkSession(sc)

  #reading the dataframe from HDFS
  df = spark.read.parquet("home/alvaro/attack.parquet")

  ##### Part 1: selecting subsets #####
  # Port and Organization
  df1 = df.select("dport", "org")
  df1.show()

  # Port and latitude2 and longitude 2
  df2 = df.select("dport", "latitude2", "longitude2")
  df2.show()


  # Port and MD5
```

```
df3 = df.select("dport", "md5")
df3.show()


##### Part 2: indexing using pipeline ####

#for the first subset
indexers = [StringIndexer(inputCol=column, outputCol=column+"_index").fit
    (df1) for column in list(set(df1.columns)) ]
pipeline = Pipeline(stages=indexers)
df_r1 = pipeline.fit(df1).transform(df1)
df_r1.show()

#for the second subset
indexers = [StringIndexer(inputCol=column, outputCol=column+"_index").fit
    (df2) for column in list(set(df2.columns)) ]
pipeline = Pipeline(stages=indexers)
df_r2 = pipeline.fit(df2).transform(df2)
df_r2 = StringIndexer(inputCol="latitude2_index","longitude2_index",
    outputCol="city_index").fit(df2).transform(df2)
df_r2.show()

#for the third subset
indexers = [StringIndexer(inputCol=column, outputCol=column+"_index").fit
    (df3) for column in list(set(df3.columns)) ]
pipeline = Pipeline(stages=indexers)
df_r3 = pipeline.fit(df3).transform(df3)
df_r3.show()

#### Part 3: Assemblying into one vector ###

#for the first subset
vecAssembler1 = VectorAssembler(inputCols=["dport_index","org_index"],
    outputCol="features")
new_df1 = vecAssembler1.transform(df_r1)
new_df1.show()

#for the second subset
vecAssembler2 = VectorAssembler(inputCols=["dport_index","city_index"],
    outputCol="features")
new_df2 = vecAssembler2.transform(df_r2)
new_df2.show()

#for the first subset
```

```
vecAssembler3 = VectorAssembler(inputCols=["dport_index","md5_index"],
    outputCol="features")
new_df3 = vecAssembler3.transform(df_r3)
new_df3.show()

#### Part 4: K-means with k = 10
kmeans_ = KMeans(k=10, seed=420)

#for the first subset
model1 = kmeans_.fit(new_df1.select('features'))
transformed1 = model1.transform(new_df1)
transformed1.show()

#for the second subset
model2 = kmeans_.fit(new_df2.select('features'))
transformed2 = model2.transform(new_df2)
transformed2.show()

#for the third subset
model3 = kmeans_.fit(new_df3.select('features'))
transformed3 = model3.transform(new_df3)
transformed3.show()


### Part 5: Saving into CSV file

#for the first subset
transformed1 = transformed1.select("dport", "org","prediction")
transformed1.write.csv("home/alvaro/org_and_port_transformed.csv")

#for the second subset
transformed2 = transformed2.select("dport", "latitude2", "longitude2","
    prediction")
transformed2.write.csv("home/alvaro/city_and_port_transformed.csv")

#for the third subset
transformed3 = transformed3.select("dport", "md5", "prediction")
transformed3.write.csv("home/alvaro/md5_and_port_transformed.csv")
```

Apêndice B

Código usado no Hive: Criando tabelas

```
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.clustering import KMeans, KMeansModel
from pyspark import SparkContext, SparkConf
from pyspark.sql import SparkSession
from pyspark.ml import Pipeline
from pyspark.ml.feature import StringIndexer
from pyspark.sql import HiveContext
import plotly.graph_objs as go
import plotly.plotly as py
import plotly.io as pio


if __name__ == "__main__":

  # criando contexto hive e spark
  conf = SparkConf().setAppName("Creating Hive Tables - TCC  lvaro  Vieira")
  sc = SparkContext(conf=conf)
  spark = SparkSession(sc)
  hive_context = HiveContext(sc)

  ##### Tabela 1: Organiza  o, porta destino e predi  o #####
  # criando tabela externa para verificar se dados ser o importados corretamente

  CREATE EXTERNAL TABLE IF NOT EXISTS org_port(
  port DOUBLE, org STRING, prediction INT)
  COMMENT 'Organization, port and prediction'
  ROW FORMAT DELIMITED
```

```
28  FIELDS TERMINATED BY ','
    STORED AS TEXTFILE
30  LOCATION '/user/root/home/alvaro/org_and_port_transformed.csv';

32  # checando se tabela foi criada corretamente
    describe org_port;
34  select * from org_port limit 5;


36

    # criando uma tabela interna com o formato ORC
38  CREATE TABLE IF NOT EXISTS org_port_(
    port INT, org STRING, prediction INT)
40  COMMENT 'Internal Table - Organization, port and prediction'
    STORED AS ORC;
42

    # carregar dados da tabela externa para a tabela interna
44  INSERT OVERWRITE TABLE org_port_ SELECT * FROM org_port;

46  # checkar se tabela interna foi criada corretamente
    describe org_port_;
48  select * from org_port_ limit 5;


50

##### Tabela 2: Organiza  o, porta destino e predi  o #####
52  # criando tabela externa para verificar se dados ser o importados
        corretamente

54  CREATE EXTERNAL TABLE IF NOT EXISTS city_port(
    port DOUBLE, lat DOUBLE, lng DOUBLE, prediction INT)
56  COMMENT 'Latitude, longitude, port and prediction'
    ROW FORMAT DELIMITED
58  FIELDS TERMINATED BY ','
    STORED AS TEXTFILE
60  LOCATION '/user/root/home/alvaro/city_and_port_transformed.csv';

62  # checando se tabela foi criada corretamente
    describe city_port;
64  select * from city_port limit 5;


66

    # criando uma tabela interna com o formato ORC
68  CREATE TABLE IF NOT EXISTS city_port_(
    port INT, lat DOUBLE, lng DOUBLE, prediction INT)
70  COMMENT 'Internal Table - Latitude, longitude, port and prediction'
    STORED AS ORC;
```

```
72
   # carregar dados da tabela externa para a tabela interna
74 INSERT OVERWRITE TABLE city_port_ SELECT * FROM city_port;

76 # checar se tabela interna foi criada corretamente
   describe city_port_;
78
   # os dados de latitude e longitude devem ser Double - converter tipos
80 ALTER TABLE city_port_ CHANGE lat lat DOUBLE;
   ALTER TABLE city_port_ CHANGE lng lng DOUBLE;
82 select * from city_port_ limit 5;

84 ##### Tabela 3: Organiza  o, porta destino e predi  o #####
   # criando tabela externa para verificar se dados ser o importados
       corretamente
86
   CREATE EXTERNAL TABLE IF NOT EXISTS md5_port(
88 port INT, md5 STRING, prediction INT)
   COMMENT 'MD5,port and prediction'
90 ROW FORMAT DELIMITED
   FIELDS TERMINATED BY ','
92 STORED AS TEXTFILE
   LOCATION '/user/root/home/alvaro/md5_and_port_transformed.csv';
94
   # checando se tabela foi criada corretamente
96 describe md5_port;
   select * from md5_port limit 5;
98


100 # criando uma tabela interna com o formato ORC
   CREATE TABLE IF NOT EXISTS md5_port_(
102 port INT, md5 STRING, prediction INT)
   COMMENT 'Internal Table - MD5,port and prediction'
104 STORED AS ORC;

106 # carregar dados da tabela externa para a tabela interna
   INSERT OVERWRITE TABLE md5_port_ SELECT * FROM md5_port;
108
   # checar se tabela interna foi criada corretamente
110 describe md5_port_;
   select * from md5_port_ limit 5;
112
```

Apêndice C

Querys utilizadas no HUE: Criando visualizações

```
##### Org and Port #####

# Analisar como K-means separou as portas
select port, prediction from org_port_

#Ver quantos ataques em cada porta
select count(org) as times, port
from org_port_
group by port
order by times desc

#Ver quantos ataques por organiza  o
criar query

##### City and Port #####

# Ver quantos ataques por "cidade"
select lat, lng, count(prediction) as times
from city_org_
group by lat, lng
order by times desc

# Ver como k-means separou por cidade
for USA
  select lat, lng, prediction from city_org_ where lat between 24.73 and
     47.57 and lng between -124.80 and -66.85

for Brazil
```

```
28   select lat, lng, prediction from city_org_ where lat between -33.80 and
         5.46 and lng between -73.87 and -34.87

30 for all
   select lat, lng, prediction from city_org_
```