EKONOMICKÁ UNIVERZITA V BRATISLAVE
FAKULTA HOSPODÁRSKEJ INFORMATIKY

Evidenčné číslo 103004 /I/2023/ 421000214251

AKO SKROTIŤ BIG DATA V PYTHONE

Diplomová práca

2023 **Bc. Juraj Šiandor**

EKONOMICKÁ UNIVERZITA V BRATISLAVE
FAKULTA HOSPODÁRSKEJ INFORMATIKY

AKO SKROTIŤ BIG DATA V PYTHONE

Diplomová práca

Študijný program:	Informačný manažment
Študijný odbor:	Ekonómia a manažment
Školiace pracovisko:	Katedra štatistiky
Vedúci záverečnej práce:	Ing. Silvia Komara, PhD.

Bratislava 2023 **Bc. Juraj Šiandor**

Ekonomická univerzita v
BratislaveFakulta hospodárskej
informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Juraj Šiandor

Študijný program: informačný manažment (Jednoodborové štúdium, inžiniersky II. st., denná forma)
Študijný odbor: ekonómia a manažment
Typ záverečnej práce: Inžinierska záverečná práca
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Ako skrotiť BigData v Pythone

Anotácia: Až 90% všetkých dát vzniklo za posledné 2 roky. Obrovské objemy dát si vyžadujú iné prístupy na spracovanie ako klasické dáta. Práve spracovanie a príprava dát je jedným z najdôležitejších krokov. Študent sa bude venovať technológiám spracovania BigData (Hadoop, Spark) v programovacom jazyku Python.

Vedúci: Ing. Silvia Komara, PhD.

Oponent: doc. Ing. Michal Páleš, PhD.

Katedra: KŠ FHI - Katedra štatistiky

Vedúci katedry: doc. Ing. Mária Vojtková, PhD.

Dátum zadania: 25.10.2021

Dátum schválenia: 15.04.2023 prof. Ing. Ivan Brezina, CSc.

osoba zodpovedná za realizáciu študijného programu

Čestné vyhlásenie

Čestne vyhlasujem, že diplomovú prácu s názvom: Ako skrotiť BigData v Pythone som vypracoval samostatne a že som uviedol všetku použitú literatúru.

Dátum: .....

podpis študenta

Poďakovanie

Touto cestou by som sa chcel poďakovať Ing. Silvia Komara, PhD. za pomoc, odborné vedenie, cenné rady a pripomienky pri vypracovaní mojej diplomovej práce.

Abstrakt

ŠIANDOR, Juraj: Ako skrotiť Big Data v Pythone – Ekonomická univerzita v Bratislave. Fakulta hospodárskej informatiky; Katedra aplikovanej informatiky. – Vedúci záverečnej práce: Ing. Silvia Komara, PhD. Bratislava: FHI, 2023, 65s.

Cieľom práce je názorná ukážka spracovania veľkých dát pomocou technológií (Hadoop, Spark) v jazyku Python. Práca je rozdelená do štyroch kapitol. Obsahuje 39 ilustrácií. Prvá kapitola je venovaná stručnému opisu súčasného stavu problematiky a základným pojmom. V druhej kapitole sa charakterizuje cieľ práce. V tretej kapitole sme sa venovali metódam a nástrojom na prácu s veľkými dátami v jazyku Python. Záverečná kapitola sa zaoberá samotným postupom tvorby našich programov a analýzy dát. Výsledkom riešenia danej problematiky sú dva funkčné programy a krátka analýza dát. Prvý program púšťame na virtuálnom stroji pomocou technológii Hadoop a druhý púšťame na kontajnerovej platforme Docker pomocou technológii Spark.

Kľúčové slová: veľké dáta, Python, virtuálny stroj, Spark, Hadoop

Abstract

ŠIANDOR, Juraj: Model for creating a database system for a company with multiple branches. - University of Economics in Bratislava. Faculty of Economics Informatics; Department of Applied Informatics. – Thesis Supervisor : Ing. Silvia Komara, PhD. Bratislava: FHI, 2023, 65s.

The main goal of the final work is to create visual demonstration of processing Big Data using technologies (Hadoop, Spark) in the Python language. The work is divided into four chapters. The work contains 39 illustrations. The first chapter is devoted to a brief description of the current state of the issue and basic terms. The aim of the work is characterized in the second chapter. In the third chapter, we dealt with methods and tools for working with big data in the Python language. The final chapter contain creation of our programs and short data analysis. Our solutions to the given issue are two functional programs and a short data analysis. We run the first program on a virtual machine using Hadoop technology and the second one on the container platform Docker, using Spark technology.

Key words: big data, Python, virtual machine, Spark.

1 Obsah

Zoznam obrázkov

Úvod

V posledných rokoch sa Python stal obľúbeným programovacím jazykom pre dátových vedcov, analytikov a inžinierov vďaka svojej všestrannosti, jednoduchosti použitia a výkonným knižniciam na analýzu dát a manipuláciu s nimi. Práca s veľkými údajmi v Pythone však môže byť náročná, pretože veľké súbory údajov často vyžadujú špecializované techniky na optimalizáciu výkonu a predchádzanie chybám súvisiacim s pamäťou. V tomto kontexte je nevyhnutné pochopiť, ako ovládať veľké údaje v Pythone, aby sme mohli efektívne spracovávať, analyzovať a vizualizovať veľké súbory údajov.

Na kontrolu veľkých dát v Pythone je k dispozícii niekoľko techník a nástrojov vrátane paralelného spracovania, distribuovaných výpočtov, streamovania dát a cloudových riešení. Každý z týchto prístupov má svoje silné a slabé stránky a výber toho správneho závisí od konkrétnych požiadaviek projektu. Napríklad paralelné spracovanie je užitočné na rozdelenie veľkého súboru údajov na menšie časti, ktoré možno spracovávať paralelne, zatiaľ čo distribuované výpočty umožňujú spúšťať výpočty na viacerých počítačoch. Na prácu s veľkými dátami v Pythone je tiež dôležité používať vhodné knižnice a rámce, ktoré dokážu efektívne spracovať veľké množiny údajov. Niektoré z populárnych knižníc na spracovanie veľkých dát v Pythone zahŕňajú Pandas, Dask, NumPy a SciPy, zatiaľ čo frameworky ako Apache Spark, Hadoop a PySpark umožňujú distribuované výpočty a paralelné spracovanie.

V tejto práci preskúmame rôzne techniky a nástroje na riadenie veľkých dát v Pythone a prediskutujeme najlepšie postupy pre ukladanie dát a spracovanie dát. Ukážeme tiež, ako používať populárne knižnice a frameworky na spracovanie veľkých dát v Pythone a poskytneme príklady prípadov použitia v reálnom svete. Na konci tejto práce by sme mali rozumieť tomu, ako ovládať veľké údaje v Pythone, a budeme môcť použiť tieto techniky a nástroje na efektívnu správu, spracovanie a analýzu veľkých súborov.

1 Súčasný stav problematiky

Organizácie generujú obrovské množstvo údajov a hľadajú spôsoby, ako dať týmto informáciám zmysel, aby získali prehľad a robili rozhodnutia podložené údajmi. Pandémia COVID-19 ešte viac urýchlila používanie veľkých dát v rôznych odvetviach, ako je zdravotníctvo, riadenie dodávateľského reťazca a elektronický obchod. V zdravotníctve sa veľké dáta používajú napríklad na sledovanie šírenia vírusu a predpovedanie prepuknutia. V manažmente dodávateľského reťazca sa veľké dáta používajú na monitorovanie toku tovaru a zabezpečenie jeho včasného doručenia. A v elektronickom obchode sa veľké dáta používajú na prispôsobenie skúseností zákazníkov a optimalizáciu online predaja. Vo všeobecnosti sa veľká dáta dajú využiť a využívajú čoraz viac v akomkoľvek odvetví.

1.1 Dátová veda

Dátová veda (data science) je skúmanie údajov za účelom získania zmysluplných poznatkov pre podnikanie. Ide o multidisciplinárny prístup, ktorý kombinuje poznatky a postupy z oblasti matematiky, štatistiky, umelej inteligencie a počítačového inžinierstva s cieľom analyzovať veľké množstvo údajov. Táto analýza pomáha odhaliť použiteľné poznatky skryté v údajoch. Tieto poznatky môže následne využiť manažment podniku pri rozhodovaní o ďalších krokoch podniku. [1]

Životný cyklus data science sa skladá z niekoľkých krokov:

- **Zber údajov**
- **Spracovanie údajov**
- **Analýza údajov**
- **Prezentácia údajov**

Zber údajov je jedným z časovo najnáročnejších krokov v životnom cykle. Predstavuje proces zhromažďovania, merania a analýzy údajov z rôznych relevantných zdrojov s cieľom nájsť odpovede na dané problémy. V podnikoch prebieha zber údajov na viacerých úrovniach. IT systémy pravidelne zhromažďujú údaje o zákazníkoch, zamestnancoch, predaji a iných aspektoch obchodných operácií pri spracovaní transakcií a zadávaní údajov. Spoločnosti tiež vykonávajú prieskumy a sledujú sociálne médiá, aby získali spätnú väzbu od zákazníkov.

Taktiež zaznamenávame údaje v reálnom čase:

- **Online streamovacích platformách** - ako sú YouTube, Twitch, Skype alebo Netflix.
- **Údaje o transakciách** - sa zhromažďujú, keď používateľ uskutoční online nákup. Týmto spôsobom získame informácie o produkte, čase nákupu a spôsoboch platby.
- **Geografické údaje** – sú údaje o polohe ľudí, vozidiel, budov, prírodných rezervácií a iných objektov, ktoré sú nepretržite zásobované satelitmi.
- **Údaje časových radov** - tento typ údajov súvisí s pozorovaním trendov a javov, ktoré sa odohrávajú práve v tomto okamihu a počas určitého časového obdobia, napríklad globálne teploty, miera úmrtnosti, úrovne znečistenia atď.
- **Prepojené údaje** - sú založené na webových technológiách HTTP, RDF, SPARQL a URI a majú umožniť sémantické spojenia medzi rôznymi databázami, aby počítače mohli správne čítať a vykonávať sémantické dotazy.

Najčastejšie spôsoby zberu údajov sú:

- **Žiadosťou o údaje** - väčšina firiem uprednostňuje priame vyžiadanie si osobných údajov od používateľov. Tieto údaje poskytujú pri vytváraní účtov na webových stránkach alebo pri nákupe online. Minimálne informácie, ktoré sa majú zhromažďovať, zahŕňajú používateľské meno a e-mailovú adresu, ale niektoré profily vyžadujú viac podrobností**.**
- ***Cookies*** - je široko používaná metóda na zhromažďovanie údajov o používateľoch, konkrétne o tom, aké webové stránky navštevujú a kedy. Poskytujú základné štatistiky o tom, ako sa webová stránka používa.
- **Sledovanie e-mailov** - sledovače e-mailov majú za úlohu poskytnúť viac informácií o akciách používateľa v poštovej schránke. Najmä sledovač e-mailov umožňuje zistiť, kedy bol e-mail otvorený. Google aj Yahoo používajú túto metódu na zistenie vzorcov správania svojich používateľov a na poskytovanie personalizovanej reklamy. [3]

Spracovanie údajov predstavuje zmenu nespracovaných informácií do žiaduceho, použiteľného a zrozumiteľného formátu, najčastejšie vo viacerých krokoch. Kroky spracovania údajov zahŕňajú zhromažďovanie, zaznamenávanie, organizovanie,

štruktúrovanie, ukladanie, získavanie, používanie a šírenie. Spracované údaje môžu mať formu obrázka, grafu, tabuľky, alebo akéhokoľvek iného formátu. Spracovanie údajov pomáha ľuďom triediť informácie rýchlo, spoľahlivo a presne. Tisíce údajov je možné spracovať v priebehu niekoľkých minút a údaje je možné organizovať tak, aby sa súbory kontrolovali a neplatné alebo poškodené údaje sa odstránili alebo minimalizovali. [4]

Spracovanie údajov pomáha tomu, aby sme dokázali zväčšiť náš úložný priestor. Spracované údaje prichádzajú vo forme, ktorej môžu ľudia ľahko porozumieť a môžu byť užitočné viacerými spôsobmi. Napríklad údaje o teplotných priemeroch alebo dokonca údaje o predaji možno previesť do grafov alebo tabuliek a potom ich prezentovať spôsobom, ktorý umožňuje predpovedať, čo sa môže stať v budúcnosti.[4]

Analytika údajov je veda o analýze údajov s cieľom vyvodiť závery o týchto informáciách. Mnohé z techník a procesov analýzy údajov boli automatizované do mechanických procesov a algoritmov, ktoré pracujú s nespracovanými údajmi pre ľudskú spotrebu. Techniky analýzy údajov môžu odhaliť trendy a metriky, ktoré by sa inak stratili v množstve informácií. Tieto informácie sa potom môžu použiť na optimalizáciu procesov, ktoré zvyšujú celkovú efektívnosť podniku alebo systému. Výrobné spoločnosti napríklad často zaznamenávajú dobu chodu, prestoje a pracovné fronty pre rôzne stroje a potom analyzujú údaje, aby mohli lepšie plánovať pracovné zaťaženie, aby stroje fungovali bližšie k maximálnej kapacite. Analýza údajov je dôležitá, pretože pomáha firmám zvýšiť ich výkonnosť. Jej začlenenie do obchodného modelu znamená, že spoločnosti môžu znížiť náklady objavením efektívnejších spôsobov podnikania a ukladaním veľkého množstva údajov. Spoločnosť môže tiež použiť analýzu údajov na lepšie obchodné rozhodnutia a pomôcť analyzovať trendy a spokojnosť zákazníkov, čo môže viesť k novým a lepším produktom a službám. [5]

Prezentácia dát pomáha potencionálnym klientom alebo publiku nestrácať čas zaoberaním sa o detaily, ale presvedčiť ich, aby investovali do spoločnosti a premenili ju na ziskovú. Nástroje na prezentáciu údajov sú výkonné komunikačné nástroje, ktoré môžu zjednodušiť pochopenie údajov tým, že ich urobia ľahko zrozumiteľnými a čitateľnými zároveň, pričom pritiahnu a udržia záujem potencionálnych klientov a efektívne prezentujú veľké množstvo zložitých údajov takým spôsobom aby im pochopil aj laik. Ak používateľ dokáže vytvoriť dômyselnú prezentáciu údajov s dostupnými súbormi faktov a čísel, potom je veľký predpoklad, že sa dostavia chcené výsledky. [6]

1.2 Big Data

Cieľom spracovanie veľkých dát je analýza veľkého množstva údajov z rôznych zdrojov a vytvoriť z nich kvalitné náhľady. Spracovanie veľkých dát si môžeme predstaviť ako nástroje a technológie používané na skladovanie, spravovanie a analýzu dát, bez toho aby sme vytvárali nejaké obmedzenia na zdroj, formát, alebo veľkosť údajov. Zvyčajne sú veľké dáta charakterizované šiestimi *V*-čkami (volume, velocity, variety, value, veracity, variability). [10]

- **Volume** - predstavuje objem dát. Môžu to byť desiatky, stovky terabajtov a vo veľkých spoločnostiach to môžu byť až petabajty. Preto je za potreby spoľahlivý systém, ktorý bude schopný pracovať aj s takýmto množstvom údajov.
- **Velocity** - znamená rýchlosť akou dokážeme generovať dáta, ako rýchlo s nimi dokážeme manipulovať a meniť ich. Napríklad video na TikToku sa v jednom dni stane virálnym a za pár dní už je irelevantné a uvoľní miesto pre ďalší trend.
- **Variety** – môžeme to preložiť ako rozmanitosť dát, je dôležité aby procesné systémy dokázali spracovávať dáta bez toho, aby ich nejako obmedzovali. Potrebujeme vedieť pracovať aj so štruktúrovanými aj neštruktúrovanými dátami, ako sú napríklad obrázky, sociálne média textové súbory a podobne.
- **Veracity** – sa vzťahuje na kvalitu a pôvod veľkých dát. V preklade to znamená pravdivosť. Vzhľadom nato, že nerobíme žiadne obmedzenia na dátach sa môže stávať, že množstvo dát je nerelevantných alebo nepravdivých, prípadne neúplných. Pri fázy spracovávania dát ich čistíme a preverujeme predtým, ako z nich začneme robiť analýzy.
- **Variability** - variabilita údajov známa aj ako rozptyl sa vzťahuje na rozloženie množiny údajov. Variabilita poskytuje používateľom spôsob, ako opísať, ako veľmi sa množiny údajov líšia a umožňuje používateľom používať štatistiky na porovnanie svojich údajov s inými súbormi naplnenými údajmi.
- **Value** – pravdepodobne najdôležitejším bodom je hodnota veľkých dát v systémoch. Pre rôznych ľudí majú dáta rôznu hodnotu v rôzny čas. Napríklad pre obchodníkov s akciami sú dôležité aktuálne hodnoty, no pre analytikov sú podstatné dáta aj do minulosti, na základe ktorých vytvárajú analýzy. Alebo na základe dát z minulosti vieme vytvoriť strojové učenie a predvídať ceny na trhu. [10]

1.2.1 Data lakes

Data lakes (dátové jazerá) sú často porovnávané s data warehouses (dátové sklady) pretože oba koncepty umožňujú skladovanie veľkých objemov dát za účelom ich transformácie do informácií. Pri data lakes sa očakáva, že budú flexibilnejšie ako data warehouses, pretože neobsahujú údaje, ktoré sú integrované podľa rovnakej schémy. Čiže akékoľvek dáta môžu byť obsiahnuté v data lake, bez ohľadu na ich pôvod, čo znamená, že jednou z hlavných výziev je umožniť akékoľvek spracovanie, ktoré sa týka týchto údajov. V data lake sú informácie, ktoré majú byť vyjadrené z dát kompletne neznáme a je na užívateľovi aby ich vyjadril na základe jeho potrieb. Toto vysvetľuje prečo sú kroky integrácie dát nazývané ELT(Extract-Load-Transform). Je veľmi dôležité aby bola zabezpečená pravdivosť údajov, keď spájame niekoľko zdrojov, pretože čistenie dát je dôležitý krok. V data warehouses tento proces prebieha pri transformačnom procese pred načítaním, čo pri data lakes nie je možné, pretože dáta sa menia až na základe príkazu od užívateľa. Avšak v praxi sa to dá dosiahnuť normalizáciou a extrakciou metadát. Toto ukazuje aké je dôležité riadenie dát a špeciálne vedenie a údržba zoznamu metadát, ak chceme dosiahnuť, aby boli dáta pravdivé a zo správnych zdrojov. [9]

	Data lakes	**Data warehouses**
Skladovanie dát	HDFS,NoSQL,Relačné dat.	Relačné databázy
Hodnota dát	Vysoká	Vysoká
Zrnitosť dát	Nespracované	Agregované
Príprava dát	Meniaca sa v priebehu	Pred integráciou
Integrácia dát	Žiadna	Kontrola kvality, filtrovanie
Transformácia dát	ELT	ETL
Informačná architektúra	Horizontálna	Vertikálna
Model	Meniaci sa v priebehu	Star, snowflake
Metadáta	Áno	Voliteľné
Metóda analýzy dát	Unikátna	Opakujúca sa
Pravidelnosť aktualizácií	V reálnom čase	Po dávkach
Architektúra	Centralizovaná, federatívna, hybridná	Centralizovaná
Používatelia	Data scientisti, developeri	Osoby, ktoré majú dáta na starosť

Tabuľka 1 Porovnanie Data lakes a data warehouses; zdroj: [9]

1.2.2 Cloud data lakes

V poslednej dobe sa používanie cloudu stáva stále populárnejšou možnosťou. Cloud poskytuje hardvér aj softvér, či už sa jedná o servery, úložiská, databázy, siete, alebo programy na analýzu. Odľahčí používateľov o potreby fyzickej údržby infraštruktúry, tým si stačí prenajať služby cloudu a oni poskytnú výpočtové zdroje a celý rad ďalších služieb na splnenie rôznych výpočtových potrieb. [26]

Cloud data lake je typ úložiska údajov, ktoré sa používa na ukladanie veľkých objemov štruktúrovaných, pološtruktúrovaných a neštruktúrovaných údajov v cloude. Data lakes sú navrhnuté tak, aby boli vysoko škálovateľné a nákladovo efektívne a možno ich použiť na ukladanie údajov z rôznych zdrojov, ako sú sociálne médiá, zariadenia internetu vecí a podnikové aplikácie. Na rozdiel od tradičných dátových skladov, ktoré sú určené pre štruktúrované dáta, sú data lakes navrhnuté tak, aby uložili a spracovali veľké objemy dát v ich natívnom formáte, bez potreby predspracovania alebo transformácie. Organizácii to uľahčuje ukladanie a analýzu údajov z rôznych zdrojov a získavanie prehľadov a hodnôt z ich údajov. [10]

Cloud data lakes sú zvyčajne postavené na službách cloudového úložiska, ako je Amazon S3, Azure Data Lake Storage alebo Google Cloud Storage, ktoré poskytujú lacné, vysoko škálovateľné úložisko pre dáta akejkoľvek veľkosti alebo typu. Okrem cloudového úložiska môžu cloudové dátové jazerá využívať aj cloudové analytické služby, ako napríklad Amazon EMR, Azure HDInsight alebo Google Dataproc, na spracovanie a analýzu údajov uložených v dátovom jazere. Niektoré z výhod cloudových dátových jazier zahŕňajú: [27]

- **Škálovateľnosť** - cloudové dátové jazerá sa môžu škálovať, aby mohli ukladať a spracovávať obrovské objemy dát, čo organizáciám uľahčuje rast a škálovanie ich možností spracovania dát.
- **Nákladová efektívnosť** - cloudové data lakes môžu byť nákladovo efektívnejšie ako tradičné dátové sklady, pretože môžu využívať lacné cloudové úložiská na ukladanie veľkých objemov dát.
- **Flexibilita -** cloudové dátové jazerá môžu ukladať dáta v ich natívnom formáte bez potreby predbežného spracovania alebo transformácie, čo organizáciám uľahčuje prácu s rôznymi typmi dát a zdrojmi.

- **Rýchlosť** - cloudové dátové jazerá môžu poskytnúť rýchly prístup k údajom, čo organizáciám umožňuje analyzovať údaje v reálnom čase, alebo takmer v reálnom čase.
- **Analýza** - cloudové dátové jazerá môžu byť integrované s cloudovými analytickými službami, ktoré môžu poskytnúť výkonné nástroje na spracovanie a analýzu veľkých objemov dát a extrahovanie prehľadov a hodnoty z týchto dát. [27]

Celkovo sú cloudové dátové jazerá dôležitým nástrojom pre organizácie, ktoré potrebujú ukladať a spracovávať veľké objemy údajov nákladovo efektívnym, škálovateľným a flexibilným spôsobom. Môžu pomôcť organizáciám získať poznatky a hodnotu z ich údajov a získať konkurenčnú výhodu na trhu. [26]

1.3 Big Data analýza

Analytika veľkých údajov sa vzťahuje na metódy, nástroje a aplikácie používané na zhromažďovanie, spracovanie a odvodzovanie poznatkov z rôznych, veľkoobjemových a vysokorýchlostných súborov údajov. Tieto súbory môžu pochádzať z rôznych zdrojov, ako je web, mobil, e-mail, sociálne médiá a sieťové inteligentné zariadenia. Často obsahujú údaje, ktoré sa generujú vysokou rýchlosťou a majú rôznu formu, od štruktúrovaných (databázové tabuľky, hárky Excelu) cez pološtruktúrované (súbory XML, webové stránky) až po neštruktúrované (obrázky, zvukové súbory). [12]

Údaje sú zapletené do každodennej štruktúry našich životov. S rozvojom mobilných zariadení, sociálnych médií a inteligentných technológií spojených s internetom, teraz prenášame viac dát ako kedykoľvek predtým – a to závratnou rýchlosťou. Vďaka analýze veľkých dát môžu organizácie teraz použiť tieto informácie na rýchle zlepšenie spôsobu práce, myslenia a poskytovania hodnoty svojim zákazníkom. S pomocou nástrojov a aplikácií nám veľké dáta môžu pomôcť získať prehľad, optimalizovať operácie a predpovedať budúce výsledky. Uvedieme si základné štyri druhy analýz. [12]

- **Deskriptívna analýza** - sumarizuje minulé údaje do formy, ktorú môžu ľudia ľahko prečítať. Pomáha to pri vytváraní prehľadov, ako sú príjmy, zisky, tržby spoločnosti atď. Pomáha tiež pri zostavovaní metrík sociálnych médií. [13]
- **Diagnostická analýza** - sa robí, aby sme pochopili, čo spôsobilo problém v prvom rade. Príkladmi sú techniky ako hĺbková analýza, dolovanie údajov a obnova údajov.

Organizácie používajú diagnostickú analýzu, pretože poskytujú hĺbkový pohľad na konkrétny problém. [13]

- **Prediktívna analýza** - tento typ analýzy skúma historické a súčasné údaje, aby mohol predpovedať budúcnosť. Prediktívna analytika využíva dolovanie údajov , AI (umelá inteligencia) a strojové učenie na analýzu aktuálnych údajov a vytváranie predpovedí o budúcnosti. Funguje na predpovedaní trendov zákazníkov, trendov na trhu atď. [13]
- **Preskriptívna analýza** - tento typ analýzy predpisuje riešenie konkrétneho problému. Perspektívna analytika pracuje s popisnou aj prediktívnou analytikou. Väčšinu času sa spolieha na AI a strojové učenie. [13]

1.4 Využitie Big Data v praxi

Ako technológia neustále napreduje, očakáva sa, že veľké dáta budú hrať čoraz významnejšiu úlohu pri podpore inovácií a zlepšovaní rozhodovania v rôznych oblastiach. Veľké dáta sa využívajú v reálnom svete v rôznych odvetviach. My si z nich uvedieme niekoľko príkladov.

1.4.1 Netflix

Snaha Netflixu o to, aby dokázali predvídať čo sa bude ich zákazníkom páčiť začala už v roku 2006, kedy ponúkli milión dolárov skupine, ktorá vytvorí najpresnejší algoritmus na predikovanie hodnotenia filmov na základe ich predchádzajúcich hodnotení. Konečný víťaz bol oznámený v roku 2009. Aj napriek neustálim zmenám a nadstavbám algoritmu sú jeho základné princípy stále rovnaké. Na začiatku boli analytici limitovaný nedostatkom informácií, pretože k dispozícii mali iba 4 premenné a to ID zákazníka, ID filmu, ID hodnotenia a dátum pozerania filmu. Od momentu, kedy sa ich primárnym spôsobom doručenia filmov stalo streamovanie, získali o zákazníkoch oveľa viac informácií. Tieto informácie umožnili dátovým analytikom vytvoriť dátové modely, ktoré neustále zásobujú zákazníkov filmami, ktoré by sa im mohli páčiť. Ďalším kľúčovým elementom toho, že nám Netflix odporúča filmy, ktoré sa nám páčia je značkovanie. Spoločnosť platí ľudí zato, že pozerajú filmy a následne ich označia elementmi, ktoré tieto filmy obsahujú. Následne vám odporučia filmy od iných produkcií, ktoré mali podobné značky ako filmy, ktoré ste si už pozreli. Vďaka tomuto značkovaniu Netflix objavil viac ako 80 000 mikrožánrov, o ktoré sa diváci zaujímajú. Následne sa spoločnosť začala zaoberať aj tvorbou filmov a nebola už len

streamovacia platforma. Vďaka množstvu mikrožánrov, ktoré objavili, zistili medzeru na trhu a začali vytvárať filmy po ktorých bol dopyt. Ich stratégia začala byť riadená skoro čisto za pomoci dát. Dokonca aj rozsah farieb na titulnom snímku filmu bol vybraný tak aby pritiahol čo najviac divákov. [14]

Na spracovanie tejto masy údajov pôvodne používali databázy Oracle, ale prešli na NoSQL a Cassandra. Tieto umožňujú komplexnejšie analýzy založené na veľkých dátach a neštruktúrovaných údajov. Kurt Brown vo svojom prejave na konferencii Strata + Hadoop World, ktorý vedie tím Data Platform v Netflixe, vysvetlil, ako sa Netflix dátová platforma neustále vyvíja. Dátová infraštruktúra Netflix zahŕňa technológie veľkých dát ako Hadoop, Hive a Pig a ďalšie nástroje business intelligence ako Teradata a MicroStrategy. Zahŕňa aj vlastné open source aplikácie a služby Netflixu Lipstick and Genie. Rovnako ako celá základná infraštruktúra Netflixu všetko beží v cloude AWS. Netflix postupne prechádza na Spark pre streamovanie, strojové učenie a analytické prípady použitia a stále pokračuje vo vývoji nových doplnkov pre svoj vlastný open-source balík. [14]

1.4.2 Rolls-Royce

Rolls-Royce vyrába obrovské motory, ktoré používajú letecké spoločnosti a zložky ozbrojených síl po celom svete. Tieto motory generujú obrovské množstvo energie. Vzhľadom nato, že je to high-tech priemysel, spoločnosť si nemôže dovoliť chyby a zlyhania, pretože by ich to stálo bilióny dolárov alebo ešte horšie ľudské životy. Preto je dôležité, aby boli schopní monitorovať stav svojich produktov, aby zistili potenciálne problémy skôr, ako nastanú. Údaje, ktoré zbierajú im umožňujú vytvárať produkty mohutnejšie, efektívnejšie a poskytovať klientom lepšie služby. Z týchto obrovských dátových súborov robia za pomoci rôznych programov vizualizácie produktov a zisťujú či boli správne alebo nesprávne nadizajnované. Veľké dáta sa používajú aj na simuláciu zaťaženia produktov. Jednotlivé produkty sú v simulácii postavené extrémnym podmienkam. Takýmto spôsobom sa dajú zistiť potencionálne riziká alebo slabiny produktov. Ako sa však tieto dáta zbierajú? Rolls-Royce na zber dát využíva stovky senzorov, ktoré zaznamenávajú každý pohyb a o každej zmene informujú inžinierov, ktorí rozhodujú ako ďalej postupovať. [14]

Údaje operátorov sú prijímané vo forme bezdrôtových prenosov z lietadla (VHF rádio, SATCOM počas letu a 5G/Wi-Fi pri vzlete a odlete) obsahuje zmes správ

o výkonnosti motora. Tieto zvyčajne zahŕňajú zábery výkonu motora v kľúčových fázach letu, ako je vzlet, kde motor má maximálny výkon, stúpanie a let (ustálený stav). Ďalšie správy poskytujú podrobnosti o akýchkoľvek zaujímavých udalostiach počas letu, kde sú dostupné vysokofrekvenčné nahrávky pred a po udalosti. Správy o údržbe generované lietadlom, správy o pohybe (čas-pečiatky a miesta) a profily celého letu poskytujú ešte viac detailov. [14]

Spoločnosť tiež generuje obrovské množstvo údajov pri výrobnom procese. Napríklad v továrni v Singapure sa generuje pol terabajtu výrobných údajov o každej jednotlivej lopatke ventilátora. Vyrobí sa tu približne 6000 lopatiek ventilátora ročne, takže to sú tri petabajty údajov o výrobe iba jedného komponentu. [14]

1.4.3 LinkedIn

Konkurencia medzi sociálnymi sieťami je tvrdá a aby LinkedIn nezaostal za ostatnými konkurentmi, tak sa musí neustále vyvíjať. Jedným z najdôležitejších nástrojov, ktoré pomáhajú spoločnosti udržať sa na čele sú Big Data. Využívajú sa pri rozhodovaní a pomáhajú tak zabezpečiť čo najlepšie služby pre svojich používateľov. LinkedIn monitoruje každú interakciu používateľa so stránkou. Avšak pri 900 miliónoch členov z viac ako 200 krajín je to ohromné množstvo údajov za každý deň. [14]

Ako aj iné sociálne siete, LinkedIn používa dáta na vytváranie takzvaných návrhov. Navrhuje nám používateľom, ktorých možno poznáme na základe rovnakých kontaktov, alebo sme pracovali v ten istý čas v rovnakej firme. Na vytváranie čo najpresnejších návrhov využíva techniky zo strojového učenia. Na lepšiu predstavu uvedieme príklad : Pred piatimi rokmi sme pracovali vo firme x a dva roky dozadu vo firme y. Nikdy si neotvárate profily kolegov z firmy x, ale často komunikujete so zamestnancami firmy y a tak LinkedIn v návrhoch uprednostní ľudí zo spoločnosti y. Tento personalizovaný prístup umožňuje vytvárať siete, ktoré pre používateľov fungujú najlepšie. LinkedIn sleduje každý pohyb svojich používateľov na stránke, od všetkých vecí, ktoré sa vám páčia, alebo ste ich zdieľali cez ponuky práce na ktoré ste klikli, až po všetky kontakty, ktorým ste napísali. Spoločnosť obsluhuje stotisíce webových stránok každú sekundu každého dňa. Všetky tieto požiadavky zahŕňajú načítanie údajov z Backendových systémov LinkedIn, ktoré zase spracovávajú milióny dopytov za sekundu. S povolením LinkedIn tiež zhromažďuje údaje o emailových kontaktoch používateľov. [14]

LinkedIn využíva technológiu stream-processingu, aby zaistil čo najaktuálnejšie informácie. Od informácií kto dostal novú prácu až po napríklad zaujímavé články, príspevky, ktoré sa páčia vašim kontaktom alebo ich zdieľajú. Stručne povedané, stránka neustále zhromažďuje a zobrazuje nové údaje pre používateľov. Jadro big dát beží na hadoope. Tisícky strojov vykonávajú map/reduce programy. Ďalšie kľúčové časti zahŕňajú Oracle, Pig, Hive, Kafka, Java a MySQL. LinkedIn tiež vyvinul svoje vlastné open-source nástroje pre spracovanie a analýzu big dát.[14]

2 Ciele práce

Cieľom tejto práce je demonštrácia spracovania veľkých dát za pomoci jazyka Python. Ako prvé si vyberieme vhodný dátový set a implementujeme techniky čistenia a predbežného spracovania údajov. Aby sme to pri takto veľkom dátovom sete zvládli, použijeme niektoré z nástrojov na spracovanie veľkých dát. Týmito nástrojmi sú:

- Apache Hadoop
- Apache Spark
- Visual Studio Code
- Jupyter Notebook
- Docker
- Hortonworks sandbox

Po získaní údajov, s ktorými budeme pracovať, prejdeme na druhý krok. Druhým krokom pre nás bude vytváranie prehľadov, optimalizácia, vizualizácie údajov a vykazovania pomocou jazyka Python, ako je použitie knižníc ako Matplotlib a Seaborn na vytváranie informatívnych a vizuálne príťažlivých grafov a tabuliek.

3 Metódy a nástroje

Pochopenie a efektívne využívanie metód a nástrojov pre spracovanie veľkých dát sú kľúčové pre organizácie. Umožňujú im získavanie cenných poznatkov a prijímanie správnych rozhodnutí. Na riešenie výziev veľkých dát boli vyvinuté rôzne metódy a nástroje, ktoré umožňujú efektívne spracovanie rozsiahlych súborov. V tejto kapitole preskúmame niektoré kľúčové metódy a nástroje používané pri spracovaní veľkých dát. Týmito nástrojmi sú MapReduce, Spark, Hadoop, NoSQL, virtuálne stroje a kontajnerové platformy.

3.1 NoSQL databázy

NoSQL je typ systému správy databáz (DBMS), ktorý je navrhnutý na spracovanie a ukladanie veľkých objemov neštruktúrovaných a pološtruktúrovaných údajov. Na rozdiel od tradičných relačných databáz, ktoré na ukladanie údajov používajú tabuľky s preddefinovanými schémami, databázy NoSQL používajú flexibilné dátové modely, ktoré sa dokážu prispôsobiť zmenám v dátových štruktúrach a sú schopné horizontálneho škálovania, aby zvládli rastúce množstvo údajov. Databázy NoSQL, známe aj ako „nielen SQL“ databázy, sú novým typom systému správy databáz, ktorý si v posledných rokoch získal na popularite. Na rozdiel od tradičných relačných databáz sú databázy NoSQL navrhnuté na spracovanie veľkého množstva neštruktúrovaných alebo pološtruktúrovaných údajov a môžu sa prispôsobiť dynamickým zmenám dátového modelu. Vďaka tomu sú databázy NoSQL vhodné pre moderné webové aplikácie, analýzy v reálnom čase a spracovanie veľkých dát. Tu je niekoľko bežných prípadov použitia databáz NoSQL v scenároch veľkých dát: [22], [23]

- **Škálovateľnosť** - databázy NoSQL sú navrhnuté tak, aby sa škálovali horizontálne, čo znamená, že dokážu spracovať veľké množstvo údajov a vysoké zaťaženie pri zápise a čítaní. Vďaka tomu sú vhodné pre veľké dátové aplikácie, kde sú objemy údajov obrovské a je potrebné ich spracovávať paralelne vo viacerých uzloch.
- **Flexibilita** - NoSQL databázy sú bez schém, čo znamená, že nevynucujú pevnú dátovú štruktúru. Vďaka tomu sú ideálne na prácu s neštruktúrovanými alebo pološtruktúrovanými údajmi, ktoré sa bežne vyskytujú v aplikáciách veľkých údajov, ako sú údaje zo sociálnych médií, údaje zo senzorov alebo protokolové údaje, ktoré nemusia presne zapadnúť do tradičnej relačnej databázy.

- **Výkon -** databázy NoSQL sú optimalizované na výkon a môžu poskytovať prístup k údajom s nízkou latenciou, čo je rozhodujúce pre spracovanie veľkých údajov v reálnom čase alebo takmer v reálnom čase, ako sú aplikácie na streamovanie údajov alebo analýzy v reálnom čase.
- **Distribuované výpočty -** mnohé databázy NoSQL sú navrhnuté tak, aby fungovali v distribuovaných alebo klastrových prostrediach, čo im umožňuje ukladať a spracovávať údaje vo viacerých uzloch alebo klastroch. To umožňuje distribuované výpočty a môže zlepšiť odolnosť voči chybám, redundanciu údajov a lokalizáciu údajov, čo sú dôležité úvahy pri spracovaní veľkých údajov.
- **Integrácia údajov** - databázy NoSQL sa dajú ľahko integrovať s inými technológiami veľkých údajov, ako sú Hadoop, Spark alebo iné rámce na spracovanie údajov, čo umožňuje bezproblémové prijímanie údajov, spracovanie a analýzy v pracovných tokoch veľkých údajov.
- **Cenovo výhodné** - databázy NoSQL sú často open source a môžu byť nákladovo efektívnejšie ako tradičné relačné databázy, pokiaľ ide o licenčné poplatky a hardvérové požiadavky. Môže to byť atraktívna možnosť pre aplikácie s veľkými dátami, kde sa berie do úvahy optimalizácia nákladov. [24]

Existuje niekoľko populárnych databáz NoSQL, ktoré sa bežne používajú vo veľkých dátových aplikáciách. Niektoré z najpoužívanejších databáz NoSQL zahŕňajú:

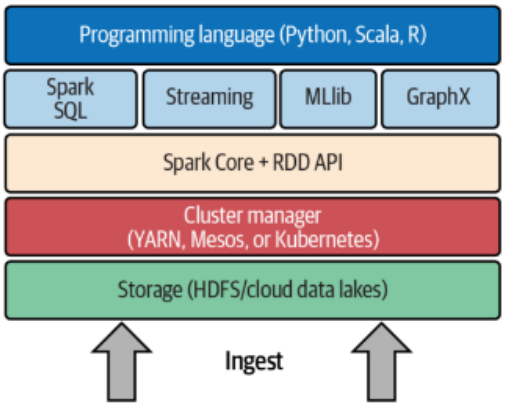
- **MongoDB** - je populárna databáza NoSQL s otvoreným zdrojovým kódom, ktorá ukladá údaje vo flexibilných dokumentoch podobných JSON. Je známy svojim vysokým výkonom, škálovateľnosťou a jednoduchým používaním a bežne sa používa na spracovanie veľkých objemov neštruktúrovaných alebo pološtruktúrovaných údajov, ako sú údaje zo sociálnych médií, údaje zo senzorov a protokolové údaje. [28]
- **Cassandra** - je vysoko škálovateľná a distribuovaná databáza NoSQL určená na spracovanie veľkého množstva údajov vo viacerých klastroch alebo uzloch. Je známy svojou vysokou dostupnosťou, odolnosťou voči chybám a lineárnou škálovateľnosťou a bežne sa používa v aplikáciách veľkých dát, ktoré vyžadujú vysokú priepustnosť zápisu a čítania, ako sú napríklad časové série údajov, analýzy a systémy odporúčaní. [29]
- **Redis** - je úložisko údajov v pamäti, ktoré sa bežne používa ako databáza NoSQL na ukladanie do vyrovnávacej pamäte, analýzy v reálnom čase, zasielanie správ a ďalšie

aplikácie náročné na údaje. Je známy svojim vysokým výkonom, nízkou latenciou prístupu a podporou rôznych dátových štruktúr, ako sú kľúč – hodnota, zoznamy, množiny a ďalšie. [30]

- **HBase** - je open source, distribuovaná, stĺpcová NoSQL databáza, ktorá je postavená na Hadoop a je navrhnutá na prácu s rozsiahlymi, riedkymi súbormi údajov. Bežne sa používa vo veľkých dátových aplikáciách na ukladanie a spracovanie obrovského množstva dát s vysokou priepustnosťou zápisu a čítania, ako sú protokolové dáta, dáta časových sérií a dáta sociálnych médií. [31]
- **Amazon DynamoDB** - je riadená databázová služba NoSQL ponúkaná spoločnosťou Amazon Web Services (AWS), ktorá je známa svojou škálovateľnosťou, výkonom s nízkou latenciou a jednoduchým používaním. Bežne sa používa vo veľkých dátových aplikáciách, ktoré vyžadujú vysokú dostupnosť, odolnosť a bezproblémovú integráciu s inými službami AWS. [32]
- **Couchbase** - je distribuovaná databáza NoSQL, ktorá kombinuje funkcie obchodov orientovaných na dokumenty a obchodov s hodnotou kľúča. Je známy svojou škálovateľnosťou, výkonom a podporou mobilných a IoT aplikácií a bežne sa používa vo veľkých dátových aplikáciách, ktoré vyžadujú synchronizáciu dát v reálnom čase a bezproblémovú integráciu s mobilnými a webovými aplikáciami. [33]

3.2 Apache Spark

Apache Spark je oproti konkurenčným nástrojom najrýchlejší, open-source nástroj na spracovanie údajov pre strojové učenie a aplikácie AI, podporovaný najväčšou open-source komunitou v oblasti veľkých dát. Je navrhnutý tak, aby poskytoval výpočtovú rýchlosť, škálovateľnosť a programovateľnosť potrebnú pre veľké dáta – konkrétne pre streamovanie

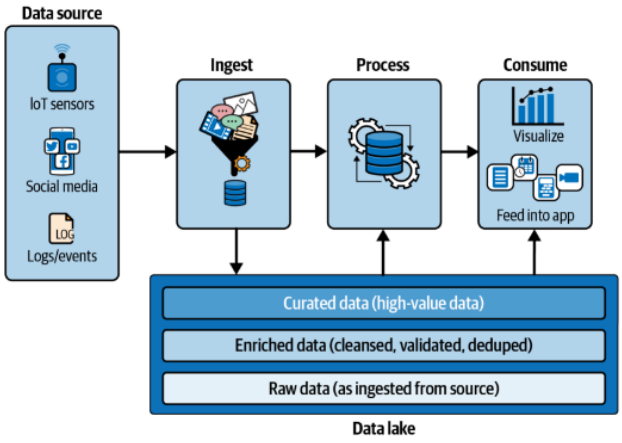


Obrázok 1 Štruktúra Apache Spark; zdroj: [10]

dát, grafové dáta, strojové učenie a aplikácie umelej inteligencie (AI) . Analytický nástroj Spark spracováva údaje 10 až 100-krát rýchlejšie ako alternatívy. Škáluje sa distribúciou spracovateľskej práce medzi veľké skupiny počítačov so zabudovaným paralelizmom a odolnosťou voči chybám. Zahŕňa dokonca rozhrania API pre programovacie jazyky, ktoré sú obľúbené medzi analytikmi údajov a vedcami údajov, vrátane Scala, Java, Python a R. [11]

Primárnou abstrakciou Sparku je odolný distribuovaný súbor údajov (RDD), ktorý predstavuje kolekciu údajov odolnú voči chybám, ktoré je možné spracovávať paralelne naprieč klastrom strojov. Spark tiež poskytuje abstrakcie vyššej úrovne, ako sú DataFrames a Datasets, ktoré sú optimalizované na spracovanie štruktúrovaných údajov a ponúkajú výraznejšie a intuitívnejšie API ako RDD. [10]

„Na rozdiel od dvojfázového procesu vykonávania v MapReduce, Spark vytvára riadený acyklický graf (DAG) na plánovanie úloh a orchestráciu pracovných uzlov v klastri. Keďže Spark pracuje a transformuje údaje v procesoch vykonávania úloh, plánovač DAG uľahčuje efektivitu organizovaním pracovných uzlov v klastri. [11]



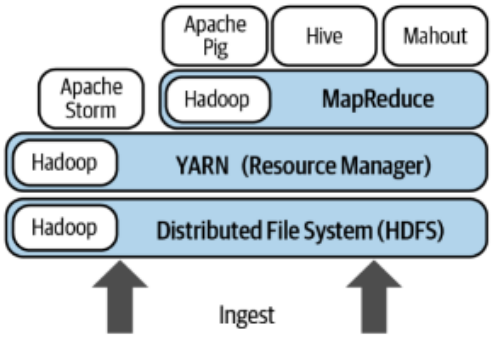
Obrázok 2 Real time processing; zdroj: [10]

Vďaka Sparku môžeme vykonávať spracovanie toku dát v reálnom čase. Tento nástroj odkazuje na špeciálne spracovanie dát zamerané na rýchlosť. Táto rýchlosť je tak veľká, že spracovanie dát sa môže zdať akoby sa dialo v reálnom čase. Napríklad si predstavme, že sme v nákupnom centre a hneď ako zapneme internet začnú nám vyskakovať reklamy z obchodov, v ktorých sme boli. Aj toto je spracovanie dát v reálnom čase. Naše mobilné zariadenie vyslalo signál o polohe a personalizované reklamy sa zmenili. Na

obrázku 1 si môžeme pozrieť ako sa dáta zo senzorov alebo sociálnych sietí spracujú a objavia sa v inej forme či už v aplikáciách alebo na webových stránkach. [10]

3.3 Apache Hadoop

„Projekt Apache™ Hadoop® vyvíja softvér s otvoreným zdrojovým kódom pre spoľahlivé, škálovateľné a distribuované výpočty. Softvérová knižnica Apache Hadoop je rámec, ktorý umožňuje distribuované spracovanie veľkých súborov údajov naprieč klastrami počítačov pomocou jednoduchých programovacích modelov. Je navrhnutý tak, aby sa škáloval od jednotlivých serverov až po tisíce strojov, z ktorých každý ponúka lokálny výpočet a úložisko. Namiesto toho, aby sa pri poskytovaní vysokej dostupnosti spoliehal na hardvér, samotná knižnica je navrhnutá tak, aby zisťovala a riešila zlyhania na aplikačnej vrstve, čím poskytuje vysoko dostupnú službu na vrchole klastra počítačov, z ktorých každý môže byť náchylný na poruchy.“ [8]



Obrázok 3 Štruktúra Apache Hadoop; zdroj: [8]

Projekt obsahuje tieto moduly:

- **Hadoop Common -** balík Hadoop Common sa považuje za základ/jadro rámca, pretože poskytuje základné služby a základné procesy, ako je abstrakcia základného operačného systému a jeho súborového systému. Hadoop Common obsahuje aj potrebné súbory Java Archive (JAR) a skripty potrebné na spustenie Hadoopu. Balík Hadoop Common tiež poskytuje zdrojový kód a dokumentáciu, ako aj sekciu príspevkov, ktorá obsahuje rôzne projekty z komunity. [8]
- **Hadoop Distributed File System (HDFS™) -** HDFS (Hadoop Distributed File System) je primárny úložný systém používaný aplikáciami Hadoop. Tento open source framework funguje tak, že rýchlo prenáša dáta medzi uzlami. Často ho využívajú spoločnosti, ktoré potrebujú spracovávať a uchovávať veľké dáta. HDFS

je kľúčovou súčasťou mnohých systémov Hadoop, pretože poskytuje prostriedky na správu veľkých dát, ako aj na podporu analýzy veľkých dát. [8]

- **Hadoop YARN -** základnou myšlienkou YARN je rozdeliť funkcie správy zdrojov a plánovania/monitorovania úloh do samostatných démonov. Myšlienkou je mať globálny ResourceManager (RM) a aplikáciu ApplicationMaster (AM). Aplikácia je buď jedna úloha, alebo viac úloh. [8]
- **Hadoop MapReduce -** MapReduce je programovací model a motor na spracovanie veľkých dát, ktorý sa používa na paralelné spracovanie veľkých súborov údajov. Pôvodne bol MapReduce jediným spúšťacím nástrojom dostupným v Hadoop. Neskôr však Hadoop pridal podporu pre ostatných, vrátane Apache Tez a Apache Spark . [25]

Kde sa Hadoop využíva?

- **Maloobchod** - keď britský maloobchodník M&S nasadil Cloudera Enterprise poháňaný Hadoopom, boli výsledkami veľmi prekvapení. Cloudera využíva podporu a služby na báze Hadoop na správu a spracovanie údajov. Krátko po implementácii cloudovej platformy spoločnosť M&S zistila, že dokáže úspešne využiť svoje údaje na oveľa lepšiu prediktívnu analýzu. To ich viedlo k efektívnejšiemu využívaniu skladov a predchádzalo vypredaniu zásob počas „neočakávaných“ špičiek dopytu a získali obrovskú výhodu oproti konkurencii. [25]
- **Financie** - Hadoop je možno vhodnejší pre finančný sektor ako ktorýkoľvek iný. Na začiatku bol softvérový rámec rýchlo upevnený na primárne použitie pri práci s pokročilými algoritmami spojenými s modelovaním rizík. Dnes je bežné, že finančné inštitúcie implementujú Hadoop, aby lepšie spravovali finančné zabezpečenie a výkonnosť aktív svojich klientov. JPMorgan Chase je len jedným z mnohých priemyselných gigantov, ktorí používajú Hadoop na správu exponenciálne rastúceho množstva údajov o zákazníkoch z celého sveta. [25]
- **Zdravotná starostlivosť** – či už súkromní alebo štátni poskytovatelia zdravotnej starostlivosti akejkoľvek veľkosti narábajú s obrovskými objemami údajov a informácií o zákazníkoch. Rámce Hadoop umožňujú lekárom, zdravotným sestrám a opatrovateľom ľahký prístup k informáciám, ktoré potrebujú a tiež uľahčuje agregáciu údajov, ktoré poskytujú užitočné informácie. To sa môže týkať záležitostí verejného zdravia, lepšej diagnostiky, zlepšenej liečby a ďalších. [25]

3.4 Jupyter Notebook

Jupyter Notebook je webová aplikácia s voľne prístupným zdrojovým kódom, ktorá poskytuje interaktívne výpočtové prostredie. Vytvára notebooky, ktoré kombinujú zdrojový kód a výstupy do jedného súboru. Tento notebook obsahuje vizualizácie, matematické rovnice, štatistické modelovanie, naratívny text a akékoľvek iné multimediálne údaje. [7]

Tento prístup s jedným dokumentom umožňuje používateľom vyvíjať, vizualizovať výsledky a pridávať informácie, grafy a taktiež vzorce, vďaka ktorým je práca zrozumiteľnejšia, opakovateľnejšia a zdieľateľnejšia. Notebooky Jupyter podporujú viac ako 40 programovacích jazykov s hlavným zameraním na Python . Keďže ide o bezplatný nástroj s otvoreným zdrojovým kódom, ktokoľvek ho môže voľne používať pre svoje projekty vedy o údajoch. Existujú dva varianty notebooku: [7]

Jupyter Notebook je originálna webová aplikácia na vytváranie a zdieľanie výpočtových dokumentov. Ponúka jednoduché, efektívne prostredie zamerané na dokumenty.

Dáva programátorom možnosť precvičiť si svoj program v prostredí, ktoré výrazne uľahčuje úpravu a úpravu konkrétnych častí kódu. Konkrétne, Jupyter Notebook dáva užívateľom možnosť spustiť časť kódu bez spustenia celého programu, aby sa zistilo, či funguje pred písaním ďalšieho riadku kódu. Tým, že umožňuje používateľom spúšťať kód krok za krokom alebo riadok po riadku, Jupyter Notebook poskytuje vynikajúci priestor na precvičenie práce s programom pred jeho odoslaním alebo zdieľaním s ostatnými. [8]

JupyterLab - jeho flexibilné rozhranie umožňuje používateľom nastaviť a zotriediť pracovné postupy v oblasti vedy o údajoch, vedeckej výpočtovej techniky, výpočtovej žurnalistiky a strojového učenia. Modulárny dizajn pozýva rozšírenia na zlepšenie a obohatenie funkčnosti. Na rozdiel od Notebooku, JupyterLab umožňuje prehliadanie csv súborov vo formáte rovnako tabuľky rovnako ako v Exceli. Dokonca umožňuje otvoriť veľké csv súbory s niekoľko miliónmi riadkov. [8]

3.5 Kontajnerové platformy

Kontajnerizácia je proces nasadenia softvéru, ktorý spája kód aplikácie so všetkými súbormi a knižnicami, ktoré potrebuje na spustenie v akejkoľvek infraštruktúre. Ak chceme na svojom počítači spustiť akúkoľvek aplikáciu, tradične by sme museli nainštalovať verziu, ktorá sa zhoduje s operačným systémom vášho počítača. Napríklad potrebujeme

nainštalovať verziu softvérového balíka pre systém Windows na počítači so systémom Windows. Pomocou kontajnerizácie však môžeme vytvoriť jeden softvérový balík alebo kontajner, ktorý beží na všetkých typoch zariadení a operačných systémov. [18]

Motor kontajnera je softvérový program, ktorý vytvára kontajnery na základe obrázkov kontajnerov. Funguje ako sprostredkovateľ medzi kontajnermi a operačným systémom, ktorý poskytuje a spravuje zdroje, ktoré aplikácia potrebuje. Napríklad kontajnerové motory môžu spravovať viacero kontajnerov v rovnakom operačnom systéme tak, že ich udržia nezávislé od základnej infraštruktúry a od seba navzájom. Obrázky kontajnerov sa stávajú kontajnermi za behu a v prípade kontajnerov Docker sa obrázky stávajú kontajnermi, keď bežia na Docker Engine. [21]

- **Docker** - alebo Docker Engine je populárny open-source kontajnerový softvér, ktorý umožňuje vývojárom softvéru vytvárať, nasadzovať a testovať kontajnerizované aplikácie na rôznych platformách. Kontajnery Docker sú samostatné balíky aplikácií a súvisiacich súborov, ktoré sú vytvorené pomocou rámca Docker. Dockerhub a Quay.io sú úložiská ponúkajúce obrázky pre váš kontajnerový engine podľa vlastného výberu. [19]
- **Linux** - je open-source operačný systém so vstavanou kontajnerovou technológiou. Kontajner Linux je samostatné prostredie, ktoré umožňuje spustenie viacerých aplikácií založených na systéme Linux na jednom hostiteľskom počítači. Vývojári softvéru používajú kontajnery Linux na nasadenie aplikácií, ktoré zapisujú alebo čítajú veľké množstvo údajov. Linuxové kontajnery neskopírujú celý operačný systém do svojho virtualizovaného prostredia. Ich cieľom je ponúknuť distribučné a dodávateľsky neutrálne prostredie pre vývoj kontajnerových technológií Linuxu. [20]
- **Kubernetes** - je populárny open-source kontajnerový orchestrátor, ktorý vývojári softvéru používajú na nasadenie, škálovanie a správu obrovského množstva mikroslužieb. Má deklaratívny model, ktorý uľahčuje automatizáciu kontajnerov. Deklaratívny model zabezpečuje, že Kubernetes podnikne príslušné kroky na splnenie požiadaviek na základe konfiguračných súborov. Umožňuje zoskupiť skupiny hostiteľov s kontajnermi Linux® a Kubernetes pomôže tieto klastre jednoducho a efektívne spravovať. [21]

3.6 Virtuálne stroje

Virtuálny stroj (VM) je virtuálna reprezentácia alebo emulácia fyzického počítača. Často sa označujú ako hosť, zatiaľ čo fyzický stroj, na ktorom bežia, sa označuje ako hostiteľ. Virtualizácia umožňuje vytvoriť viacero virtuálnych strojov, každý s vlastným operačným systémom (OS) a aplikáciami, na jednom fyzickom stroji. VM nemôže priamo interagovať s fyzickým počítačom. Namiesto toho potrebuje ľahkú softvérovú vrstvu nazývanú hypervízor na koordináciu medzi ním a základným fyzickým hardvérom. Hypervízor prideľuje fyzické výpočtové zdroje – ako sú procesory, pamäť a úložisko – každému virtuálnemu stroju. Udržuje každý VM oddelený od ostatných, takže sa navzájom nerušia. [16]

Technológia virtuálnych strojov sa často využíva v lokálnych a cloudových prostrediach. Verejné cloudové služby v poslednej dobe využívajú virtuálne stroje na poskytovanie virtuálnych aplikačných zdrojov viacerým používateľom naraz, čo umožňuje ešte efektívnejšie a flexibilnejšie výpočty. Virtuálne stroje umožňujú firme spustiť operačný systém, ktorý sa v okne aplikácie na pracovnej ploche správa ako úplne samostatný počítač. Virtuálne počítače môžu byť nasadené tak, aby vyhovovali rôznym úrovniam potrieb výpočtového výkonu, spúšťali softvér, ktorý vyžaduje iný operačný systém, alebo testovali aplikácie v bezpečnom prostredí v izolovanom priestore. Okrem toho môžu virtuálne stroje vykonávať špecifické úlohy považované za príliš riskantné na vykonávanie v hostiteľskom prostredí, ako je napríklad prístup k údajom infikovaným vírusom alebo testovanie operačných systémov. Keďže virtuálny stroj je oddelený od zvyšku systému, softvér vo virtuálnom stroji nemôže zasahovať do hostiteľského počítača. [15]

Typy virtualizácií:

- **Virtualizácia aplikácií** - virtualizácia aplikácií pomáha používateľovi získať vzdialený prístup k aplikácii zo servera. Server ukladá všetky osobné informácie a ďalšie charakteristiky aplikácie, ale stále môže bežať na lokálnej pracovnej stanici cez internet. Príkladom môže byť používateľ, ktorý potrebuje spustiť dve rôzne verzie toho istého softvéru. Technológie, ktoré využívajú virtualizáciu aplikácií, sú hostované aplikácie a zabalené aplikácie. [17]
- **Virtualizácia siete** - na rovnakej fyzickej sieti je možné vytvoriť viacero podsietí spojením zariadení do jedného, softvérovo založeného virtuálneho sieťového prostriedku. Medzi výhody patrí zvýšená spoľahlivosť, rýchlosť siete, bezpečnosť a

lepšie monitorovanie využitia dát. Virtualizácia siete môže byť dobrou voľbou pre spoločnosti s veľkým objemom používateľov, ktorí potrebujú nepretržitý prístup. [15]

- **Virtualizácia desktopu** - tento bežný typ virtualizácie oddeľuje desktopové prostredie od fyzického zariadenia a ukladá desktop na vzdialenom serveri, čo umožňuje užívateľom pristupovať k svojim desktopom odkiaľkoľvek na akomkoľvek zariadení. Okrem ľahkej dostupnosti patrí medzi výhody virtuálnych desktopov lepšie zabezpečenie dát, úspora nákladov na softvérové licencie a aktualizácie a jednoduchá správa. [15]
- **Virtualizácia úložiska** - virtualizácia úložiska je pole serverov, ktoré sú riadené systémom virtuálneho úložiska. Umožňuje spravovať úložisko z viacerých zdrojov a využívať ho ako jediné úložisko. Softvér na virtualizáciu úložísk zachováva plynulé operácie, konzistentný výkon a nepretržitú sadu pokročilých funkcií napriek zmenám, poruchám a rozdielom v základnom vybavení. [17]
- **Virtualizácia servera -** ide o druh virtualizácie, pri ktorej dochádza k maskovaniu serverových zdrojov. Tu je centrálny server (fyzický server) rozdelený na viacero rôznych virtuálnych serverov zmenou identifikačného čísla a procesorov. Takže každý systém môže prevádzkovať svoje operačné systémy izolovaným spôsobom. [17]

Hypervízor virtuálneho stroja, známy aj ako virtualizačný manažér alebo monitor virtuálneho stroja (VMM), je softvér, ktorý umožňuje súčasné spustenie viacerých operačných systémov na jednom hostiteľskom stroji. Hypervízor vytvára virtuálne stroje (VM), ktoré sa správajú ako nezávislé a izolované stroje, z ktorých každý má svoj vlastný CPU, pamäť, úložisko a sieťové zdroje. [34]

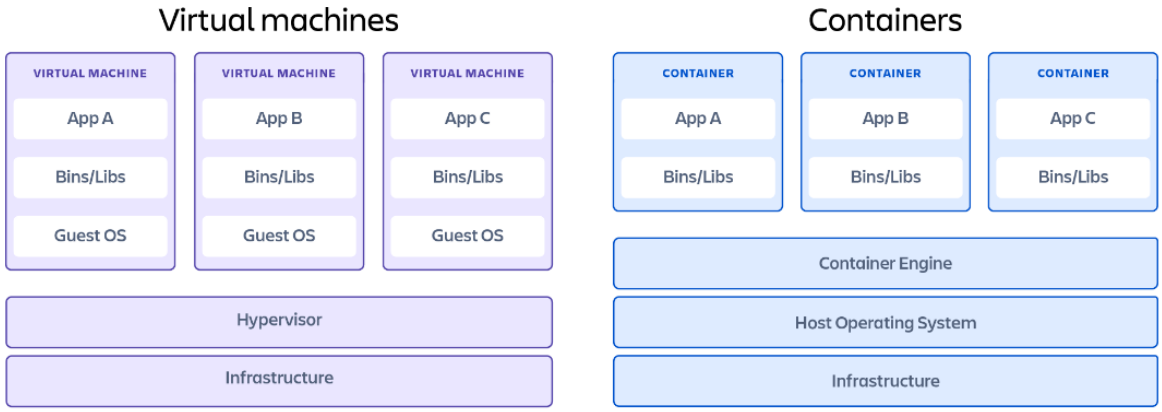
Existujú dva typy hypervízorov:

- **Hypervízory typu 1** - známe aj ako holé hypervízory, bežia priamo na hardvéri hostiteľa a priamo riadia hardvérové prostriedky, aby hosťujúcim operačným systémom poskytovali možnosti virtualizácie. Príklady hypervízorov typu 1 zahŕňajú VMware ESXi, Microsoft Hyper-V a Citrix XenServer.
- **Hypervízory typu 2** - známe aj ako hostované hypervízory, bežia nad existujúcim operačným systémom a poskytujú hosťujúcim operačným systémom možnosti

virtualizácie. Príklady hypervízorov typu 2 zahŕňajú Oracle VirtualBox, VMware Workstation a Parallels Desktop. [34]

3.7 Virtuálne stroje vs kontajnerové platformy

Ako prvé treba povedať, že virtuálne stroje aj kontajnery sú veľmi silné a výkonné nástroje. Obidve poskytujú izolované prostredia na bezpečný chod procesov, líšia sa však svojimi špecifickými účelmi. Kontajnerová aplikácia má priamejší prístup k hardvéru ako aplikácia spustená vo virtuálnom počítači, vďaka čomu sú kontajnery vhodné pre ľahké prípady použitia.



Obrázok 4 Porovnanie VM a Containers; zdroj:[35]

Plusy kontajnerov:

- **Rýchlosť iterácie** - keďže kontajnery sú ľahké a obsahujú iba softvér vysokej úrovne, veľmi rýchlo sa upravujú a opakujú.
- **Robustný ekosystém** - väčšina prevádzkových systémov kontajnerov ponúka hosťované verejné úložisko vopred vyrobených kontajnerov. Tieto kontajnerové úložiská obsahujú mnoho populárnych softvérových aplikácií, ako sú databázy alebo systémy na odosielanie správ, a možno ich okamžite stiahnuť a spustiť, čo šetrí čas vývojovým tímom. [16]

Mínusy kontajnerov

- **Zdieľané hostiteľské exploity** - všetky kontajnery zdieľajú rovnaký základný hardvérový systém pod vrstvou operačného systému, je možné, že exploit v jednom kontajneri by sa mohol vymaniť z kontajnera a ovplyvniť zdieľaný hardvér.

Najpopulárnejšie kontajnery majú verejné úložiská vopred vytvorených kontajnerov. Pri používaní jedného z týchto verejných obrázkov existuje bezpečnostné riziko, pretože môžu obsahovať exploity alebo môžu byť náchylné na napadnutie. [16]

Plusy VM

- **Zabezpečenie úplnej izolácie** - virtuálne počítače bežia izolovane ako úplne samostatný systém. To znamená, že virtuálne stroje sú imúnne voči akémukoľvek zneužitiu alebo rušeniu zo strany iných virtuálnych strojov na zdieľanom hostiteľovi. Jednotlivý virtuálny stroj môže byť stále unesený exploitom, ale zneužitý virtuálny stroj bude izolovaný a nebude môcť kontaminovať žiadne iné susedné virtuálne stroje. [36]
- **Interaktívny vývoj** - kontajnery sú zvyčajne statické definície očakávaných závislostí a konfigurácie potrebnej na spustenie kontajnera. Virtuálne stroje sú dynamickejšie a možno ich interaktívne rozvíjať. Po zadaní základnej hardvérovej definície pre virtuálny stroj sa s virtuálnym strojom môže zaobchádzať ako s obyčajným počítačom. Softvér je možné manuálne nainštalovať do virtuálneho stroja a virtuálny stroj možno zosnímať na zachytenie aktuálneho stavu konfigurácie. Snímky virtuálneho stroja možno použiť na obnovenie virtuálneho stroja do daného bodu v čase alebo na spustenie ďalších virtuálnych strojov s touto konfiguráciou. [36]

Zápory VM

- **Rýchlosť iterácie** - virtuálne stroje sú časovo náročné na zostavenie a regeneráciu, pretože zahŕňajú systém plného zásobníka. Akékoľvek úpravy snímky virtuálneho počítača môžu trvať dlho, kým sa regenerujú a overia, či sa správajú podľa očakávania. [17]
- **Náklady na veľkosť úložiska** - virtuálne stroje môžu zaberať veľa úložného priestoru. Môžu rýchlo narásť na veľkosť niekoľkých gigabajtov. To môže viesť k problémom s nedostatkom miesta na disku na hostiteľskom počítači virtuálnych strojov. [17]

VM	Kontajnery
Zaberá veľa pamäte	Zaberá málo pamäte
Obmedzený výkon	Výkon podľa potreby
Každý VM beží vo svojom vlastnom OS	Všetky kontajnery zdieľajú hostiteľský OS
Virtualizácia na hardvérovej úrovni	Virtualizácia OS
Čas spustenia v minútach	Čas spustenia v milisekundách
Prideľuje požadovanú pamäť	Vyžaduje menej miesta v pamäti
Úplne izolované, a teda bezpečnejšie	Izolácia procesu, menej bezpečná

Tabuľka 2 Porovnanie VM a kontajnerov; zdroj: vlastné spracovanie

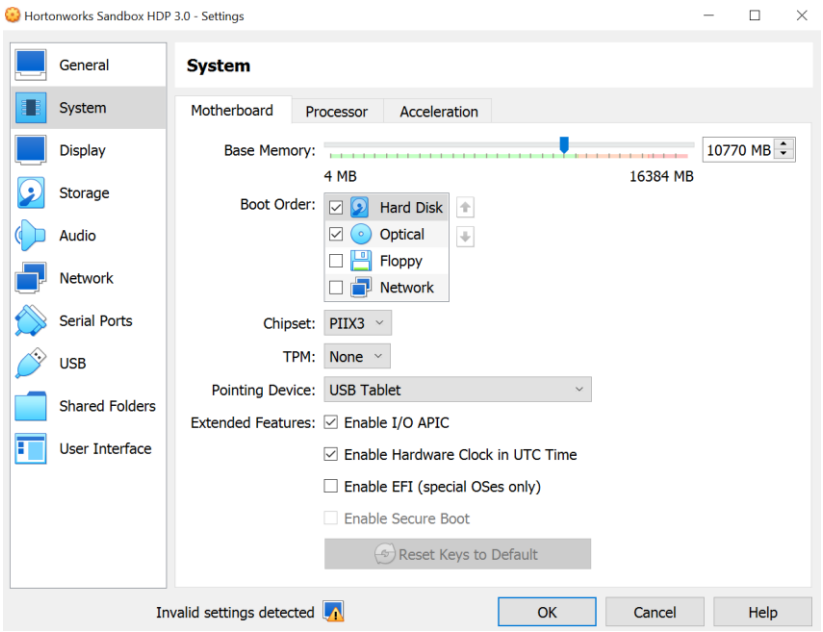
Napriek svojej popularite kontajnery úplne nenahradili VM. V mnohých prípadoch kontajnery dopĺňajú používanie VM. Ak chcete otestovať aplikáciu, ktorá môže ohroziť celý váš operačný systém alebo potrebujete zdieľať hardvér medzi službami spustenými na rôznych operačných systémoch, potrebujete VM. Pretože všetky vaše kontajnery na danom počítači zdieľajú rovnaké jadro, je pre škodlivý kód jednoduchšie kompromitovať celý počítač. Ak by ste si aj napriek všetkým uvedeným informáciách nevedeli vybrať, ktoré z týchto platforiem sú lepšie, môžete sa pozrieť do tabuľky č. 2 a rozhodnúť sa na základe vlastností, ktoré hľadáte. [14]

4 Výsledky práce

Výsledkom práce je preskúmanie možností spracovania veľkých dát v jazyku Python. Výstupom sú dva programy. Prvý program je mapreduce program v Hadoope. Tento program je napísaný v jazyku Python a púšťame ho na virtuálnom stroji Hortonworks sandbox. Pomôže nám upraviť veľký dátový súbor do takej podoby, v ktorej sa bude dať vykonávať dátová analýza. Následne si na upravenom súbore ukážeme grafy pomocou knižníc v jazyku Python. Druhý program je tiež napísaný v jazyku Python. Tento program využíva distribuovaný systém Apache Spark. Púšťame ho v kontajnerovej platforme s názvom Docker. Umožní nám rýchle vyhľadávanie modelov áut podľa zadaného názvu.

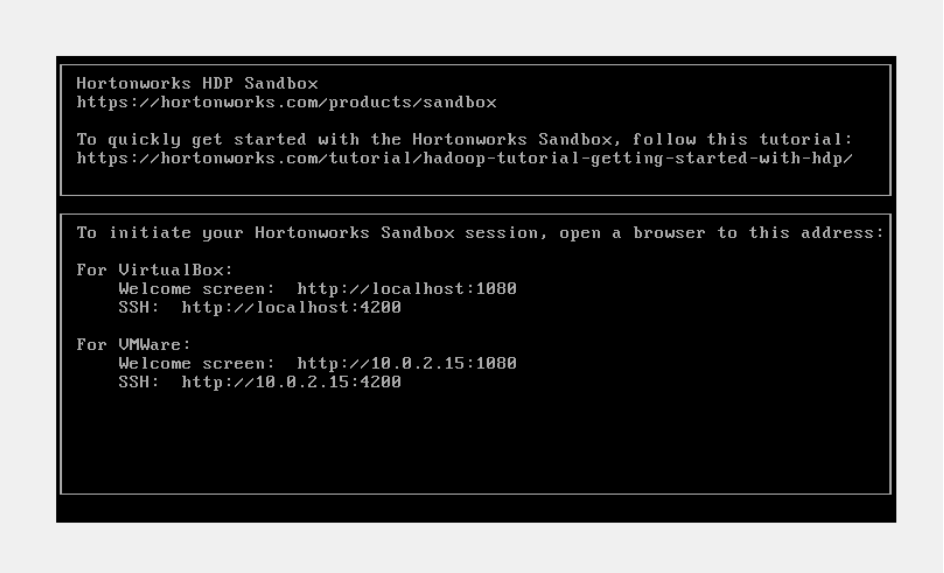
4.1 Inštalácia a nastavenie Hortonworks sandbox

- Stiahneme si Hortonworks sandbox z Cloudery https://www.cloudera.com/downloads.html (treba si vybrať verziu virtualbox)
- Ak nemáme stiahnutý virtualbox od Oracle je potrebné ho stiahnuť a nainštalovať https://www.virtualbox.org/
- Hortonworks spustíme tak, že ho vložíme do ikonky virtualboxu a ten ho otvorí
- Default nastavenia bývajú dobré, ale odporúča sa programu dať aspoň 8GB pamäte a 4 jadrá, neodporúča sa mu dať toľko pamäti, aby na ukazovateli bol v červenom poli.



Obrázok 5 Systémové nastavenie; zdroj: vlastné spracovanie

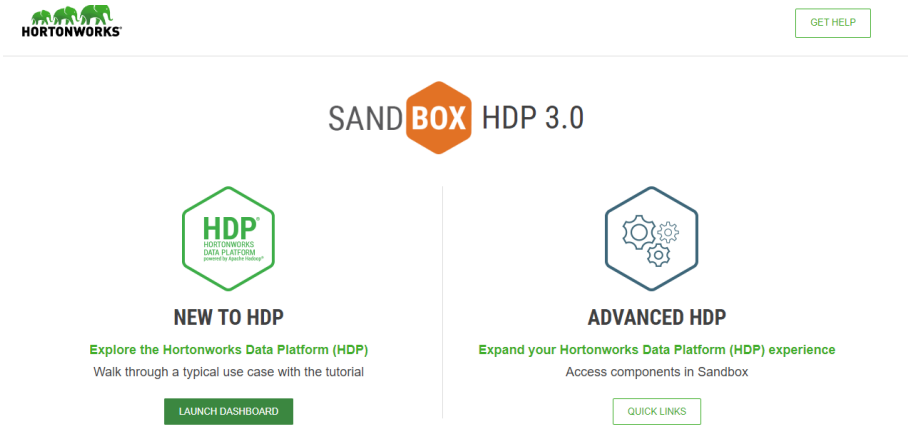
Program spustíme tlačidlom štart alebo dvojklikom. Prvýkrát to môže trvať dlhšie. Zobrazí sa nám okno, ktoré môžeme vidieť na obrázku 5. Toto okno nás odkazuje na localhost:1080



Obrázok 6 Domovská obrazovka; zdroj: vlastné spracovanie

Nastavenie Hortonworks sandbox:

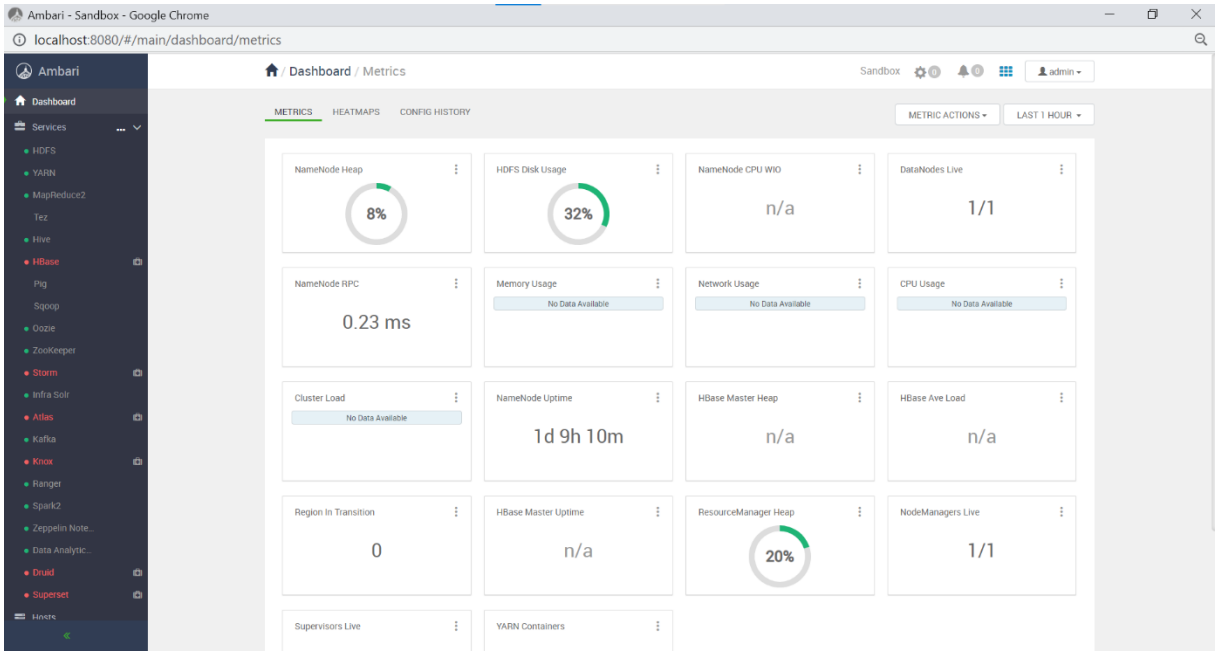
- Na localhoste:1080 si vyberieme launch dashboard



Obrázok 7 Sandbox dashboard; zdroj: vlastné spracovanie

- Prihlasovacie údaje sú prednastavené maria_dev heslo maria_dev, alebo raj_ops a heslo raj_ops

- Na prácu s hortonworks môžeme využiť linuxovú konzolu na localhost: 4200
- Prihlásime sa ako root a heslo hadoop v prípade, ak sme už nastavovali SSH kanál tak heslo, ktoré máme. Aby sme sa dostali k oprávneniam admina zadáme príkaz: Ambari-admin-password-reset. Obrázok 7 zobrazuje ako vyzerá okno programu ak sa nám náš virtuálny stroj podarí úspešne zapnúť.



Obrázok 8 Ambari Metrics; zdroj: vlastné spracovanie

Aby mohli používatelia pristupovať na HDFS úložisko treba im priradiť oprávnenia (permissions).

To docielime nasledovným postupom:

- V navigačnom menu: "admin" → "Manage Ambari" → Users → "Add Users"
- Vytvoríme používateľa s menom "hdfs" (heslo si môžeme vytvoriť aké chceme)
- Priradíme oprávnenia Cluster a Ambari admin.

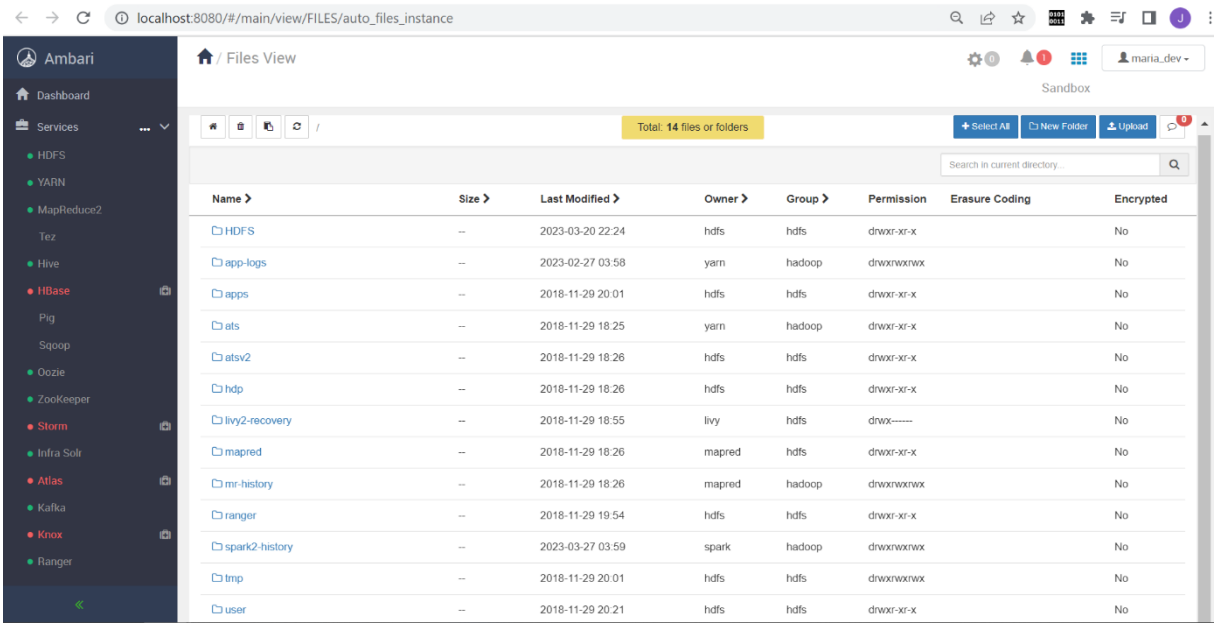
Aby sme mohli súbory na HDFS úložisku čítať a upravovať, potrebujeme vytvoriť náhľad. **Ten vytvoríme v dvoch krokoch.**

- V navigačnom menu: "admin" → "Manage Ambari" → Views → "AUTO_FILES_INSTANCE" → Edit "Permissions" (upraviť oprávenia).
- Tu nastavíme oprávnenia pre jednotlivých používateľov.

Pre pohodlnejšiu prácu s Hortonworks si stiahneme konzolu a nainštalujeme Python. Konzola na komunikáciu s SSH môže byť napríklad: https://www.putty.org/. Prihlásime sa ako maria_dev@localhost na jeden z dostupných portov napr. 2222.

4.2 Mapreduce program v Hadoope

Začneme výberom vhodného súboru. Rozhodli sme sa pre Covid dáta z jednotlivých krajín, ktoré ukazujú vývoj po dňoch. Zapneme si Hortonworks sandbox a Putty konzolu rovnako ako sme uviedli v kapitole 4.1. Prvý problém nastáva, keď chceme uložiť vybraný súbor na náš Hadoop file system. Existuje viacero spôsobov. Ak chceme súbor sťahovať z internetu použijeme príkaz *wget*. Ak ho chceme dostať z nášho lokálneho systému na HDFS, tak ho musíme nahrať na Ambari web a premiestniť ho na HDFS. V pravom hornom rohu nájdeme files view-obrázok .. .



Obrázok 9 Ambari files view; zdroj: vlastné spracovanie

Tu môžeme nájsť všetky súbory uložené na našom lokálnom systéme. Rovnako tu môžeme nastaviť oprávnenia na prácu so súborom pre jednotlivých používateľov. Súbor s dátami nahráme kam potrebujeme. Používateľa maria_dev vytvoril zložku s názvom work a sem nahral súbor s dátami.

Aby sme videli tento súbor aj na HDFS potrebujeme do konzoly napísať príkaz: `../../bin/hadoop fs -get /user/maria_dev/work/Covid.csv`

V tomto príkaze musíme zadať miesto uloženia v hadoope a cestu k súboru na lokálnom systéme. Následne sa musíme rozhodnúť čo chceme s daným súborom robiť a ktoré údaje z neho potrebujeme. Ja som sa rozhodol, že pre analýzu z dát potrebujem hlavne posledný deň sledovania údajov. V každej krajine sa doba sledovania líšila. Tento problém môžeme vyriešiť tak, že si vyberieme stĺpec z dátového súboru a nájdeme najväčšiu hodnotu pre každú krajinu. Na túto úlohu slúži **mapreduce** program.

Tento program sa skladá z dvoch častí. Prvou časťou je ***mapper***. Úlohou mappera je spracovať vstupné údaje. Vstupné údaje sú vo všeobecnosti vo forme súboru alebo adresára a sú uložené v systéme súborov Hadoop (HDFS). Vstupný súbor sa odovzdá funkcii mappera. Ten prejde riadok po riadku. Mapper spracuje údaje a vytvorí niekoľko malých kúskov údajov. Toto je úsek mappera v kóde:

```
def mapper(self, _, line):
        row = next(csv.reader([line]))
        Location = row[0]
        tdeaths = row[1]
        yield (Location, (tdeaths, row))
```

Úlohou Reducera je spracovať údaje, ktoré prichádzajú z mappera. Po spracovaní vytvorí nový súbor výstupov, ktorý sa dočasne uloží do HDFS. V mojom prípade zredukuje súbor na maximálne hodnoty v riadku ***Tdeaths*** podľa jednotlivých krajín. Ak by bola hodnota 0 nechá v riadku prázdnu hodnotu.

```
  def reducer(self, Location, values):
        max_tdeaths = 0
        max_row = None
        for value in values:
            tdeaths = value[0]
            if tdeaths:
                try:
                    tdeaths = int(tdeaths)
                except ValueError:
                    continue
                if tdeaths > max_tdeaths:
                    max_tdeaths = tdeaths
                    max_row = value[1]
        yield None, max_row or ['', '', '', '']
```

Zopakujeme proces ukladania na HDFS. Rovnako ako pri Covid.csv si uložíme náš mapreduce program pomocou príkazu:

```
../../bin/hadoop fs -get /user/maria_dev/work/Max1.py
```

Keď už máme funkčný mapreduce program a k nemu súbor s dátami obe uložené na HDFS, môžeme ich vyskúšať pustiť lokálne, či nenastanú nejaké komplikácie. To docielime tak, že do konzoly napíšeme príkaz: `python Max.py Covid.csv`
Ak sa program úspešne vykoná, tak by nám mal začať vybiehať výpis súboru. Podrobnosti o vykonaní mapreduce programu si môžeme pozrieť na obrázkoch 10 a 11.

```
Map-Reduce Framework
        CPU time spent (ms)=23930
        Combine input records=0
        Combine output records=0
        Failed Shuffles=0
        GC time elapsed (ms)=799
        Input split bytes=338
        Map input records=126247
        Map output bytes=66595547
        Map output materialized bytes=67100547
        Map output records=126247
        Merged Map outputs=2
        Peak Map Physical memory (bytes)=1503305728
        Peak Map Virtual memory (bytes)=5657460736
        Peak Reduce Physical memory (bytes)=294776832
        Peak Reduce Virtual memory (bytes)=2645237760
        Physical memory (bytes) snapshot=1832910848
        Reduce input groups=234
        Reduce input records=126247
        Reduce output records=234
        Reduce shuffle bytes=67100547
        Shuffled Maps =2
        Spilled Records=252494
        Total committed heap usage (bytes)=1655177216
        Virtual memory (bytes) snapshot=8324419584
```

Obrázok 10 Mapreduce output; zdroj: vlastné s pracovanie

```
File System Counters
        FILE: Number of bytes read=67100541
        FILE: Number of bytes written=134918981
        FILE: Number of large read operations=0
        FILE: Number of read operations=0
        FILE: Number of write operations=0
        HDFS: Number of bytes read=36622399
        HDFS: Number of bytes written=100732
        HDFS: Number of large read operations=0
        HDFS: Number of read operations=11
        HDFS: Number of write operations=2
Job Counters
        Data-local map tasks=2
        Launched map tasks=2
        Launched reduce tasks=1
        Total megabyte-milliseconds taken by all map tasks=150912000
        Total megabyte-milliseconds taken by all reduce tasks=19000320
        Total time spent by all map tasks (ms)=147375
        Total time spent by all maps in occupied slots (ms)=589500
        Total time spent by all reduce tasks (ms)=18555
        Total time spent by all reduces in occupied slots (ms)=74220
        Total vcore-milliseconds taken by all map tasks=147375
        Total vcore-milliseconds taken by all reduce tasks=18555
```

Obrázok 11 Mapreduce outpu 2; zdroj: vlastné s pracovanie

Vzhľadom nato, že všetko funguje môžeme tento program pustiť na hadoop clustri. Na to, aby sme ho pustili ho musíme manuálne naviesť do hadoop-streaming.jar, ktorý túto úlohu vykoná. Na začiatku príkazu do konzoly zadáme python mapreduce program a na konci príkazu súbor, na ktorom sa úloha vykoná.

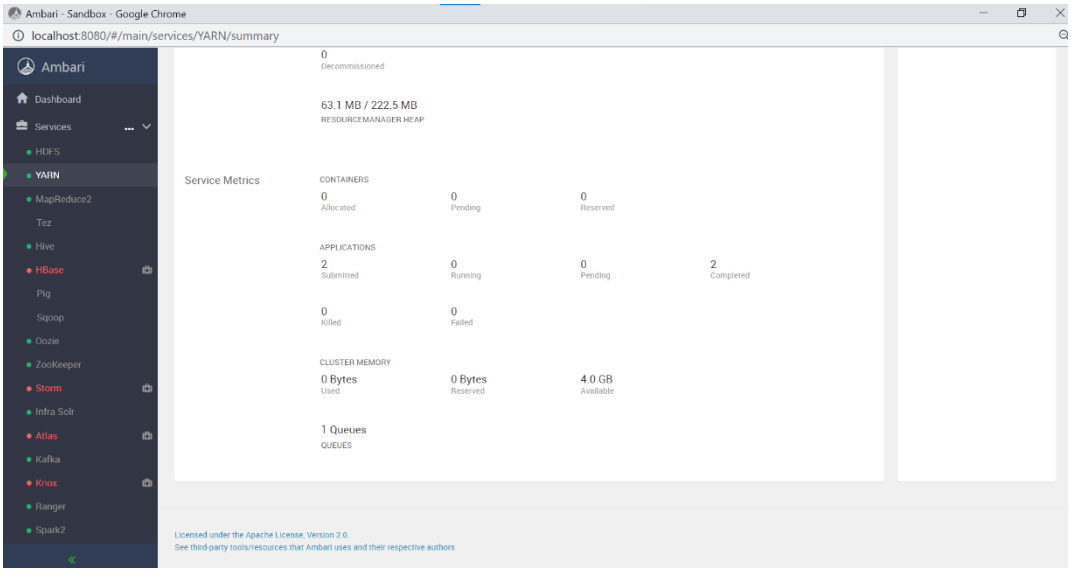
```
python Max.py -r hadoop --hadoop-streaming-jar
/usr/hdp/current/hadoop-mapreduce-client/hadoop-streaming.jar
Covid.csv
```

Príkaz bol úspešne vykonaný.

```
Job job_1679847667581_0002 running in uber mode : false
 map 0% reduce 0%
 map 43% reduce 0%
 map 67% reduce 0%
 map 100% reduce 0%
 map 100% reduce 89%
 map 100% reduce 100%
Job job_1679847667581_0002 completed successfully
Output directory: hdfs:///user/maria_dev/tmp/mrjob/Max.maria_dev.20230327.012940.147904/output
```

Obrázok 12 Mapreduce Hadoop cluster; zdroj: vlastné spracovanie

Potvrdiť si to môžeme aj na webovom rozhraní Hortonworksu. V metrikách yarnu si môžeme pozrieť 2 požiadavky na yarn a obidve boli úspešne vykonané.



Obrázok 13 Hadoop YARN; zdroj: vlastné spracovanie

4.3 Prieskumná analýza

Ak už máme dáta k dispozícii, potrebujeme ich spracovať do takej formy, aby čo boli čo najkvalitnejšie. Samotné spracovanie dát je pritom kľúčovým krokom od ktorého závisí kvalita výsledného modelu. Na tento účel slúži exploratívna analýza údajov (EDA). Prieskumná analýza údajov (EDA) je jedna z najdôležitejších činností v rutine dátového analytika alebo vedca. Umožňuje nám do hĺbky pochopiť súbor údajov, definovať alebo vyvrátiť hypotézy a vytvárať prediktívne modely. Používa techniky manipulácie s údajmi a niekoľko štatistických nástrojov na opísanie a pochopenie vzťahu medzi premennými a toho, ako môžu ovplyvniť podnikanie.

EDA pozostáva z niekoľkých krokov:

- Importovanie množiny údajov
- Pochopenie celkového obrazu („big picture“)
- Príprava dát
- Pochopenie premenných
- Štúdium vzťahov medzi premennými
- Brainstorming

4.3.1 Importovanie množiny údajov

Priebeh analýzy údajov začína importom alebo vytvorením pracovného súboru údajov. Okamžite potom začína fáza prieskumnej analýzy. Import súboru údajov v Pythone je s pomocou knižnice `Pandas` jednoduchý. Ak má náš súbor formát .csv, môžeme ho okamžite použiť.

```
df = pd.read_csv('output2.csv')
```

Obrázok 14 Načítanie súboru; zdroj: vlastné spracovanie

`df` je skratka pre dataframe, čo je objekt `Pandas` podobný hárku programu Excel. Funkcia `read_csv` sa berie ako vstup k súboru, ktorý chceme čítať. V jazyku Python existuje niekoľko možností importovať dáta do programu.

Uvádzame niekoľko najbežnejších možností:

- Importovanie pomocou knižnice `Pandas` ktorá odporuje importovanie dát z rôznych formátov, ako sú CSV, Excel, SQL databázy, JSON a mnoho ďalších. Na import dát z CSV súboru by ste mohli použiť nasledujúci kód:

- Importovanie dát priamo z internetu, napríklad priamo z `Github`, `Our World in Data` a podobne.
- Používanie knižníc, ktoré poskytujú integrované datasety, napríklad knižnica `Quandl` poskytuje prístup k rôznym finančným a ekonomickým dátam.

Na demonštráciu analýzy údajov použijeme súbor output2.csv, ktorý je našim výstupom mapreduce programu.

4.3.2 Pochopenie celkového obrazu

Teraz si opíšeme načítané dáta. Môžeme použiť napríklad metódu `describe()` ak použijeme túto metódu dostaneme iba opisnú štatistiku číselných hodnôt. Ak chceme tento príkaz vykonať pre všetky stĺpce, vrátane slovných (kategoriálnych) premenných tak ako atribút funkcie použijeme `include='all'`. Výstup tohto príkazuje vidíme na obr. 15

```
In [24]:  1 df.describe(include='all')
```

Out[24]:

	Location	Tdeaths	Date	Date Range Selected	Excess Mortality	Excess Mortality Cumulative	Excess Mortality Cumulative Absolute	Excess Mortality Cumulative Per Million	Extreme Poverty	Female Smokers	...	Reproduction Rate	Stringen Inde
count	188	1.880000e+02	188	188	0.0	0.0	0.0	0.0	121.000000	140.000000	...	52.000000	48.0000
unique	188	NaN	23	1	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	Na
top	Afghanistan	NaN	25/10/2021	True	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	Na
freq	1	NaN	121	188	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	Na
mean	NaN	3.458615e+04	NaN	NaN	NaN	NaN	NaN	NaN	13.961157	10.325000	...	0.822500	42.7175(
std	NaN	1.329479e+05	NaN	NaN	NaN	NaN	NaN	NaN	20.502953	10.474032	...	0.315811	18.3197(
min	NaN	1.000000e+00	NaN	NaN	NaN	NaN	NaN	NaN	0.100000	0.100000	...	0.060000	8.3300(
25%	NaN	4.412500e+02	NaN	NaN	NaN	NaN	NaN	NaN	0.600000	1.900000	...	0.645000	28.4700(
50%	NaN	2.834000e+03	NaN	NaN	NaN	NaN	NaN	NaN	2.200000	5.850000	...	0.850000	43.0550(
75%	NaN	1.465275e+04	NaN	NaN	NaN	NaN	NaN	NaN	21.400000	18.925000	...	1.057500	53.2400(
max	NaN	1.167433e+06	NaN	NaN	NaN	NaN	NaN	NaN	77.600000	44.000000	...	1.470000	90.7400(

11 rows × 72 columns

Obrázok 15 Načítanie súboru; zdroj: vlastné spracovanie

Pre každý stĺpec, ktorý predstavuje konkrétnu premennú, môžeme jednotlivé opisné charakteristiky ako celkový počet hodnôt danej premennej, priemer, smerodajnú odchýlku, maximálne a minimálne hodnoty, kvartily a podobne. Napríklad môžeme vidieť, že stĺpec *Location* obsahuje 188 hodnôt a každá z nich je jedinečná. Ďalej napríklad vidíme, že niektoré premenné v našom datasete neobsahujú žiadne pozorovania (count = 0). Tieto premenné preto nemajú zmysel pre ďalšiu analýzu a môžeme ich v tomto kroku vylúčiť.

Ak na množinu údajov použijeme funkciu `.shape`, Pandas nám vráti počet riadkov a počet stĺpcov v našom *.csv* súbore. V našom prípade to znamená, že dataset obsahuje 195 riadkov a 72 stĺpcov, t.j 72 premenných (obr. 16).

```
In [4]:  1 df.shape
Out[4]: (195, 72)
```

Obrázok 16 Shape (tvar); zdroj: vlastné spracovanie

Dve z používaných funkcií v Pandas sú `.head()` a `.tail()`. Tieto dva príkazy nám umožňujú zobraziť ľubovoľný počet riadkov (štandardne 5) od začiatku `(.head)` alebo

```
In [5]:  1 df.head()
```

Out[5]:

	Location	Tdeaths	Date	Date Range Selected	Excess Mortality	Excess Mortality Cumulative	Excess Mortality Cumulative Absolute	Excess Mortality Cumulative Per Million	Extreme Poverty	Female Smokers	...	Reproduction Rate	Stringency Index	Total Boosters
0	Afghanistan	7260.0	25/10/2021	True	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN
1	Africa	217122.0	25/10/2021	True	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN
2	Albania	2880.0	25/10/2021	True	NaN	NaN	NaN	NaN	1.1	7.1	...	NaN	NaN	NaN
3	Algeria	5894.0	25/10/2021	True	NaN	NaN	NaN	NaN	0.5	0.7	...	NaN	NaN	NaN
4	Andorra	130.0	01/09/2021	True	NaN	NaN	NaN	NaN	NaN	29.0	...	0.86	51.85	NaN

5 rows × 72 columns

Obrázok 17 Head; zdroj: vlastné spracovanie

od konca `(.tail)` množiny údajov. Sú veľmi užitočné pre rýchly náhľad na množinu údajov.

Metóda `.info()` vytlačí informácie o množine údajov vrátane celkového počtu riadkov, informácii o type údajov, o počte pozorovaní danej premennej a nenulových hodnôt.

Ako vidíme vo výstupe na obrázku č. 18, súhrn obsahuje zoznam všetkých stĺpcov s ich typmi údajov a počtom nenulových hodnôt v každom stĺpci.

```
In [25]:  1 df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 188 entries, 0 to 187
Data columns (total 72 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   Location                                188 non-null    object
 1   Tdeaths                                 188 non-null    float64
 2   Date                                    188 non-null    object
 3   Date Range Selected                     188 non-null    object
 4   Excess Mortality                        0 non-null      float64
 5   Excess Mortality Cumulative             0 non-null      float64
 6   Excess Mortality Cumulative Absolute    0 non-null      float64
 7   Excess Mortality Cumulative Per Million 0 non-null      float64
 8   Extreme Poverty                         121 non-null    float64
 9   Female Smokers                          140 non-null    float64
 10  Hosp Patients                           2 non-null      float64
 11  Hosp Patients Per Million               2 non-null      float64
 12  Icu Patients                            1 non-null      float64
 13  Icu Patients Per Million                1 non-null      float64
 14  Iso Code                                188 non-null    object
 15  Location (copy)                         188 non-null    object
 16  Location (geo)                          188 non-null    object
 17  Male Smokers                            138 non-null    float64
```

Obrázok 18 Informácie o údajoch; zdroj: vlastné spracovanie

4.3.3 *Príprava a čistenie údajov*

V tejto fáze chceme začať čistiť náš súbor, aby sme mohli pokračovať v analýze. **Niektoré z otázok, ktoré si položíme, sú:**

- Existujú nejaké zbytočné alebo nadbytočné premenné?
- Existujú nejaké duplicitné stĺpce?
- Má názvoslovie zmysel?
- Sú nejaké nové premenné, ktoré chceme vytvoriť?
- Máme v súbore chýbajúce hodnoty?

Ako prvé si všimol, že v našom súbore máme lokácie ako sú Ázia, Európe, svet a podobne. Preto je potrebné ich zo súboru vylúčiť, aby nedochádzalo k duplicite. Ukážku kódu uvádza obrázok 19.

```
In [94]:
# List of values to drop
values_to_drop = ["Africa", "Europe", "United States", "Asia", "World", "International", "European Union"]

# Drop rows with specified values in the "Location" column
df = df[~df["Location"].isin(values_to_drop)]

# Reset the index
df.reset_index(drop=True, inplace=True)
```

Obrázok 19 Odstránenie duplicít; zdroj: vlastné spracovanie

Ďalej na kontrolu duplicitných riadkov môžeme použiť príkaz `df.duplicated().sum()`. Tento príkaz nám nájde všetky duplicitné riadky, spočíta ich a následne vypíše počet duplicitných riadkov v našej množine údajov. Z výstupu na obrázku č. 20 vyplýva, že náš súbor duplicitné riadky neobsahuje.

```
In [27]:
df.duplicated().sum()
Out[27]: 0
```

Obrázok 20 Hľadanie duplicít; zdroj: vlastné spracovanie

Manipulácia s chýbajúcimi hodnotami. Jednoduchý spôsob, ako skontrolovať chýbajúce hodnoty, je použiť metódu `isnull`. Dostaneme dátový rámec s hodnotami true (1) a false (0), teda hodnoty sčítame a vidíme, v ktorom stĺpci sa nachádzajú chýbajúce hodnoty (obrázok 21).

```
In [32]:
df.isnull().sum()
Out[32]: Location                         0
         Tdeaths                          0
         Date                             0
         Date Range Selected              0
         Excess Mortality               188
                                        ...
         Total Cases Per Million          0
         Total Deaths Per Million         0
         Tvaccinations                  121
         Total Vaccinations Per Hundred 121
```

Obrázok 21 Hľadanie chýbajúcich hodnôt; zdroj: vlastné spracovanie

S chýbajúcimi hodnotami sa vieme vysporiadať s dvomi spôsobmi:

- **Odstránenie chýbajúcich hodnôt**
  - **Odstránenie riadkov** - môžeme odstrániť riadky s neprijateľnými množstvom chýbajúcich hodnôt. Dá sa to urobiť pomocou metódy *dropna*. Týmto spôsobom vylúčime riadky, ktoré obsahujú chýbajúce hodnoty (obrázok č. 22).
  - **Odstránenie stĺpcov** - ak stĺpec obsahuje väčšiu časť chýbajúcich hodnôt, má zmysel ho celý odstrániť, pretože odhadu chýbajúcich hodnôt nemožno dôverovať. Pretože sa vypočíta z malej vzorky údajov a merania by mohli byť nepresné .

```
In [3]:
# Drop rows with all empty values
df.dropna(how='all', inplace=True)
```

Obrázok 22 Odstránenie prázdnych riadkov; zdroj: vlastné spracovanie

- **Odhadnúť chýbajúce hodnoty** - Ak chýba iba prijateľné percento hodnôt, môžeme hodnoty odhadnúť.
  - **Dá sa to urobiť vyplnením strednou hodnotou** - funguje to tak, že sa vypočíta priemer/medián chýbajúcich hodnôt v stĺpci a potom sa chýbajúce hodnoty v každom stĺpci nahradia samostatne a nezávisle od ostatných. Dá sa použiť len s číselnými údajmi. V našom prípade (obrázok č. 23) sme doplnili chýbajúce hodnoty v stĺpci Median Age priemerom. A vypísali sme 5 hodnôt po doplnení.

```
In [12]:
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy='mean')
df['Median Age'] = imputer.fit_transform(df[['Median Age']])

print("After imputation:\n", df.loc[:, ['Location', 'Median Age']])
```

```
After imputation:
         Location  Median Age
0     Afghanistan   18.600000
1         Albania   38.000000
2         Algeria   29.100000
3         Andorra   30.294382
4          Angola   16.800000
..            ...         ...
183     Venezuela   29.000000
184       Vietnam   32.600000
185         Yemen   20.300000
186        Zambia   17.700000
187      Zimbabwe   19.600000

[188 rows x 2 columns]
```

Obrázok 23 Doplnenie prázdnych hodnôt; zdroj: vlastné spracovanie

  - **Druhá možnosť je doplnenie najčastejších hodnôt** - je to ďalšia štatistická stratégia na pripočítanie chýbajúcich hodnôt. Pracuje s kategorickými funkciami (reťazce alebo číselné reprezentácie) tak, že chýbajúce údaje nahrádza najčastejšími hodnotami v každom stĺpci.

- **Doplnenie pomocou najbližšieho suseda** - algoritmus K-najbližších susedov je algoritmus, ktorý sa používa na jednoduchú klasifikáciu. Algoritmus používa podobnosť funkcií na predpovedanie hodnôt akýchkoľvek nových údajových bodov.
- **Doplnenie podľa reťazenej rovnice (MICE)** - tento typ imputácie funguje tak, že chýbajúce údaje sa doplnia viackrát. Viacnásobné imputácie (MI) sú oveľa lepšie ako jednoduché imputácie, pretože merajú neistotu chýbajúcich hodnôt lepším spôsobom.
- **Doplnenie pomocou hlbokého učenia (Datawig)** - je to knižnica, ktorá sa učí modely strojového učenia pomocou hlbokých neurónových sietí na doplnenie chýbajúcich hodnôt v dátovom rámci.

Keď znovu použijeme príkaz `df.describe()`, na obrázku č. 24 si môžeme si všimnúť, že už máme rovnaký počet hodnôt ako v stĺpci `Tcases`. Numerické hodnoty Python často vypisuje vo formáte vedeckých hodnôt. Výpis môžeme upraviť použitím anonymnej funkcie `lambda` a zadaním želaného formátu. Okrem toho sme si vybrali iba tri stĺpce a to sú `Tcases, Median Age` a `Tdeaths`.

```
In [13]:
1 df.describe(include='all')
2 pd.set_option('display.float_format', lambda x: '%.2f' % x)
3 cols = ['Tcases', 'Median Age', 'Tdeaths']
4 print(df[cols].describe())
```

```
            Tcases  Median Age    Tdeaths
count       188.00      188.00     188.00
mean   1551712.45       30.29   34586.15
std    5756296.72        8.95  132947.86
min          4.00       15.10       1.00
25%      25962.00       22.35     441.25
50%     192767.50       30.29    2834.00
75%     725066.50       38.17   14652.75
max   54756977.00       48.20 1167433.00
```

Obrázok 24 Výpis hodnôt po doplnení; zdroj: vlastné spracovanie

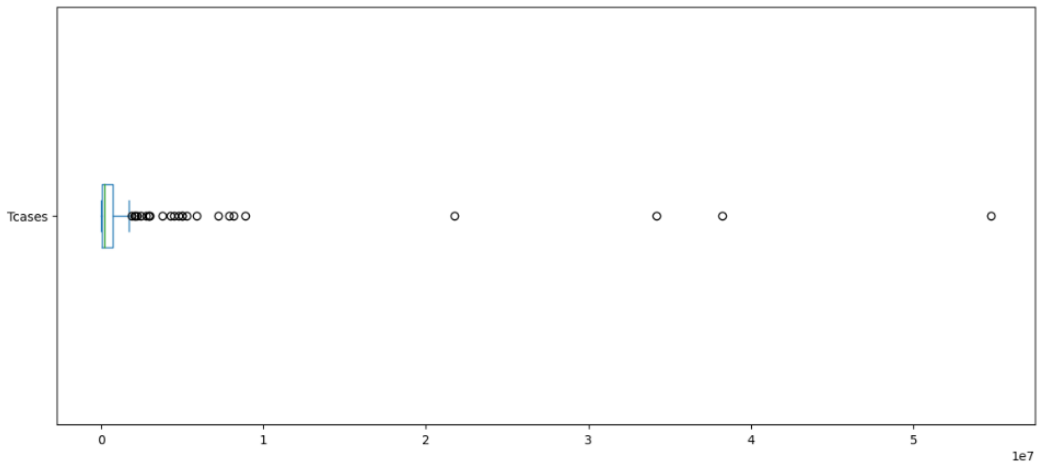
4.3.4 *Identifikácia odľahlých hodnôt*

Pri skúmaní údajov sú odľahlé hodnoty extrémne hodnoty v rámci súboru údajov. To znamená, že odľahlé údajové body sa značne líšia od očakávaných hodnôt – buď sú oveľa väčšie, alebo výrazne menšie. Pre údaje, ktoré sa riadia normálnym rozdelením, sa hodnoty, ktoré spadajú o viac ako tri štandardné odchýlky od priemeru, zvyčajne považujú za odľahlé hodnoty.

Možnosti detekcie odľahlých pozorovaní:

- **Nakreslíme si graf ideálne box plot(krabicový graf obrázok č. 25)** - začneme vytvorením krabicového grafu pre stĺpec Tcases. Krabicový graf nám umožňuje identifikovať jednorozmerné odľahlé hodnoty alebo odľahlé hodnoty pre jednu premennú. Odľahlé hodnoty sú v grafe zobrazené ako body, vďaka čomu sú ľahko viditeľné.

```
In [54]:  1 df['Tcases'].plot(kind='box', vert=False, figsize=(14,6))
Out[54]: <AxesSubplot: >
```



Obrázok 25 Box Plot; zdroj: vlastné spracovanie

- **Pravidlo kvartilového rozpätia (obrázok č. 26)** - kvartilové rozpätie (IQR) -rozdiel medzi horným a dolným kvartilom IQR = Q 0,75 – Q 0,25. Použitím IQR vypočítame dolnú (Q 0,25 -1,5*IQR) a hornú hranicu (Q 0,75 +1,5*IQR). Všetky pozorovania pod dolnou hranicou a nad hornou hranicou sú odľahlé pozorovania.

```
In [70]:  1 a = df['Tdeaths'].quantile(0.99865)
          2 a2 = df['Tdeaths'].quantile(0.00135)
          3 df_without_outliers_t = df[df['Tdeaths']<a]
          4 df_without_outliers = df_without_outliers_t[df_without_outliers_t['Tdeaths']>a2]
          5 df_without_outliers.describe(include='all')
```

Out[70]:

	Location	Tdeaths	Date	Date Range Selected	Excess Mortality	Excess Mortality Cumulative	Excess Mortality Cumulative Absolute	Excess Mortality Cumulative Per Million	Extreme Poverty	Female Smokers	...	Reproduction Rate	Stringency Index	
count	186	186.00	186	186	0.00	0.00	0.00	0.00	120.00	139.00	...	52.00	47.00	
unique	186	NaN	22	1	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	
top	Afghanistan	NaN	25/10/2021	True	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	
freq	1	NaN	120	186	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	
mean	NaN	28681.52	NaN	NaN	NaN	NaN	NaN	NaN	13.97	10.38	...	0.82	43.15	24
std	NaN	104344.20	NaN	NaN	NaN	NaN	NaN	NaN	20.59	10.49	...	0.32	18.26	44
min	NaN	3.00	NaN	NaN	NaN	NaN	NaN	NaN	0.10	0.10	...	0.06	8.33	
25%	NaN	464.00	NaN	NaN	NaN	NaN	NaN	NaN	0.57	1.90	...	0.65	29.16	
50%	NaN	2834.00	NaN	NaN	NaN	NaN	NaN	NaN	2.20	5.90	...	0.85	43.52	4
75%	NaN	14375.75	NaN	NaN	NaN	NaN	NaN	NaN	21.67	19.05	...	1.06	54.63	16
max	NaN	1114402.00	NaN	NaN	NaN	NaN	NaN	NaN	77.60	44.00	...	1.47	90.74	152

11 rows × 72 columns

Obrázok 26 Kvartilové rozpätie; zdroj: vlastné spracovanie

- **Z-skóre –** Z-skóre sa tiež nazýva štandardné skóre. Toto skóre pomáha pochopiť, či je hodnota údajov väčšia alebo menšia ako priemer a ako ďaleko je od priemeru. Presnejšie, Z-skóre hovorí, koľko štandardných odchýlok od dátového bodu je od priemeru. Na obrázku č. 27 sme si na ukážku vypísali prvých a posledných 5 hodnôt Z-skóre pre stĺpec `Tcases`.

```
In [69]:
1 from scipy import stats
2 z=np.abs(stats.zscore(df.Tcases))
3 print(z)

0     0.24
1     0.24
2     0.23
3     0.27
4     0.26
      ...
183   0.20
184   0.11
185   0.27
186   0.23
187   0.25
Name: Tcases, Length: 188, dtype: float64
```

Obrázok 27 Z skóre; zdroj: vlastné spracovanie

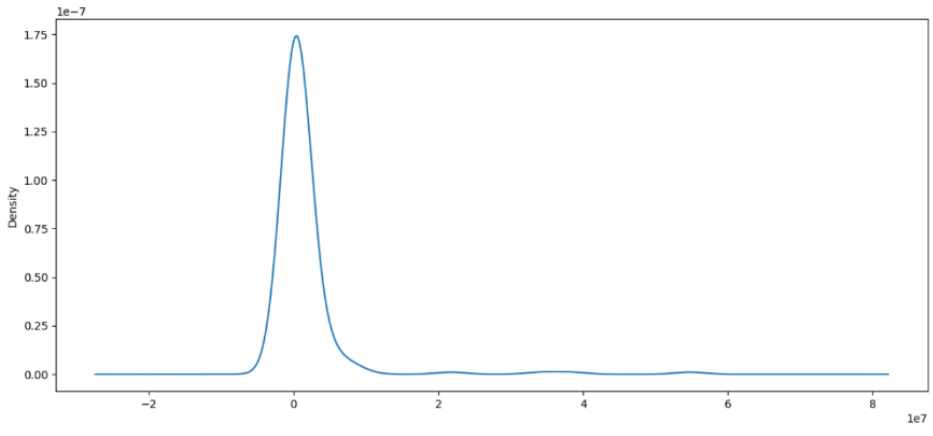
4.3.5 *Možnosti grafického zobrazenia premenných*

Zatiaľ čo v predchádzajúcich podkapitolách sme sa snažili opísať a upraviť celý dataset, teraz sa snažíme presne popísať jednotlivé premenné, jednotlivé premenné za pomoci vizualizácii údajov. Na samotnú vizualizáciu môžeme využiť knižnice ako `Matplotlib, Seaborn` alebo knižnicu pre interaktívne grafické zobrazenia `Plotly.`

<u>My si uvedieme niektoré z nich spolu s ukážkou:</u>

- **KDE Plot (obr. 28)** - opísaný ako Kernel Density Estimate sa používa na vizualizáciu pravdepodobnosti hustoty spojitej premennej.

```
In [58]:
1 df['Tcases'].plot(kind='kde', figsize=(14,6))

Out[58]: <AxesSubplot: ylabel='Density'>
```

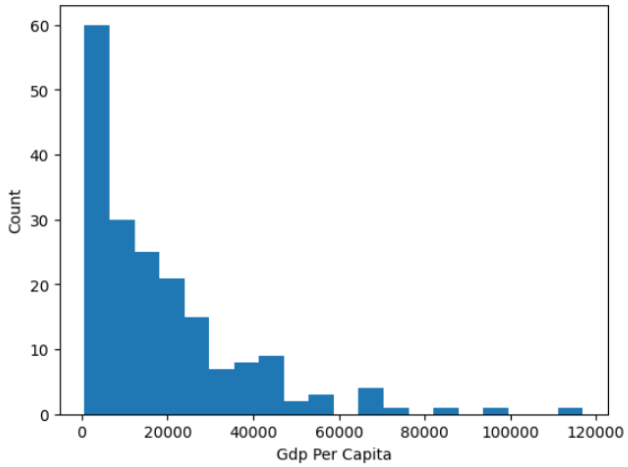


Obrázok 28 KDE Plot; zdroj: vlastné spracovanie

Zobrazuje hustotu pravdepodobnosti pri rôznych hodnotách v spojitej premennej. Môžeme tiež vykresliť jeden graf pre viacero vzoriek, čo pomáha pri efektívnejšej vizualizácii údajov.

- **Histogram (obrázok č. 29)** - histogramy nám pomáhajú identifikovať, kde sú hodnoty sústredené, aké sú extrémy a či existujú nejaké medzery alebo neobvyklé hodnoty. Histogram zobrazuje rozdelenie spojitej premennej. Dokáže odhaliť frekvenčné rozdelenie pre jednu premennú v jednorozmernej analýze.

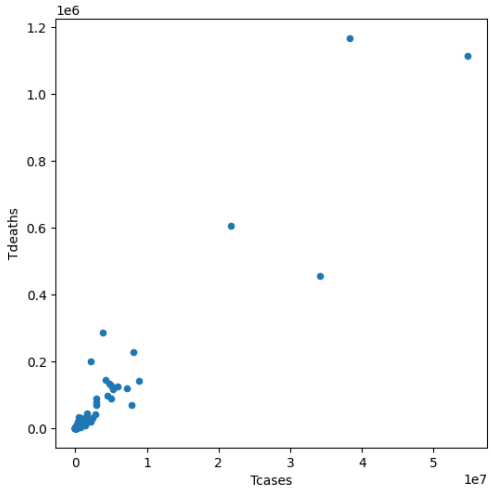
```
In [92]:
1 plt.hist(df['Gdp Per Capita'], bins=20)
2 plt.xlabel('Gdp Per Capita')
3 plt.ylabel('Count')
4 plt.show()
```



Obrázok 29 Histogram; zdroj: vlastné spracovanie

- **Bodový graf (obrázok č. 30)** - ak si chceme pozrieť vzťah dvoch premenných použijeme `.plot(scatter)`. Najčastejšie sa používa na nájdenie korelácií medzi dvoma spojitými premennými. Tu uvidíme bodový graf pre počet prípadov a počet úmrtí.

```
In [35]:
1 df.plot(kind='scatter', x='Tcases', y='Tdeaths', figsize=(6,6))
Out[35]: <AxesSubplot: xlabel='Tcases', ylabel='Tdeaths'>
```

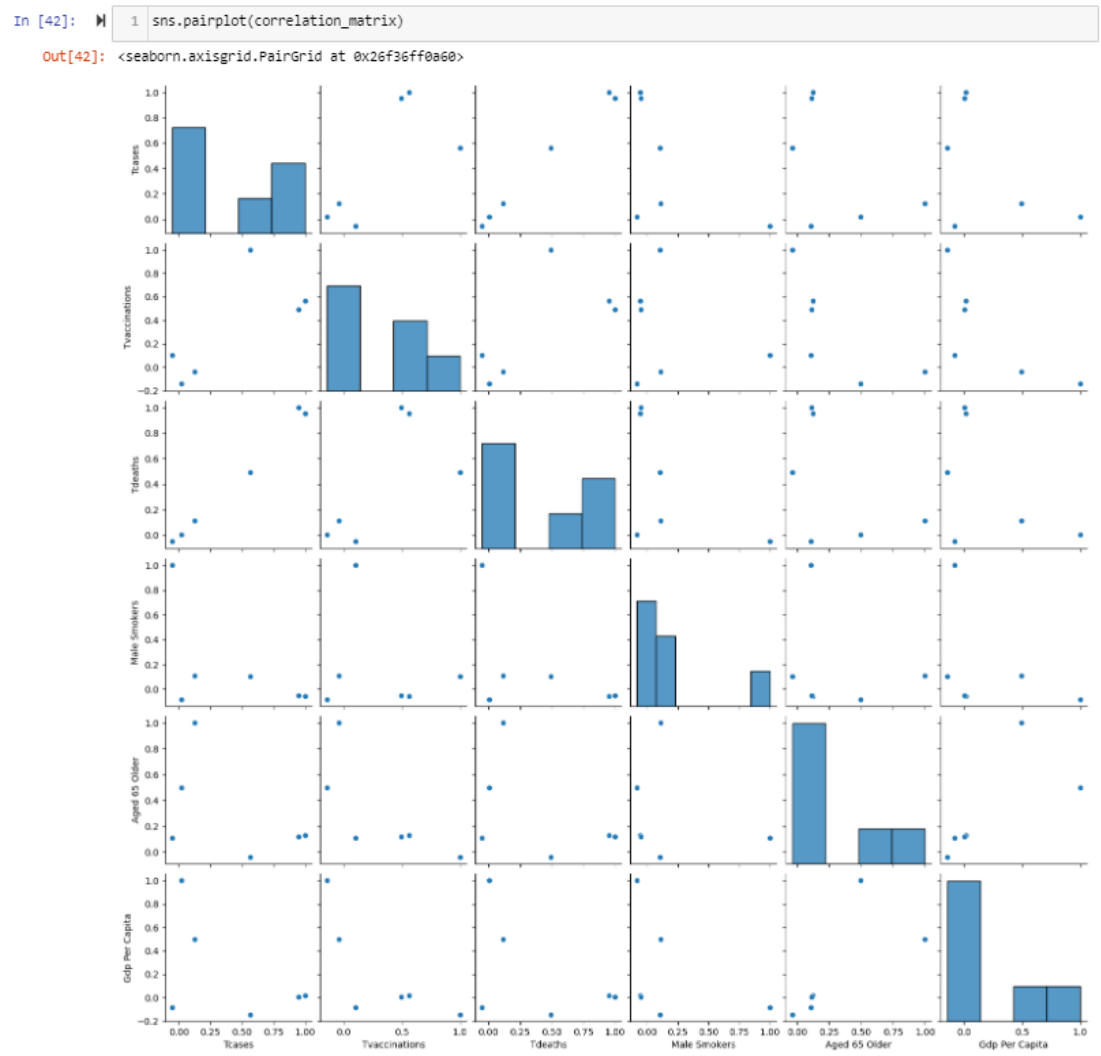


Obrázok 30 Bodový graf; zdroj: vlastné spracovanie

4.3.6 *Štúdium vzťahov medzi premennými*

Teraz sa posnažíme nájsť zaujímavé vzťahy, ktoré ukazujú vplyv jednej premennej na druhú premennú. Môžeme použiť vizualizácie, ako sú bodové grafy a tepelné mapy, aby sme získali prehľad o koreláciách.

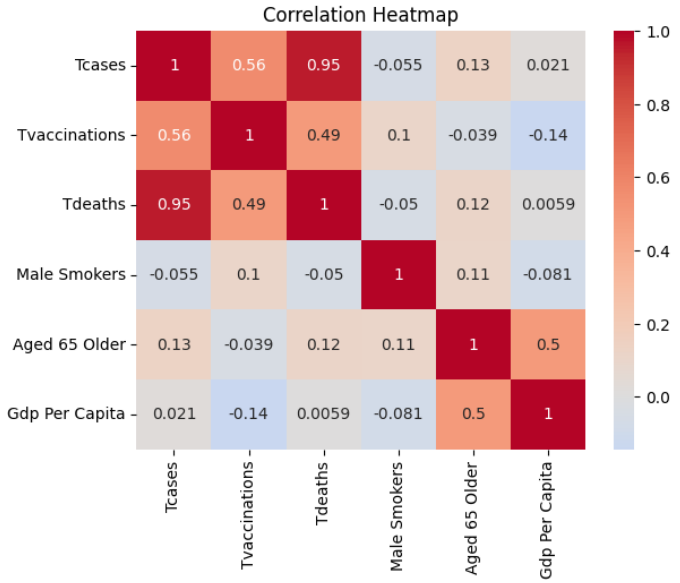
Ďalej budeme vzťahy skúmať pomocou `Seaborn` a `pairplot` obrázok č. 31. Ako si môžeme všimnúť, párový graf zobrazuje všetky premenné proti sebe v bodovom grafe. Je veľmi užitočný na pochopenie najdôležitejších vzťahov bez toho, aby ste museli každú jednu kombináciu prechádzať ručne . Nanešťastie musíme brať do úvahy, že je to výpočtovo náročné, takže sa najlepšie hodí pre súbory údajov s relatívne nízkym počtom premenných. Keďže máme v súbore až 72 premenných na ukážku sme použili 6 premenných z korelačnej matice.



Obrázok 31 Párový graf; zdroj: vlastné spracovanie

Teraz sa pozrieme, ako nám Seaborn môže opäť pomôcť rozšíriť náš prieskum vďaka tepelnej mape (obrázok č. 32). Na vytvorenie tepelnej mapy použijeme korelačnú maticu z číselných premenných, ktoré nás zaujímajú. Tepelná mapa je užitočná, pretože nám umožňuje efektívne pochopiť, ktoré premenné spolu silne korelujú. Zaujímavosťou je, že ak by sme odrezali polovicu tepelnej mapy pozdĺž diagonálnej čiary označenej jednotkami, nestratili by sme žiadne informácie. Na našej tepelnej mape môžeme vidieť silnú koreláciu medzi premennými Tcases a Tdeaths a napríklad žiadnu koreláciu medzi male smokers a Gdp Per Capita.

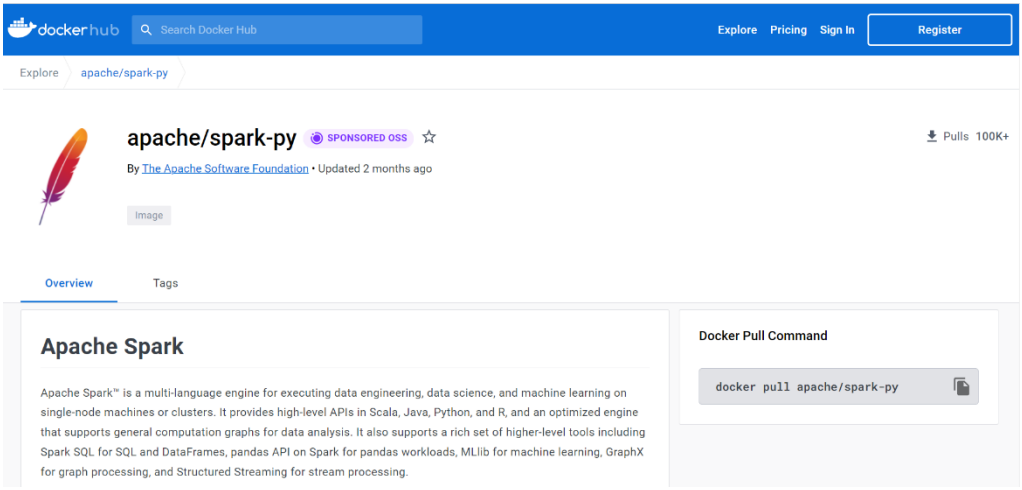
```
In [34]:
# Create a heatmap using seaborn
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', center=0)


plt.title('Correlation Heatmap')
plt.show()
```



Obrázok 32 Tepelná mapa; zdroj: vlastné spracovanie

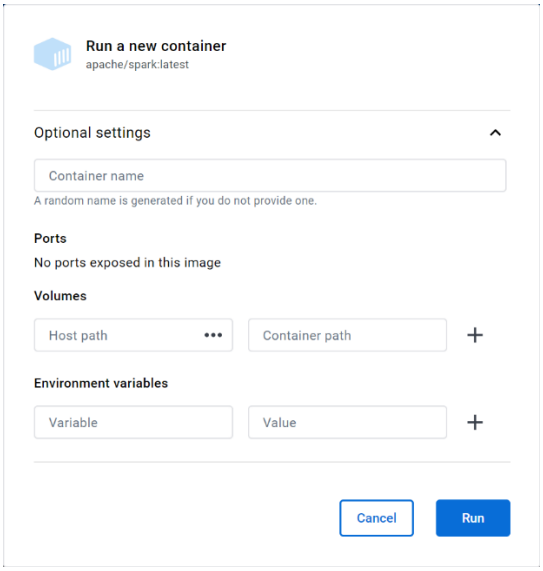
4.4 Nastavenie Dockeru a Visual Studia

Ako druhý spôsob pre spracovanie veľkých dát si ukážeme Apache Spark. Na prácu so Sparkom využijeme aplikácie Docker a Visual Studio Code. Ako prvé bolo treba stiahnuť kontajner s potrebnými aplikáciami a prepojiť ho s Visual Studiom. Na oficiálnej stránke hub.docker.com (obrázok č. 33) sú už vytvorené kontajnery s balíčkami aplikácií podľa našej potreby. Nemusíme všetko od základu nastavovať iba si vyberieme najvhodnejší balíček a stiahneme ho za pomoci príkazového riadku. Do príkazového riadku vložíme príkaz uvedený na stránke pre daný balíček.



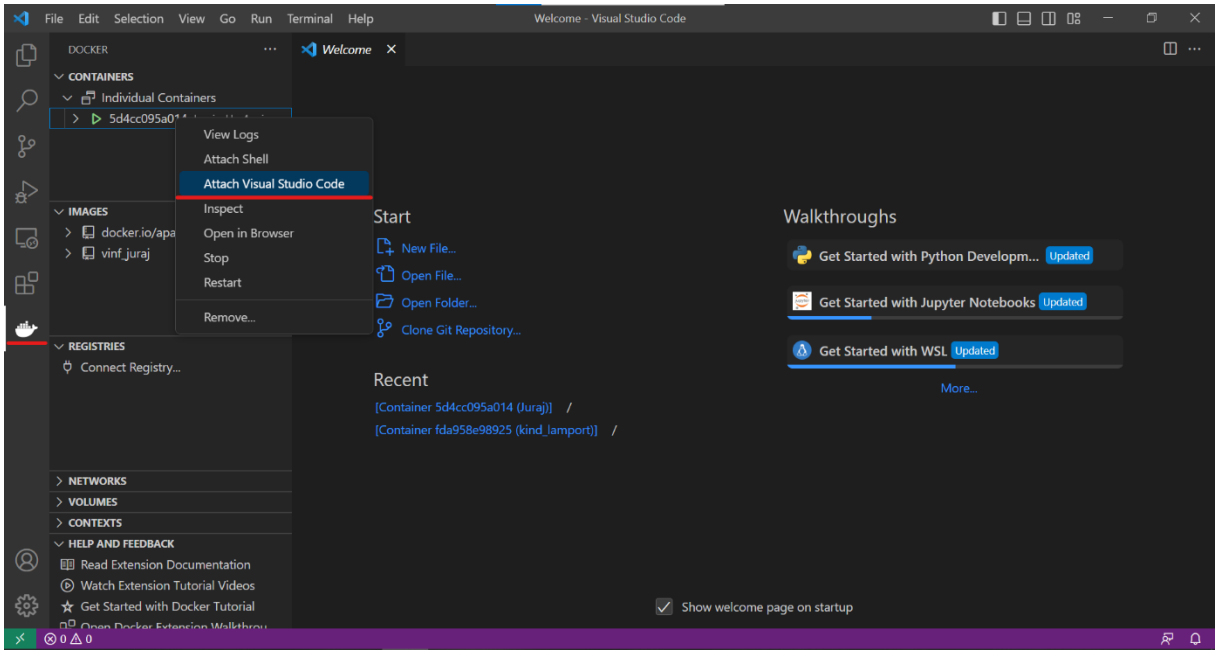
Obrázok 33 apache/spark-py; zdroj: vlastné spracovanie

Ak sa nám úspešne podarí balík stiahnuť. Zobrazí sa nám v Docker aplikácií. Po spustení môžeme nastaviť meno, porty, cestu a prostredie. Po prvom spustení takzvaného obrázku sa nám automaticky vytvorí aj kontajner.



Obrázok 34 Nastavenie kontajnera; zdroj: vlastné spracovanie

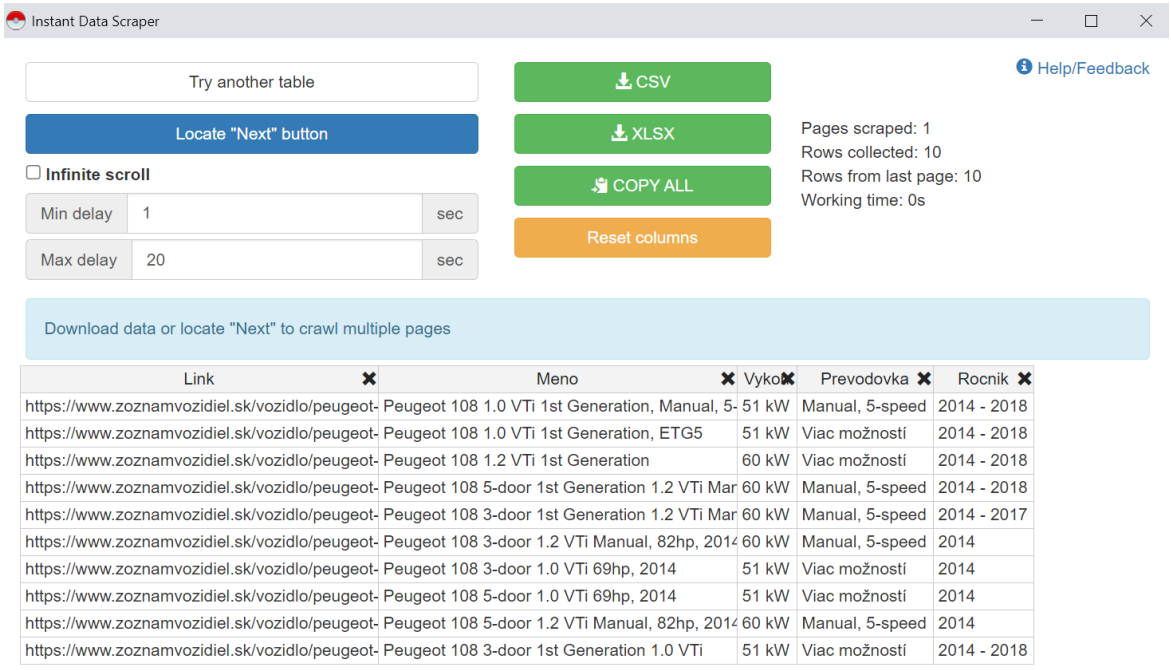
Po vytvorení kontajnera pustiť za pomoci tlačidla run. Aby sme v tomto kontajneri mohli vytvárať funkčné programy, potrebujeme ho prepojiť s Visual studiom. Vo Visual Studiu si stiahneme doplnok *Docker*. V tomto doplnku vidíme všetky naše existujúce kontajnery. Vyberieme si ten, ktorý ideme používať a pravým tlačidlom nájdeme možnosť Attach Visual Studio Code. To nám umožní pracovať so súbormi v príslušnom kontajneri.



Obrázok 35 Prepojenie Docker a Visual Studio Code; zdroj: vlastné spracovanie

4.5 Vyhľadávanie za pomoci Apache Spark

Ako prvé potrebujeme stanoviť aký program ideme vytvárať. Rozhodol som sa pre vyhľadávanie podľa kritérií vo veľkom datasete. Ako fanúšika áut ma zaujímajú rôzne modely áut. Existuje webová stránka, ktorá obsahuje všetky modely áut od všetkých svetových značiek. Aby sme tieto údaje stiahli z webovej stránky, použijeme takzvaný „*data scraper*“. Je to program na sťahovanie dát z internetu. Existuje viacero druhov. My použijeme *Instant Data Scraper*, ktorý je voľne dostupný ako doplnok do Google Chrome. Otvoríme ho na webovej stránke, z ktorej chceme dáta stiahnuť. Zvolíme si, ktoré údaje chceme sťahovať. Vyberieme si Link, meno, výkon, prevodovku a ročník výroby.



Link	Meno	Vyko	Prevodovka	Rocnik
https://www.zoznamvozidiel.sk/vozidlo/peugeot-	Peugeot 108 1.0 VTi 1st Generation, Manual, 5-	51 kW	Manual, 5-speed	2014 - 2018
https://www.zoznamvozidiel.sk/vozidlo/peugeot-	Peugeot 108 1.0 VTi 1st Generation, ETG5	51 kW	Viac možností	2014 - 2018
https://www.zoznamvozidiel.sk/vozidlo/peugeot-	Peugeot 108 1.2 VTi 1st Generation	60 kW	Viac možností	2014 - 2018
https://www.zoznamvozidiel.sk/vozidlo/peugeot-	Peugeot 108 5-door 1st Generation 1.2 VTi Mar	60 kW	Manual, 5-speed	2014 - 2018
https://www.zoznamvozidiel.sk/vozidlo/peugeot-	Peugeot 108 3-door 1st Generation 1.2 VTi Mar	60 kW	Manual, 5-speed	2014 - 2017
https://www.zoznamvozidiel.sk/vozidlo/peugeot-	Peugeot 108 3-door 1.2 VTi Manual, 82hp, 201	60 kW	Manual, 5-speed	2014
https://www.zoznamvozidiel.sk/vozidlo/peugeot-	Peugeot 108 3-door 1.0 VTi 69hp, 2014	51 kW	Viac možností	2014
https://www.zoznamvozidiel.sk/vozidlo/peugeot-	Peugeot 108 5-door 1.0 VTi 69hp, 2014	51 kW	Viac možností	2014
https://www.zoznamvozidiel.sk/vozidlo/peugeot-	Peugeot 108 5-door 1.2 VTi Manual, 82hp, 201	60 kW	Manual, 5-speed	2014
https://www.zoznamvozidiel.sk/vozidlo/peugeot-	Peugeot 108 3-door 1st Generation 1.0 VTi	51 kW	Viac možností	2014 - 2018

Obrázok 36 Datascrape; zdroj: vlastné spracovanie

Vďaka linku budeme vedieť zistiť bližšie špecifikácie o autách. Aby data scraper vedel automaticky prechádzať medzi jednotlivými stranami, ukážeme mu tlačidlo next. Počet jedinečných údajov je 648179. Po úspešnom stiahnutí si vyberieme do akého formátu si chceme dáta uložiť. Pre lepšiu prácu formát CSV. V predchádzajúcej časti sme prepojili programy Docker a Visual Studio Code. Teraz môžeme stiahnuté dáta uložiť do nášho kontajneru a začať písať kód. Najprv som napísal kód v jazyku Python.

```
from difflib import SequenceMatcher


with open('/home/cars/zoznamvozidiel.csv', 'r',
encoding='utf-8') as f:
    lines = f.readlines()


# skip the first row which contains column names
lines = lines[1:]


# create a list of dictionaries representing each car
cars = []
for line in lines:
    parts = line.strip().split(',')
```

```
    cars.append({'Name': parts[0], 'Link': parts[1], 'Vykon': parts[2], 'Rocnik': parts[3], 'Prevodovka': parts[4], 'Rocnik': parts[5],})


# define the search criteria
search_term = input("Enter a car name: ").lower()


# find the car with the highest percentage of similarity
best_car = None
best_score = 0
for car in cars:
    score = SequenceMatcher(None, search_term, car['Name'].lower()).ratio()
    if score > best_score:
        best_car = car
        best_score = score


# display the result
if best_car:
    print("The car that best matches your search term is:")
    for key, value in best_car.items():
        print(f"{key}: {value.encode('utf-8')}")
else:
    print("No car was found that matches your search term.")
```

Úlohou tohto kódu je zistiť od užívateľa aký model auta hľadá a po zadaní prehľadá riadok po riadku. Pri každom riadku vypočíta percentuálnu zhodu názvu so zadanou hodnotou od používateľa. Program vypíše v termináli model auta s najväčšou zhodou. Ak sa zadaná hodnota nezhoduje so žiadnym modelom, na terminál bude vypísané:

```
"No car was found that matches your search term."
```

Ďalším krokom bude pretvorenie klasického Python programu na program spustiteľný v Sparku. Podobne ako v programe tvorenom pre Hadoop vytvoríme mapper a reducer. V tomto programe môžeme použiť až dva mappery a reducery, pretože v jednom prípade definujeme kritériá vyhľadávania a v druhom hľadám najväčšiu zhodu.

```
from difflib import SequenceMatcher
from pyspark.sql import SparkSession
import sys

# define the search criteria
search_term = (" ".join(sys.argv[1:])).lower()

def list_mapper(line):
    parts = line.strip().split(',')
    return [{'Name': parts[0], 'Link': parts[1], 'Vykon': parts[2], 'Rocnik': parts[3], 'Prevodovka': parts[4], 'Rocnik': parts[5]}]

def list_reducer(x, y):
    x = x+y
    return x


def compare_mapper(car):
    score = SequenceMatcher(None, search_term, car['Name'].lower()).ratio()
    return {"car": car, "score": score}

def compare_reducer(x, y):
    if x["score"] > y["score"] :
        return x
    else:
        return y

if __name__ == '__main__':

    spark = SparkSession\
            .builder\
```

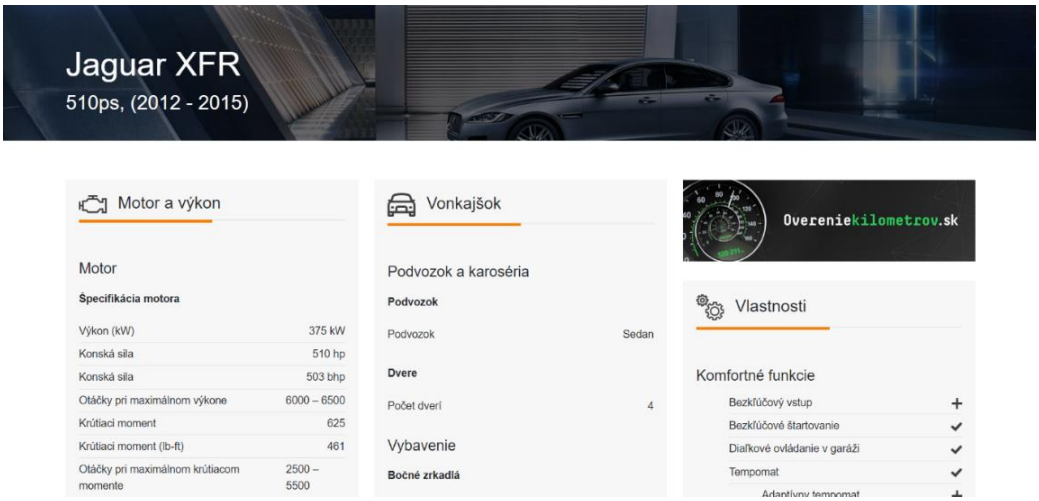
```
            .appName("PythonPi")\
            .getOrCreate()       #inicialization and setup of
spark
    spark.sparkContext.setLogLevel('ERROR')
    partitions = 2

    with      open('/home/cars/zoznamvozidiel.csv',      'r',
encoding='utf-8') as f:
        lines = f.readlines()

    # skip the first row which contains column names
    lines = lines[1:]

    # create a list of dictionaries representing each car
    cars       =       spark.sparkContext.parallelize(lines,
partitions).map(list_mapper).reduce(list_reducer)
    if len(sys.argv)>1:
        best_car    =    spark.sparkContext.parallelize(cars,
partitions).map(compare_mapper).reduce(compare_reducer)
        best_car = best_car["car"]

        # display the result
        if best_car:
            print("The car that best matches your search term
is:")
            for key, value in best_car.items():
                print(f"{key}: {value.encode('utf-8')}")

        else:
            print("No car was found that matches your search
term.")
    else:
        print("Missing name of car we are looking for.")
```

Program spustíme zadaním príkazu do terminálu, na konci príkazu musíme zadať hľadaný výraz. Následne budú údaje o danom modeli vypísané na termináli (obrázok č. 37)

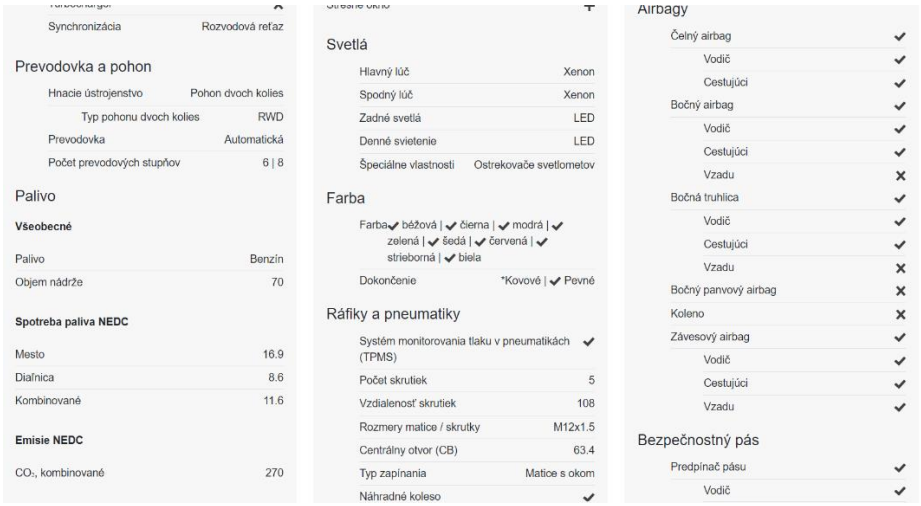
```
spark-submit --driver-memory 25g /home/cars/CarsSpark.py Jaguar XFR
```

Obrázok 37 Výpis údajov; zdroj: vlastné spracovanie

Pre viac podrobností o danom modeli auta môžeme ísť na vypísaný link, kde nájdeme všetky špecifikácie o aute



Obrázok 38 Špecifikácia vozidla; zdroj: vlastné spracovanie



Obrázok 39 Špecifikácia vozidla 2; zdroj: vlastné spracovanie

Záver

Spracovanie veľkých dát v Pythone vyžaduje a dôkladné pochopenie špeciálnych techník a nástrojov, ktoré dokážu optimalizovať výkon a zabezpečiť efektívne spracovanie dát. Počas tejto práce sme skúmali rôzne prístupy, ako napríklad paralelné spracovanie, distribuované výpočty, streamovanie údajov a cloudové riešenia, pričom každý má svoje jedinečné silné a slabé stránky. Taktiež sme si popísali populárne knižnice a rámce, ako sú Apache Spark, Hadoop a PySpark, ktoré umožňujú efektívne spracovanie veľkých dát v Pythone.

Využitím vhodných postupov na ukladanie a spracovanie údajov a využitím výkonu knižníc a rámcov môžu dátoví vedci, analytici a inžinieri odomknúť plný potenciál jazyku Pythonu. Od rozdelenia veľkých súborov údajov na menšie časti na paralelné spracovanie až po využitie viacerých uzlov clustra na distribuované výpočty. S hlbokým pochopením toho, ako narábať s veľkými údajmi v Pythone, môžu odborníci riešiť problémy v reálnom svete a odvodiť zmysluplné poznatky z rozsiahlych súborov údajov. Či už ide o analýzu veľkých súborov údajov pre business intelligence, spracovanie obrovského množstva vedeckých údajov na výskum alebo využitie veľkých údajov pre aplikácie strojového učenia. Znalosti a zručnosti získané z tejto práce umožnia odborníkom efektívne spravovať, spracovávať a analyzovať veľké údaje v Pythone.

Prvá kapitola sme venovali stručnému opisu súčasného stavu problematiky a základným pojmom. V druhej kapitole sme charakterizovali cieľ práce. V tretej kapitole sme sa venovali metódam a nástrojom na prácu s veľkými dátami v jazyku Python. V záverečnej kapitole sme sa zaoberali samotným postupom tvorby našich programov a analýzy dát. Výsledkom riešenia danej problematiky sú dva funkčné programy a analýza dát. Prvý program sme púšťali na virtuálnom stroji pomocou technológii Hadoop a druhý na kontajnerovej platforme Docker pomocou technológii Spark.

Cieľ práce sa nám podarilo splniť. Vytvorili sme funkčný Mapreduce program, ktorý sme púšťali v Apache Hadoop a taktiež sa nám podarilo vytvoriť program, ktorý obsahoval mapreduce funkcie v Sparku. Táto práca nám pomohla zlepšiť sa v Pythone, rozšírila nám obzory v spracovaní veľkých dát a taktiež nás naučila využívať základné príkazy v Linuxe.

Zoznam použitej literatúry

[1] IBM 2022. *data-science* In ibm.com [online]. [citované dňa 3.1.2023]. Dostupné na internete: https://www.ibm.com/topics/data-science

[2] CRAIG STEDMAN 2022. *Data-collection* In techtarget.com [online]. [citované dňa 3.1.2023]. Dostupné na: https://www.techtarget.com/searchcio/definition/data-collection.

[3] VIKTORIA RUBAN 2022. *How-is-big-data-collected* In computools.com [online]. [citované dňa 13.1.2023]. Dostupné na internete: https://computools.com/how-is-big-data-collected/

[4] UTSONLINE 2022. *Data-processing* In studyonline.uts [online]. [citované dňa 13.1.2023]. Dostupné na internete: https://studyonline.uts.edu.au/blog/what-data-processing.

[5] CRAIG STEDMAN 2022. *Data-analytics* In techtarget.com [online]. [citované dňa 1.2.2023]. Dostupné na internete: https://www.techtarget.com/searchdataman agement/definition/data-analytics.

[6] A UNIT OF MEDHYA ANALYTICS SOLUTIONS 2022. *Data-presentation-types-importance* In analyticstraininghub [online]. [citované dňa 1.2.2023]. Dostupné na internete: https://analyticstraininghub.com/data-presentation-types-importance/.

[7] SHANIKA WICKRAMASIGHNE 2022. *Jupyter Notebooks for Data Analytics In* bmc.com [online]. [citované dňa 1.2.2023]. Dostupné na internete: https://www.bmc.com/blogs/installing-jupyter-for-big-data-and-analytics/.

[8] JUPYTER 2022. *Jupyterhub* In jupyter.org [online]. [citované dňa 1.2.2023]. Dostupné na internete: https://jupyter.org/.

[9] RUKMANI, GOPALAN: *The Cloud Data Lake*, United States of America: O´reilly Media 2022. 213 s. ISBN 978-1-098-11658-3

[10] LAURENT, Anne – LAUENRT Dominique – MADERA, Cédrine: *Data Lakes*, United Kingdom: ISTE Ltd 2020. 219 s. ISBN 978-1-78630-585-5

[11] IBM 2022. *apache-spark In* ibm.com [online]. [citované dňa 18.2.2023]. Dostupné na internete: https://www.ibm.com/topics/apache-spark

[12] AZURE 2022. *cloud-computing-dictionary* In azure.microsoft.com [online]. [citované dňa 18.2.2023]. Dostupné na internete: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is- big-data-analytics/

[13] SIMPLILEARN 2023. *What is big data analytics and why it is important?* In simplilearn.com [online]. [citované dňa 18.2.2023]. Dostupné na internete: https://www.simplilearn.com/what-is-big-data-analytics-article

[14] MARR, Bernard: BIG DATA IN PRACTICE, United Kingdom: John Wiley and Sons Ltd, 2016. 301 s. ISBN 978-1-119-23138-7

[15] VMWARE 2023. *What is virtual machine?* In vmware.com [online]. [citované dňa 21.2.2023]. Dostupné na internete: https://www.vmware.com/topics/glossary/content/virtual-_machine.html

[16] IBM 2022. *Virtual machines* In ibm.com [online]. [citované dňa 21.2.2023]. Dostupné na internete: https://www.ibm.com/topics/virtual-machines

[17] NAMRATA BISHT 2023. *Virtualization in Cloud Computing and Types* In geeksforgeeks.org [online]. [citované dňa 26.2.2023]. Dostupné na internete: https://www.geeksforgeeks.org/virtualization-cloud-computing-types/

[18] AMAZON WEB SERVICES 2023. *What is containerization* In aws.amazon.com [online]. [citované dňa 1.3.2023]. Dostupné na internete: https://aws.amazon.com/what-is/containerization/

[19] RED HAT 2022. *What is Docker?* In opensource.com [online]. [citované dňa 1.3.2023]. Dostupné na internete: https://opensource.com/resources/what-docker

[20] LINUX 2022. *Container and virtualization tools* In linuxcointainers.org [online]. [citované dňa 1.3.2023]. Dostupné na internete: https://linuxcontainers.org/

[21] RED HAT 2020. *What is Kubernetes?* In redhat.com [online]. [citované 20.3.2023]. Dostupné na internete: https://www.redhat.com/en/topics/containers/what-is-kubernetes

[22] SARAN SHARMA 2023. *Introduction to NOSQL* In geeksforgeeks.org [online]. [citované dňa 20.3.2023]. Dostupné na internete: https://www.geeksforgeeks.org/introduction-to-nosql/

[23] TRUSTRADIUS 2023. *NOSQL databases* In trustradius.com [online]. [citované dňa 20.3.2023]. Dostupné na internete: https://www.trustradius.com/nosql-databases

[24] IBM 2022. *NoSQL-databases* In ibm.com [online]. [citované dňa 20.3.2023]. Dostupné na internete: https://www.ibm.com/topics/nosql-databases

[25] DATABRICKS 2021. *Glossary Hadoop* In databricks.com [online]. [citované dňa 12.4.2023]. Dostupné na internete: https://www.databricks.com/glossary/hadoop

[26] SNOWFLAKE INC 2022. *Cloud data lake* In snowflake.com [online]. [citované dňa 12.4.2023]. Dostupné na internete: https://www.snowflake.com/guides/cloud-data-lake

[27] TIM KING 2022. *The Best Cloud Data Lake Solutions* In solutionsreview [online]. [citované dňa 12.4.2023]. Dostupné na internete: https://solutionsreview.com/data-management/the-best-cloud-data-lake-solutions/

[28] MONGODB 2022. *Mongodb* [online]. [citované dňa 15.4.2023]. Dostupné na internete: https://www.mongodb.com/

[29] CASSANDRA 2022 *Apache Cassandra* In cassandra [online]. [citované dňa 15.4.2023]. Dostupné na internete: https://cassandra.apache.org/_/index.html

[30] REDIS 2022. *Redis* In redis.io [online]. [citované dňa 15.4.2023]. Dostupné na internete: https://redis.io/

[31] HBASE 2023. *Hbase* In hbase.apache.org [online]. [citované dňa 15.4.2023]. Dostupné na internete: https://hbase.apache.org/

[32] AMAZON WEB SERVICES *Dynamodb* in aws.amazon.com [online]. [citované dňa 15.4.2023]. Dostupné na internete: https://aws.amazon.com/dynamodb/

[33] COUCHBASE 2022. *Couchbase* In couchbase.com [online]. [citované dňa 15.4.2023]. Dostupné na internete: https://www.couchbase.com/

[34] KEITH SHAW 2022. *What is a virtual machine, and why are they so useful?* In networkworld.com [online]. [citované dňa 18.4.2023]. Dostupné na internete: https://www.networkworld.com/article/3583508/what-is-a-virtual- machine-and-why-are-they-so-useful.html

[35] BUCHANAN IAN 2022. *Containers vs VMS* In atlassian.com [online]. [citované dňa 18.4.2023]. Dostupné na internete: https://www.atlassian.com/microservices/cloud-computing/containers-vs-vms

[36] STEGNER BEN 2020. *Practical Reasons to Start Using a Virtual Machine* In makeuseof.comm [online]. [citované dňa 18.4.2023]. Dostupné na: https://www.makeuseof.com/tag/reasons-start-using-virtual-machine/