Conceitos de Data & Analytics II

1.Técnicas para Processamento de Dados

Quando se fala de Big Data e Data Analytics precisa-se entender diversos conceitos. Neste tópico fala-se das técnicas de processamento de dados **Batch** e **Stream**. O processamento em lote, ou do inglês Batch processing, é o processamento que acontece em blocos de dados que já foram armazenados durante um período de tempo, ou seja, é o processamento de diversas transações que ocorreram num sistema de origem e serão processadas para outros sistemas em conjunto ou em bloco[3]. Por exemplo: processar todas as transações que ocorreram em uma semana na empresa. O Hadoop MapReduce é um dos melhores frameworks para processar dados em lotes. Um processamento em lote pode ser apenas a transferência de dados em blocos até um processamento com transformações nos dados. Um exemplo de processamento em lote com transformações é o ETL (Extração, Transformação, Carregamento, do inglês Extract Transform Load) que é o processamento de blocos de dados em etapas de Extração, Transformação e Carga. ETLs são comumente utilizado para construir um data warehouse (DW) e Business Intelligence (BI). De acordo com a SAS, nesse processo os dados são retirados (extraídos) de um sistema-fonte, convertidos (transformados) em um formato que possa ser analisado, e armazenados (carregados) em um armazém ou outro sistema. Vale ressaltar que algumas ferramentas não fazem o ETL tradicional e sim uma variação, o ELT.

O processamento stream trabalha com fluxos de dados que são capturados em tempo real e processados com latência mínima para o sistema de destino. Neste processamento uma ou um pequeno conjunto de transações é processada e enviada ao sistema de destino. De acordo com [3] o processamento stream é uma chave de ouro na busca por resultados analíticos em tempo real ou muito próximas do tempo real. O processamento stream permite processar dados em tempo real conforme eles chegam e detectar rapidamente as condições dentro de um pequeno período de tempo a partir do momento em que os dados são recebidos.

A tabela abaixo apresenta quais perguntas podem ser respondidas por cada técnica de processamento em Big Data:

Análise	**Processamento Batch**	**Processamento Stream**
Análise de Logs	O que deu errado uma hora atrás?	Ocorreu um problema, tome uma ação corretiva / notifique o administrador
Análise de Faturamento	Quanto foi gasto no último ciclo de faturamento?	Notifique o usuário de que o limite de faturamento está se aproximando
Preferências do Cliente	Quais anúncios foram mais eficazes ontem?	Qual anúncio o cliente deve ver agora?
Fraude	Trilha de auditoria / evidência de fraude	Pare a fraude agora

2. Business Intelligence (BI)

Inteligência de negócios, do inglês Business Intelligence (BI), de acordo com o glossário do Gartner é um termo guarda-chuvas ou abrangente que inclui os aplicações, infraestrutura, ferramentas e boas práticas que permitem o acesso e a análise de informações para melhorar e otimizar as decisões. Ou seja, resumidamente, o BI é um conjunto de teorias, metodologias, práticas, processos, tecnologias e estruturas para desenvolver uma inteligência ao negócio.

Tecnologias BI fornecem a visão do histórico das operações de negócios, assim como a visão atual e as possíveis previsões. As principais funções de um BI são: relatórios, análises, mineração de dados, processamento de eventos complexos, gerenciamento de desempenho dos negócios, benchmarking, mineração de dados, análises previsíveis e análises prescritivas.

Para saber mais sobre BI e a sua evolução clique aqui.

3. Data Warehouse (DW)

Armazém de Dados, ou do inglês Data Warehouse, tem por objetivo a centralização dos dados retirados de diversas origens para facilitar ou simplificar o consumo futuro. É um repositório centralizado otimizado para análises [4]. De um DW pode-se extrair:

- Relatórios
- Relatórios ad-hoc
- Análises

Analistas de negócios, cientistas de dados e tomadores de decisões acessam os dados por meio de ferramentas de inteligência de negócios (BI), clientes SQL e outros aplicativos de análise[4]. Um DW possui diversos Data marts e um Data mart é um pequeno DW ou uma pequena parte de um DW. De acordo com a Amazon [4], um data mart é um data warehouse que atende às necessidades de uma equipe ou unidade de negócios específica, como finanças, marketing ou vendas. O data mart é menor, mais focado e pode conter resumos de dados para atender melhor à comunidade de usuários. No mesmo artigo a Amazon apresenta a tabela comparativa:

Características	Data warehouse	Data mart
Escopo	Várias áreas centralizadas e integradas	Uma área específica e descentralizada
Usuários	De toda a organização	Uma única comunidade ou departamento
Fonte de dados	Muitas fontes	Uma ou poucas fontes, ou uma parte dos dados já coletados em um data warehouse
Tamanho	Grandes, pode variar de centenas de gigabytes a petabytes	Pequenos, normalmente até algumas dezenas de gigabytes
Projeto	De cima para baixo	De baixo para cima
Detalhes dos dados	Dados completos e detalhados	Pode manter dados resumidos

Existem várias diferenças entre um banco transacional e um DW. Enquanto um banco de dados transacional (OLTP) utiliza normalização de formato de dados, um DW utiliza dados em formado de-normalizados para aumentar o desempenho das consultas e torna-se mais intuitivo para os utilizadores. A esta maneira de distribuir os dados se denomina modelagem dimensional. A modelagem dmensional é um tipo de modelagem entidade-relacionamento (ER). Uma modelagem dimensional é composta de 2 tipos de tabelas chamadas Dimensões e Fatos. As dimensões se referem ao contexto enquanto que os fatos se referem às medidas/métricas. De acordo com a IBM, os dados de um DW são agrupados ao redor de fatos, que são campos numéricos que podem ser agregados e analisados e dimensões que são filtros de negócios e campos de agrupamento. As dimensões podem ser consideradas Filtros ou expressões Group By. Abaixo um exemplo de DW:

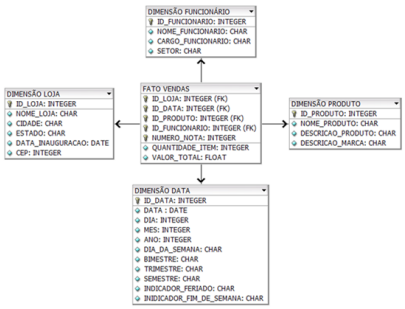


Figura 1 - Exemplo de modelo multidimensional

Slowly Changing Dimension (SCD)

Em um DW, pode-se ter a necessidade de armazenamento de dados históricos. Para atender a este objetivo, pode-se utilizar a técnica de Slowly Changing Dimension (SCD) ou Dimensões que Mudam Lentamente, em português. De acordo com a Oracle [5], um SCD é uma dimensão que armazena e gerencia dados atuais e históricos ao longo do tempo em um data warehouse. Existem diversos tipos de SCD, aqui apresentamos os 4 tipos mais relevantes:

- SCD Tipo 1 (Sobreposição): é o comportamento pradão de um objeto, sem versionamento. Em um SCD Tipo 1, os novos dados substituem os dados existentes. Assim, os dados existentes são perdidos, pois não são armazenados em nenhum outro lugar. Ou seja, não é feito o versionamento do registro modificado. Este é o tipo de dimensão padrão.
- SCD Tipo 2 (Criação de novo registro): é a técnica mais utilizada para atualizações de dimensões. Mantém o histórico completo dos valores. Quando o valor de um atributo é alterado, o registro atual é fechado. Um novo registro é criado com os valores de dados alterados e esse novo registro se torna o registro vigente/atual. Cada registro contém o tempo efetivo e o tempo de expiração para identificar o período de tempo entre o qual o registro estava ativo.

- SCD Tipo 3 (Criação de novo campo): armazena duas versões de valores para determinados atributos. Cada registro armazena o valor anterior e o valor atual do atributo. Quando o valor de qualquer um dos atributos selecionados é alterado, o valor atual é armazenado como o valor antigo e o novo valor se torna o valor atual. Ou seja, cria uma coluna nova para cada coluna modificada.
- SCD Tipo 6 ou SCD Híbrido: pode combinar todos os tipos de SCD anteriores.

4. Mineração de Dados

Mineração de Dados, ou no inglês Data mining, de acordo com o glossário do gartner, é o processo de descobrir correlações, padrões e tendências significativos analisando grandes quantidades de dados armazenados em repositórios. A mineração de dados emprega tecnologias de reconhecimento de padrões, bem como técnicas estatísticas e matemáticas. De acordo com a SAS[10], o processo de minerar dados para descobrir conexões escondidas e prever tendências futuras tem uma longa história. Por vezes chamado de "descoberta de conhecimento em bancos de dados", o termo "mineração" só foi cunhado nos anos 1990, mas sua base compreende três disciplinas científicas entrelaçadas que existem há tempos: estatística (o estudo numérico das relações entre dados), inteligência artificial (inteligência exibida por softwares e/ou máquinas, que se assemelha à humana) e machine learning (algoritmos que podem aprender com dados para realizar previsões). A tecnologia de mineração de dados continua evoluindo para acompanhar o potencial ilimitado do big data e a computação de baixo custo.

Ciência de Dados vs Mineração de Dados

A grande diferença está na profundidade que cada uma tem. Mineração de dados é uma técnica enquanto que Ciência de dados é uma área multidisciplinar[11].

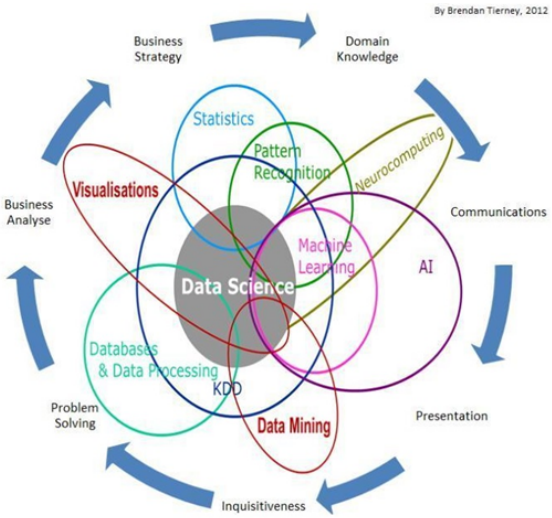


Figura 2 - Ciência de Dados é multidisciplinar

A tabela abaixo[12], faz uma comparação mais detalhada.

Área	Mineração de Dados	Ciência de Dados
O que é	Uma técnica	Uma área
Foco	Processo de Negócios	Estudo Científico
Objetivo	Tornar os dados mais usáveis	Construir produtos centrados em dados para uma organização
Abrangência	Mineração de Dados pode ser um subconjunto de Ciência de Dados, uma vez que mineração faz parte do pipeline de Ciência de Dados	Multidisciplinar - Consiste de visualizações de dados, Ciência Socio-computacional, Mineração de Dados, Processamento de linguagem natural, etc
Tipo do Dado	Maioria estruturada	Todos os formatos de dados

5. Machine Learning

Machine Learning ou aprendizado de máquina é uma espécie de inteligência artificial que é responsável por fornecer aos computadores a capacidade de aprender sobre novos conjuntos de dados sem serem programados por meio de uma fonte explícita[11]. De acordo com o glossário do Gartner, são algoritmos avançados de aprendizado de máquina compostos de muitas tecnologias (como deep learning, redes neurais e processamento de linguagem natural), usadas em aprendizado supervisionado e não supervisionado, que operam guiados por lições de informações existentes.

A SAS[13] explica os métodos de Machine Learning:

- Algoritmos de aprendizado supervisionado são treinados por meio de exemplos rotulados, como uma entrada na qual a saída desejada é conhecida. Por exemplo, um equipamento poderia ter pontos de dados rotulados como “F” (falha) ou “E” (executa). O algoritmo de aprendizado recebe um conjunto de entradas junto com as saídas corretas correspondentes, e aprende ao comparar a saída real com as saídas corretas para encontrar erros. Ele, então, modifica o modelo de acordo. Através de métodos como classificação, regressão e gradient boosting, o aprendizado supervisionado utiliza padrões para prever os valores de rótulos em dados não-rotulados adicionais. O aprendizado supervisionado é comumente empregado em aplicações nas quais dados históricos preveem eventos futuros prováveis. Por exemplo, ele pode antecipar quando transações via cartão de crédito são passíveis de fraude ou qual segurado tende a reivindicar sua apólice.
- O aprendizado não-supervisionado é utilizado contra dados que não possuem rótulos históricos. A "resposta certa" não é informada ao sistema. O algoritmo deve descobrir o que está sendo mostrado. O objetivo é explorar os dados e encontrar alguma estrutura dentro deles. O aprendizado não-supervisionado funciona bem com dados transacionais. Por exemplo, ele pode identificar segmentos de clientes com atributos similares que podem, então, ser tratados de modo igualmente similar em campanhas de marketing; ou ele pode encontrar os principais atributos que separam segmentos distintos de clientes. Técnicas populares incluem mapas auto-organizáveis, mapeamento por proximidade, agrupamento k-means e decomposição em valores singulares. Esses algoritmos também são utilizados para segmentar tópicos de texto, recomendar itens e identificar pontos discrepantes nos dados.
- O aprendizado semi-supervisionado é utilizado para as mesmas aplicações que o aprendizado supervisionado. Mas este aqui manipula tanto dados rotulados quanto não-rotulados para treinamento – normalmente uma pequena quantidade de dados rotulados com uma grande quantidade de dados não-rotulados (porque dados sem rótulos são mais baratos e demandam menos esforços para serem adquiridos). Esse tipo de aprendizado pode ser empregado com métodos como classificação, regressão e previsão. O aprendizado semi-supervisionado é útil quando o custo associado à rotulação é muito alto para possibilitar um processo de treinamento totalmente rotulado. Exemplos básicos incluem a identificação do rosto de uma pessoa em uma webcam.
- O aprendizado por reforço é normalmente utilizado em robótica, jogos e navegação. Com ele, o algoritmo descobre através de testes do tipo 'tentativa e erro' quais ações rendem as maiores recompensas. Este tipo de aprendizado possui três componentes principais: o agente (o aprendiz ou tomador de decisão), o ambiente (tudo com que o agente interage) e ações (o que o agente pode fazer). O objetivo é que o agente escolha ações que maximizem a recompensa esperada em um período de tempo determinado. O agente atingirá o objetivo muito mais rápido se seguir uma boa política. Então o foco do aprendizado por reforço é descobrir a melhor política.

6. Deep Learning

Deep learning é um tipo de machine learning com o objetivo de treinar computadores para realizar tarefas como seres humanos, o que inclui reconhecimento de fala, identificação de imagem e previsões. Em vez de organizar os dados para serem executados através de equações predefinidas, o deep learning configura parâmetros básicos sobre os dados e treina o computador para aprender sozinho através do reconhecimento padrões em várias camadas de processamento[14]. Por exemplo, o deep learning é usado para classificar imagens, reconhecer fala, detectar objetos e descrever conteúdo. Sistemas como Siri e Cortana são parcialmente alimentados por deep learning.

7. Relatórios

Conceito bastante conhecido, porém muitas pessoas confundem e criam gráficos acreditando que estão fazendo relatórios. Relatório é um documento que apresenta informações em um formato organizado para um público específico e propósito. Embora os resumos dos relatórios possam ser entregues oralmente, os relatórios completos são quase sempre na forma de documentos escritos.

8. Dashboards

Dashboard basicamente é um indicador de negócios, pode ser um número ou um gráfico. Um conjunto de dashboards chama-se painel de dados, e é uma importante ferramenta para gerenciamento de informações que visualmente, analisa e exibe os principais indicadores de desempenho, ou do inglês Key Performance Indicators (KPI) e/ou métricas de dados principais para monitorar a integridade de uma empresa, departamento ou processo específico. Eles são personalizáveis para atender às necessidades específicas de um departamento e empresa. Por trás dp dashboard, um painel se conecta a seus arquivos, anexos, serviços e APIs, exibindo todos esses dados na forma de tabelas, gráficos de linhas, gráficos de barras, indicadores entra outras formas de visualização.

Figura 3 - Exemplo de um painel de dashboards

9. Internet das coisas (IoT)

Internet das coisas, do inglês Internet of Things (IoT), de acordo com o Gartner é a rede de objetos físicos que contém tecnologia embarcada para comunicar e sentir ou interagir com seus estados internos ou com o ambiente externo. Em outras palavras, são dispositivos físicos (veículos, videogames, edifícios, câmeras, sensores e outros dotados de tecnologia embarcada conexão com a rede) capaz de coletar e transmitir dados.

10. API

Uma API, do inglês Application Program Interface, é um conjunto de rotinas, protocolos e ferramentas para criar aplicativos de software. Basicamente, uma API especifica como os componentes de software devem interagir[15]. Além disso, as APIs são usadas ao programar componentes da interface gráfica do usuário (GUI). Uma API define a maneira correta para um desenvolvedor escrever um programa.

As APIs são compostas de dois elementos. O primeiro é uma especificação que descreve como as informações são trocadas entre os programas, realizadas através de um request/solicitação de processamento e retornando os dados solicitados[16]. O segundo é uma interface de software escrita para que essa especificação esteja publicada de alguma forma para que o uso ocorra da forma correta. APIs são importantes para que um sistema interaja com outros sistemas sempre do mesmo formato.

De acordo com Margaret Rouse[16] existem 3 tipos básicos de APIs:

- local: As APIs locais são a forma de comunicação básica. Geralmente são serviços de sistema operacional (SO) ou de middleware para programas. Exemplo: JDBC

- web-like: As APIs da Web são projetadas para representar recursos amplamente usados, como páginas HTML, e são acessadas usando um protocolo HTTP simples. Qualquer URL da web ativa uma API da web. As APIs da Web são geralmente chamadas de REST (representational state transfer) ou RESTful ou SOAP.
- program-like: As APIs de programa são baseadas na tecnologia de chamada de Remote Procedure Call (RPC) que faz com que um componente de programa remoto pareça ser local para o restante do software. Um exemplo são as APIs, de arquitetura orientada a serviços (SOA).

11. Métodos de acesso à Banco de Dados

No passado, havia uma dezena de produtos para bases de dados disponíveis, e cada um se comunicando com as aplicações em sua própria linguagem. Se a aplicação necessitasse falar a uma nova base de dados, seria necessário "ensiná-la" essa nova linguagem. Por causa da confusão causada pela proliferação de APIs nativas de acesso às bases de dados proprietárias, surgiu a ideia de uma API de acesso universal às bases de dados.

A questão de se ter que aprender várias linguagens de APIs de diversos fornecedores foi resolvido através da criação de uma linguagem padrão única de acesso. Assim, historicamente, a comunidade da área de bancos de dados formou um consórcio, chamado SAG (**S**tandard Query Language **A**ccess **G**roup) constituído por vários fornecedores, para prover um padrão unificado de linguagem para acesso à base de dados remotas. Assim, nasceu SQL CLI (Standard Query Language Call Level Interface) definido pelo SAG.

Uma CLI não é uma nova linguagem de consulta, é, simplesmente, uma interface procedural para SQL. As aplicações usam CLI para submeter *statements* SQL para um sistema gerenciador de base de dados (DBMS). Você usa SQL para realizar consultas e atualizações numa base de dados. Assim, você pode pensar em CLI, exatamente como um invólucro para SQL (SQL wrapper). Uma CLI nem adiciona, nem subtrai poder de SQL. CLI é, simplesmente, um mecanismo para submissão de *statements* SQL.

O CLI requer o uso de *drives* de base de dados que aceitam uma chamada CLI e traduzem essa chamada para a linguagem nativa de acesso aos servidores de base de dados. Com o *driver* apropriado, cada aplicação-fonte de dados pode funcionar como um servidor-CLI, e pode, então, ser acessada por ferramentas *front-end* e programas-clientes que usam a CLI. Uma CLI requer um *driver* para cada base de dados para qual ela se conecta. Cada *driver* deve ser escrito para uma tecnologia específica, usando os métodos de acesso existentes para bases de dados. A CLI provê um *driver manager* que fala com um driver através de uma SPI (Service Provider Interface).

ODBC da Microsoft

A API padrão Windows *Open Database Connectivity* (ODBC) da Microsoft é uma versão estendida da CLI SAG. Em agosto de 1992, a Microsoft liberou o ODBC 1.0 SDK. ODBC foi desenvolvida para criar um único padrão de acesso à base de dados no ambiente WINDOWS. A ideia de ODBC foi criar uma maneira comum de acesso usando SQL (*ODBC Driver Manager*), de forma que uma aplicação-cliente pudesse acessar bases de dados de diferentes fornecedores, como por exemplo, Oracle Database, SQL Server (Microsoft), DB2 (IBM), entre outras, através de seus *drivers* correspondentes. Veja a figura 1 que mostra as camadas de ODBC (Layers of ODBC). ODBC foi a resposta shrink-wrapped (um pacote restrito) para acesso à base de dados sob a plataforma WINDOWS. ODBC é uma API procedural. ODBC lida com chamadas `a API. Embora a indústria tenha aceito ODBC como o meio primário para acessar bases de dados em WINDOWS, este produto não traduz bem no mundo de Java, que é um mundo orientado a objeto. ODBC é uma interface em C; ela não pode usada "com é" em Java.

ODBC existe para prover uma camada de abstração para linguagens como C e C++, bem como, ferramentas de desenvolvimento populares como DELPHI, POWER BUILDER e VISUAL BASIC. Mas, ODBC não provê a independência de plataforma do sistema operacional como Java.

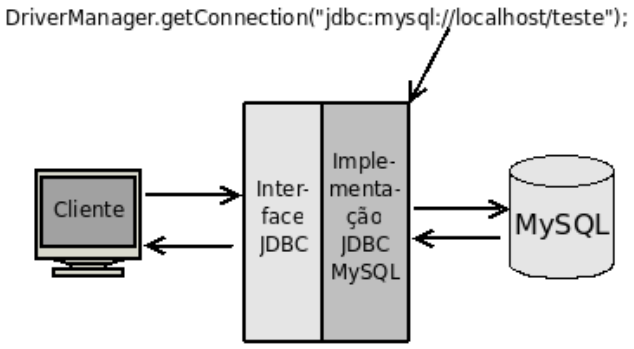
ODBC tem muitas desvantagens. A mais séria é que é uma especificação controlada pela Microsoft, e está constantemente sendo estendido. Seu futuro é também incerto, dado o compromisso corrente da Microsoft com o OLE/DB, que introduz um paradigma diferente de programação – é baseado em objeto e não procedural. Os *drivers* atuais têm diferentes níveis de compatibilidade, que não são bem documentados. As camadas ODBC introduzem uma porção de *overhead* (especialmente para atualizações e inserções SQL), e não são nunca tão rápidos quanto as APIs nativas. Para funções simples de leitura-somente, os drivers ODBC estão agora em torno de 10% da performance de um driver nativo.

JDBC (Java Database Connectivity)

Trabalhando com as empresas líderes no campo de bases de dados (Sybase, Informix, IBM e outras), JavaSoft desenvolveu um API única para acesso à base de dados. O objetivo foi prover uma interface orientada a objeto, para acessar uma base de dados relacional. Como parte desse processo, eles guardaram três principais metas em mente:

- JDBC deveria ser uma API a nível de SQL. Bases de Dados SQL é o modelo de aplicação que mais aparece como cliente/servidor. Uma API a nível de SQL significa que JDBC permite-nos construir *statements* SQL e embuti-las dentro de chamadas de uma API Java.
- JDBC deveria capitalizar a experiência de APIs de bases de dados existentes. De fato, JavaSoft projetou sobre os aspectos de sucesso da API de ODBC.
- JDBC deveria ser simples. JDBC tenta ser tão simples quanto possível, enquanto provendo desenvolvedores com a máxima flexibilidade.

JDBC é um conjunto de classes e interfaces em Java, que proporcionam uma interface similar a ODBC para bases de dados SQL. Você pode usar JDBC, de dentro de seus programas Java, para acessar quase todos as bases de dados SQL, incluindo ORACLE 7, SYBASE, DB2, SQL SERVER, ACCESS, FOXBASE, PARADOX, PROGRESS, SQLBase e XDR. Drivers JDBC estão disponíveis em Symantec, Intersolv, IBM, JavaSoft, e Inprise (Borland e Visigenic).



Referências

[1] Laudon, Kenneth; Jane (2011). Sistemas de Informação gerenciais

[2] NoSQL, Base VS ACID e Teorema CAP , acessado em 01/04/2019

[3] Big Data Battle : Batch Processing vs Stream Processing , acessado em 01/04/2019

[4] https://aws.amazon.com/pt/data-warehouse/

[5] Slowly Changing Dimensions

[6] LaPlante, Alice; Ben Sharma(2014). Architecting Data Lakes.

[7] O que é Big Data? | Oracle Brasil

[8] Big Data: What it is and why it matters

[9] O que é Data Science? | Oficina da Net

[10] O que é mineração de dados?

[11] Difference of Data Science, Machine Learning and Data Mining - DataScienceCentral.com

[12] What are the differences between Data Science and Data Mining, are they same?

[13] Machine learning: o que é e qual sua importância?

[14] Deep learning: o que é e qual sua importância?

[15] What is an API (Application Program Interface)? | Webopedia

[16] What Is an API (Application Program Interface)?

[17] Enterprise Service Bus

[18] Zato | O que ESB e SOA são, afinal?

[19] What Is Messaging? (The Java EE 6 Tutorial)

[20] Integração por Mensageria (Messaging)

[21] https://thinksis.com/wp-content/uploads/2018/10/Nexla_Whitepaper_Introduction-to-Big-Data-Formats-Saket-Saurabh.pdf