Aplicação de ferramentas de Big Data para o estudo da preferência das pessoas em ficar em casa em tempos de pandemia

Ana Filipa Pereira[1*]

[1*]Departamento de Informática, Universidade do Minho, Braga, Portugal.

Corresponding author(s). E-mail(s): pg46978@alunos.uminho.pt;

Abstract

A pandemia de *Covid-19* é dos assuntos que mais marcou os últimos anos, tendo impactado significativamente vários aspetos do quotidiano. Assim sendo, neste artigo pretende-se analisar impacto que diversos factores tiveram na preferências das pessoas em ficar em casa, comparativamente ao período de pré-pandemia. Para isto, serão explorados vários *datasets*, além de serem abordadas várias ferramentas de *Big Data* e sua aplicação no caso de uso em questão. Com isto, o objetivo reside no desenvolvimento de uma arquitetura que consiga abordar de forma eficaz o *use case* levantado.

Keywords: Big Data, Caso de uso, Covid-19, Dados, Ferramentas, Processamento, Arquitetura

1 Introdução

É inegável que a pandemia de *Covid-19* mudou por completo a forma como temos vivido o dia-a-dia. Atividades, que outrora eram feitas com regularidade, ficaram condicionadas pelas várias restrições que foram sendo impostas pelo governo ou pelo simples receio das pessoas em sair de casa. Ora, a preferência de alguém em ficar em casa, comparativamente aos tempos antes da pandemia é, sem dúvida, impactada por diversos aspetos. Por um lado, a obrigatoriedade do uso de máscara na via pública e em diversos estabelecimentos bem como a proibição dos ajuntamentos provocam receio nas pessoas, principalmente as de

maior risco, de ficarem expostas. Por outro lado, o surgimento e rápido avanço do processo de vacinação em alguns países, poderá implicar um aumento da confiança dos indíviduos em retomar, da melhor forma possível, as suas atividades diárias da mesma forma que fariam no período pré-pandemia. Assim sendo, ao longo deste documento irá ser desenvolvida uma possível arquitetura que, a partir de *datasets* encontrados, consiga mostrar de que forma a preferência das pessoas em ficar em casa é afetada com os vários aspetos referidos.

Por fim, é importante também salientar que há duas abordagens que podem ser consideradas, sendo elas efetuar previsões recorrendo a modelos de *Machine Learning*, ou então, fazer visualização de dados, sendo esta última a abordagem tida em conta neste artigo.

2 Metodologias

As ferramentas que serão apresentadas e exploradas ao longo do presente documento foram encontradas em alguns *websites* [1, 2] e artigos científicos [3] nos quais era referido quais eram as mais populares atualmente e quais as mais usadas por empresas conhecidas a nível mundial. Por outro lado, a procura das ferramentas deu-se de acordo com a sua função, ou seja, procuraram-se ferramentas para extração de conhecimento, para visualização de dados, para junção dos *datasets* ou para processamento e análise.

Posto isto, é importante também realçar a forma como foi encontrado o *use case* explorado neste documento , tendo a abordagem sido de *bottom up*. Isto deve-se ao facto de que a partir de um *dataset* inicial fornecido foram encontrados outros que o complementassem e, com base nas informações que estes possuiam, escolheu-se o caso de uso que já se referiu anteriormente.

3 Estado da Arte

Ao longo deste tópico irão ser exploradas diversas ferramentas de *Big Data*, bem como as suas vantagens e desvantagens e assim concluir quais as mais utéis para o caso de uso em questão. É de realçar que as funções destas ferramentas dividem-se em quatro categorias : junção de *datasets*, armazenamento, extração de conhecimento e visualização de dados.

3.1 Pandas

O *Pandas* é uma biblioteca *open source* do Python, destinada à análise e manipulação de dados. Esta ferramenta é conhecida pelo seu desempenho e eficiência sendo que facilita a realização de tarefas tais como: limpeza e normalização de dados, junção de *datasets*, visualização ou análise estatística. Para além disto, o *Pandas* introduziu também novos objetos de armazenamento de dados sendo eles o *DataFrame* e o *Pandas Series*. [4] Desta forma,a partir de dados brutos de várias fontes(SQL server, ficheiro Excel, csv e outros) é criado um destes objetos de armazenamento. Por outro lado, com o *Pandas* grande

parte das tarefas fazem-se em poucas linhas de código. Apesar de isto, quando se usa esta ferramenta, os objetos *Panda Series* ou *Dataframe* ficam guardados em memória, o que condiciona a quantidade de informação que pode ser processada de cada vez.

3.2 HDFS

O HDFS, sigla de *Hadoop Distributed File System*, é o sistema de ficheiros utilizado pela *framework opensource Hadoop*. Ora é de realçar que o*Apache Hadoop* assenta numa arquitetura descentralizada sendo que esta, por sua vez, pressupõe a existência de *clusters* cujos nodos são máquinas (computadores,servidores e outros). Isto dá ao HDFS um caráter distribuído, tornando-o uma ferramenta bastante confiável na medida em que, caso um nó do *cluster* fique inativo, o serviço não fica comprometido, pois os outros nós assumem as funcionalidades daquele que deixou de funcionar. [5] Além disto, esta ferramenta realiza a replicação de dados, ou seja, cada vez que um ficheiro é guardado em HDFS, este é dividido em blocos, sendo cada um dos blocos replicado.Desta forma, tanto o bloco original como as suas cópias são armazenados em diferentes localizações, o que pressupõe que em caso de falha de um dos nós do *cluster*, os dados que este armazena, estão disponíveis em outra máquina. Outra característica interessante do HDFS é a sua flexibilidade na medida em que permite o armazenamento de vários tipos de dados (estruturados ou não estruturados).Apesar disto, esta ferramenta apresenta também algumas desvantagens tais como o HDFS não permitir a modificação de dados, apenas o seu acréscimo, ou ainda o facto de o HDFS possuir uma rotina de manutenção de ficheiros e do próprio serviço que acaba por provocar *overhead* de processamento. [6, 7]

3.3 Apache Cassandra

O *Apache Cassandra* ou apenas Cassandra é uma base de dados não relacional capaz de armazenar grandes volumes de dados e de lidar com cetenas de milhares de operações *write* por segundo. É de salientar que o *Cassandra* possui uma arquitetura descentralizada circular e que foi desenvolvida tendo como princípios fundamentais a escalabilidade, performance e disponibilidade contínua. Esta arquitetura revela-se tolerante a faltas visto que não há um ponto único de falha além de ser também altamente confiável dado que o *Cassandra* efetua a replicação dados e distribuição destas cópias pelos vários nodos do *cluster*. [8] No entanto, esta base de dados apresenta algumas desvantagens. Por um lado, o *Cassandra* não apresenta elevada consistência visto que permite que se armazene a mesma informação várias vezes. Além disto, ao contrário do que acontece com as operações de *write*, as operações de *read* não são tão rápidas e eficientes.

3.4 Apache MapReduce

O *MapReduce* é um modelo de programação da *framework Hadoop* e responsável pelo processamento paralelo em grande escala de dados guardados em HDFS. Este processamento é feito nas várias máquinas de um *cluster Hadoop*. Tendo isto em conta, é possível entender que ao dividir partes dos ficheiros por várias localizações e ao permitir que estes sejam processados em simultâneo (paralelamente), há um aumento significativo de rapidez e eficácia dos seviços fornecidos por esta ferramenta.Além disto, o *MapReduce* apresenta também grande flexibilidade visto que permite o processamento de dados tanto estruturados como não estruturados. É importante também realçar que o *MapReduce* pressupõe o processamento de informações em disco. [9, 10] Por um lado, isto é uma vantagem pois não há um consumo muito elevado de memória. No entanto, operações feitas em disco são consideravelmente mais lentas do que aquelas efetuadas em memória RAM. [6, 7] ALém disto, o MapReduce é muito pouco eficiente em tarefas de processamento iterativas.

3.5 Apache Spark

O *Apache Spark* é uma ferramenta ETL (*Extract, transform, load*) *open source* capaz de processar grandes volumes de dados de forma paralela e distribuída. Esta ferramenta recorre a *in memory caching* e permite uma execução de queries mais rápida e eficaz. O *Spark* é mais eficiente que outras abordagens de *Big Data*, como por exemplo o *Map Reduce* popularizado pelo *Apache Hadoop*, dado que corre em memória RAM e não em disco. Além disto, esta ferramenta permite também distribuir tarefas de processamento de dados por vária máquinas (daí ser distribuída). [11**?**] Além do *Spark* ser uma ferramenta de rápido processamento, paralela, eficiente e com múltiplas funcionalidades, possui também outras vantagens tais como, permitir a programação em três linguagens distintas sendo elas o Java,Python e Scala ou ainda o facto de ser uma ferramenta fácil de usar visto que disponibiliza várias APIs para o efeito. No entanto, há também algumas desvantagens a ter em conta. O *Apache Spark* não possui o seu próprio sistema de gestão de ficheiros, sendo que para isto recorre ao *Hadoop* ou a plataformas *cloud-based*. Por outro lado, o *Spark* não tem grande capacidade de lidar com múltiplos utilizadores ao mesmo tempo. Outro problema que surge é o facto de que esta ferramenta é muito custosa,sendo o consumo de memória *RAM* muito elevado.[12]

3.6 Xplenty

A *Xplenty* é uma ferramenta ETL(*Extract, transform, load*) na qual é possivel criar um *pipeline* que consiste em integrar, processar e preparar dados para análise. Esta ferramenta de *Big Data* é bastante flexível uma vez que é capaz de se conectar a diversas fontes de dados. Isto que dizer, que com a *Xplenty* é possível , por exemplo, ler dados de um sistema de armazenamento em nuvem e, após o seu processamento e análise, armazená-los numa base de dados relacional. Além disto, é importante também realçar que esta ferramenta

permite efetuar a visualização destes dados bem como agendar tarefas de processamento (*scheduling*). No entanto, há medida que a quantidade de dados ou o número de *data sources* aumenta, algumas funcionalidades da ferramenta poderão perder performance. [13]

3.7 Power BI

O *Power BI* é uma ferramenta de ***B**usiness **I**ntelligence*, *cloud-based* e desenvolvida pela *Microsoft*, cujo objetivo é analisar dados e visualizá-los através de diversos elementos gráficos personalizáveis. [14] Ora, isto significa que com o *Power BI*, é possível acessar vastos volumes de dados, das mais diversas fontes (armazenamento na cloud, ficheiros csv, XML, Excel e outros) e transformá-los ema gráficos, tabelas ou diagramas. Desta forma, o utilizador conseguirá melhor entender e interpretar os dados que tem ao seu dispor. É ainda de realçar que para cada *dataset*, é possível criar um report dinãmico que vão atualizando à medida que os próprios dados mudam. No entanto, uma desvantagem desta ferramenta é que o processo de personalização de *dashboards* (página com gráficos, tabelas ou outros elementos gráficos) ainda é bastante manual além de ser repetitivo. Por outro lado, a capacidade de processamento do *Power BI* é limitada na versão gratuita desta ferramenta. [15]

3.8 Tableau

Assim como o *Power BI*, o *Tableau* é uma ferramenta de e ***B**usiness **I**ntelligence*, de visualização e análise de dados. Esta é capaz de produzir *dashboards* e *reports* bastante intuitivos que auxiliam os utilizadores na interpretação de dados. [16] Além disto, esta ferramenta é capaz de lidar com dados de diversas fontes(Excel, SQL server, XML, Google Cloud e outros), além de permitir também a realização de um número limitado de tarefas de tratamento e processamento de dados. É ainda importante salientar que o *Tableau* utiliza uma estrutura de armazenamento orientada às colunas o que possibilita um acesso mais eficiente aos dados guardados.No entanto, uma desvantagem que esta ferramenta apresenta é o facto de ser bastante cara comparada com outras opções no mercado.Além disto, outro ponto negativo do *Tableau* é o facto de que apenas as suas versões mais recentes possuem a funcionalidade de histórico. As versões mais antigas não permitem fazer *backup* de, por exemplo, *dashboards*. [15, 17]

4 Resultados

4.1 Análise dos Datasets

Para o desenvolvimento deste projeto prático, foi fornecido pela equipa docente um *dataset* inicial relacionado com a temática da *Covid-19*. Neste *dataset* os dados encontram-se organizados por país e por dia ou seja, uma determinada informação é relativa a um determinado dia num determinado país. É de realçar que o período temporal explorado é semelhante ao período da atual pandemia,

ou seja, as datas mais antigas remontam ao ínicio do ano de 2020 e extendem-se até ao presente (março de 2022).Assim sendo, existem colunas que referem o número de mortos e de novos casos por dia e por país, bem como o número cumulativo de mortes e de casos verificados.

Para além deste *dataset* inicial, foi pedido que se encontrassem alguns *datasets* que o complementassem. Desta forma, foram encontrados quatro *datasets* adicionais que serão explorados ao longo deste tópico.

Ora, o primeiro *dataset* [18], que se encontra em formato *JSON*, e cuja fonte é a plataforma *Our World in Data*, contém dados relativos ao processo de vacinação em diversos países do mundo. De forma mais detalhada, o conteúdo do *dataset* é o seguinte: o nome e código Id do país; a data em que cada entrada na tabela foi registada; o número total de vacinas aplicadas no país até à data; o número de pessoas que levaram pelo menos uma vacina no país até à data; número de pessoas totalmente vacinadas (por exemplo,levaram as duas doses) no país até à data; o número de vacinas aplicadas num dado dia;número de pessoas que levaram pelo menos uma vacina num dado dia;percentagem de vacinas aplicadas no país até à data; percentagem de pessoas que levaram pelo menos uma vacina até à data; percentagem de pessoas totalmente vacinadas até à data;razão em ppm (parte por milhão) de vacinas aplicadas até à data;nome de todas as marcas de vacinas utilizadas no país; nome da fonte onde os dados foram obtidos;*website* das fontes.

Já o segundo *dataset* [19] encontrado, cuja fonte é a plataforma *Our World in Data*, contém dados relativos às restrições do uso de máscara. Assim, para um determinado dia, num determinado país, a restrição imposta é de uma de 5 opções: (0) não houve restrições impostas; (1) o uso de mácara é recomendado mas não obrigatório; (2) Obrigatório em alguns espaços públicos ou em locais onde o distanciamento social não é possível; (3) Obrigatório em **todos** os espaços públicos e em locais onde o distanciamento social não é possível; (4) Obrigatório usar em qualquer local independentemente se há pessoas presentes ou não.

O terceiro *dataset* [20], cuja fonte é também a plataforma *Our World in Data*, possui dados referentes às restrições impostas aos ajuntamentos de indivíduos.Desta forma, para um determinado país, num determinado dia, a restrição imposta é uma de 5 opções: (0) não houve restrições impostas; (1) restrições nos ajuntamentos com mais de 1000 pessoas; (2) restrições nos ajuntamentos entre 100 e 1000 pessoas; (3) restrições nos ajuntamentos entre 10 e 100 pessoas; (4) restrições nos ajuntamentos com mais de 10 pessoas.

Por último, o quarto *dataset* [21] explorado,foi encontrado na plataforma *Kaggle* e contém dados relativos à preferência das pessoas em ficar em casa. Desta forma, para um determinado país,num determinado dia, está associado um valor númerico que indica o aumento ou diminuição em percentagem que se verificou de indivíduos que preferem ficar em casa em vez de sair, comparativamente ao período pré-pandemia.

4.2 Escolha das ferramentas

Tendo sido feita a análise de diversas ferramentas de *Big Data*, é agora preciso selecionar aquelas que farão parte do *pipeline*.

Em primeiro lugar, para efetuar a junção dos vários *datasets* e assim criar um *dataset* único, será utilizada a biblioteca *Pandas*. Isto deve-se essencialmente à facilidade de uso e flexibilidade desta ferramenta. É de salientar que o *Apache Spark* também serviria para esta fase do *pipeline* visto que disponibiliza uma API designada *Koalas*, semelhante ao *Pandas*. No entanto, tendo em conta o cenário deste artigo, e visto que não há a preocupação da escalabilidade e o volume de dados a tratar é relativamente pequeno, a biblioteca *Pandas* será uma ferramenta mais poderosa.

Já para a fase de armazenamento dos dados, tanto o *Apache Cassandra* com o HDFS seriam opções plausíveis, no entanto no cenário em questão, o HDFS é a ferramenta escolhida. Visto que o objetivo é apenas guardar os vários *datasets*, um sistema de ficheiros é suficiente.É de realçar que o *Cassandra* seria uma ferramenta mais vantajosa que o HDFS se fosse utilizada para processamento de dados em tempo real ou numa aplicação *always on* (exigência de alta disponibilidade) .

No que diz respeito ao tratamento e processamento do *dataset*, foi escolhida a ferramenta *Spark* por ser mais fácil de utilizar que o *MapReduce* além de possuir melhor performance. É importante salientar que será usado mais especificamente O *PySpark*, que é a API Python do *Apache Spark*.

Na fase de visualização de dados, selecionou-se a ferramenta *Power BI*, visto que é bastante utilizada atualmente, além de ser fácil de usar e de existir bastante documentação sobre ela. É de realçar que o *Tableau* seria uma ferramenta mais vantajosa caso se estivesse a trabalhar com dados em *real time*.

É também de salientar que não se escolheu a ferramenta *Xplenty* visto que não é uma ferramenta tão utilizada e conhecida como as outras que foram referidas e por isso não é disponibilizada muita documentação sobre ela.

4.3 Arquitetura

Ao longo deste subtópico iremos abordar detalhadamente todos as fases de uma possível proposta de *pipeline*(Figura 1).

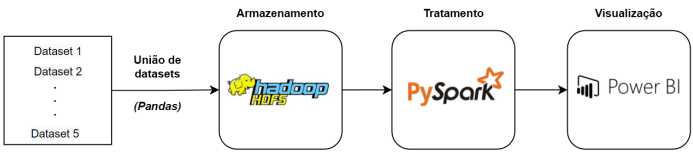

Figura 1: Esquema da arquitetura proposta

Começemos então pela **primeira fase** da arquitetura que consiste em unir os *datasets* num único *dataset* geral. Para isto, e como já se referiu irá ser

usada biblioteca *Pandas* do *Python*. No entanto, é preciso ter em conta que existem ficheiros tanto em formato csv como em formato Json. Para tratar deste problema, primeiro coloca-se os dados dos ficheiros csv e Json em objetos *Dataframe* usando-se, para isso, as funções *read_csv()* e *read_json()*, respetivamente. Ora, após isto, dá-se a fase do*merge* (junção) dos vários *datasets*, sendo que esta união é feita pelos código ISO dos países **e** pelas datas. É de notar que em todos os *datasets*, as colunas que vão ser utilizadas para fazer a junção devem ter o mesmo nome. Assim sendo, uma hipótese seria que as colunas com o código dos países devam ser chamadas "ISO_Code" e as colunas com as datas deve ser chamada "Date". Além disto, tem de ser ter em conta o formato das datas que é diferente em alguns dos *datasets* pelo que é necessário uniformizá-las. Para tal usa-se a função *strftime* que recebe como argumento o formato desejado. Por outro lado, nos vários *datasets* os códigos ISO também não estão uniformes.A título de exemplo, surge tanto USA como US para representar os Estados Unidos. Uma forma, de tratamento seria utilizar a biblioteca *PyCountry*, que faz o mapeamento dos países para o um código ISO único.Além disto, para o caso dos *datasets* que não têm uma coluna com o código do país, o uso desta biblioteca também resolveria o problema.

Posto isto, basta chamar a função *merge* iterativamente para ir juntando os vários *datasets* num só. É de realçar que nesta fase, não houve qualquer tratamento de *missing values* ou geração de dados sintéticos. Isto foi decidido para que os dados que forem armazenados sejam o mais próximo possível dos originais.

A **segunda fase** do *pipeline* é bastante simples, sendo que apenas consiste em armazenar os *datasets* da fase anterior em HDFS. Para tal, basta instanciar um objeto cliente HDFS no mesmo ambiente onde se realizou a fase anterior, como por exemplo um *Jupyter Notebook*. Após isto,utiliza-se a função *write()* para escrever os *dataframes* , no formato csv, no sistema de ficheiros local. [22]

Já na **terceira fase** do *pipeline*, o objetivo consiste em efetuar o tratamento do *dataset* final para este ficar pronto para a fase de visualização. Em primeiro lugar, é preciso importar, do sistema de ficheiros local, o *dataset* final. Para isto, cria-se um objeto *Spark Session* e utiliza-se a sua função *read.csv()* que recebe como argumento o caminho para ficheiro csv pretentido. Desta forma, o *dataset* lido será convertido num objeto *Dataframe*. [23]

Posto isto,fará sentido analisar que colunas poderão não ser relevantes para o estudo do problema e assim removê-las, sendo que para isto utiliza-se a função *drop*. Um possível exemplo de coluna que poderia ser retirada é a coluna que indica o *website* da fonte dos registos. Posto isto, é necessário tratar dos *missing values* recorrendo para isso à geração de dados sintéticos. Por um lado, uma possível abordagem seria agrupar os dados por país (usando a função *Groupby*) e, usando a função *sequence*, preencher as datas em falta. Além disto, e relativamente ao aos dados da vacinação, todos os registos obtidos antes de dezembro de 2020 devem ser colocados a zero , visto que só a partir desta data, é que processo de vacinação iniciou. Ainda no caso das colunas que possuem dados em falta, se esses dados forem númericos contínuos uma

solução seria fazer agrupamentos por país, calcular a mediana (usando a função *median()*) para cada país e preencher os *missing values* com este valor. O uso da mediana, deve-se ao facto que esta métrica é menos impactada pela presença de *outliers*.Por outro lado, se a coluna apresentar a falta de dados categóricos, a abordagem seria a mesma, mas, em vez, da mediana, usar-se-ia, por exemplo, a moda (usa-se a função *most_common()*).

É importante referir que, certamente, outros métodos de tratamento terão se ser aplicados ao *dataset* no momento em que ele estiver a ser tratado. No entanto, há confiança de que o *Pyspark*, sendo uma ferramenta ETL bastante poderosa, seja capaz de efetuar as mais diversas operações de processamento de dados. Por fim, este *dataset* processado será armazenado novamente em HDFS. Para tal basta utilizar a função *write.csv()* do *dataframe* que contêm o conteúdo*dataset* e que recebe como argumento o caminho para o destino, ou seja, o caminho para o sistema de ficheiros local.

Por último, a **quarta fase** consiste na visualização de dados utilizando a ferramenta *Power BI*. Para começar, é necessário importar o *dataset* armazenado em HDFS. Para tal, basta selecionar a opção de conectar ao *Hadoop File System*.A partir daqui, pode-se criar um *dashboard* no qual serão inseridos vários elementos gráficos através do quais é possível analisar com detalhe o *use case* escolhido. De forma mais detalhada, isto significa que através de gráficos temporais, *scatter plots* e outros elementos disponibilizados pela ferramenta, consegue-se observar a correlação existente (ou inexistente) entre, por exemplo :

- Evolução do processo de vacinação e a preferência das pessoas em ficar de casa.
- Aumento de mortes/número de casos a preferência das pessoas em ficar de casa.
- As restrições impostas ao uso de máscara e a preferência das pessoas em ficar de casa.
- As restrições impostas aos ajuntamentos e a preferência das pessoas em ficar de casa.

Seria também interessante avaliar estas correlações tanto por país, como por período de tempo, ou seja, verificar em que países ou meses do ano as pessoas revelam maior preferência em ficar em casa.

5 Conclusão

É inegável que atualmente a área de *Big Data* ganha cada vez mais importância e relevo, visto que as empresas geram e consomem dados a um ritmo acelerado. Estes dados são fundamentais pois, a partir deles, é possível extrair informações e conhecimento que permitem aumentar a produtividade destas empresas, melhor entender as necessidades de clientes ou ainda ganhar vantagens competitivas no mercado.No entanto, o processo de transformar *raw data* em algo com significado, pressupõe o uso de ferramentas de *Big Data*.

Assim sendo, ao longo do presente artigo científico foi explorada a temática da pademia Covid-19, mais precisamente o impacto de fatores como a vacinação ou as restrições de uso de máscara na preferência das pessoas em ficar em casa. Este caso de uso foi abordado, recorrendo ao estudo e análise de diversos *datasets* além da aplicação de ferramentas de *Big Data* nos mesmos. Após se avaliar diversas ferramentas e selecionar as que seriam mais vantajosas, desenvolveu-se a uma possível proposta de arquitetura que contempla todos os passo necessários para a exploração deste *use case*. Desta forma, considera-se que este documento foi elaborado com sucesso, havendo confiança nas ferramentas escolhidas e no *pipeline* que se elaborou.

References

[1] Top 10 Big Data Tools (Big Data Analytics Tools) (2022). https://www.softwaretesttips.com/big-data-tools/

[2] Pratt, K.: 17 top big data tools and technologies to know about in 2022 (2022). https://www.techtarget.com/searchdatamanagement/feature/15-big-data-tools-and-technologies-to-know-about

[3] Reddy, C., Singh, D.: A survey on platforms for big data analytics (2014)

[4] McKinney, W.: pandas: powerful python data analysis toolkit (2012)

[5] Google: HDFS vs. Cloud Storage: Pros, cons and migration tips. https://cloud.google.com/blog/products/storage-data-transfer/hdfs-vs-cloud-storage-pros-cons-and-migration-tips

[6] Bappalige, S.: An introduction to Apache Hadoop for big data (2014). https://opensource.com/life/14/8/intro-apache-hadoop-big-data

[7] Gangodkar, D., Ghazi, M.: Hadoop, mapreduce and hdfs: A developers perspective (2015)

[8] Lakshman, A., Malik, P.: Cassandra - a decentralized structured storage system (2010)

[9] IBM: What is MapReduce? https://www.ibm.com/topics/mapreduce

[10] Hadoop. https://databricks.com/glossary/hadoop

[11] Joseph, R.: What is Spark? https://chartio.com/learn/data-analytics/what-is-spark/

[12] Joseph, R.: Apache Spark Pros and Cons (2021). https://www.knowledgehut.com/blog/big-data/apache-spark-advantages-disadvantages

[13] Xplenty: SecurETL Platform Overview and Review. https://www.datawarehouse4u.info/reviews/cloud-data-integration-software/xplenty-overview-and-review

[14] Ferrari, A., Russo, M.: Introducing microsoft power bi (2016)

[15] Power BI vs Tableau. https://www.educba.com/power-bi-vs-tableau/

[16] Batt, S., Harmon, R., Tomolonis, P.: Learning tableau: A data visualization tool (2020)

[17] What Are the Pros Cons of Tableau? https://thinklytics.com/what-are-the-pros-cons-of-tableau/

[18] COVID-19 World Vaccination Progress. https://raw.githubusercontent.com/owid/covid-19-data/master/public/data/vaccinations/vaccinations.json

[19] Face covering policies during the COVID-19 pandemic. https://ourworldindata.org/grapher/face-covering-policies-covid

[20] Restrictions on public gatherings in the COVID-19 pandemic. https://ourworldindata.org/grapher/public-gathering-rules-covid

[21] Worldwide Residential Mobility in COVID-19. https://www.kaggle.com/aestheteaman01/people-staying-in-home-during-covid19

[22] Python HDFS. https://tahiriamine9.medium.com/python-hdfs-cd822199799e

[23] How to read data from HDFS in Pyspark. https://www.projectpro.io/recipes/read-data-from-hdfs-pyspark