DEPARTAMENTO DE CIÊNCIAS E TECNOLOGIAS

MESTRADO EM ENGENHARIA INFORMÁTICA E DE TELECOMUNICAÇÕES

UNIVERSIDADE AUTÓNOMA DE LISBOA

"LUÍS DE CAMÕES"

ENHANCING CITIZENS ON SCIENCE INFORMATION QUALITY THROUGH RATING OF DATA

Dissertação para a obtenção do grau de Mestre em Engenharia Informática e de Telecomunicações

Autor: João António Pinto Nascimento

Orientador: Professor Doutor Daniel de Matos Silvestre

Número do candidato: 30003572

Setembro de 2020

Lisboa

Agradecimentos

Agradeço ao meu Orientador da Tese Professor Doutor Daniel Silvestre, pela sua dedicação, apoio, orientação e disponibilidade e a todos os professores que lecionaram este mestrado, que de forma direta ou indireta contribuíram para o sucesso ao nível das cadeiras e por consequência à conclusão desta tese. Dirijo um particular agradecimento ao Vítor Rodrigues, CEO da empresa Magic Beans, pela compreensão, ajuda, apoio e até financiamento dos testes necessários. Sem esta ajuda seria impossível a realização dos mesmos. Agradeço ainda a todos os colegas da Empresa Magic Beans. Um especial agradecimento aos meus pais e irmão, devido a toda a motivação e apoio que me passaram neste percurso exigente e à Marta Gaspar pela motivação e apoio prestado. E por fim a todos os meus amigos e colegas de turma do Mestrado um grande obrigado.

Resumo

Com o objetivo de automatizar tarefas, muitos classificadores têm sido propostos, com diferentes compromissos entre performance e robustez. Quando se pretende classificar um *stream* de dados extraídos sobre um processo cujo estado está em constante evolução, o desconhecimento da dinâmica do sistema e informação incorreta dificultam a tarefa de identificar que subconjunto dos dados são confiáveis e devem ser utilizados. Este problema é central em aplicações como a deteção de fogos florestais, presença de lixo nas praias, medições de ruído e poluição nas cidades e outras para as quais os cidadãos sejam chamados a fornecer dados através de sistemas computacionais. Utilizadores mal-intencionados, dados corrompidos por ruído, sensores baratos, entre outros, podem conduzir a erros na deteção.

Esta tese visa desenvolver um serviço capaz de detetar anomalias em dados e prever os valores seguintes, tendo em conta o historial. Partindo do pressuposto que os dados serão fornecidos através de uma aplicação mobile para alertas de incêndios florestais e deteção de lixo nas praias, foi criada uma aplicação de *backend* capaz de suportar o volume expectável de dados bem como executar o seu processamento. A infraestrutura desenvolvida para *AWS (Amazon Web Services)* contém várias ferramentas de deteção de anomalias e previsão, entre elas o *AWS Sagemaker* (Ferramenta de *machine learning*) e o *AWS S3 (Amazon Simple Storage Service*, armazenamento de objetos).

Através das experiências efetuadas foi possível a deteção de anomalias, bem como a previsão de valores futuros, num *dataset* composto por 22695 entradas de medições de temperatura de um componente interno de uma grande máquina industrial provenientes do *Numenta Anomaly Benchmark (NAB)*.

Palavras-chave: Citizen Cience; Citizen Application; AWS; AWS Sagemaker; AWS S3.

Abstract

To automate tasks, many classifiers have been proposed, with different compromises between performance and robustness. To classify a stream of data extracted from a process whose state is evolving, unknown system dynamics and incorrect information render the task of identifying the reliable subset of data that should be used. This problem is central in applications such as forest fires detection, finding garbage on beaches, noise and pollution measurements in cities and others for which citizens are called to provide data through computational systems. Malicious users, data corrupted by noise, cheap sensors, among others, can lead to errors in the detection.

This thesis aims to develop a service capable of detecting anomalies in data and predicting future values, considering history. Assuming that the data will be provided through a mobile application for forest fire alerts and beach garbage detection, a backend application was created capable of supporting the expected volume of data as well as performing its processing. The infrastructure developed for AWS (Amazon Web Services) contains several anomaly detection and forecasting tools including AWS Sagemaker (Machine Learning Tool) and AWS S3 (Amazon Simple Storage Service).

Through the experiments carried out it was possible to detect anomalies in a dataset used (22695 different temperatures from the Numenta Anomaly Benchmark (NAB), these concern data from a temperature sensor of an internal component of a large industrial machine) and to predict future values.

Keywords: Citizen Cience; Citizen Application; AWS; AWS Sagemaker; AWS S3.

Índice

Índice de Figuras

Lista de Siglas e Acrónimos (opcional)

AWS	Amazon Web Service
Saas	Software as a Service
Paas	Platform as a Service
Iaas	Infrastructure as a Service
RCF	Random Cut Forest
S3	Simple Service Storage
ODK	Open Data Kit
CC	Cloud Computing
NIST	National Institute of Standards and Technology
CDN	Content Delivery Network
EC2	Elastic Cloud Computing
RRCF	Robust Random Cut Forest
RRCT	Robust Random Cut Tree
CODISP	Collusive Displacement
MAE	Mean Absolute Error
MSE	Mean Squared Error
RMSE	Root Mean Squared Error

1 Introdução

Com o aumento do número de incêndios e lixo nos oceanos, têm sido desenvolvidos sistemas de alerta com dados provenientes dos cidadãos. Dois exemplos deste tipo de sistemas são a aplicação *STAYAWAY COVID* e a *Fireloc*. A aplicação *STAYAWAY COVID* tem com principal funcionalidade alertar o utilizador de exposições, consideradas de elevado risco, a outros utilizadores da aplicação a quem foi, entretanto, diagnosticada a COVID-19. É mais do que de uma solução de rastreio, denomina-se de um sistema de notificação da exposição individual a fatores de risco de contágio. A *Fireloc* tem como finalidade a identificação, posicionamento e monitorização de incêndios florestais com dados provenientes de multidões, ou seja. No entanto, estas necessitam de mecanismos de defesa contra dados erróneos que de outra forma tornariam ineficaz os sistemas de deteção atuais. Esta necessidade é comprovada com o aumento de tragédias no planeta, como a poluição dos oceanos, o número elevado de incêndios, terramotos, entre outras. É urgente reduzir os tempos de resposta (através de bombeiros, Proteção civil, entre outros), resultantes, na maioria das vezes, do alerta tardio.

Por sua vez, uma resposta/deteção rápida representa uma abordagem com maior sucesso a um incêndio por o encontrar ainda numa fase embrionária. Por todas estas razões, é da maior importância o desenvolvimento de uma ferramenta que permita a classificação dos dados de alerta recebidos ao longo do tempo, tendo a consciência que poderão existir anomalias e erros nos dispositivos, bem como a existência de dados fraudulentos por parte de indivíduos com o intuito de eludir as autoridades.

A recolha de dados por parte dos cidadãos, através de aplicações desenhadas com o objetivo de ter *Citizen science* tem uma longa história, com origem na Sociedade *Audubon*, há um século atrás. Nessa primeira experiência, a contagem de aves na época natalícia foi efetuada através de voluntários que observaram e contaram o número de espécies de aves durante as férias de inverno [1]. Tendo como base dados recolhidos por utilizadores, enviados ao longo do tempo, é então possível desenvolver sistemas talhados para a resposta aos problemas identificados nesta tese. Contudo, a questão de aferir quais as informações fidedignas permanece central à eficácia de uma aplicação mobile deste tipo.

O problema principal desta tese consiste em desenvolver um ambiente de backend capaz de ser eficiente no processamento de dados e na deteção de dados incorretos. Começaremos por abordar no capítulo 2 o tema *Citizen Cience*, sendo definido o termo bem como as aplicações

que são baseadas neste conceito. Também é discutida a privacidade e motivações que levam as pessoas a participar no fornecimento de dados, ilustrando aplicações e as suas funcionalidades.

No capítulo 3 é abordado o o conceito de *Cloud Computing,* e porque tornam este paradigma tão vantajoso para diversos projetos, seja de pequena até grande dimensão. São também indicados os diferentes modelos existentes de *Cloud Computing,* evidenciando as suas diferenças e especificidades. Por fim são abordados os diversos tipos de *Cloud Computing* (*Cloud* Privada, Publica, Comunitária e Híbrida) e para que fins são mais adequados.

Outra componente fundamental para a solução proposta consiste no *Amazon Web Services (AWS),* apresentado no capítulo 4. O serviço *AWS Sagemaker* é introduzido como um dos principais pois permite a execução de algoritmos como o *Random Cut Forest (RCF),* que será utilizado na solução.

No Capítulo 5 é abordada a definição do problema, justificando-se a importância da utilização de *Cloud Computing* para a solução proposta.

No Capítulo 6 menciona-se a solução proposta, onde é descrita uma arquitetura ideal, e depois a que foi implementada, apenas com os serviços essenciais de forma a apresentar um custo mais atrativo e ideal para projetos de mais baixo custo. É também abordado o algoritmo *Random Cut Forest (RCF)* utilizado para a deteção de anomalias em dados como este foi implementado passo a passo, inclusivamente é possível verificar os resultados obtidos. Também foi elaborada uma solução adicional, tendo em conta a regressão linear da reta, onde é possível visualizar passo a passo a implementação utilizada e previsões efetuadas com a mesma.

Por fim, no Capítulo 7 apresenta-se a conclusão. É feito um levantamento de toda a tese, são abordados os resultados e o seu significado. São também explicadas as vantagens e as limitações da solução.

2. Definição do problema

Tendo em conta o panorama atual a deteção de anomalias (dados incorretos) tem sido um desafio, principalmente quando se tem em consideração dados que mudam ao longo do tempo. Os dados do *dataset* que foram utilizados foram recolhidos ao logo de 3 meses em intervalos regulares pelo menos por um utilizador. As métricas utilizadas para esta recolha basearam-se na temperatura e no tempo (temperaturas registadas ao longo do tempo).

Um dos problemas é não se ter dados classificados do *stream* para treinar, em vezes disso serão dados estáticos. Outra questão é a performance, é necessária para este tipo de soluções que necessita de uma capacidade computacional elevada.

Antes de se abordar a solução desenvolvida é necessária a introdução de vários conceitos, que vão ser abordados nas secções que se seguem. Conceitos como *Citizen on science* e *Cloud Computing* são bastante relevantes.

3. Citizen on Science

"*Citizen on Science* is a form of research collaboration between researchers and volunteers to address real-world problems. Members of the public who volunteer for these collaborations participate in different phases of the research and in a *variety of ways*." [1]

O avanço das tecnologias de informação móvel em ambientes urbanos tem contribuído para uma melhoria significativa na deteção de intrusões nos dados, proporcionando diversos canais para os cientistas recolherem dados e permitindo também que os cidadãos se envolvam em projetos científicos. Os dispositivos móveis são uma ótima ferramenta para a recolha de dados. [1]

A maior disponibilidade de dispositivos móveis, facilidade de criação de aplicações móveis, tarifários de dados móveis incluídos, disponibilizaram novas ferramentas para que os voluntários participem em pesquisas no domínio ambiental e de biodiversidade. Vários projetos têm recorrido a dispositivos móveis tendo por base o conceito de *Citizen on Science*, utilizando recursos como a câmara, o microfone e dados de localização *GPS*. [2]

3.1. Gestão de Dados e Privacidade de Dados

A gestão de dados tornou-se um dos desafios centrais que surgiu com o crescimento de projetos de *Citizen on Science*. Um ponto importante é a privacidade dos dados, embora os cientistas estejam naturalmente inclinados para a captura de dados, deve-se capturar o mínimo de dados pessoais possíveis, que correspondam às necessidades mínimas do projeto. [2]

Estes dados são a nossa pegada digital, incorporando detalhes da vida quotidiana. Dados de localização são uma parte fundamental dos dados que são recolhidos por *Citizen Applications*, que quando combinados com data e hora dos dados levantam uma série de preocupações adicionais sobre a privacidade. [1]

Os participantes têm de ter meios para indicar como os seus dados podem ou não ser usados ou partilhados. Por exemplo, se os pontos de onde são esses dados forem disponibilizados num mapa público, é fundamental que os participantes compreendam e que tenham dado consentimento para isso, dado que as observações partilhadas podem revelar locais domésticos ou outros detalhes pessoais, mesmo sendo um utilizador anónimo. [2]

Os gestores do projeto são responsáveis por garantir a transmissão segura de dados e do armazenamento destes. Os dados pessoais devem ser apagados quando deixam de ser necessários para cumprir os objetivos do projeto.

3.2. Motivação para a Participação

Atrair voluntários é fundamental para o sucesso deste tipo de projetos envolvendo tecnologia, pelo que os fatores motivacionais têm de ser considerados. Investigadores descobriram que as motivações intrínsecas e coletivas como, interesses pessoais, o gozo, o seguir as normas sociais, ou reconhecimento são fatores dominantes, tais como o interesse em apoiar organizações favoritas [3], enquanto motivos de recompensa são indiscutivelmente menos salientes. [1]

Um bom exemplo é o caso da *Ushahidi* que é um ambiente de relatórios colaborativo baseado na Web onde são fornecidos dados pelos cidadãos. Outro exemplo é o *Open Data Kit (ODK)* que ajuda as organizações autoras a gerir soluções de recolha de dados móveis. [1]

Plataformas em que a comunidade ajuda na identificação/validação das observações, tais como *Natusfera* 30 (Figura 1), são um bom exemplo de como apoiar a aprendizagem não estruturada, e uma ferramenta poderosa para envolver as pessoas não treinadas que vão aprender progressivamente com a ajuda da comunidade. [2]

3.3. Qualidade de Dados

Citizen on Science depende de voluntários e não de utilizadores certificados e com experiência, o que potencialmente levanta questões sobre a qualidade dos dados. Embora os dados recolhidos por amadores possam ser de alta qualidade, o contrário também pode acontecer devido a submissões acidentais/maliciosas dos mesmos. [1]

Uma das formas mais comuns de avaliar o que foi implementado em projetos de *Citizen on Science* e verificar se os dados recolhidos são bons é a comparação destes dados recolhidos por voluntários. [3] Isto, não é o suficiente para implementar projetos de Citizen on Science, daí que de seguida, é necessária uma análise dos resultados da investigação. É comum os mesmos aplicarem-se apenas localmente (ao nível de por exemplo uma empresa), em grande parte porque muitos projetos acontecem a nível local, sendo executados por indivíduos e organizações que têm como principal objetivo a resolução de problemas que têm no momento

e não têm grande interesse ou não dão prioridade à comunicação com as comunidades mais amplas de *Citizen on Science*. Na maioria dos relatórios e publicações desta área, são descritas técnicas e conhecimentos relacionados com projetos em que tenham trabalhado, mas não aqueles que não correram da melhor forma. Devido a esta falta de comunicação, muita aprendizagem é perdida e muitos recursos são gastos de forma despropositada tendo como consequência custos de oportunidade enormes. [3]

De forma a aumentar a qualidade dos dados, existe a necessidade de se executar uma validação cruzada com outras fontes. Outro exemplo que pode ajudar na validação de dados é a avaliação de dupla abordagem, como pedir em formato de caneta e papel documentação das medições, para além de relatórios baseados em aplicações móveis que também podem ser utilizados. A comparação com outros dados pode revelar valores atípicos e estabelecer um nível geral de confiança. A contabilidade de dupla abordagem permite a identificação de discrepâncias nos relatórios, e, portanto, a medição potencial de problemas de dados. Além disto, dupla contabilidade permite um retorno em caso de mau funcionamento de aplicações móveis e de transmissão de dados, o que inclui pessoas sem ou incompatíveis com smartphones. [2]

Embora os dados recolhidos através de *Citizen Applications* sejam informações com qualidade, estas representam também alguns desafios em projetos, implementações e análise, que incluem:

- O desenvolvimento de ferramentas de recolha de dados de forma a ser maximizada, enquanto se mantêm a manutenção de elevados padrões e qualidade dos dados;
- A aplicação de novas técnicas de análise e visualização que podem revelar com precisão padrões nestes dados.

  [4]

No caso da ecologia por exemplo, a necessidade de dados em *Citizen on Science* na conservação de espécies parte de uma compreensão da distribuição, abundância, preferências de habitat e movimentos de organismos nas áreas, em todas as regiões geográficas e por períodos mais longos de tempo. Tipicamente apenas observadores humanos podem com segurança identificar organismos ao nível da espécie. Com os levantamentos feitos por observadores profissionais é possível identificar com precisão padrões de espécies consoante as ocorrências ao nível espacial e local, por espaços temporais curtos. Mas, devido ao custo e disponibilidade de especialistas não é possível que os mesmos recolham dados durante todo o

ano, sendo que para estes serem recolhidos em larga escala são necessários voluntários. Devido ao acesso à Internet e tecnologias de informação tem sido possível o desenvolvimento de mais projetos de *Citizen on Science*, como o *Galaxy Zoo*. A recolha geral de dados em *Citizen on Science* apresenta múltiplos desafios de informática: [4]

- A aquisição e gestão de grandes volumes de dados requer planeamento, não só da infraestrutura que os vai gerir, assim como de métodos adequados que mantenham a motivação de voluntários;
- O controlo e manipulação da qualidade, dada a variação entre observadores, falsos positivos devido à deteção de organismos imperfeitos e distribuições, contêm muitas vezes irregularidades nos dados, no espaço e no tempo;
- A modelagem adequada de fenómenos ecológicos e escalas espaciais e temporais mais amplas são um desafio, porque os organismos participam em diferentes dinâmicas, processos dependentes de escala, incluindo a dispersão, invasão, migração, predação, exclusão e interações competitivas. Qualquer um que pode variar através do espaço e tempo.

  [4]

O *eBird* é atualmente o maior projeto de *Citizen on Science* na área da ecológica, atualmente recolhe cerca de 2 a 3 milhões de novas espécies, registado mensalmente em todo o planeta. [4]

A quantidade de dados é importante, dado que aumenta a probabilidade de obtermos dados verdadeiros. A qualidade de dados é essencial, daí que se deve limitar a entrada de dados incorretos. É então necessário software, bases de dados e ferramentas na infraestrutura que suportem dezenas de milhares de dados provenientes de voluntários. [4]

Para obtermos mais dados, os protocolos de recolha dos mesmos não devem ser muito complexos, para que os voluntários tenham facilidade e vontade de participar. [4]

As aplicações móveis para um ambiente de apoio e monitorização da biodiversidade são frequentemente utilizadas para registar a presença e localização de espécies nativas e invasoras, a data, georreferências diferentes e eventos biológicos, tais como reprodução, e para identificar padrões de cobertura do solo ou fundo do mar, como indicado anteriormente. Assim, dada a sua utilidade também poderão ser usadas para deteção de lixo nos oceanos e incêndios entre outros. [2]

Para atingir o sucesso desejável, a *Citizen on Science* é necessário um número suficiente de participantes, durante um período prolongado. [2]

3.4. Aprendizagem Sobre Ciência: apoio à aprendizagem partilhada

Plataformas em que a comunidade ajuda na identificação/validação das observações, tais como *Natusfera* 30 (Figura 1), são um bom exemplo de como apoiar a aprendizagem não estruturada, e uma ferramenta poderosa para envolver as pessoas não treinadas que vão aprender progressivamente com a ajuda da comunidade. [5]

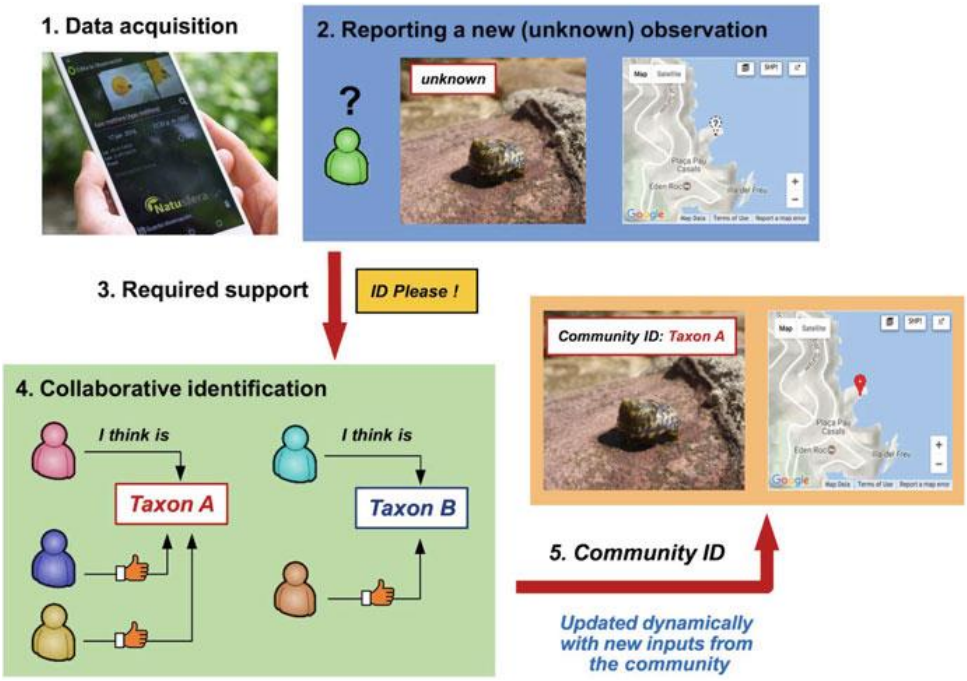


Figura 1 - Processo de aquisição de dados com recurso a fotografias e geo localização. [2]

4. Cloud Computing

4.1. Definição

Existem diversas definições de *Cloud Computing (CC)*, uma das melhores, foi dada pelo *National Institute of Standards and Technology (NIST).*

O NIST definiu como sendo um modelo que permite o acesso a um conjunto de serviços e recursos de computação configuráveis (por exemplo, redes, servidores, armazenamento, aplicações e serviços) que podem ser rapidamente fornecidos e libertados com o esforço de gestão mínima ou interação do *provider* de serviço. É caracterizada também por ter características de self-service em que um utilizador acede através da rede de acordo com as suas necessidades. [5]

4.2. Características

Cloud Computing incorpora uma série de características que lhe permitiu ganhar todo o protagonismo que tem neste momento. As principais características são *on-demand Self-Service* em que se um consumidor que utilize serviços de *Cloud*, se verificar que necessita de mais recursos, pode aumentar as capacidades computacionais, por exemplo armazenamento em rede, sem que seja necessário a intervenção humana. *Ubiquitous Network Access* onde os serviços de *Cloud* podem ser acedidos, através da rede e de acordo com mecanismos que promovem as plataformas heterogêneas (exemplo: telemóveis, computador, *tablet*). *Resource Pooling* em que os recursos de *Cloud Computing* estão reunidos geograficamente. Os seus recursos virtuais são dinamicamente atribuídos ou reatribuídos conforme as exigências do consumidor. O Consumidor não possui controlo nem a informação do local exato, de onde estão a ser fornecidos os seus recursos (exemplo: Memória, processamento, armazenamento). Apenas é possível ter uma informação mais ampla como o país em que se encontra, continente ou *Data Center*. Outra característica é a *rapid elasticity* definindo-se como a capacidade de um serviço de aumentar a quantidade de um recurso quando é necessário, de forma rápida e eficiente. Um Consumidor de *Cloud*, ficará com a perceção de que a *Cloud* parece ser infinita, pois o mesmo pode adquirir mais ou menos poder computacional conforme seja necessário para as suas aplicações. Esta é uma das características que torna tão atrativa a *Cloud Computing*. Por fim *measured service*, em que todos os serviços na *Cloud* são controlados e monitorizados de forma automática (processamento, largura de banda, contas de utilizador, entre outros). Desta forma, a utilização

do consumidor é otimizada, bem como auxilia o fornecedor na contabilização e faturação do consumo. Adicionalmente, existe uma maior transparência aquando da cobrança do serviço. [6]

4.3. Modelos de Serviço

Com o aumento de popularidade de *Cloud Computing*, surgiu a necessidade da criação de modelos de adoção de forma a ser vantajoso e inovador a aquisição de *Cloud Computing*. Foram então os modelos de serviço, designado por Software as a Service (SaaS), Platform as a Service (PaaS) e Insfrastructure as a Service (IaaS).

4.3.1. Software as a Service (SaaS)

Um serviço SaaS disponibiliza aos utilizadores uma conexão a aplicações ou serviços via internet. Os Consumidores/utilizadores compram junto dos fornecedores de serviços *Cloud*, a capacidade de aceder e utilizar uma aplicação ou um serviço presente na *Cloud*. O fornecedor de serviços *Cloud* gere o hardware e Software e garante a disponibilidade e segurança da aplicação ou serviço fornecido. A informação necessária da interação entre o consumidor e o serviço apresenta-se como parte do serviço na *Cloud*, onde apenas é pago o que é utilizado. Este modelo é flexível e permite por exemplo, aquisição de serviços de email de forma mais simplificada sem que o principal foco seja a configuração do serviço adquirido. [7] [8] [9]

Saas tem várias vantagens entre elas, obter acesso a aplicações sofisticadas, só é pago o que é utilizado, pode ser utilizado *software* cliente gratuito, facilidade de trabalho para os utilizadores e acesso a dados de aplicações em qualquer lugar. [7] [9]

4.3.2. Platform as a Service (PaaS)

Os consumidores/utilizadores compram o acesso a plataformas, onde lhes é permitido desenvolver software e aplicações na *Cloud* para uso próprio. Os recursos são comprados junto do fornecedor de serviços *Cloud*, e os utilizadores acedem aos mesmo através de uma ligação à internet (segura). [10]

O serviço PaaS permite evitar custos na compra e gestão de licenças de software, infraestruturas que suportam as aplicações, entre outros recursos. O utilizador gere as aplicações e serviços que desenvolve, o fornecedor de serviços *Cloud* por sua vez gere os restantes. [9] [8] [10]

O *Paas* tem a vantagem de que, é gasto menos tempo a programar, mantendo as mesmas equipas, acrescenta capacidades de desenvolvimento, desenvolve para várias plataformas, utiliza ferramentas sofisticadas sem custos e gere o ciclo de vida das aplicações. [10]

4.3.3. Infrastructure as a Service (IaaS):

Os consumidores controlam e gerem os sistemas quanto a sistemas operacionais, aplicações, armazenamento e conectividade de rede, mas não controlam a infraestrutura de nuvem. [8] [9]

É possível visualizar na figura 2 a representação dos modelos de serviço.

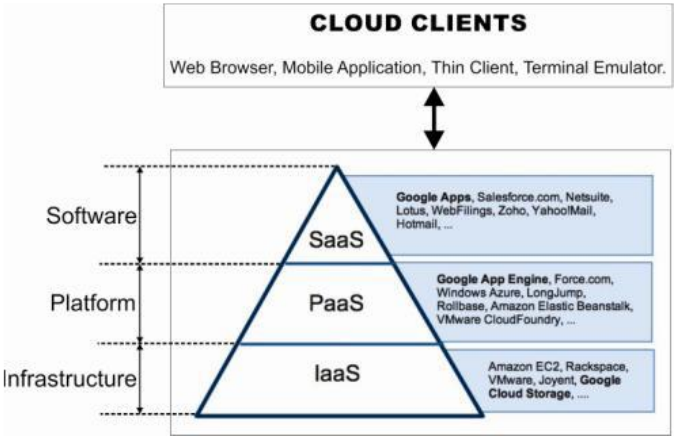


Figura 2 - Infraestruturas, SaaS, PaaS e IaaS. [11]

A escolha do modelo de serviço que vai ser utilizado é um passo bastante importante, mas para além disso é necessário definir de que forma será efetuada essa mesma implementação.

4.4. Modelos de Implementação

Existem diversos modelos de implementação de *Cloud Computing*, todos eles utilizado em caso específicos, como a c*loud* privada em que uma infraestrutura de *Cloud* Privada é destinada para uso exclusivo de uma organização. Pode ser gerida e pertencente a uma organização ("*in-house*) ou por uma terceira entidade ("*third party*"). Pode existir dentro ou fora das instalações da empresa. *Cloud* comunitária onde uma infraestrutura é destinada a organizações que partilham de interesses idênticos. Pode ser gerida e pertencente a uma ou mais organizações, ou então por uma terceira entidade. Pode existir dentro ou fora das instalações da empresa. *Cloud* pública sendo destinada ao uso do público em geral. Isto permite que um consumidor possa desenvolver e desdobrar um serviço, em *Cloud* sem tantos encargos

financeiros. Pode ser gerida e pertencer a uma empresa, organização governamental ou uma terceira entidade. Existe na infraestrutura/instalações de um fornecedor de *Cloud*. Por fim c*loud* híbrida onde a infraestrutura é composta por duas ou mais infraestruturas de *Cloud* (Privadas, Comunitárias e Públicas). Com esta estrutura é possível mover dados entre os diferentes tipos de *Cloud*, dentro de uma *Cloud* Híbrida. Estas permanecem como exclusivas, mas ainda assim estão conectadas por uma tecnologia padronizada, o que permitirá então a passagem de dados entre as diferentes infraestruturas de *Cloud.* [12]

5. Amazon Web Services

A *Amazon Web Services (AWS)* é o *provider* de serviços *cloud* mais adotada e mais abrangente do mundo [13]. Esta plataforma permite oferecer mais de 175 serviços completos em *datacenter* espalhados por todo o mundo. Tem milhões de clientes, entre eles *startups*, grandes empresas e os maiores órgãos governamentais. Todos estes órgãos usam a *AWS* para reduzirem custos, ficarem mais ágeis e inovarem com rapidez. [14]

Neste momento a *AWS* abrange 69 zonas de disponibilidade, está presente em 22 regiões geográficas, anunciou mais treze zonas de disponibilidade e mais quatro regiões (Espanha, Itália, África do Sul e Indonésia. [15]

5.1. AWS CloudFront

Amazon CloudFront fornece uma rede de distribuição de conteúdo *(CDN)* com destino à entrega de conteúdo a serviço web. *Amazon CloudFront* possibilita fazer por exemplo de um site dinâmico ou estático de conteúdo disponível a partir de uma rede global de pontos de presença. *Amazon CloudFront* armazena o conteúdo nos pontos de Presença por um determinado tempo especificado na configuração. Oferece também suporte a todo o tipo de arquivos que podem ser enviados através de *HTTP*. Incluindo páginas web dinâmicas, quaisquer arquivos estáticos que fazem parte de uma aplicação *web*, como o website imagens, downloads de software, entre outros. [16]

Amazon CloudFront é otimizado para trabalhar com outros serviços da Amazon, como a *Amazon S3*, *Amazon EC2*, *Amazon Elastic Load Balancing*, e *Amazon Route 53*. É projetado para fornecer uma baixa latência e uma elevada largura de banda no envio de conteúdos. A distribuição do conteúdo ao utilizador final é acelerada através da escolha de um ponto de presença com melhor desempenho da rede mundial disponível (atualmente já existe um ponto

de presença em Portugal. [16] É possível a visualização na figura 3 de um esquema de funcionamento.

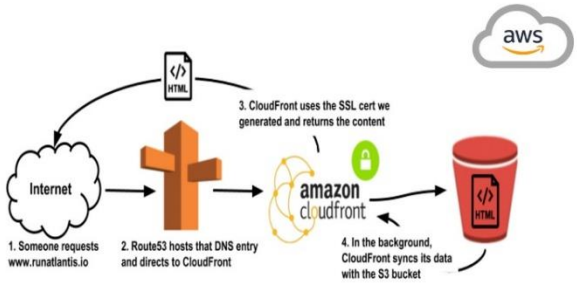


Figura 3 - Exemplo de um acesso a um AWS S3 (bucket) através do AWS CloudFront. [28]

5.2. AWS S3 (Simple Storage Service)

Amazon S3 (Simple service Storage) é um *storage* de objetos altamente durável (durabilidade 99,999999999%, *11 nines*). Um dos muitos casos de uso para o *Amazon S3* é o armazenamento e distribuição de conteúdos *web* estáticos, dado que cada um tem um endereço *URL HTTP* único. Outra utilidade do Amazon S3 é servir de origem para uma rede de distribuição de conteúdo (*CDN*), como o *Amazon CloudFront*. [16]

Por causa da elasticidade (capacidade de se adaptar a uma carga de trabalho através do provisionamento e desaprovisionamento de recursos automaticamente) do Amazon S3, este funciona bem para sites estáticos de rápido crescimento e guardar grandes volumes de dados, gerados por utilizadores, como sites de patilha de fotos e vídeo. Amazon S3 oferece uma solução altamente disponível e *highly scalable* para sites apenas com conteúdo estático, incluindo arquivos *HTML*, imagens, vídeos e scripts do lado do cliente, tais como *JavaScript*. É também usado como *storage* de dados para análise em larga escala, como de operações financeiras, fluxo de cliques, fotos e transcodificação de media (figura 4). Dada a sua escalabilidade é possível o acesso aos dados de vários dispositivos diferentes ao mesmo tempo, sem que a mesma seja restringida por uma única conexão. [16]

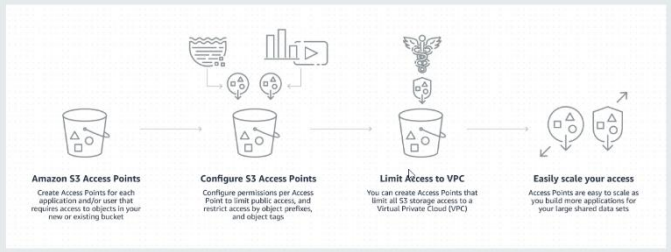


Figura 4 - Algumas Características do AWS S3. [30]

5.3. AWS Sagemaker

O *Amazon SageMaker* é um serviço composto por vários algoritmos de *machine learning.* Através deste é possível criar e treinar modelos com rapidez e facilidade, podendo posteriormente, exportar para um ambiente de produção. Este serviço oferece uma instância de *notebook Jupyter* integrada, facilitando assim o acesso a bases de dados destinadas à exploração e análise, sem que seja necessária a gestão de servidores. Oferece ainda algoritmos comuns de *Machine Learning* otimizados para execução de alta performance de grandes volumes de dados num ambiente distribuído. Suportando de forma nativa algoritmos e estruturas próprios do utilizador, o *Amazon SageMaker* oferece opções flexíveis de treino distribuído ajustando-se a fluxos de trabalho específicos. [17]

Através do *Amazon SageMaker Studio* que é um ambiente de desenvolvimento integrado (*IDE*) é possível criar, treinar, testar, implementar modelos de *machine learning(*figura 5). Através de uma única interface permite: [18]

- Escrever e executar código em blocos de anotações *Jupyter*; [18]
- Criar e treinar modelos de *machine learning*; [18]
- Implementar modelos e monitorizar o desempenho das previsões; [18]
- Acompanhar e testar os testes de *machine learning*. [18]

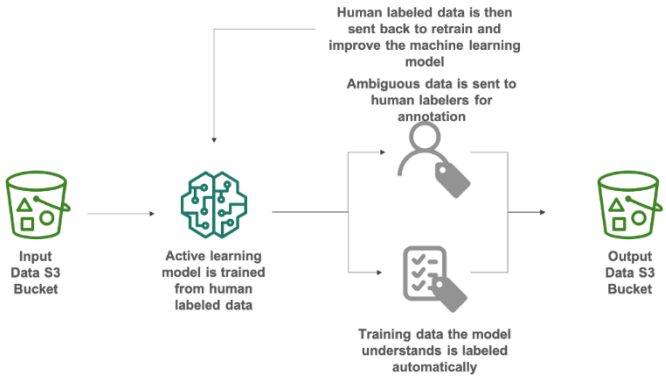


Figura 5 - Exemplo do AWS Sagemaker ler dados de um AWS S3 Bucket. [29]

5.3.1. – O que é deteção de anomalias?

Outliers/anomalias num conjunto de dados são entradas (input) que não se encaixam com as outras, ou seja, que estão fora do intervalo esperado de valores. Como tal, precisam de ser tratados com cuidado para uso direto (monitorização ou durante o pré-processamento, ou seja, *pipeline* de *machine Learning*). Uma tarefa de classificação pode ser transformada num problema de deteção de *outliers*, se as classes forem altamente distorcidas. Essas entradas representam estados que não devem ser observados e, portanto, também são chamados de anomalias. [19]

Há uma grande variedade de algoritmos que são utilizados para deteção de outliers:

- Os baseados em distribuições e quantis (*z-score*, *digest-t*);
- Os baseados na densidade (*DBSCAN*);
- Os baseados em árvores de isolamento (*iForest*, *RCF*).

Para séries temporais: algoritmos clássicos que não levam em consideração a temporalidade dos dados, bem como algoritmos preditivos. Para este último, considera-se um ponto como normal se o resíduo entre a previsão e o valor real for pequeno. [19]

5.3.2. Algoritmos baseados em isolamento

Os algoritmos de deteção de anomalias baseiam-se no aspeto em que pontos anómalos escassos e diferentes de uma distribuição normal, podem ser isolados. De forma reduzir-se o tempo de execução e de recursos usados, este tipo de algoritmos tomam algumas decisões aleatórias em relação à escolha de recursos, ou seja, provisiona recurso se precisar, o que explica a sua popularidade ao lidar com dados de alta dimensão ou ao implementar uma aplicação de *streaming*. [19]

O *iForest* foi o primeiro algoritmo a utilizar este esquema, que tem como objetivo a construção de uma floresta de árvores binárias. Sendo que cada árvore é construída num subconjunto (nó) dos dados disponíveis. É selecionada aleatoriamente uma dimensão, e efetua-se um corte aleatório ao longo dessa, é repetido nos dois subconjuntos (nós) resultantes, até que todos os pontos desta amostra de dados fiquem isolados numa folha da árvore. [19]

Quanto mais próximo um ponto estiver da raiz de uma árvore ou quanto menos níveis tiver a árvore, aumenta a probabilidade de ser considerado uma anomalia. Zero (0) é o ponto normal (não anómalo), e um (1) o ponto anómalo). [19]

5.3.3. Algoritmo Random Cut Forest (RCF)

O *Amazon SageMaker* utiliza um algoritmo não supervisionado (conjunto de dados utilizado não possui nenhum tipo de rótulo) para deteção dados anormais num conjunto de dados. Esse algoritmo é o *Random Cut Forest (RCF).* Estes dados anormais podem ser detetados através de picos inesperados nos dados em espaços temporais, pausas na periodicidade ou dados inclassificáveis. Estes dados anormais são facilmente visualizados num gráfico, isto porque são normalmente distinguíveis dos restantes “normais”. [20]

O RCF associa uma pontuação de anomalia a cada conjunto de dados. Um valor pequeno representa dados fidedignos, embora a definição de um threshold seja dependente da aplicação em questão. Um valor típico considerado costuma ser a distância de três vezes o desvio padrão em relação à média. [20]

5.3.3.1. Interface de entrada (input)/saída (output) para o algoritmo RCF

O *Random Cut Forest* do *Amazon SageMaker* é compatível com os canais de dados *train* (treino) e *test* (teste). O canal de teste é opcional e é utilizado para calcular métricas de precisão, *recall*, exatidão e pontuação *F1* em dados rotulados. Os dados de treino e teste podem ser carregados no formato de *texto/csv*. ou *application/x-recordio-protobuf*. No caso dos dados de teste (*text/csv*), o conteúdo destes dados deve ser especificado como *text/csv;label_size=1*, o rótulo de anomalias é representado pela primeira coluna de cada linha: "*1*" para dados anormais, e "*0*" para dados normais. Existe a possibilidade de utilizar o modo de Ficheiro ou de *Pipe* para treinar modelos *RCF* em dados com o formato *CSV* ou *recordIO-wrapped-protobuf*. [20]

Para inferência, o *RCF* é compatível com os seguintes tipos de conteúdo de dados de entrada:

- *Application/x-recordio-protobuf*; [20]
- *Text/csv*; [20]
- *Application/json*. [20]

O formato de saída da inferência do *RCF* é *application/json* ou *application/x-recordio-protobuf*. Cada registo de saída tem associado as pontuações ao nível das anomalias nos dados de entrada. [20]

O objetivo do algoritmo *RCF* é criar várias árvores formando uma floresta, sendo que cada árvore é criada através de cada amostra. [20]

Por exemplo, é dada a entrada de uma amostra e de dados aleatórios. Esta amostra é dividida pelas árvores que constituem a floresta. As árvores recebem e organizam os dados num subconjunto de pontos, numa árvore *k-d*. As pontuações das anomalias atribuídas aos conjuntos de dados pela árvore são definidas como as alterações esperadas na complexidade dessa árvore, resultando na inclusão do ponto à árvore. De forma, a ser possível a atribuição de uma pontuação anómala, é calculada a pontuação média de cada árvore pelo *Random Cut Forest* escalando os resultados, tendo em conta as amostras do *Random Cut Forest*. [20]

5.3.3.2. Obter amostra de dados de forma aleatória

Como primeiro passo no algoritmo *Random Cut Forest* é necessária a obtenção de uma amostra aleatória de dados de treino. Ou seja, uma amostra de tamanho *K* do total de *N* pontos de dados (conjuntos de dados). Caso os dados de treino sejam pequenos é possível utilizá-los todos, desenhando aleatoriamente *K* elementos desse conjunto. Normalmente isso não acontece, daí ser necessário utilizar uma técnica chamada de amostragem de reservatório (*reservoir sampling*). [20]

Esta técnica consiste num algoritmo para o desenho eficiente de amostras de dados aleatórios num conjunto de dados $S = \{S_1, \dots, S_N\}$, sendo que estes elementos apenas podem ser observados individualmente ou em lotes. Mesmo quando *N* não é conhecido à priori, a amostragem de reservatório pode ser utilizada. [20]

Por exemplo caso uma amostra seja solicitada, como quando *K=1* , o algoritmo será:

Algoritmo: Amostragem de reservatório

- Entrada: conjunto de dados ou *streaming* de dados $S = \{S_1, \dots, S_N\}$
- Inicializar a amostra aleatória X= S_1
- Para cada amostra observada $S_n, n = 2, \dots, N$:
- Escolha um número aleatório uniforme $\xi \in [0,1]$
- Se $\xi < 1/n$
- Definir $X = S_n$
- Retornar *X*

[20]

Este algoritmo seleciona uma amostra aleatória, de modo que $P(X = S_n) = 1/N$ para todos os $n = 1, \dots, N$. Quando $K > 1$, o algoritmo é mais complicado. É preciso ter em conta que, é necessário estabelecer uma distinção entre a amostragem aleatória que está com substituições e a que está sem as mesmas. [20]

5.3.3.3. Treinar um modelo RFC e produzir inferências

Como segundo passo do algoritmo *Random Cut Forest* é necessário, através da amostra de dados aleatórios (Figura 6) construir uma floresta de corte aleatório (*Random Cut Forest*). Os dados da amostra são divididos de igual forma pela quantidade de árvores da floresta. De modo a organizar-se recursivamente a partição nas árvores binárias, estas particionam o domínio de dados em caixas delimitadoras. [20]

Figura 6 - Conjunto de dados bidimensionais recebido onde é possível observar que a maioria dos dados estão agrupados (azul), exceto um conjunto de dados que está mais afastado (laranja). A árvore é inicializada com um nó raiz. [20]

Estes dados são organizados numa árvore pelo algoritmo *RCF*, que os organiza numa árvore, onde é calculada uma caixa delimitadora nos dados, selecionando assim uma dimensão aleatória, atribuindo mais peso às dimensões com maior "variação". Depois, de forma aleatória é determinada a posição de "corte" híper-plano através da dimensão. Estes dois subespaços que resultam destes passos definem a própria subárvore, como é possível visualizar na figura 7. [20]

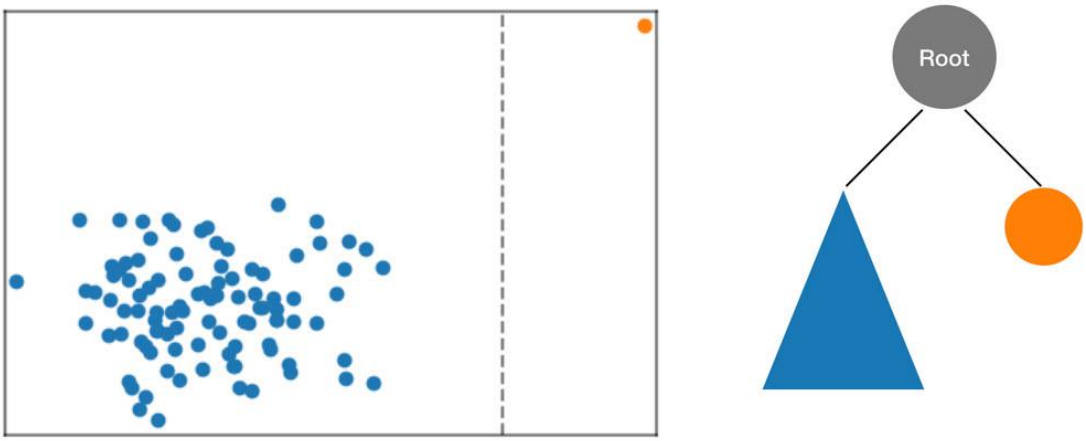


Figura 7 - Como é possível visualizar corte ocorre para separar um ponto isolado do restante da amostra. É mais provável que os dados anómalos residam no ponto isolado (direita) do que na árvore à esquerda. [20]

O processo do cálculo de caixas delimitadoras é repetido até que todas as folhas da árvore representem um único ponto de dados da amostra. Se o ponto único estiver suficientemente longe, a probabilidade de que um corte aleatório resulte no seu isolamento é bastante maior. [20]

Na execução de inferência com um modelo *RCF* treinado, a pontuação de anomalia final é registada como a média entre as pontuações registadas por cada árvore. É frequente o novo ponto de dados ainda não residir na árvore. Para determinar a pontuação associada ao novo ponto, o ponto de dados é inserido na árvore em questão, que, por sua vez, é eficientemente (e temporariamente) remontada de uma maneira equivalente ao processo de treino descrito acima. Ou seja, a árvore resultante é como se o ponto de dados de entrada fosse um membro da amostra, usada para construir a árvore no começo. A pontuação registada é inversamente proporcional à profundidade do ponto de entrada na árvore. [20]

5.3.3.4. Escolher híper-parâmetros

Os principais híper parâmetros que são utilizados de forma a se ajustar o modelo *RCF* são *num_trees* e *num_samples_per_tree (num_samples_per_tree* está relacionado à densidade esperada de anomalias no conjunto de dados, este deve ser escolhido de forma a que *1/ num_samples_per_tree* fique próximo da proporção entre dados anómalos e dados normais).

Ao aumentar o *num_trees*, o ruído observado nas pontuações anómalas será reduzido, dado que a pontuação final é o resultado da média das pontuações registadas em cada árvore. O recomendado é a utilização de 100 árvores e ajustar quando necessário de forma a manter o equilíbrio entre o ruído das pontuações e a complexidade do modelo. [20]

5.3.3.5. Diagrama Funcional

O diagrama disponibilizado abaixo demonstra a forma como os componentes do *Sagemaker* se encaixam;

1. Fazer o upload do conjunto de dados para a *AWS* (por exemplo para uma *AWS S3 bucket*);
2. Criar *uma notebook Jupyter* no *Sagemaker*;
3. Treinar o modelo de *Machine Learning* com o conjunto de dados;
4. Hospedar o modelo de *Machine Learning*;
5. Criar um terminal que permita obter previsões do modelo;
6. Executar o *notebook* e gere previsões.

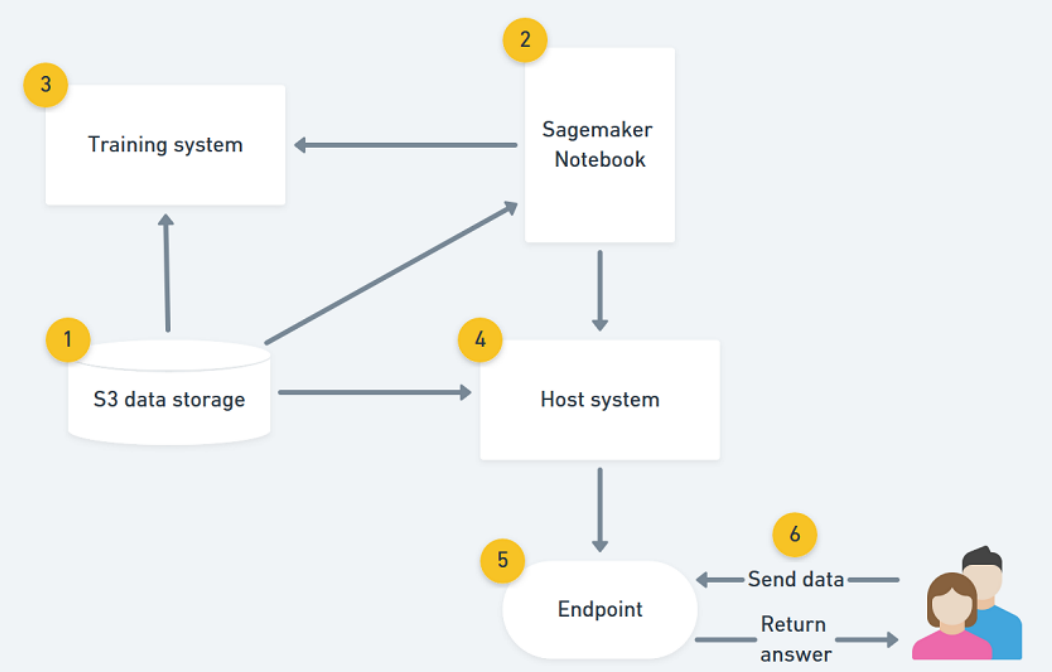


Figura 8 - Como o Sagemaker funciona (esquema de funcionamento). [21]

Uma árvore é uma maneira ordenada de armazenar dados numéricos, sendo que par se criar uma árvore, é necessário subdividir aleatoriamente os conjuntos dos pontos conseguindo

assim isolar o ponto que está a ser testado, conseguindo determinar se o ponto é uma anomalia ou não. Quando se subdivide os pontos de dados, é criado um novo nível na árvore. [21]

Subdividir menos vezes os pontos de dados, antes de isolar o ponto de dados de destino, aumenta a probabilidade de ser uma anomalia. Vão ser explicados dois exemplos, o exemplo 1 e o exemplo 2, no primeiro o ponto de dados de destino parece ser uma anomalia, no segundo, o ponto de dados de destino não é uma anomalia. [21]

Exemplo 1

É possível visualizar na Figura 9 que estão representados seis pontos a cinza que simbolizam dados extraídos aleatoriamente de um conjunto de dados. O ponto branco representa o ponto de dados que está a ser testado para se determinar se é uma anomalia ou não. É visível que o ponto branco fica mais distante dos outros valores, indicando assim que pode ser uma anomalia.

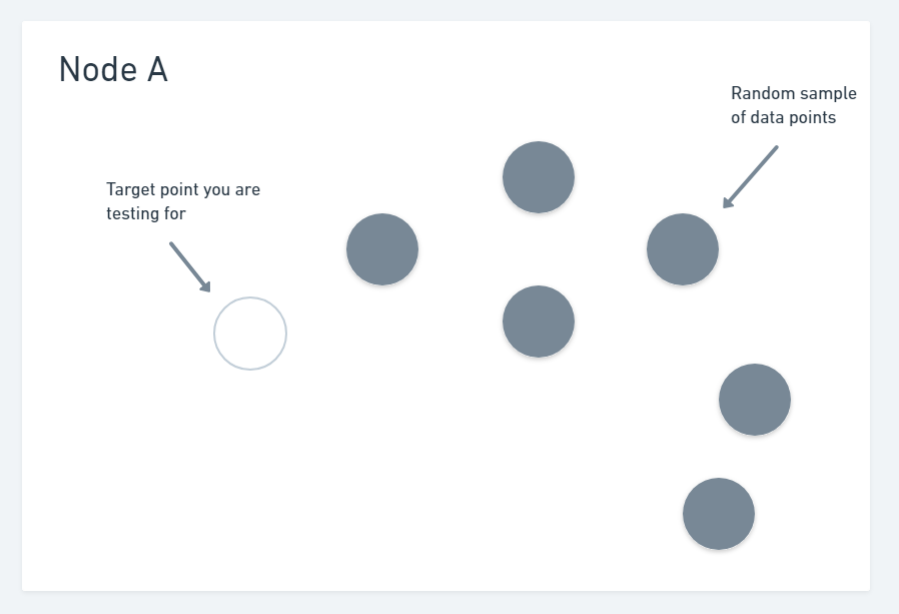


Figura 9 - Exemplo 1: pontos de dados de nível 1. [21]

Na figura 10 está representado o nível superior da árvore, ou seja, um nó único que representa todos os pontos de dados de um conjunto de dados (incluindo o ponto de dados de

destino que está a ser testado). O primeiro é representado sempre a cinza porque é representativo de todos os pontos de dados de um conjunto de dados. [21]

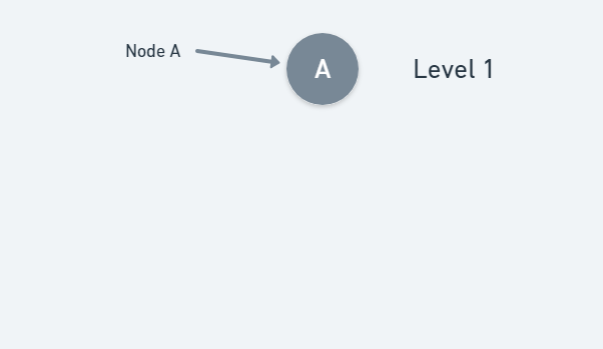


Figura 10 - Exemplo 1: árvore de nível 1. [21]

Na Figura 11 estão representados os pontos de dados após a primeira subdivisão (reta a dividir o nó B do C). A reta que subdivide os dados é inserida aleatoriamente através dos pontos de dados. Cada lado da subdivisão representa um nó na árvore. [21]

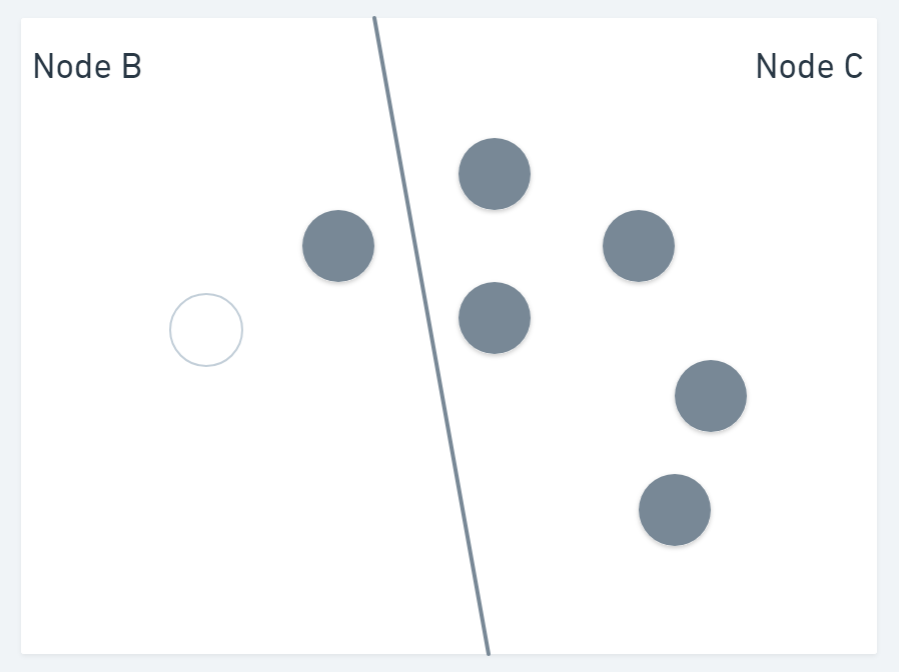


Figura 11 - Exemplo 1: pontos de dados de nível 2. [21]

Na figura 12 mostra o nível 2 da árvore, ou seja, o próximo nível, onde é possível verificar que os nós B e C representam respetivamente o lado esquerdo e direto da figura 5. Estes dois nós da árvore são representados na cor cinza dado que os dois lados do diagrama subdivididos na figura anterior contêm pelo menos um ponto cinza. [21]

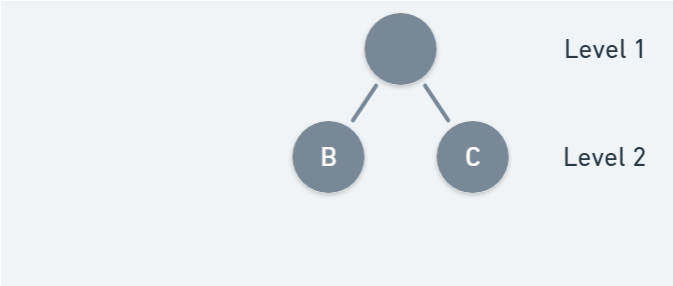


Figura 12 - Exemplo 1: árvore de nível 2. [21]

O próximo passo é subdividir ainda mais a parte do diagrama que contém o ponto de dados de destino. Como pode ser visualizado na figura 13. É possível verificar que o nó B foi subdividido novamente dando origem a mais dois nós, ao D e E.

Dado que o nó E contém apenas o ponto de dados de destino, nenhuma subdivisão adicional é necessária (isto porque já foi isolado). [21]

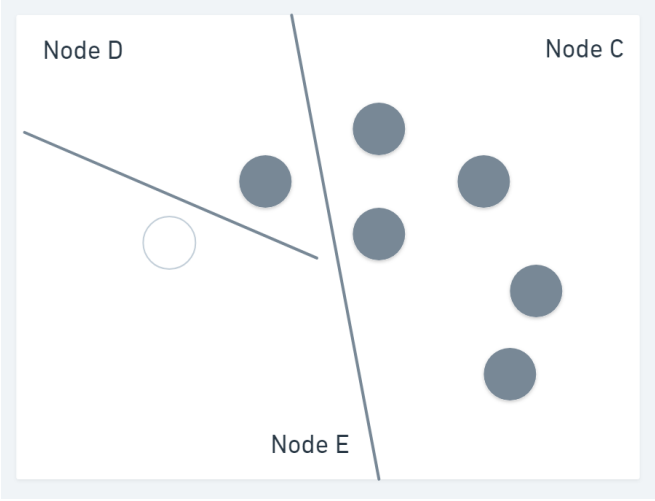


Figura 13 - Exemplo 1: pontos de dados de nível 3. [21]

Na próxima figura (figura 14) é possível visualizar a árvore final. O nó E é mostrado em branco porque contém o ponto de dados de destino. Esta árvore tem três níveis e quanto menor é a árvore, a probabilidade de o ponto ser uma anomalia é maior. Neste caso a probabilidade de ser uma anomalia é alta. [21]

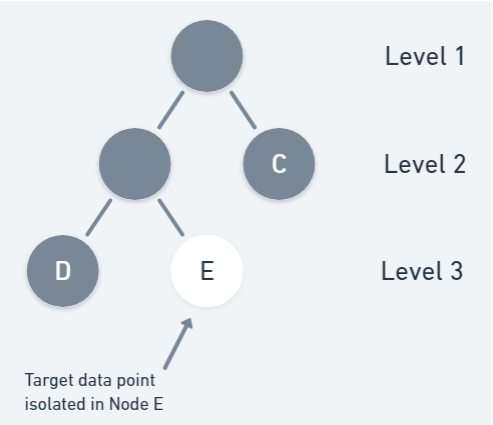


Figura 14 - Exemplo 1: árvore de nível 3. [21]

Exemplo 2

Como segundo exemplo, os dados selecionados de forma aleatória e estão mais próximos do ponto de dados de destino (ponto branco). O ponto de dados de destino é o mesmo utilizado no exemplo anterior. De salientar que os pontos de dados (representados a cinza) estão mais próximos do ponto de dados de destino do que no primeiro exemplo. [21]

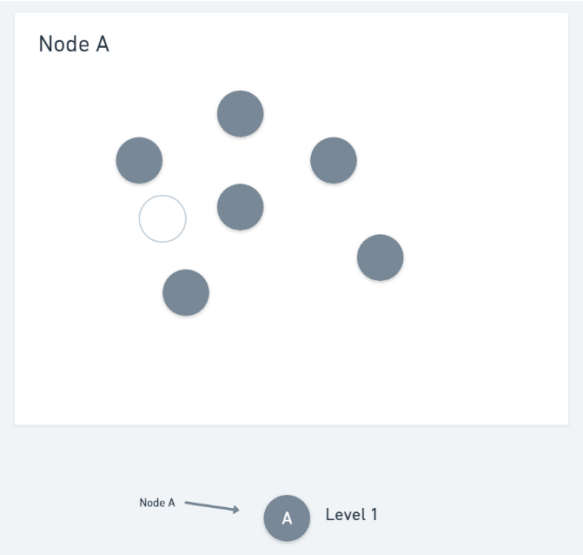


Figura 15 - Nó nível 1 representativa de todo os dados da desta amostra. [21]

Os dados são divididos então também em duas secções, denominadas nó B e C. Como pode ser visualizado na figura 16. [21]

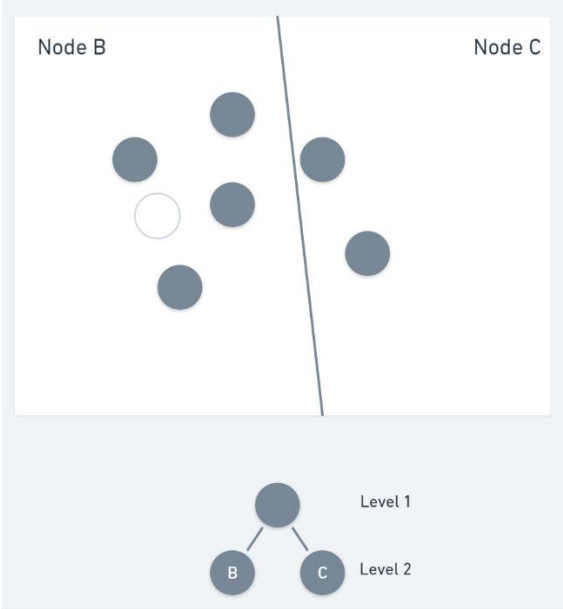


Figura 16 - É possível visualizar a nossa árvore nível 2 com os nós B e C. [21]

Depois, é necessário dividir novamente a secção que contém o ponto de dados de destino, ou seja, o nó B, criando assim o nó D e E, adicionando um novo nível à arvore (nível 3). [21]

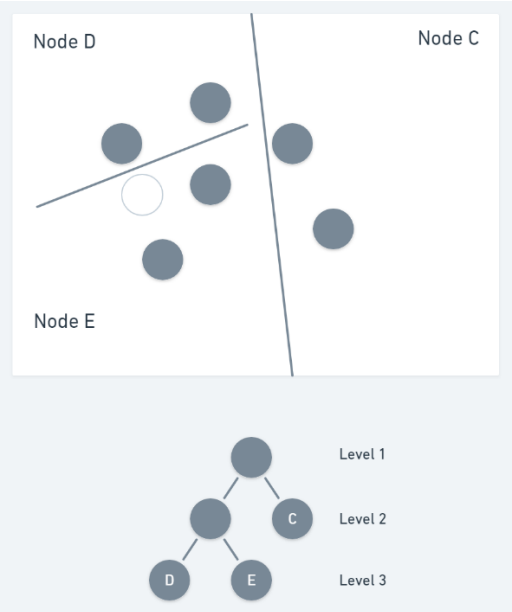


Figura 17 - É possível visualizar a árvore nível 3 que contem os nós D e E. [21]

É novamente dividido o nó com o ponto de dados de destino (nó E), que dará origem aos nós F e G. como demonstrado na figura 18. [21]

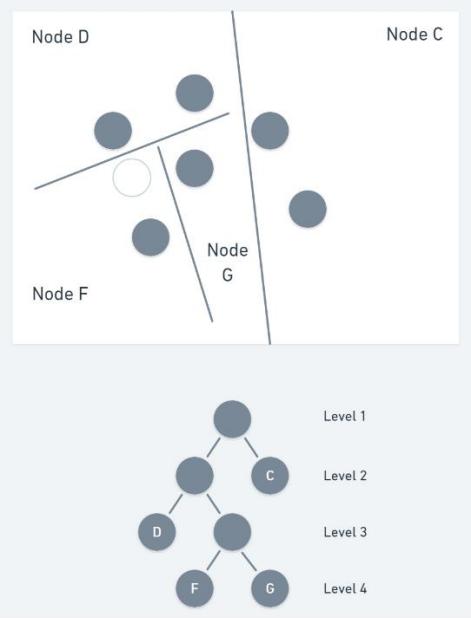


Figura 18 - Árvore nível 4 que contem os nós F e G. [21]

Dado que o ponto de dados de destino está no nó F este é dividido novamente originando os nós H e J. Com foi isolado o ponto de dados de destino no nó J não são necessárias mais divisões. A árvore/diagrama ficou assim com cinco níveis, podendo tirar assim a conclusão de que provavelmente o nosso ponto de dados não será uma anomalia. [21]

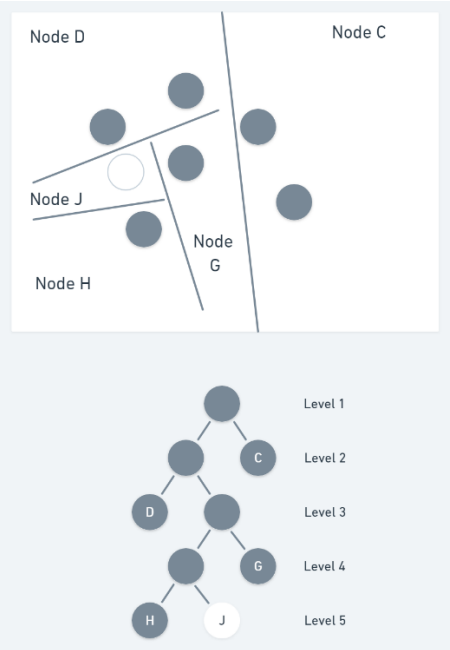


Figura 19 - Esta figura representa a árvore final, que contem cinco níveis (vai até ao nó J). [21]

O último passo vai se executado através do algoritmo *Random Cut Forest*, este último passo consiste na combinação de todas as árvores da floresta. É necessário ter especial atenção a amostras que contenham um valor elevado de árvores pequenas, isto porque a probabilidade de o ponto de dados de destino seja uma anomalia aumenta consideravelmente, se o número de árvores pequenas for reduzido já é mais improvável que seja uma anomalia. [21]

5.3.3.6. Explicação Matemática

É explorada uma abordagem aleatória. A randomização é uma ferramenta poderosa e conhecida por ser valiosa na aprendizagem (*Machine Learning*) supervisionada.

Definição 1 A *Robust Random Cut Tree (RRCT)* o conjunto *S* é gerado da seguinte maneira: [22]

- Escolher uma dimensão aleatória proporcional a $\frac{l_i}{\sum_j l_j}$, onde $l_i = max_{x \in S} x_i - min_{x \in S} x_i$.
- Escolher $X_i \sim Uniform[min_{x \in S} x_i, max_{x \in S} x_i]$.
- Deixar $S_1 = \{x | x \in S,\ x_i \leq X_i\}$ e $S_2 = S \backslash S_1$ e a recursão em *S1* e *S2*.

Robust Random Cut Forest (RRCF) é uma coleção de *RRCTs* independentes. [22]

A abordagem difere do procedimento acima na Etapa (1) e escolhe a dimensão para cortar uniformemente aleatoriamente. Discutimos esse algoritmo com mais detalhes na Seção 4.1 e é fornecida uma comparação extensiva. [22]

Em relação à questão (2), questiona-se: A estrutura de dados do *RRCF* contém informações suficientes que são independentes das especificidades do algoritmo de construção de árvores? Foi provado que a estrutura de dados *RRCF* preserva aproximadamente distâncias no seguinte sentido: [22]

Teorema 1 Considere-se o algoritmo na Definição 1. O peso de um nó numa árvore é a soma correspondente de dimensões $\sum_i l_i$. Considerando dois pontos *u*, *v*$\in$ *S*, é definida a distância da árvore entre *u* e *v* sendo o peso do antecessor (pai) menos comum de u,v. Então, a distância da árvore é sempre pelo menos a distância de *Manhattan* $L_1\ (u, v)$ e, expectavelmente no máximo $O\left(d \log \frac{|S|}{L_1(u,v)}\right)$ e tempos $L_1\ (u, v)$. [22]

O **Teorema 1** fornece uma baixa distância de extensão preservando a incorporação, remanescente do *Johnson-Lindenstrauss Lemma (Johnson & Lindenstrauss, 1984*), utilizando projeções aleatórias para distâncias $L_2()$ (que tem uma dependência muito melhor de *d*). Se um ponto estiver longe de outros (como é no caso das anomalias), o mesmo continuará a sê-lo sendo pelo menos o mais longe possível numa *Random Cut Tree*. A prova do **Teorema 1** segue as mesmas linhas da prova de aproximação de espaços métricos finitos por uma coleção de árvores. [22]

O teorema mostra que, se houver muito espaço vazio em torno de um ponto, ou seja, $\gamma = \min_v L1(u, v)$ é grande, então é necessário isolar o ponto dentro $O(d \log|S|/\gamma)$ níveis da

raiz (root). Já que para qualquer $p \geq 1$, a distância normal satisfaz $d^{1-1/p}\ L_p(u,v) \geq L_1(u,v) \geq L_p(u,v)$ e, daí que, o isolamento mais cedo aplica-se a todas as grandes distâncias $L_p(\,)$ simultaneamente. Isto apresenta um ponto para o sucesso do algoritmo da floresta de isolamento original em dados de dimensões reduzidas a moderados, dado que *d* é pequeno e a probabilidade de escolher uma dimensão não é tão importante se estes constituem uma quantidade reduzida. Com isto, um conjunto *RRCF* contém informações suficientes permitindo determinar anomalias baseadas na distância, sem que seja necessário colocar o foco nas especificidades da função de distância. Além disso, as escalas de distância são ajustadas à medida, tendo como base os espaços vazios entre pontos, pois as duas caixas delimitadoras podem encolher após o corte. [22]

Suponha-se que existe especial interesse no problema de manutenção da amostra de produzir uma árvore aleatoriamente (com a probabilidade correta) de *T (S − {x})* ou de *T (S ∪ {x})*. [22]

Teorema 2 Dada uma árvore T desenhada de acordo com T (S), se for excluído o nó contendo o ponto isolado *x* e o seu progenitor (figura 20), então a árvore resultante *T'* tem a mesma probabilidade como se fosse retirada de *T(S-{x})*. Da mesma forma, é possível produzir uma árvore *T''* como se desenhada aleatoriamente a partir de *T (S ∪{x})* é o tempo *O(d)* vezes a profundidade máxima de *T*, que normalmente é sublinear em | *T* |. [22]

O **Teorema 2** apresenta um comportamento intuitivamente natural quando os pontos são excluídos (como representado na abaixo). Se se inserir *x*, serão realizadas mais algumas operações para se excluir *x*, então não só preservamos as distribuições como as árvores permanecem muito próximas umas das outras, como se a inserção nunca tivesse acontecido. [22]

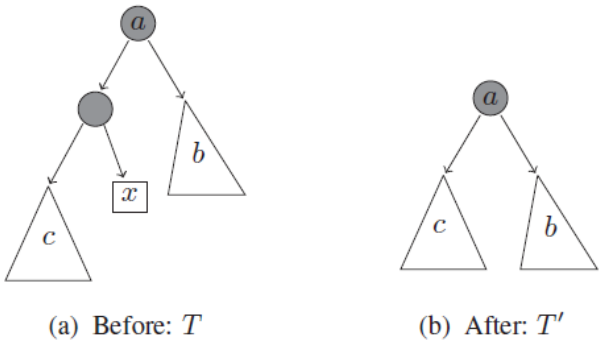


Figura 20 - Representação decrescente das árvores. [22]

O comportamento natural das exclusões não é verdadeiro se não se escolher as dimensões da construção do *RRCF*. Por exemplo, se se escolher as dimensões uniformemente e aleatoriamente, suponha-se que se constrói uma árvore para *(1, 0), (E, E), (0, 1)* onde *1>>E>0* e exclui-se *(1, 0)*. A probabilidade de obter uma árvore sobre os dois pontos restantes que usa um separador vertical é *3/4 - ∈/2* e não *1/2* conforme desejado. A probabilidade de se obter essa árvore no processo *RRCF* (depois da aplicação do Teorema 2) é *1 - ∈*, conforme desejado. Este comportamento natural sobre "apagar" também não é verdadeiro para a maioria dos métodos de particionamento do espaço. [22]

Teorema 3 É possível ser mantida uma árvore aleatória sobre uma amostra *S*, mesmo que a amostra *S* seja atualizada dinamicamente para transmitir dados utilizando o tempo de atualização sublinear e o espaço *O(d|S|)*. Pode-se usar reservar uma amostragem (de forma a se manter uma amostra aleatória uniforme de tamanho |*S*| ou uma amostra aleatória ponderada de tamanho recente | *S* |, em espaço proporcional a | *S* | *on the fly*. [22]

O processo de amostragem aleatória agora é ortogonal a partir da construção de uma *Robust Random Cut Forest*. Por exemplo, para se produzir uma amostra de tamanho *ρ|S|* para *ρ<1*, numa amostragem aleatória uniforme, é possível realizar amostragem direta por rejeição, na amostra tendenciosa para o tempo recente, é necessário excluir os pontos de menor prioridade *(1-ρ)|S|*. Essa noção de redução da amostra por através de exclusões é suportada pelo Teorema 2, mesmo para taxas de amostragem reduzida que são determinadas após a construção de árvores durante o pós-processamento. [22]

Teorema 4 Dada uma árvore *T(S)* para a amostra *S*, se existir um procedimento que reduz a amostragem por exclusão, tem-se um algoritmo que fornece simultaneamente uma árvore com redução da amostragem para cada taxa de amostragem reduzida. [22]

Os teoremas 3 e 4 juntos separam a noção de amostragem da tarefa de análise, daí que eliminam a necessidade de ajustar o tamanho da amostra como parâmetro inicial. [22]

Além disso, a manutenção dinâmica de árvores no Teorema 3 fornece um mecanismo para responder a perguntas contra factuais, conforme indicado no Teorema abaixo, ou seja, o Teorema 5. [22]

Teorema 5 Dada uma árvore *T(S)* para a amostra *S* e um ponto *p*, podemos eficientemente através da computação criar uma árvore aleatória em $T(S \cup \{p\})$. Isto permite responder a certas questões como, por exemplo, qual teria sido a profundidade esperada se tivesse sido incluída na amostra? [22]

A capacidade de responder a questões contra factuais como esta é essencial para determinar anomalias. Intuitivamente, rotulado um ponto *p* como uma anomalia quando a distribuição conjunta de incluir o ponto é significativamente diferente da distribuição que o exclui. O teorema 5 permite prever com eficiência a distribuição conjunta, incluindo o ponto *p*. Em vez de medir o efeito dos pontos de dados amostrados em *p* para determinar seu rótulo (como é medido por noções como profundidade esperada), deve-se medir o efeito de *p* nos pontos amostrados. Isto leva à definição de anomalias. [22]

5.3.3.6.1. Anomalias

Considere-se as seguintes hipóteses:

1. Normalmente uma anomalia é fácil de descrever, por exemplo se considerarmos o António a utilizar um chapéu vermelho num mar de chapéus escuros. Embora possa ser difícil encontrar o António no meio da multidão, se esquecermos os rostos e despertarmos a atenção na cor, o reconhecimento da anomalia é bastante simples. [22]
2. Uma anomalia torna-se mais difícil de descrever o restante dos dados, se o António não estivesse a usar o chapéu vermelho, talvez não se admitisse a possibilidade de que os chapéus possam ser coloridos. Em essência, uma anomalia desloca a nossa atenção da observação normal para essa nova. [22]

Um dos pontos mais importantes é, então, quantificar a mudança de atenção. Suponha-se a atribuição dos ramos esquerdos ao *bit 0* e direitos ao *bit 1* numa árvore de uma *Random Cut Forest*. [22]

Ao considerar os bits que especificam um ponto (excluindo os bits necessários para armazenar os valores do atributo do próprio ponto). Isto define a complexidade de um modelo aleatório *MT* que, neste caso, corresponde a uma árvore *T* que se ajusta aos dados iniciais. Logo, o número de bits necessários para expressar um ponto corresponde à sua profundidade na árvore. [22]

Dado um conjunto de pontos *Z* e um ponto $y \in Z$ *seja f (y, Z, T)* a profundidade de *y* na árvore *T*. Considere-se agora a árvore produzida pela exclusão de *x* como no Teorema 2 como

T (Z - {x}). De notar que os dados *T* e *x* na árvore *T (Z− {x})* é determinada exclusivamente. [22]

A profundidade de *y* em *T (Z - {x})* é deixada ser *f (y, Z - {x}, T)* (é permitida "cair" a qualificação da árvore nesta notação, pois esta é definida exclusivamente). [22]

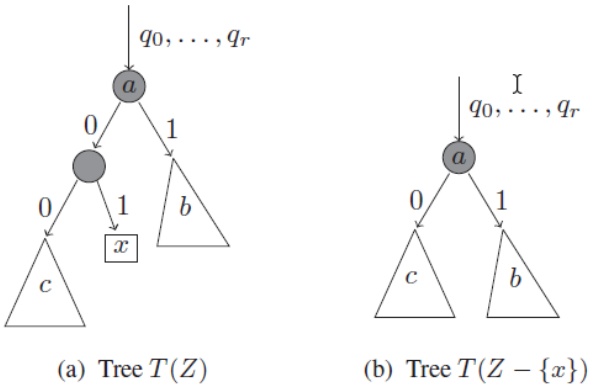


Figura 21 - Correspondência das árvores. [22]

Considerando agora um ponto *y* na subárvore *c* na Figura 2(a). A representação de bits em *T* seria *q0, ... , qr, 0, 0, ...* . A complexidade do modelo, demonstrada como *|M(T)|* o número de bits necessário para escrever a descrição de todos os pontos *y* na árvore *T* será então, $|M(T)| = \sum_{y \in Z} f(y, Z, T)$. Se se retirar *x*, a nova complexidade do modelo será

$$|M(T')| = \sum_{y \in Z - \{x\}} f(y, Z - \{x\}, T')$$

onde *T' = T(Z − {x})* é uma árvore sobre *Z - {x}*. Se se considerar a mudança esperada na complexidade do modelo num modelo aleatório. [22] Entretanto, como existe um mapeamento de muitos para um de *T (Z) a T (Z - {x})* como consequência do Teorema 2, é possível expressar a segunda soma sobre *T (Z)* em vez de *T '= T (Z - {x})* :

$$E_{T(T)}[|M(T)|] - E_{T(Z-\{x\})}[|M(TZ - \{x\})|]$$

$$= \sum_{T} \sum_{y \in Z - \{x\}} \Pr[T] \left(f(y, Z, T) - f(y, Z - \{x\}, T') \right) + \sum_{T} \Pr[T] f(x, Z, T)$$

(1)

[22]

Definição 2 Definindo o deslocamento do bit ou deslocamento de um ponto *x* para ser o aumento na complexidade do modelo de todos os outros pontos, ou seja, para um conjunto Z, para aprisionar a externalidade introduzida por *x*, é necessário definir onde *T '= T (Z - {x}),*

$$DISP(x,Z) = \sum_{T,y\in Z-\{x\}} \Pr[T]\big(f(y,Z,T) - f(y,Z-\{x\},T')\big)$$

De Observar que a mudança total na complexidade do modelo é *DISP(x,Z)+* $g(x,Z)$ onde $g(x,Z) = \sum_T \Pr[T]\, f(x,Z,T)$ é a profundidade esperada do ponto *x* num modelo aleatório é a profundidade esperada do ponto *x* num modelo aleatório. Em vez de indicar que as anomalias correspondem a $g(\,)$, foca-se em valores maiores de *DISP ( ).* O deslocamento do nome é mais claro com base no seguinte ponto: [22]

Ponto 1 - O deslocamento esperado causado por um ponto *x* é o número esperado de pontos no nó irmão do nó folha que contém *x*, quando a divisão é feita de acordo com o algoritmo da Definição 1. [22]

Deficiências

Enquanto a definição 2 aponta para uma possível definição de uma anomalia, a definição conforme declarada não é robusta para duplicações ou quase duplicações. Ao considerar um cluster denso e um ponto *p* longe do cluster. [22]

O deslocamento de *p* será grande. Mas se existir um ponto *q* muito próximo de *p*, o deslocamento de *q* na presença de *p* é pequeno. Esse fenômeno é conhecido como *outlier masking*. Duplicações e quase duplicações são naturais e, portanto, a semântica de qualquer algoritmo de deteção de anomalias deve tolerá-los. [22]

Resiliência duplicada

Tendo em consideração a noção de que o António tem alguns amigos que o ajudam a se esconder, esses amigos são conspiradores e se se quiser livrar de todos os conspiradores, a descrição muda significativamente. Especificamente, em vez de apenas remover o ponto *x*, removemos um conjunto *C* com $x \in C$. Análogo à Equação (1), [22]

$$E_{T(Z)}[|M(T)|] - E_{T(Z-C)}[|M(T(Z-C))|]$$
$$= DISP(C,Z) + \sum_T \sum_{y\in C} \Pr[T]\, f(y,Z,T) \qquad (2)$$

onde *DISP (C, Z)* é a noção de deslocação extensiva a subconjuntos, como, por exemplo *T '' = T(Z - C),* [22]

$$\sum_{T,y\in Z-C} \Pr[T]\,\big(f(y,Z,T) - f(y,Z-C,T'')\big) \qquad (3)$$

Na ausência de qualquer conhecimento de domínio, parece que o deslocamento deve ser atribuído igualmente a todos os pontos em *C*. Logo, uma escolha natural de determinar *C* parece ser *max DISP(C,Z)/ |C|* sujeito a x $\in C \subseteq Z$. No entanto, existem 2 problemas:

- Existem muitos subconjuntos *C*;
- Numa configuração de *streaming*, é provável que se use uma amostra $S \subset Z$.

[22]

Logo, a escolha supostamente natural não se estende às amostras. Para evitar ambos os problemas, é permitido que a escolha de *C* seja diferente para diferentes amostras *S*, na verdade é permitido que a António colabore com diferentes membros em diferentes testes, isto dá origem à seguinte definição:

[22]

Definição 3 O Deslocamento Colusivo (*Collusive Displacement*) de *x* indicado pelo *CODISP(x,Z, |S|)* de um ponto *x* é definido como:

$$E_{S \subseteq Z,T}\left[max_{x \in C \subseteq S} \frac{1}{|C|} \sum_{y \in S-C} \left(f(y,S,T) - f(y,S-C,T'')\right)\right]$$

[22]

Ponto 2 – O *CODISP (x, Z, | S |)* pode ser estimado com eficiência.
Enquanto *CODISP (x, Z, | S |)* depende de | *S* |, a dependência não é grave. [22]

Definição 4 – *Outliers* (anomalias) correspondem a grandes *CODISP()*. [22]

5.3.3.6.2. Forest Maintenance on a Stream

Será discutido como as *Robust Random Cut Tree* podem ser mantidas dinamicamente. Depois, que a *RRCF(S)* seja uma distribuição por árvores através da Definição 1 em *S*. Observe-se as seguintes operações:

- Inserção: Dado *T* retirado da distribuição *RRCF(S)* e $p \in S$ produz um *T '* retirado de *RRCF(S* $\cup \{p\}$*)*.
- Eliminação: Dado *T* retirado da distribuição *RRCF(S)* e $p \in S$ produz um *T'* retirado de *RRCF(S-{p})*. É necessária a seguinte observação simples.

[22]

Observação 1 A separação de um conjunto de pontos *S* e *p* com recurso a um corte paralelo ao eixo é possível se e só se for possível separar a caixa delimitadora mínima alinhada ao eixo *B (S)* e *p* utilizando um corte paralelo ao eixo. [22]

O próximo ponto oferece uma propriedade estrutural sobre as árvores *RRCF*. É de interesse atualizações incrementais com o mínimo de alterações possível num conjunto de árvores. Observe-se que, dada uma árvore específica, têm-se dois casos exaustivos:

(i) O novo ponto a ser excluído (inserido respetivamente) não é separado pelo primeiro corte; [22]

(ii) O novo ponto é excluído (o respetivo inserido) é separado pelo primeiro corte. O Ponto 3 refere isto para coleções de árvores (não apenas para uma única árvore) que satisfaçam (i) e (ii) respetivamente. [22]

Ponto 3 Dados o ponto *p* e o conjunto de pontos *S* com uma caixa delimitadora mínima de eixo paralelo *B (S)* tal que $p \notin B$:

(i) Qualquer que seja a dimensão *i*, a probabilidade de escolher um corte paralelo de eixo numa dimensão *i* que divide *S* utilizando o algoritmo de floresta de isolamento ponderado (*Weighted Isolation Forest Algorithm*) é exatamente a mesma que a probabilidade condicional de escolher um corte paralelo de eixo que divide $S \cup \{p\}$ na dimensão *i*, condicionada a não isolar *p* de todos os pontos de *S*.

(ii) Dada uma árvore aleatória da *RRCF(S* $\cup \{p\}$*)*, condicionada ao facto de o primeiro corte isolar *p* de todos os pontos de *S*, o resto da árvore é uma árvore aleatória na *RRCF(S).*

[22]

5.3.3.6.2.1. Exclusão de pontos

Algoritmo 1 Algoritmo ForgetPoint.

1: Encontrar o nó *v* na árvore em que *p* está isolado em *T*.

2: Deixar que *u* seja o irmão de *v*. Eliminar o pai de *v* e de *u* e substituir esse pai por *u* (ou seja, reduz-se o caminho de *u* até a raiz).

3: Atualizar todas as caixas delimitadoras a partir dos *u* (novo) pai para cima, este estado não é necessário para supressões, mas é útil para inserções.

4: Retornar a árvore modificada *T '*.

[22]

Ponto 4 Se *T* foi extraído da distribuição *RRCF (S)*, o algoritmo 1 produz uma árvore *T '* que é desenhada aleatoriamente a partir da distribuição de probabilidade *RRCF (S - {p}).* [22]

Ponto 5 A operação de exclusão pode ser realizada no tempo *O(d)* vezes a profundidade do ponto *p*. [22]

De notar que se se excluir um ponto aleatório da árvore, o tempo de execução da operação de exclusão será *O(d)* vezes a profundidade esperada de qualquer ponto. Da mesma forma que se se excluir pontos cuja profundidade é mais rasa que a maioria dos pontos da árvore, é possível melhorar o tempo de execução do ponto 5. [22]

5.3.3.6.2.2. Inserção de pontos

Dada uma árvore *T* do *RRCF(S)*, produz-se uma árvore *T '* a partir da distribuição *RRCF (S ∪{p})*. O algoritmo é fornecido no algoritmo 2 (algoritmo a baixo). Junta-se as decisões que refletem a mesma divisão em *T '* e em *T*, desde que *p* não esteja fora de uma caixa delimitadora em *T*. Até este momento está-se a realizar os mesmos passos como na construção da floresta em *S ∪{p}*, com a mesma probabilidade. [22]

Ponto 6 Se *T* foi extraído da distribuição *RRCF (S)*, o algoritmo 1 produz uma árvore *T '* que é desenhada aleatoriamente a partir da distribuição de probabilidade *RRCF (S ∪{p})*. [22]

5.3.3.6.3. Floresta de isolamento e outros trabalhos relacionados

5.3.3.6.3.1. O algoritmo da floresta de isolamento

O algoritmo de isolamento florestal (*Isolation Forest Algorithm*) utiliza um conjunto de árvores semelhante às construídas na Definição 1, com a modificação de que a dimensão a cortar é escolhida uniformemente ao acaso. Dado um novo ponto *p*, este algoritmo segue os cortes e calcula a profundidade média do ponto através de um conjunto de árvores. O ponto é rotulado como uma anomalia se a pontuação exceder um limiar, o que corresponde à

profundidade média ser pequena em comparação com *log* |*S*| onde *S* é uma amostra de tamanho adequado dos dados. [22]

A vantagem da floresta de isolamento é que diferentes dimensões são tratadas de forma independente e o algoritmo é invariável para escalar diferentes dimensões de forma diferente. [22]

Algoritmo 2 Algoritmo InsertPoint.

1: Considere-se um conjunto de pontos *S′* e uma árvore *T(S′).* Quere-se inserir 'p' e produzir a árvore *T′(S′ ∪{p}).*

2: Se *S'* = ∅ então retorna-se um nó contendo o nó único *p*.

3: Caso contrário, *S′* tem uma caixa delimitadora $B(S') = [x_1^l, x_1^h]\text{x}[x_2^l, x_2^h]\text{x}\ldots[x_d^l, x_d^h]$. Que $x_i^l \leq x_i^h$ para todos os *i*.

4: Para todos os *i* fica $\hat{x}_i^l = \min\{p_i, x_i^l\}$ e $\hat{x}_i^h = \max\{x_i^h, p_i\}$.

5: Escolher um número aleatório $r \in \left[0, \sum_i (\hat{x}_i^h - \hat{x}_i^l)\right]$.

6: Este *r* corresponde a uma escolha específica de um corte na construção da *RRCF (S′ ∪ {p}).* Por exemplo, pode-se calcular $\arg\min\{j | \sum_{i=1}^{j} (\hat{x}_i^h - \hat{x}_i^l) \geq r\}$ e o corte (*cut*) corresponde à escolha $\hat{x}_j^l + \sum_{i=1}^{j} (\hat{x}_i^h - \hat{x}_i^l) - r$ na dimensão *j*.

7: Se este corte separar *S′* e *p* (isto é, não estiver no intervalo $\left[x_j^l, x_j^h\right]$) então, é possível utilizá-lo como o primeiro corte para *T′(S′ ∪{p})*. Cria-se um nó, um lado do corte é *p* e o outro lado é a árvore *T(S′).*

8: Se este corte não separar *S′* e *p*, então manda-se fora este o corte. Escolhe-se exatamente a mesma dimensão que *T(S′)* em *T′(S′∪{p})* e o mesmo valor do corte escolhido por *T(S′)* e efetua-se a divisão. O ponto *p* vai para um dos lados, com o subconjunto *S″*. Repete-se este procedimento com uma caixa delimitadora menor *B(S″)* de *S″*. Para o outro lado, utiliza-se a mesma subárvore que em *T(S′).*

9: Nos dois casos, atualiza-se a caixa delimitadora de *T ′*.

[22]

6. Solução Proposta

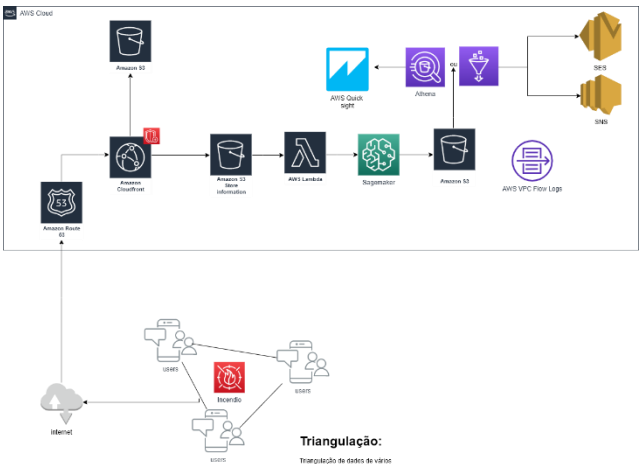


Figura 22 - Infraestrutura responsável por analisar os dados enviados pelos utilizadores e detetar, efetivamente, consoante os dados que recebeu se se trada de um incendio ou não.

A solução proposta para a análise de dados enviados pelos cidadãos no âmbito de projeto de alerta de fogos florestais consiste na aplicação de um classificador para detetar anomalias, sejam causadas por erros nos sensores ou por utilizadores mal-intencionados. Neste sentido a arquitetura representada na Figura 22 é a ideal para cumprir estes objetivos. Os dados recolhidos são recebidos pelo *AWS Cloudfront*, sendo por sua vez acelerados reduzindo as latências e armazenados num *AWS S3 Bucket*. Um modelo de *Random Cut Forest* é executado no *AWS Sagemaker* e também testado a possibilidade de utilizar a regressão linear como previsão futura. Os dados processados são armazenados num *AWS S3 Bucket*, permitindo *querys* diversas ou a sua visualização no *AWS Quicksight*.

Infelizmente não foi possível reproduzir a arquitetura acima descrita, devido aos custos envolvidos. Foi elaborada uma arquitetura mais simples, mantendo a parte mais importante de análise (*AWS Sagemaker*) com o *Random Cut Forest* e a previsão usando uma regressão linear.

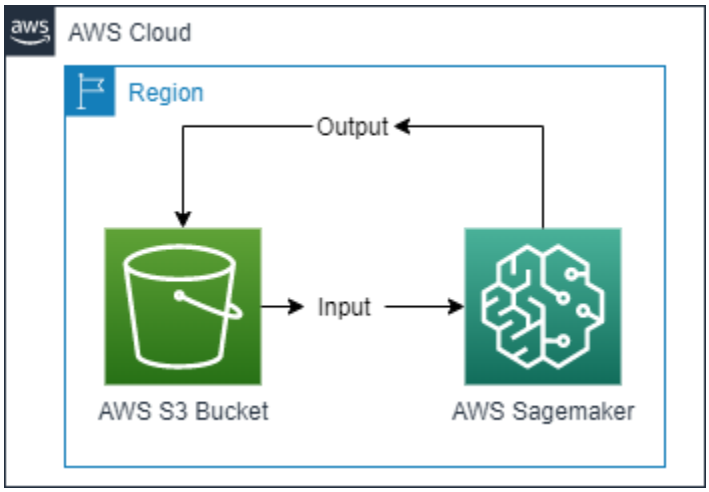


Figura 23 - Arquitetura implementada.

Para validação da estratégia proposta foi utilizado um *dataset* com 22695 temperaturas diferentes provenientes do *Numenta Anomaly Benchmark (NAB).* Estes são relativos a dados do sensor de temperatura de um componente interno de uma grande máquina industrial, comportando valores ao longo de 3 meses. Após a criação do *AWS S3 Bucket* contendo o *dataset,* foi criado um *website* estático privado com o objetivo de se utilizarem os dados através de um *notebook instance*, esta instância é denominada de *ml.t2.medium* (*2vCPUS* e *4GB RAM*).

Após ser efetuada a análise do dados dentro da *notebook instance* será possível exportar os mesmos em vários formatos (p.e. *PDF* e *HTML*).

6.1. Aplicação do Algoritmo Random Cut Forest

O *dataset* original foi inspecionado, de forma a que sejam verificadas as existências de alguns padrões ou estruturas subjacentes que se podem fornecer ao modelo como “pistas” ou avaliar se existe ruído que possa ser pré-processado. [23] [24] [25]

```
In [3]: temp_data.head()
```

Out[3]:

	timestamp	value
0	2013-12-02 21:15:00	73.967322
1	2013-12-02 21:20:00	74.935882
2	2013-12-02 21:25:00	76.124162
3	2013-12-02 21:30:00	78.140707
4	2013-12-02 21:35:00	79.329836

Figura 24 - Inspeção efetuada ao dataset.

Depois da inspeção do *dataset*, pode ser visualizado graficamente com a biblioteca *bokeh*, como pode ser visualizado na Figura 25.

```
In [21]: import bokeh
         import bokeh.io
         from bokeh.models import HoverTool
         bokeh.io.output_notebook()
         from bokeh.plotting import figure, show, output_file, ColumnDataSource
         from bokeh.models.formatters import DatetimeTickFormatter
         import bokeh.palettes
         temp_data['timestamp'] = pd.to_datetime(temp_data['timestamp'])
         output_file("datetime.html")
         date = temp_data['timestamp']
         temp = temp_data['value']

         x = date.values
         y = temp.values
         score = []

         string_x = list(map(str, x))
         source = ColumnDataSource(temp_data)
         source.add(temp_data['timestamp'].apply(lambda d: d.strftime('%Y-%m-%d %H:%M:%S')), 'event_date_formatted')
         #Hover Tool

         hover = HoverTool(
             names = ["temp"],
             tooltips=[
                 ( 'date',   '@event_date_formatted'),
                 ( 'temperature',  '$y' ),
             ],


                 # mostrar uma tooltip sempre que o cursor estiver verticalmente em linha com um glyph

             mode='vline'
         )

         p = figure(x_axis_type='datetime', plot_width=900, plot_height=400, tools=[hover, 'pan','wheel_zoom','box_zoom','reset'])
         p.line( name="temp", x='timestamp',y= 'value',source=source, line_width=2,color='navy', alpha=0.5)
         show(p)
```

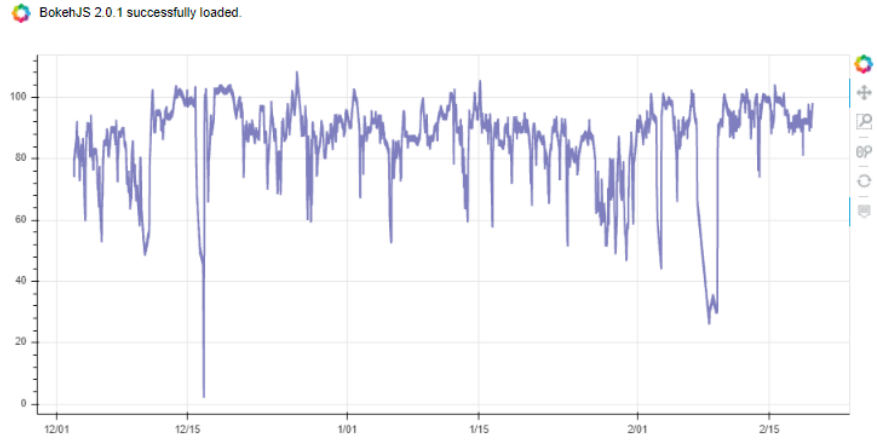


Figura 25 - Visualização dos dados presentes no dataset gráficamente.

Neste momento os dados são armazenados num *AWS S3 Bucket*.

O Algoritmo *Random Cut Forest* aceita os dados no formato *RecordIO Protobuf*, que através de funções do *SageMaker Python API* auxiliam a converteção mais facilmente os dados utilizados para este formato. Na figura 26 é possível a visualização de como os dados são convertidos e enviados para *o AWS S3 Bucket* com um prefixo Amazon S3 de destino, especificado no início do *notebook* na parte do *Setup AWS Credentials*. [23] [24] [25]

```
In [22]: def convert_and_upload_training_data(ndarray, bucket, prefix, filename='data.pbr'):
             import boto3
             import os
             from sagemaker.amazon.common import numpy_to_record_serializer

             # converter o numpy array para o formatp Protobuf RecordIO
             serializer = numpy_to_record_serializer()
             buffer = serializer(ndarray)

             # Upload para o S3
             s3_object = os.path.join(prefix, 'train', filename)
             boto3.Session().resource('s3').Bucket(bucket).Object(s3_object).upload_fileobj(buffer)

             s3_path = 's3://{}/{}'.format(bucket, s3_object)
             return s3_path

         #RCV só funciona com base num conjunto de valores.
         s3_train_data = convert_and_upload_training_data(
             temp_data['value'].values.reshape(-1,1),
             bucket,
             prefix)
         print('Uploaded data to {}'.format(s3_train_data))

         Uploaded data to s3://ssmbucket-mb/rcf-benchmarks/train/data.pbr
```

Figura 26 - Dados convertidos e enviados para o AWS S3 Bucket.

Até a este instante foi efetuado *upload* do *dataset* para o *AWS S3 Bucket*. Em seguida, foi configurado um *training job* do *AWS SageMaker* de forma a se utilizar o algoritmo *Random Cut Forest (RCF)* no *dataset* presente no *AWS S3 Bucket.* [23] [24] [25]

A próxima etapa é então a especificação da localização da imagem do Docker que contem o algoritmo do *SageMaker Random Cut Forest*. Para se minimizar a latência de comunicação, a AWS fornece um *container* com destino a cada região, na qual o *SageMaker* está disponível. É visível na figura 27 o *container* escolhido tendo em conta a região onde está a ser executado o *notebook* (Neste caso está a se executado na Irlanda). [23] [24] [25]

Particularmente para um training job o *AWS Sagemaker Random Cut Forest (RCT)* são os seguintes Hiperparâmetros:

- *num_samples_per_tree* - o número de pontos de dados amostrados aleatoriamente enviados para cada árvore. De forma geral, *1/núm_samples_per_tree* aproxima a razão estimada de anomalias dos pontos normais no conjunto de dados.
- *num_trees* - o número de árvores a criar na floresta, cada árvore aprende um modelo separado a partir de diferentes amostras de dados. O modelo de floresta completa utiliza a pontuação média prevista das anomalias de cada árvore constituinte.
- *feature_dim* - a dimensão de cada ponto de dados.

[23] [24] [25]

Além dos hiperparâmetros do modelo *Random Cut Forest (RCF)*, fornecem-se parâmetros adicionais que definem outros elementos como o tipo de instância *AWS EC2* onde será efetuado o treino do modelo, o *AWS S3 Bucket* contem os dados e a função de acesso AWS. O tipo de instância recomendada para ser lançada a instância de treino/endpoint são das famílias *ml.m4, ml.c4, ou ml.c5*. [23] [24] [25]

In [23]:
```
%%time
session = sagemaker.Session()

# Especificar a localização do training container
container = '438346466558.dkr.ecr.eu-west-1.amazonaws.com/randomcutforest:latest'

# Especificar informação geral sobre o training job
rcf = sagemaker.estimator.Estimator(
    container,
    execution_role,
    input_mode='File',
    output_path='s3://{}/{}/output'.format(bucket, prefix),
    train_instance_count=1,
    train_instance_type='ml.m4.xlarge',
    sagemaker_session=session,
)

# Definir hiperparâmetros
rcf.set_hyperparameters(
    num_samples_per_tree = 500,
    num_trees = 100,
    feature_dim = 1,
)

# O treino do RCF requer dados fragmentados.

s3_train_input = sagemaker.session.s3_input(
    s3_train_data,
    distribution='ShardedByS3Key',
    content_type='application/x-recordio-protobuf',
)


# Executar o training job sobre os dados de entrada armazenados no S3
rcf.fit({'train': s3_train_input})
```

Figura 27 - Especificação da localização do container, informação do training job, hiperparâmetros do algogoritmo RCF.

```
2020-07-24 15:13:24 Completed - Training job completed
Training seconds: 76
Billable seconds: 76
CPU times: user 614 ms, sys: 39.9 ms, total: 654 ms
Wall time: 4min 42s
```

Figura 28 - Output a indicar que o training job foi concluido com sucesso.

É possível visualizar no final dos *logs* do *output* do *script* executado anteriormente "Job Complete", isso significa que o treino foi concluído com sucesso e o modelo RCF ficou armazenado no caminho especificado para o *output*. Também é possível visualizar informações e o *status* de um *training job* utilizando a consola *AWS SageMaker*. [23] [24] [25]

```
In [24]: print('Training job name: {}'.format(rcf.latest_training_job.job_name))

         Training job name: randomcutforest-2020-07-24-15-09-16-810
```

Figura 29 - Visualização do training job.

Um modelo treinado através do algoritmo *Random Cut Forest (RCF)* não faz seja o que for por si próprio, é necessário utilizar o modelo que calculamos para fazer inferências sobre os dados. Isto significa calcular as pontuações das anomalias a partir dos pontos de dados das séries temporais de *input*. [23] [24] [25]

É necessário criar um *endpoint* de inferência utilizando a função *AWS Sagemaker Python SDK deploy()* a partir do trabalho que definido acima. Especificou-se o tipo de instância onde a inferência é processada, bem como um número inicial de instâncias para *spin up*. A instância utilizada foi uma *ml.m4.xlarge (4vCPUs e 16GB RAM)*. [23] [24] [25]

```
In [25]: rcf_inference = rcf.deploy(
             initial_instance_count=1,
             instance_type='ml.m4.xlarge',
         )
```

Figura 30 - Especificação da instância onde é processada a inferência.

Com o *Endpoint* de inferência *Sagemaker RCF*, já é possível confirmar a configuração e o *status* do *Endpoint* na consola do *AWS SageMaker*. [23] [24] [25]

```
In [26]: print('Endpoint name: {}'.format(rcf_inference.endpoint))

         Endpoint name: randomcutforest-2020-07-24-15-09-16-810
```

Figura 31 - Nome do Endpoint criado.

É possível passar dados numa variedade de formatos para o *Endpoint* de inferência. Neste caso, demonstra-se a passagem de dados em formato *CSV*. Existem outros formatos disponíveis sendo estes formatos *JSON* e *Protobuf RecordIO*. Foram utilizados *SageMaker Python SDK utilities csv_serializer* e *json_deserializer* ao configurar o *endpoint* de inferência. [23] [24] [25]

```
In [27]: from sagemaker.predictor import csv_serializer, json_deserializer

         rcf_inference.content_type = 'text/csv'
         rcf_inference.serializer = csv_serializer
         rcf_inference.deserializer = json_deserializer
```

Figura 32 - Passagem de dados em formato CSV e utilização das utilities csv_serializer e json_deserializer.

Vão ser enviados os conjuntos de dados de treino, em formato *CSV*, para o *Endpoint* de inferência para que se possa detetar automaticamente as anomalias que foram visualizadas nas tramas acima. O serializador e o desserializador (*serializer and deserializer*) tratam automaticamente da conversão do *datatype* dos *Numpy NDArrays*. [23] [24] [25]

```
In [28]: import time
         prediction_data = pd.read_csv(data_filename, delimiter=',')
         prediction_data_numpy = prediction_data['value'].values.reshape(-1,1)
         results = rcf_inference.predict(prediction_data_numpy)
```

Figura 33 - Conjuntos de dados de treino enviados, em formato CSV, para o Endpoint de inferência para que se possa detetar automaticamente as anomalias.

Vão ser calculados e delineados os resultados das anomalias (pontuação das anomalias) a partir de todo o conjunto de dados das temperaturas. [23] [24] [25]

In [29]:

```
from bokeh.models import Range1d, LinearAxis
def prediction(data, threshold=None):
    prediction_data = data
    prediction_data['timestamp'] = pd.to_datetime(prediction_data['timestamp'])
    output_file("datetime.html")
    temp = prediction_data['value']
    scores = prediction_data['score']

    hover = HoverTool(
        names = ["temp","score"],
        tooltips=[
            ( 'date',   '@event_date_formatted'),
            ( 'temp',  '@temp_y' ),
           ( 'Score',  '@score_y' ),
        ],
    )
    source = ColumnDataSource(prediction_data)
    source.add(prediction_data['timestamp'].apply(lambda d: d.strftime('%Y-%m-%d %H:%M:%S')), 'event_date_formatted')
    source.add(temp, 'temp_y')
    source.add(scores, 'score_y')

    p = figure(x_axis_type='datetime', plot_width=1000, plot_height=350, tools=[hover, 'pan','wheel_zoom','box_zoom','reset'])
    p.line( name = "temp", x='timestamp',y= 'value',source=source, line_width=2,color='navy', alpha=0.5, legend_label="Temperatu
re")
    p.extra_y_ranges = {"Anomaly": Range1d(start=0, end=10)}
    p.add_layout(LinearAxis(y_range_name="Anomaly"), 'right')
    p.line( name = "score", x='timestamp',y='score',source=source, line_width=2,color='red', alpha=0.5, y_range_name="Anomaly",l
egend_label="Score")
    p.legend.location = "top_left"
    p.legend.click_policy="hide"
    p.title.text = "Anomaly Detection for Device Temperature"



    #Seleccionar as pontuações das anomalias mais elevadas
    score_mean = prediction_data['score'].mean()
    score_std = prediction_data['score'].std()
    if not threshold:
        score_cutoff = score_mean + 3*score_std
    else:
        score_cutoff = threshold
    print("The best threshold value is: " + str(score_cutoff))
    anomalies = prediction_data[prediction_data['score'] > score_cutoff]
    sorted_anomalies = anomalies.sort_values(by=['score'], ascending=False)
    print(sorted_anomalies)
    source = ColumnDataSource(prediction_data)
    p.circle( sorted_anomalies['timestamp'],sorted_anomalies['score'], line_width=8,color='black', alpha=0.5, y_range_name="Anom
aly")
    p.circle( sorted_anomalies['timestamp'],sorted_anomalies['value'], line_width=5,color='black')


    show(p)

def process_data(file):
    prediction_data = pd.read_csv(file, delimiter=',')
    prediction_data_numpy = prediction_data['value'].values.reshape(-1,1)
    results = rcf_inference.predict(prediction_data_numpy)
    scores = [datum['score'] for datum in results['scores']]
    prediction_data['score'] = pd.Series(scores, index=prediction_data.index)
    return prediction_data
```

Figura 34 - Calcular e delinear os resultados das anomalias.

Fazer previsões em todo o conjunto de dados:

```
In [30]: prediction(process_data('s3://ssmbucket-mb/rcf-benchmarks/train/machine_temperature_system_failure.csv'))
```

```
The best threshold value is: 6.017795865428086
                timestamp      value     score
3986  2013-12-16 17:25:00   2.084721  6.848094
3985  2013-12-16 17:20:00   4.117241  6.825141
3984  2013-12-16 17:15:00   6.440238  6.794717
3983  2013-12-16 17:10:00   6.918645  6.788762
3981  2013-12-16 17:00:00   9.633952  6.751497
...                   ...        ...       ...
19381 2014-02-08 03:20:00  40.654995  6.025140
19387 2014-02-08 03:50:00  40.696344  6.024148
19398 2014-02-08 04:45:00  40.727206  6.023083
3992  2013-12-16 17:55:00  40.782224  6.021679
19382 2014-02-08 03:25:00  40.796344  6.021225

[414 rows x 3 columns]
```

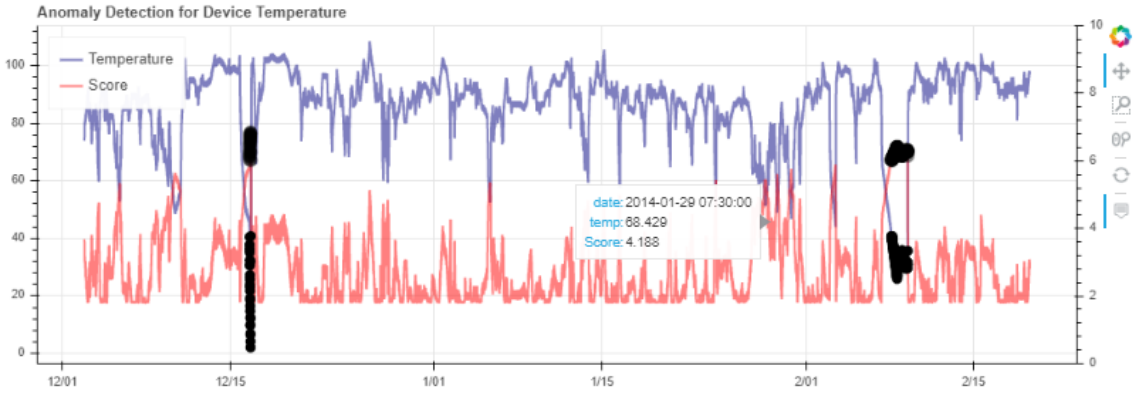


Figura 35 - Previsão em todo o conjunto de dados.

Fazer previsões sobre um intervalo de dados:

```
In [31]: prediction(process_data('s3://ssmbucket-mb/rcf-benchmarks/train/temp_predictions.csv'), 6.01)

The best threshold value is: 6.01
              timestamp      value     score
209 2013-12-16 17:25:00   2.084721  6.848094
208 2013-12-16 17:20:00   4.117241  6.825141
207 2013-12-16 17:15:00   6.440238  6.794717
206 2013-12-16 17:10:00   6.918645  6.788762
204 2013-12-16 17:00:00   9.633952  6.751497
205 2013-12-16 17:05:00  10.001966  6.746475
210 2013-12-16 17:30:00  12.120381  6.716422
203 2013-12-16 16:55:00  12.414093  6.712093
202 2013-12-16 16:50:00  14.671060  6.679224
201 2013-12-16 16:45:00  16.457109  6.649704
200 2013-12-16 16:40:00  18.591971  6.611621
199 2013-12-16 16:35:00  19.277179  6.598647
198 2013-12-16 16:30:00  21.737032  6.554823
197 2013-12-16 16:25:00  22.983839  6.519860
196 2013-12-16 16:20:00  23.829044  6.497073
195 2013-12-16 16:15:00  25.680672  6.441804
194 2013-12-16 16:10:00  27.212228  6.400193
193 2013-12-16 16:05:00  30.504461  6.298110
192 2013-12-16 16:00:00  30.699649  6.293280
211 2013-12-16 17:35:00  32.001703  6.255197
191 2013-12-16 15:55:00  32.454943  6.240659
190 2013-12-16 15:50:00  35.072456  6.172791
189 2013-12-16 15:45:00  36.249653  6.140396
188 2013-12-16 15:40:00  37.791275  6.112991
187 2013-12-16 15:35:00  40.461427  6.032286
215 2013-12-16 17:55:00  40.782224  6.021679
```

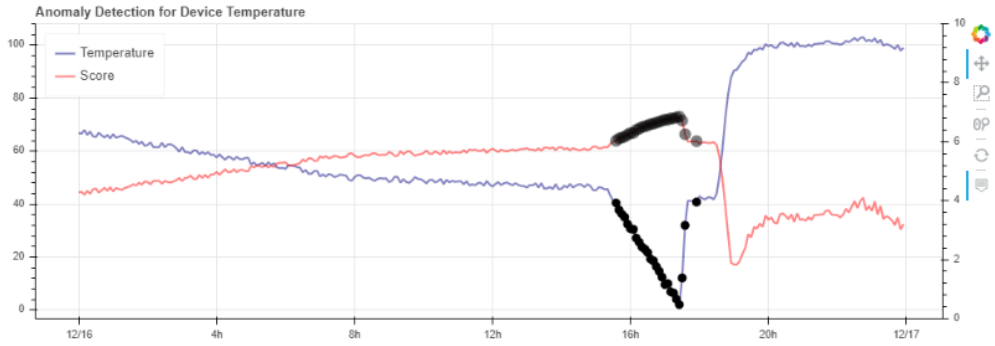


Figura 36 - Previsões sobre um novo dado.

De indicar que o *score* da anomalia atinge picos onde há um ponto de dados anómalo, bem como em alguns lugares onde os nossos olhos não são tão precisos. [23] [24] [25]

Foi imprimido e desenhado todos e qualquer ponto de dados com pontuações superiores a 3 desvios padrão (aproximadamente 99,9º percentil) da pontuação média.

É possível verificar que possivelmente a primeira anomalia foi um *planned shutdown*. Em relação às outras anomalias no meio, verifica-se uma mudança subtil, mas ainda assim observável no comportamento, indicando o início real de um problema que levou a uma eventual falha do sistema. [23] [24] [25]

É possível verificar que em 2013/12/16 o sistema de forma planeada foi desligado e em 2014/02/08 existiu uma falha grave.

Verifique-se que o algoritmo conseguiu capturar estes eventos. Abaixo estão adicionadas estas anomalias ao gráfico de pontuação. [23] [24] [25]

Com as escolhas atuais de Hiperparâmetros nota-se que o limiar de três desvio padrão, embora capaz de capturar as anomalias conhecidas, bem como as aparentes no gráfico, é bastante sensível a comportamentos anómalos. A adição de árvores ao modelo *Random Cut Forest (RCF)* do *AWS SageMaker* poderia suavizar os resultados, bem como utilizar um conjunto de dados maior. [23] [24] [25]

6.2. Sliding Windows Regressão linear

A regressão linear executa a tarefa de prever um valor da variável dependente *y* com base numa variável independente *x*, logo, esta técnica de regressão descobre uma relação linear entre *x* (input) e *y* (output).. É também calculada também a correlação entre conjuntos de dados. Um dos objetivos da solução é então prever as temperaturas seguintes tendo em conta as atuais. É necessário importar algumas bibliotecas que vão ser uteis no cálculos e gráficos elaborados mais à frente. [26] [27]

```
In [2]: import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as seabornInstance
        from sklearn.model_selection import train_test_split
        from sklearn.linear_model import LinearRegression
        from sklearn import metrics
        %matplotlib inline
```

Figura 37 - Bibliotecas importadas.

De seguida é necessário importar os conjuntos de dados CSV.

```
In [9]: dataset = pd.read_csv("s3://ssmbucket-mb/rcf-benchmarks/train/machine_temperature_system_failure.csv", delimiter=',')
```

Figura 38 - Importar os dados CSV do AWS S3 Bucket onde estão.

Após importarem-se os dados é necessário efetuar-se uma inspeção aos dados e verificar-se quantas colunas e linhas têm. [26] [27]

```
In [10]: dataset.shape
Out[10]: (22695, 2)
```

Figura 39 - É possível visualizar-se que o dataset atual tem 22695 linhas e 2 colunas.

Também é possível a visualização dos detalhes estatísticos do conjunto de dados que se está a utilizar. [26] [27]

```
In [11]: dataset.describe()
```

Out[11]:

	value
count	22695.000000
mean	85.926498
std	13.746912
min	2.084721
25%	83.080078
50%	89.408246
75%	94.016252
max	108.510543

Figura 40 - Detalhes estatísticos em relação ao conjunto de dados utilizado.

Na figura 41 é possível visualizarem-se os dados num gráfico.

```
In [13]: dataset.plot(x='timestamp', y='value', style='o')
         plt.title('Temp along time')
         plt.xlabel('Timestamp')
         plt.ylabel('Temp')
         plt.show()
```

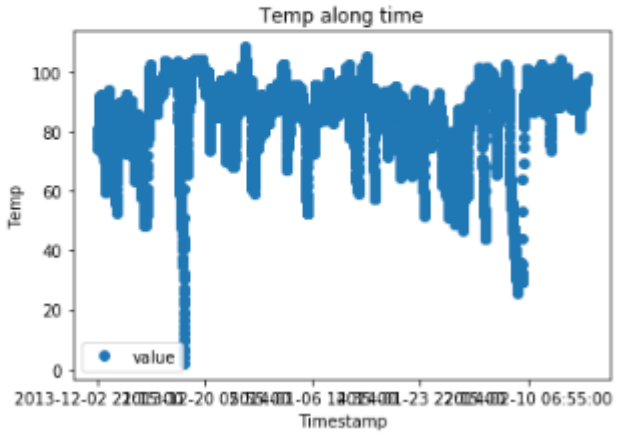


Figura 41 - Neste gráfico é possível a visualização de todos das a temperaturas representadas no dataset utilizado.

A figura 42 é mostra a temperatura máxima média. [26] [27]

```
In [14]: plt.figure(figsize=(15,10))
         plt.tight_layout()
         seabornInstance.distplot(dataset['value'])
Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x7f8844702c18>
```

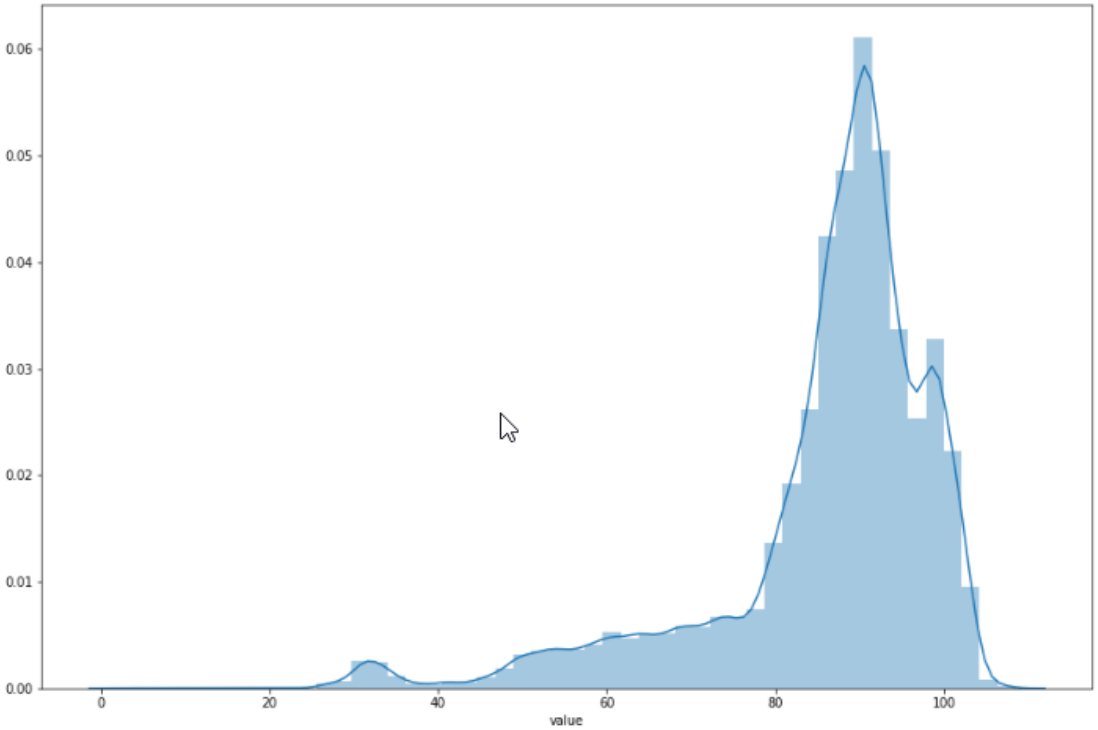


Figura 42 - É possível visualizar que a temperatura máxima média se situa entre aproximadamente 87 e 92.

Os dados são divididos em atributos e rótulos, os atributos são as variáveis independentes, enquanto os rótulos são variáveis dependentes cujos valores devem ser previstos. Neste conjunto de dados existem apenas duas colunas. O objetivo é prever o *value*, dependendo do *timestamp* registado, logo o conjunto de atributos consistirá na coluna *timestamp*, que é armazenada na variável X, e o rótulo será na coluna *value*, que é armazenada na variável y. É necessário converter o *timestamp* num valor numérico. [26] [27]

```
In [45]: import datetime as dt
         myvar = pd.to_datetime(dataset['timestamp']) #converter a data para um valor numerico
         myvar=myvar.map(dt.datetime.toordinal)
         X = myvar.values.reshape(-1,1)
         y = dataset['value'].values.reshape(-1,1)

         0        735204
         1        735204
         2        735204
         3        735204
         4        735204
                   ...
         22690    735283
         22691    735283
         22692    735283
         22693    735283
         22694    735283
         Name: timestamp, Length: 22695, dtype: int64
```

Figura 43 - Dividir os dados em atributos e rótulos. Conversão da data para um valor numérico.

Em seguida o *dataset* é dividido da seguinte forma, 80% dos dados no conjunto de treino, e 20% dos dados são testados, usando a variável *test_size* . [26] [27]

```
In [46]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

Figura 44 - Divisão dos dados para treino e para testes.

De seguida o treino da regressão linear requer:

```
In [47]: regressor = LinearRegression()
         regressor.fit(X_train, y_train)

Out[47]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

Figura 45 - Treino do algoritmo.

A previsão executa-se através do comando:

```
In [50]: y_pred = regressor.predict(X_test)
```

Figura 46 - Efetuar previsões.

Após se efetuarem as previsões foram comparados os valores de *output X_test* com os valores previstos. [26] [27]

```
In [51]: df = pd.DataFrame({'Actual': y_test.flatten(), 'Predicted': y_pred.flatten()})
         df
```

Out[51]:

	Actual	Predicted
0	85.664319	86.201261
1	30.080704	85.468252
2	88.653841	85.468252
3	62.231677	85.659471
4	57.554582	86.519960
...	...	...
4534	88.012166	85.978171
4535	80.353425	86.567765
4536	93.144866	86.089716
4537	82.031584	86.153456
4538	83.447442	86.551830

4539 rows × 2 columns

Figura 47 - Comparação entre os valores atuais e os previstos.

Também é possível visualizar a comparação efetuada anteriormente num gráfico de barras. De indicar que como o número de registos é muito grande, para fins de representação, apenas são apresentados 25 registos. [26] [27]

```
In [52]: df1 = df.head(25)
         df1.plot(kind='bar',figsize=(16,10))
         plt.grid(which='major', linestyle='-', linewidth='0.5', color='green')
         plt.grid(which='minor', linestyle=':', linewidth='0.5', color='black')
         plt.show()
```

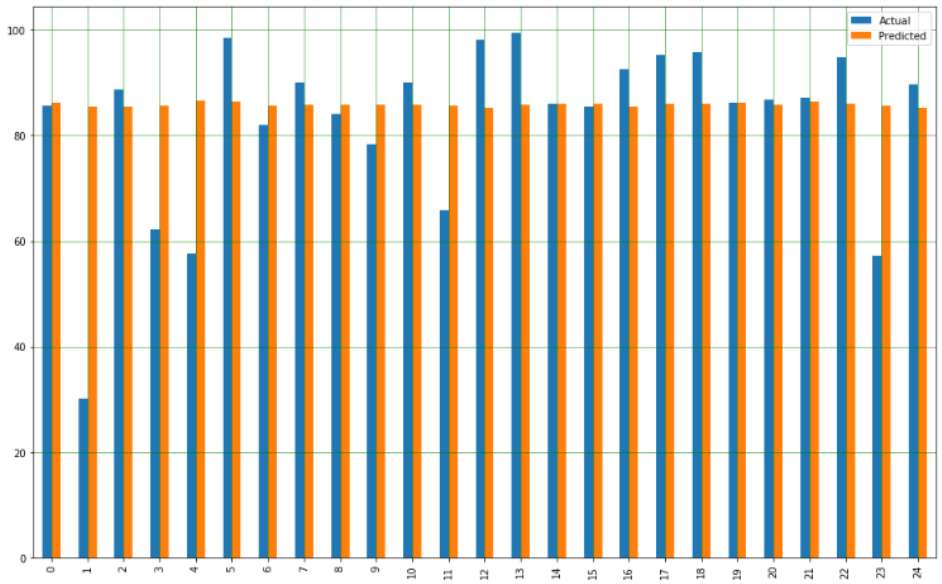


Figura 48 - Representação gráfica dos valores atuais e dos previstos.

Embora o modelo não seja muito preciso pontualmente mais em concreto nas posições 1, 3, 4, 11 e 23, as percentagens previstas são próximas das reais. [26] [27]

```
In [53]: plt.scatter(X_test, y_test,  color='gray')
         plt.plot(X_test, y_pred, color='red', linewidth=2)
         plt.show()
```

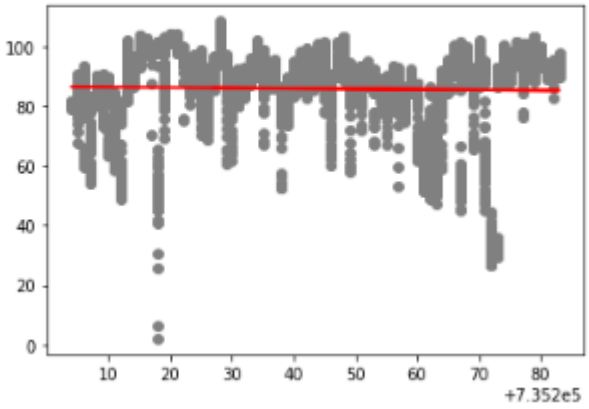


Figura 49 - traçar a reta no gráfico.

O passo final é avaliar o desempenho do algoritmo. Esta etapa é particularmente importante para comparar o desempenho de diferentes algoritmos num conjunto de dados específico. Para algoritmos de regressão, há três métricas de avaliação que costumam ser utilizadas:

- Mean Absolute Error (MAE) - é a média do valor absoluto dos erros;
- Mean Squared Error (MSE) - é a média dos erros quadráticos;
- Root Mean Squared Error (RMSE) - é a raiz quadrada da média dos erros quadráticos. [26] [27]

```
In [54]: print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
         print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
         print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))

Mean Absolute Error: 9.694472177780346
Mean Squared Error: 193.29992456505667
Root Mean Squared Error: 13.903234320296002
```

Figura 50 - Cálculo dos valores de MAE, MSE e RMSE.

Através dos valores do *Mean Absolute Error (MAE), Mean Squared Error (MSE)* e *Root Mean Squared Error (RMSE)* é possível verificar que entre o valor previsto e o atual

vão 9,69 unidades (MAE). Destaca-se também a existência também de valores atípicos (com grandes variações) de algumas temperaturas, como pode ser visualizado através do valor de 193,29 do MSE. Constata-se através do RMSE que existe um valor de erro de 13,90.

Estes valores tão elevados poderão justificar-se devido à janela utilizada não ter sido a melhor, dado que foi considerado todo o conjunto de dados e não um pequeno intervalo de tempo. É importante também realçar que existem temperaturas que oscilam de forma muito drástica em alguns intervalos de tempo. Por fim, possivelmente, para se obter valores mais aceitáveis seria necessário considerar um conjunto maior de dados.

7. Conclusões

Esta tese visou o problema de deteção de informação incorreta por parte dos utilizadores num *stream* de dados variantes no tempo. O problema assume maior importância dados os vários projetos de *Citizen on Science* com foco na deteção de fogos florestais, identificação de lixo nos oceanos e linha costeira, bem como outros cenários relacionados com a biodiversidade. Este estudo centrou-se na eliminação de dados errados vindos dos utilizadores, de forma a se oberem dados mais fidedignos. Para atingir este objetivo foi fundamental utilizar uma *framework* capaz de processar dados de dimensões consideráveis e em ambiente *Cloud* dada a proporção deste tipo de aplicações. *Cloud Computing* tem melhorado os negócios informáticos de uma forma generalizada. Com as muitas vantagens e possibilidades oferecidas por esta tecnologia, permitiu o desenvolvimento tanto desta solução, assim como no futuro possibilitará o de outras soluções cada vez mais elaboradas e de complexidade elevada, permitindo também a evolução da tecnologia de todas as áreas. Neste caso, uma das aplicações diretas poderá ser em relação à deteção de incêndios e de dados mal-intencionados enviados por utilizadores.

Foi desenvolvida uma solução com sucesso, que teve por base um *dataset* que simula os dados de temperaturas recebidas ao longo do tempo de *x* em *x* minutos, para analisar estes dados foi utilizado o algoritmo de *Random Cut Forest* com a finalidade da deteção de anomalias desenvolvido pela *Amazon Web Services* e o método *Sliding Windows* Regressão linear, de forma a ser possível o cálculo da correlação entre dados e uma previsão de valores possíveis.

Os resultados foram positivos em relação à deteção de anomalias, foi possível a visualização das mesmas em que, em alguns casos não era percetível ao olho humano e através do algoritmo *Random Cut Forest* foi possível a sua deteção. Em relação ao *Sliding Windows* Regressão linear, a implementação também foi concluída com sucesso, conseguindo a previsão de valores futuros e dos *Mean Absolute Error (MAE), Mean Squared Error (MSE)* e *Root Mean Squared Error (RMSE).* Em relação aos valores de MAE, MSE e RMSE, estes não foram os melhores. Este fato deve-se a que a janela utilizada não foi efetivamente a melhor tendo em conta que se utilizou a totalidade dos dados e não apenas pequenos intervalos. Não foi possível fazer mais testes com intervalos mais pequenos, devido ao "*budget"* disponibilizado para o desenvolvimento da infraestrutura e de testes que foram necessários efetuar.

8. Trabalho Futuro

Como trabalho futuro deixo a sugestão de se realizarem testes idênticos aos que foram feitos nesta tese, mas em intervalos de tempo mais reduzidos, apenas com uma amostra da totalidade do *dataset*, isto poderá permitir previsões mais aproximadas nos casos de grandes variações.

9. Bibliografia

[1] J. M. E. P. Sunyoung Kim, "Citizen Science," *Sensr: Evaluating a Flexible Framework for Authoring,* 23–27 02 2013.

[2] M. G. A. A. L. C. B. C. O. D. M. H. R. K. C. K. J. P. A. R. S. S. U. S. Soledad Luna, "Developing Mobile Applications for Citizen Science," *Developing Mobile Applications for Environmental and Biodiversity Citizen Science: Considerations and Recommendations,* 20 06 2018.

[3] T. M. C. G. E. A. J. M.-R. G. E. Elizabeth R. Ellwood Guest Editors, "Citizen science and conservation: Recommendations for a rapidly moving field," *Elsevier,* nº BIOC-06994, p. 4, 08 October 2016.

[4] D. F. R. A. H. D. S. W.-K. W. a. S. K. Wesley M. Hochachka, "Special Issue: Ecological and evolutionary informatics," *Data-intensive science applied to broad-scale citizen science,* pp. 130-137, 02 2012.

[5] T. G. R. M. B. A. S. P. D. G. U. S. f. S. a. a. D. Peter Mell, "The NIST Definition of Cloud Computing," *Recommendations of the National Institute of Standards and Technology,* 27 04 2012.

[6] F. Pinheiro, "CLOUD COMPUTING," 2012. [Online]. Available: https://www.gta.ufrj.br/ensino/eel879/trabalhos_vf_2010_2/fernando/caracteristicas.html.

[7] Microsoft Azure, "What is SaaS?," [Online]. Available: https://azure.microsoft.com/en-us/overview/what-is-saas/.

[8] Dialogic, "Introduction to Cloud Computing," 10 2017. [Online]. Available: https://www.dialogic.com/~/media/products/docs/whitepapers/12023-cloud-computing-wp.pdf. [Acedido em 15 01 2020].

[9] Amazon Web services, "Types of Cloud Computing," [Online]. Available: https://aws.amazon.com/pt/types-of-cloud-computing/. [Acedido em 06 01 2020].

[10] Microsoft Azure, "What is PaaS?," [Online]. Available: https://azure.microsoft.com/en-us/overview/what-is-paas/. [Acedido em 16 01 2020].

E. C. O. K. ‘. U. C. Awodele Oludele, “On the Evolution of Virtualization and
[11] Cloud Computing: A Review,” *Journal of Computer Sciences and Applications,* vol. 2, nº 3, p. 1, 2014.

Dialogic Corporation, “Introduction to Cloud Computing,” 2010. [Online].
[12] Available: http://www.dialogic.com/~/media/products/docs/whitepapers/12023-cloud-computing-wp.pdf.

Gartner, “2020 Magic Quadrant for Cloud,” AWS - Amazon Web Services, 1 09
[13] 2020. [Online]. Available: https://pages.awscloud.com/GLOBAL-multi-DL-gartner-mq-cips-2020-learn.html?pg=LWIAWS. [Acedido em 10 09 2020].

AWS, “Cloud computing with AWS,” 2020. [Online]. Available:
[14] https://aws.amazon.com/pt/what-is-aws/?nc1=h_ls. [Acedido em 17 01 2020].

AWS, “Global Infrastructure,” 2020. [Online]. Available:
[15] https://aws.amazon.com/about-aws/global-infrastructure/?nc1=h_ls. [Acedido em 16 01 2020].

J. B. e. S. Kotecha, “Storage Options in the AWS Cloud,” *Storage Options in the*
[16] *AWS Cloud,* pp. 18-19, 10 2013.

AWS, “What Is Amazon SageMaker?,” AWS - Amazon Web Services, 2020.
[17] [Online]. Available: https://docs.aws.amazon.com/sagemaker/latest/dg/whatis.html. [Acedido em 18 01 2020].

AWS, “Get Started with Amazon SageMaker Studio,” AWS - Amazon Web
[18] Services, 2020. [Online]. Available: https://docs.aws.amazon.com/sagemaker/latest/dg/gs-studio.html. [Acedido em 18 01 2020].

MFG Labs, “Medium,” 25 April 2019. [Online]. Available:
[19] https://medium.com/@mfg_labs/anomaly-detection-in-hostile-environment-28e4ff875621. [Acedido em 06 04 2020].

AWS - Amazon Web Services, “Amazon SageMAker,” 2020. [Online].
[20] Available: https://docs.aws.amazon.com/sagemaker/latest/dg/sagemaker-dg.pdf#randomcutforest. [Acedido em 07 04 2020].

D. H. a. R. Nichol, “Machine Learning for Business,” *The Random Cut Forest*
[21] *Algorithm,* 14 02 2019.

[22] N. M. G. R. O. S. Sudipto Guha, “Robust Random Cut Forest Based Anomaly Detection On Streams,” *Robust Random Cut Forest Based Anomaly Detection On Streams*, 2016.

[23] W. Badr, “Detect Anomalies in Your Data with Amazon SageMaker (Level 300),” Amazon Web Services (AWS), 24 08 2018. [Online]. Available: https://www.youtube.com/watch?v=yx1vf3uapX8&t=1s. [Acedido em 07 2020].

[24] J. D. M. L. K. a. M. J. Chris Swierczewski, “Use the built-in Amazon SageMaker Random Cut Forest algorithm for anomaly detection,” Amazon Web Services (AWS), 25 04 2018. [Online]. Available: https://aws.amazon.com/pt/blogs/machine-learning/use-the-built-in-amazon-sagemaker-random-cut-forest-algorithm-for-anomaly-detection/. [Acedido em 07 2020].

[25] W. Badr, “Random-Cut-Forest,” Will Badr, 23 10 2019. [Online]. Available: https://github.com/wmlba/Random-Cut-Forest. [Acedido em 07 2020].

[26] N. S. Chauhan, “A beginner’s guide to Linear Regression in Python with Scikit-Learn,” Medium, 25 02 2019. [Online]. Available: https://towardsdatascience.com/a-beginners-guide-to-linear-regression-in-python-with-scikit-learn-83a8f7ae2b4f. [Acedido em 07 2020].

[27] A. Agarwal, “Linear Regression using Python,” Medium, 5 10 2018. [Online]. Available: https://towardsdatascience.com/linear-regression-using-python-b136c91bf0a2. [Acedido em 07 2020].

[28] dinfratechsource, “Host a Static Site on AWS, using S3 and CloudFront,” dinfratechsource, 28 07 2019. [Online]. Available: https://dinfratechsource.com/2019/07/28/host-a-static-site-on-aws-using-s3-and-cloudfront/. [Acedido em 09 01 2020].

[29] eCloudture, “Labeling Data with SageMaker Ground Truth,” eCloudture, 20 03 2019. [Online]. Available: https://www.ecloudture.com/en/labeling-data-with-sagemaker-ground-truth-2/. [Acedido em 09 01 2020].

[30] AWS, “Amazon S3,” AWS - Amazon Web Services, 2020. [Online]. Available: https://aws.amazon.com/s3/?sc_channel=PS&sc_campaign=acquisition_PT&sc_publisher=google&sc_medium=ACQ-P%7CPS-GO%7CBrand%7CDesktop%7CSU%7CStorage%7CS3%7CPT%7CEN%7CText&sc

_content=s3_e&sc_detail=aws%20s3&sc_category=Storage&sc_segment=293614596615&sc_matcht. [Acedido em 17 01 2020].

Anexo 01 – Random Cut Forest (RCF), Código

```
        import boto3
import re
import sagemaker
import seaborn as sns

execution_role = sagemaker.get_execution_role()

sess = sagemaker.Session()
bucket = 'ssmbucket-mb'
prefix = 'rcf-benchmarks'
```

In [4]:

```
%%time

import pandas as pd
import urllib.request

data_filename = 'http://ssmbucket-mb.s3-website-eu-west-1.amazonaws.com/rcf
-benchmarks/train/temp_predictions.csv'
data_source = 'http://ssmbucket-mb.s3-website-eu-west-1.amazonaws.com/rcf-b
enchmarks/train/machine_temperature_system_failure.csv'

temp_data = pd.read_csv(data_source, delimiter=',')
prediction_data = pd.read_csv(data_filename, delimiter=',')
CPU times: user 29.9 ms, sys: 809 µs, total: 30.7 ms
Wall time: 179 ms
```

In [3]:

```
temp_data.head()
```

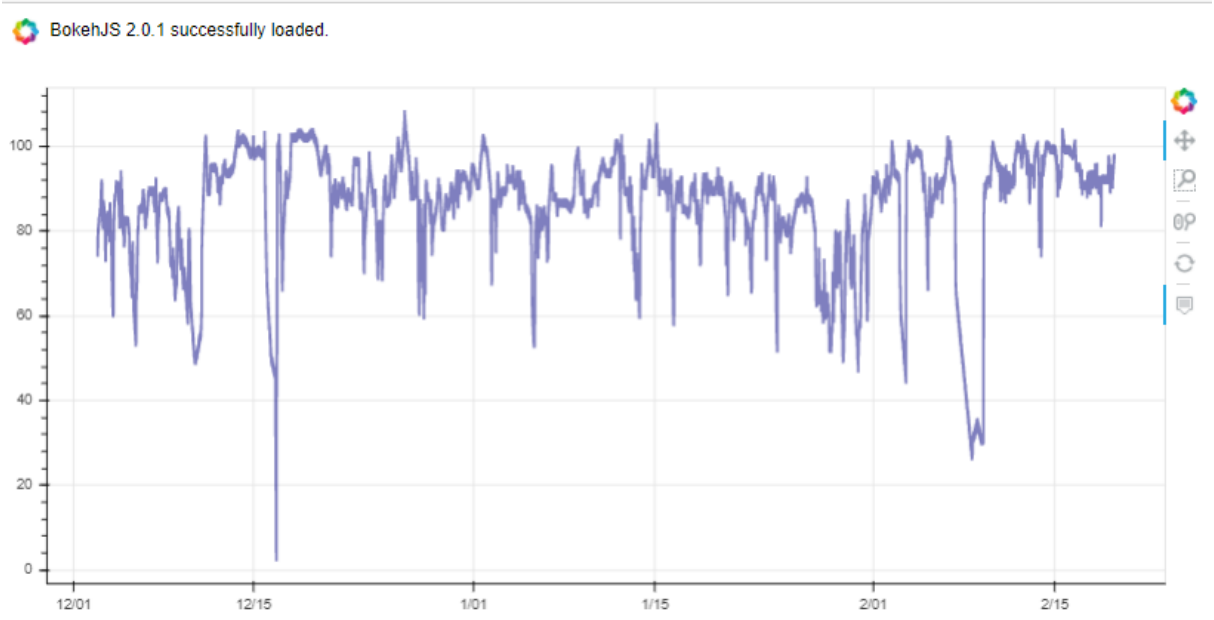
Out[3]:

	timestamp	value
0	2013-12-02 21:15:00	73.967322
1	2013-12-02 21:20:00	74.935882
2	2013-12-02 21:25:00	76.124162
3	2013-12-02 21:30:00	78.140707
4	2013-12-02 21:35:00	79.329836

In [21]:

```
import bokeh
import bokeh.io
from bokeh.models import HoverTool
bokeh.io.output_notebook()
from bokeh.plotting import figure, show, output_file, ColumnDataSource
from bokeh.models.formatters import DatetimeTickFormatter
import bokeh.palettes
temp_data['timestamp'] = pd.to_datetime(temp_data['timestamp'])
output_file("datetime.html")
date = temp_data['timestamp']
temp = temp_data['value']

x = date.values
y = temp.values
score = []

string_x = list(map(str, x))
source = ColumnDataSource(temp_data)
source.add(temp_data['timestamp'].apply(lambda d: d.strftime('%Y-%m-%d %H:%
M:%S')), 'event_date_formatted')

hover = HoverTool(
    names = ["temp"],
    tooltips=[
        ( 'date',   '@event_date_formatted'),
        ( 'temperature',  '$y' ),
    ],
    mode='vline'
)

p = figure(x_axis_type='datetime', plot_width=900, plot_height=400, tools=[
hover, 'pan','wheel_zoom','box_zoom','reset'])
p.line( name="temp", x='timestamp',y= 'value',source=source, line_width=2,c
olor='navy', alpha=0.5)
show(p)
```

Out[21]:


BokehJS 2.0.1 successfully loaded.
100
80
60
40
20
0
12/01
12/15
1/01
1/15
2/01
2/15

In [22]:

```
def convert_and_upload_training_data(ndarray, bucket, prefix, filename='dat
a.pbr'):
    import boto3
    import os
    from sagemaker.amazon.common import numpy_to_record_serializer

    serializer = numpy_to_record_serializer()
    buffer = serializer(ndarray)

    s3_object = os.path.join(prefix, 'train', filename)
    boto3.Session().resource('s3').Bucket(bucket).Object(s3_object).upload_
fileobj(buffer)

    s3_path = 's3://{}/{}'.format(bucket, s3_object)
    return s3_path

s3_train_data = convert_and_upload_training_data(
    temp_data['value'].values.reshape(-1,1),
    bucket,
    prefix)
print('Uploaded data to {}'.format(s3_train_data))
```

Out[22]:

```
Uploaded data to s3://ssmbucket-mb/rcf-benchmarks/train/data.pbr
```

In [23]:

```
%%time
session = sagemaker.Session()

container = '438346466558.dkr.ecr.eu-west-1.amazonaws.com/randomcutfore
st:latest'

rcf = sagemaker.estimator.Estimator(
    container,
    execution_role,
    input_mode='File',
    output_path='s3://{}/{}/output'.format(bucket, prefix),
    train_instance_count=1,
    train_instance_type='ml.m4.xlarge',
    sagemaker_session=session,
)

rcf.set_hyperparameters(
    num_samples_per_tree = 500,
    num_trees = 100,
    feature_dim = 1,
)
```

```
s3_train_input = sagemaker.session.s3_input(
    s3_train_data,
    distribution='ShardedByS3Key',
    content_type='application/x-recordio-protobuf',
)
```

```
rcf.fit({'train': s3_train_input})
```

Out[23]:

```
2020-07-24 15:13:24 Completed - Training job completed
Training seconds: 76
Billable seconds: 76
CPU times: user 614 ms, sys: 39.9 ms, total: 654 ms
Wall time: 4min 42s
```

In [24]:

```
print('Training job name: {}'.format(rcf.latest_training_job.job_name))
```

Out[24]:

```
Training job name: randomcutforest-2020-07-24-15-09-16-810
```

In [25]:

```
rcf_inference = rcf.deploy(
    initial_instance_count=1,
    instance_type='ml.m4.xlarge',
)
```

Out[25]:

```
---------------!
```

In [26]:

```
print('Endpoint name: {}'.format(rcf_inference.endpoint))
```

Out[26]:

```
Endpoint name: randomcutforest-2020-07-24-15-09-16-810
```

In [27]:

```
from sagemaker.predictor import csv_serializer, json_deserializer

rcf_inference.content_type = 'text/csv'
rcf_inference.serializer = csv_serializer
rcf_inference.deserializer = json_deserializer
```

In [28]:

```
import time
prediction_data = pd.read_csv(data_filename, delimiter=',')
prediction_data_numpy = prediction_data['value'].values.reshape(-1,1)
results = rcf_inference.predict(prediction_data_numpy)
```

In [29]:

```
from bokeh.models import Range1d, LinearAxis
def prediction(data, threshold=None):
    prediction_data = data
    prediction_data['timestamp'] = pd.to_datetime(prediction_data['timestam
p'])
    output_file("datetime.html")
    temp = prediction_data['value']
    scores = prediction_data['score']

    hover = HoverTool(
        names = ["temp","score"],
        tooltips=[
            ( 'date',   '@event_date_formatted'),
            ( 'temp',  '@temp_y' ),
           ( 'Score',  '@score_y' ),
        ],
    )
    source = ColumnDataSource(prediction_data)
    source.add(prediction_data['timestamp'].apply(lambda d: d.strftime('%Y-
%m-%d %H:%M:%S')), 'event_date_formatted')
    source.add(temp, 'temp_y')
    source.add(scores, 'score_y')

    p = figure(x_axis_type='datetime', plot_width=1000, plot_height=350, to
ols=[hover, 'pan','wheel_zoom','box_zoom','reset'])
    p.line( name = "temp", x='timestamp',y= 'value',source=source, line_wid
th=2,color='navy', alpha=0.5, legend_label="Temperature")
    p.extra_y_ranges = {"Anomaly": Range1d(start=0, end=10)}
    p.add_layout(LinearAxis(y_range_name="Anomaly"), 'right')
    p.line( name = "score", x='timestamp',y='score',source=source, line_wid
th=2,color='red', alpha=0.5, y_range_name="Anomaly",legend_label="Score")
    p.legend.location = "top_left"
    p.legend.click_policy="hide"
    p.title.text = "Anomaly Detection for Device Temperature"



    score_mean = prediction_data['score'].mean()
    score_std = prediction_data['score'].std()
```

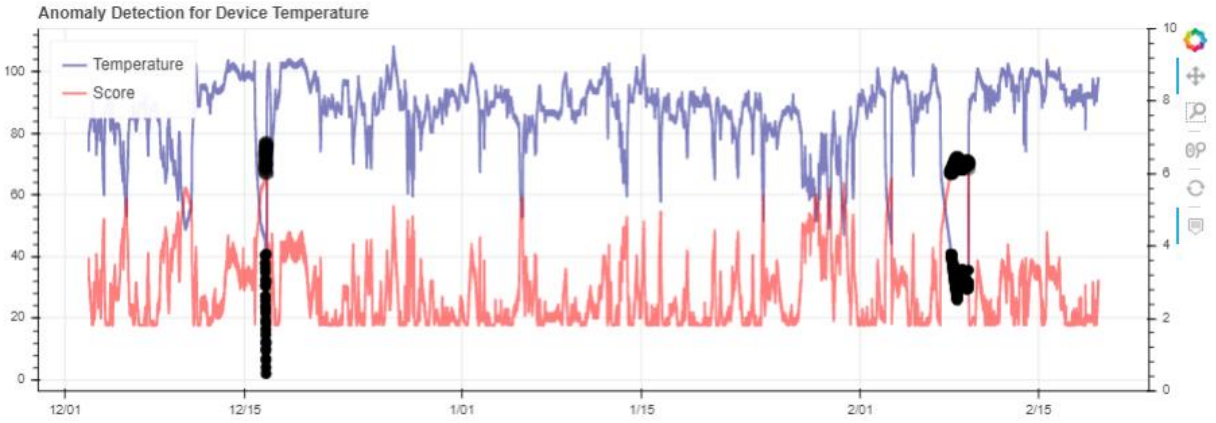
```
    if not threshold:
        score_cutoff = score_mean + 3*score_std
    else:
        score_cutoff = threshold
    print("The best threshold value is: " + str(score_cutoff))
    anomalies = prediction_data[prediction_data['score'] > score_cutoff]
    sorted_anomalies = anomalies.sort_values(by=['score'], ascending=False)
    print(sorted_anomalies)
    source = ColumnDataSource(prediction_data)
    p.circle( sorted_anomalies['timestamp'],sorted_anomalies['score'], line
_width=8,color='black', alpha=0.5, y_range_name="Anomaly")
    p.circle( sorted_anomalies['timestamp'],sorted_anomalies['value'], line
_width=5,color='black')


    show(p)

def process_data(file):
    prediction_data = pd.read_csv(file, delimiter=',')
    prediction_data_numpy = prediction_data['value'].values.reshape(-1,1)
    results = rcf_inference.predict(prediction_data_numpy)
    scores = [datum['score'] for datum in results['scores']]
    prediction_data['score'] = pd.Series(scores, index=prediction_data.inde
x)
    return prediction_data
```

In [30]:

```
prediction(process_data('s3://ssmbucket-mb/rcf-benchmarks/train/machine_tem
perature_system_failure.csv'))
```

Out[30]:

```
The best threshold value is: 6.017795865428086
              timestamp      value     score
3986  2013-12-16 17:25:00   2.084721  6.848094
3985  2013-12-16 17:20:00   4.117241  6.825141
3984  2013-12-16 17:15:00   6.440238  6.794717
3983  2013-12-16 17:10:00   6.918645  6.788762
3981  2013-12-16 17:00:00   9.633952  6.751497
...                   ...        ...       ...
19381 2014-02-08 03:20:00  40.654995  6.025140
19387 2014-02-08 03:50:00  40.696344  6.024148
19398 2014-02-08 04:45:00  40.727206  6.023083
3992  2013-12-16 17:55:00  40.782224  6.021679
19382 2014-02-08 03:25:00  40.796344  6.021225
```

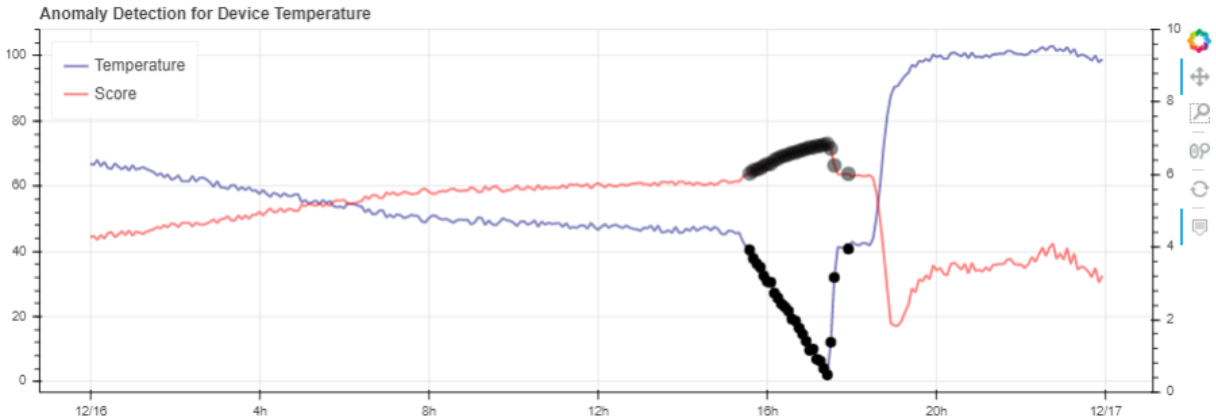


In [31]:

```
prediction(process_data('s3://ssmbucket-mb/rcf-benchmarks/train/temp_predictions.csv'), 6.01)
```

Out[31]:

```
The best threshold value is: 6.01
              timestamp      value     score
209 2013-12-16 17:25:00   2.084721  6.848094
208 2013-12-16 17:20:00   4.117241  6.825141
207 2013-12-16 17:15:00   6.440238  6.794717
206 2013-12-16 17:10:00   6.918645  6.788762
204 2013-12-16 17:00:00   9.633952  6.751497
205 2013-12-16 17:05:00  10.001966  6.746475
210 2013-12-16 17:30:00  12.120381  6.716422
203 2013-12-16 16:55:00  12.414093  6.712093
202 2013-12-16 16:50:00  14.671060  6.679224
201 2013-12-16 16:45:00  16.457109  6.649704
200 2013-12-16 16:40:00  18.591971  6.611621
199 2013-12-16 16:35:00  19.277179  6.598647
198 2013-12-16 16:30:00  21.737032  6.554823
197 2013-12-16 16:25:00  22.983839  6.519860
196 2013-12-16 16:20:00  23.829044  6.497073
195 2013-12-16 16:15:00  25.680672  6.441804
194 2013-12-16 16:10:00  27.212228  6.400193
193 2013-12-16 16:05:00  30.504461  6.298110
192 2013-12-16 16:00:00  30.699649  6.293280
211 2013-12-16 17:35:00  32.001703  6.255197
191 2013-12-16 15:55:00  32.454943  6.240659
190 2013-12-16 15:50:00  35.072456  6.172791
189 2013-12-16 15:45:00  36.249653  6.140396
188 2013-12-16 15:40:00  37.791275  6.112991
187 2013-12-16 15:35:00  40.461427  6.032286
215 2013-12-16 17:55:00  40.782224  6.021679
```



Anexo 02 – Sliding Windows Regressão Linear

In [2]:

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as seabornInstance
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
%matplotlib inline
```

In [9]:

```
dataset = pd.read_csv("s3://ssmbucket-mb/rcf-benchmarks/train/machine_tempe
rature_system_failure.csv", delimiter=',')
```

In [10]:

```
dataset.shape
```

Out[10]:

```
(22695, 2)
```

In [11]:

```
dataset.describe()
```

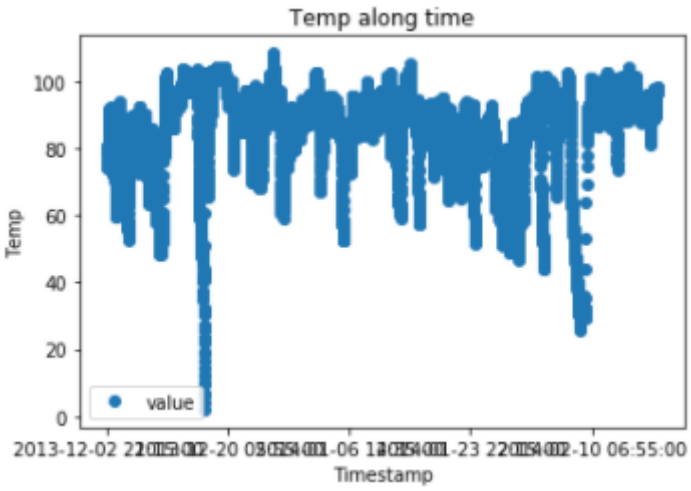
Out[11]:

	value
count	22695.000000
mean	85.926498
std	13.746912
min	2.084721
25%	83.080078
50%	89.408246
75%	94.016252
max	108.510543

In [13]:

```
dataset.plot(x='timestamp', y='value', style='o')
plt.title('Temp along time')
plt.xlabel('Timestamp')
plt.ylabel('Temp')
plt.show()
```

Out[13]:

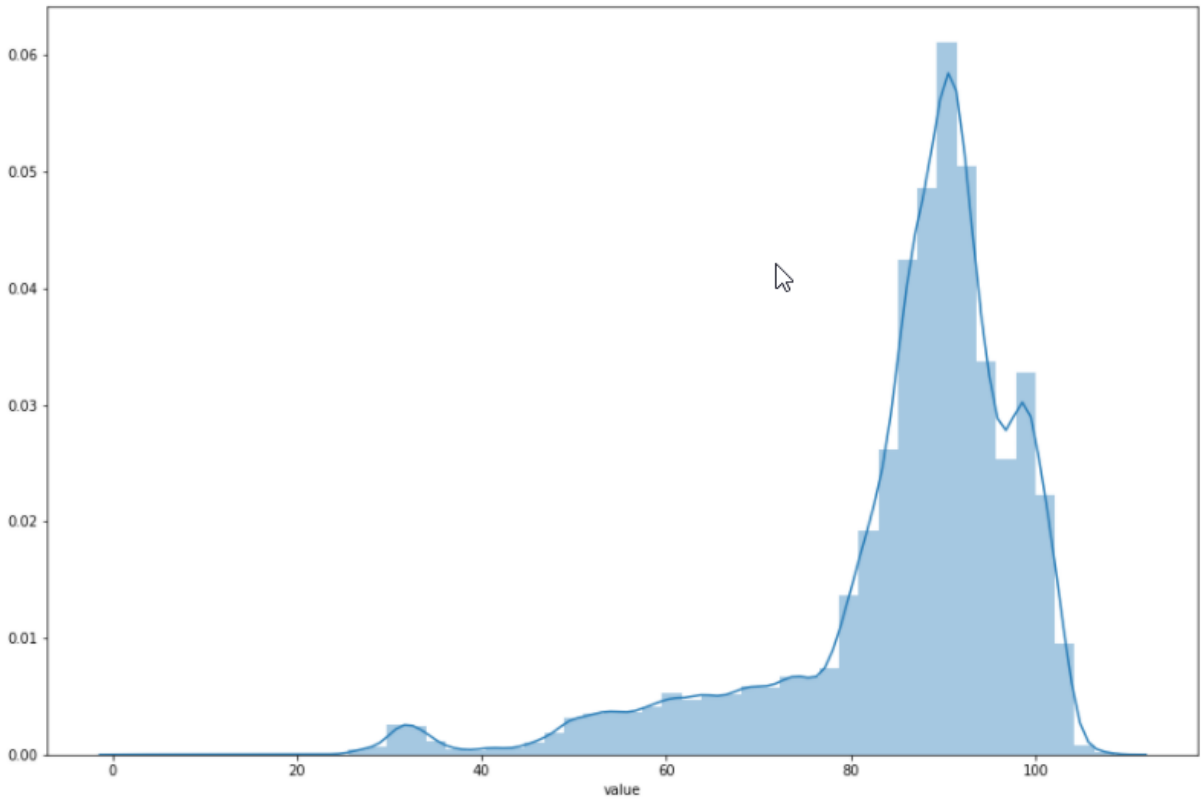


In [14]:

```
plt.figure(figsize=(15,10))
plt.tight_layout()
seabornInstance.distplot(dataset['value'])
```

Out[14]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8844702c18>
```



In [45]:

```
import datetime as dt
```

```
myvar = pd.to_datetime(dataset['timestamp']) #converter a data para um valo
r numerico
myvar=myvar.map(dt.datetime.toordinal)
X = myvar.values.reshape(-1,1)
y = dataset['value'].values.reshape(-1,1)
```

Out[45]:

```
0        735204
1        735204
2        735204
3        735204
4        735204
          ...
22690    735283
22691    735283
22692    735283
22693    735283
22694    735283
Name: timestamp, Length: 22695, dtype: int64
```

In [46]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra
ndom_state=0)
```

In [47]:

```
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

Out[47]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=Fa
lse)
```

In [48]:

```
print(regressor.intercept_)
```

Out[48]:

```
[11802.02938327]
```

In [49]:

```
print(regressor.coef_)
```

Out[49]:

```
[[-0.01593498]]
```

In [50]:

```
y_pred = regressor.predict(X_test)
```

In [51]:

```
df = pd.DataFrame({'Actual': y_test.flatten(), 'Predicted': y_pred.flatten(
)})
df
```

Out[51]:

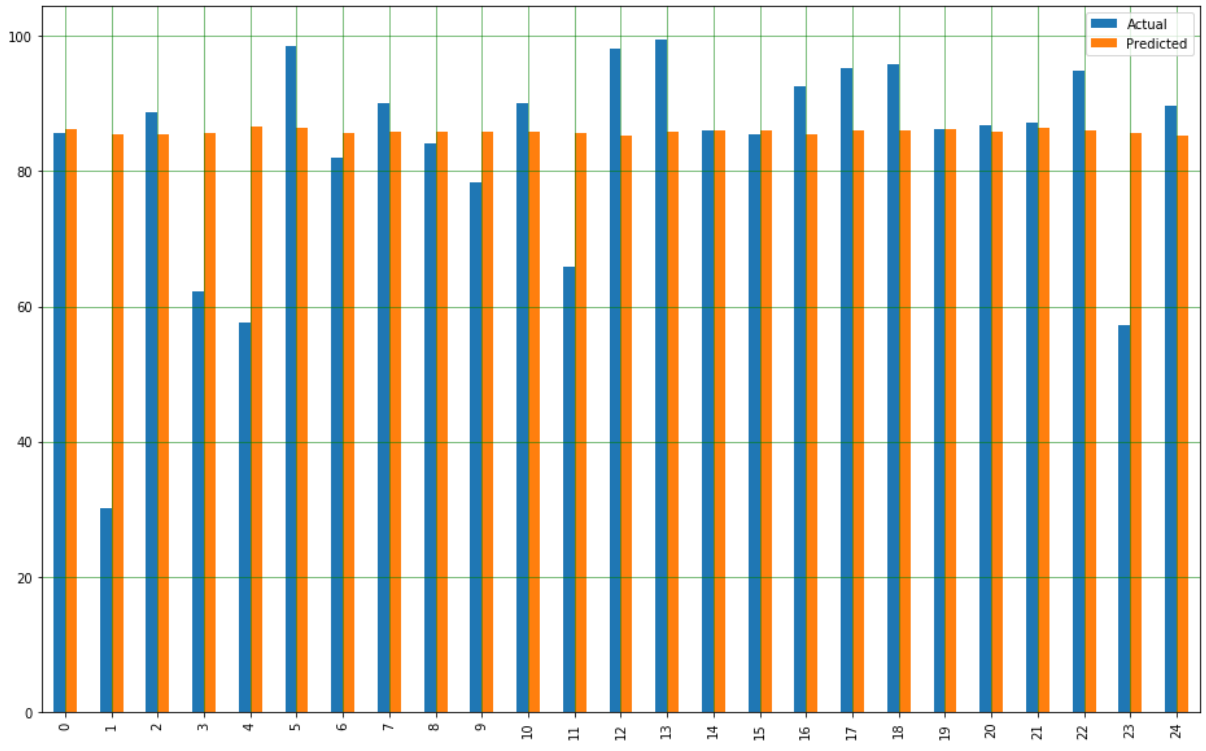
	Actual	Predicted
0	85.664319	86.201261
1	30.080704	85.468252
2	88.653841	85.468252
3	62.231677	85.659471
4	57.554582	86.519960
...	...	...
4534	88.012166	85.978171
4535	80.353425	86.567765
4536	93.144866	86.089716
4537	82.031584	86.153456
4538	83.447442	86.551830

4539 rows × 2 columns

In [52]:

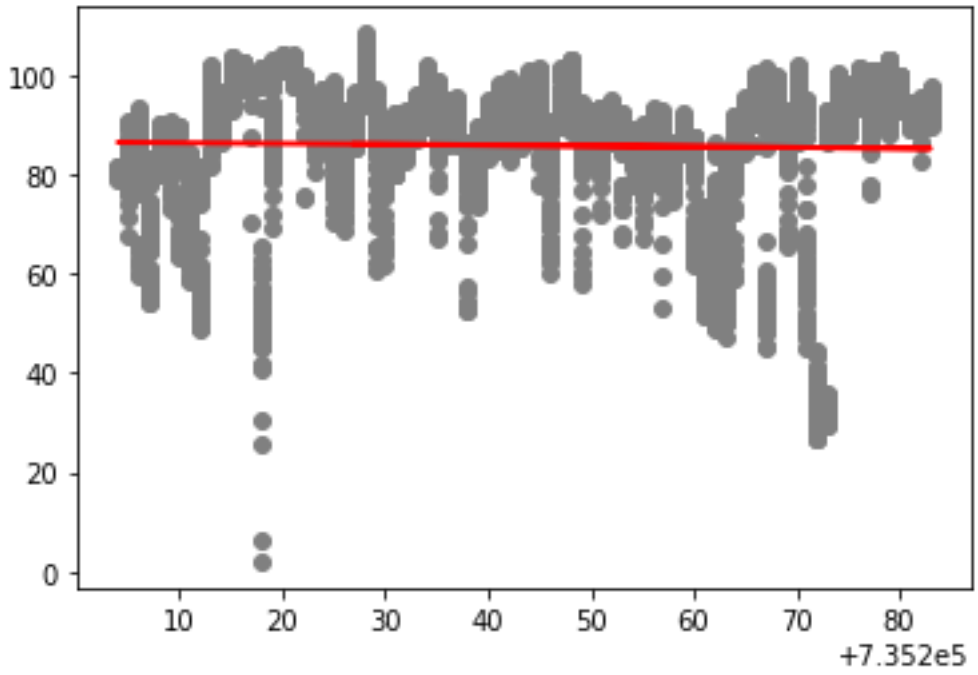
```
df1 = df.head(25)
df1.plot(kind='bar',figsize=(16,10))
plt.grid(which='major', linestyle='-', linewidth='0.5', color='green')
plt.grid(which='minor', linestyle=':', linewidth='0.5', color='black')
plt.show()
```

Out[52]:



In [53]:

```
plt.scatter(X_test, y_test,  color='gray')
plt.plot(X_test, y_pred, color='red', linewidth=2)
plt.show()
```



In [54]:

```
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test
, y_pred)))
```

Out[54]:

```
Mean Absolute Error: 9.694472177780346
Mean Squared Error: 193.29992456505667
Root Mean Squared Error: 13.903234320296002
```