MSc project that will submitted in partial fulfilment of a University of Greenwich Master's Degree

Federated learning decentralised Securing and Standardising Data before Cloud Deployment using Hadoop.

Data Pipeline Pre-Cloud Edition

Name:	Cornateanu Laurentiu
Student:	ID: 001295276
Course of Study:	Big Data & Business Intelligence
Date Submitted:	18.12.2024
Submission Date:	22.12.2024

Supervisor:	Dr. Tuan Vuong
Topic Area:	Big Data Analytics

Keywords associated with the project:

Competitive Advantage
Data Security & Privacy
Efficaciousness
Model Training
Customisation
Asset usage
Customisation
Sklearn (standardisation

MSc Modules studied that contribute towards this project:

Data Visualisation
Data Warehousing and Business Intelligence
Big Data
Applied Machine Learning
Programming Fundamentals for Data Science
Advanced Database Technologies
User Centred Web Engineering
Clouds, Grids and Virtualisation

Thesis

This dissertation is about the Big Data revolution, where the volume of data is fantastic and growing exponentially. One crucial fact must be remembered: at the same time, data clutter, which refers to data not relevant to be analysed, represents 70%. To better understand what happens in two scenarios, before/after data uploading to the cloud, the focus is on research on data processing before uploading it to the cloud or any other storage medium. How to optimise data volume and have processing efficiency by cleaning, validating and converting data efficiently. The concept reduces storage costs, improving Performance when data is manipulated somehow. Data clutter affects business, productivity suffers, and more than this, the environment has a negative footprint.

All the arguments we made predict that the demand for energy in data centres will increase in the next 5 years by at least 60%. (Goldman Sachs, 2024)

Big Data is a special kind of data. The research mainly focuses on a challenging problem data storage efficiency. The project will explore a comprehensive side of data pre-processing using Hadoop single node and pre-cloud ETL. Execute It will Standardise/Normalise solving/secure data storage using Parquet in an FL. This concept optimises costs, increases data privacy compliance, and ensures robust data security and efficiency. ETL (Extract, Transform, Load) processes locally before uploading data to the cloud is a pivotal approach in data management that enhances performance, cost-effectiveness, and data quality. This methodology allows organizations to pre-process and refine their data in a controlled environment, addressing critical issues such as data consistency and integrity before cloud integration. The growing use of cloud-based data solutions necessitates efficient local ETL operations for firms seeking to leverage data analytics efficiently and competitively in today's data-driven environment.

This concept highlights the methodology for combining on-premises and cloud resources to create a scalable, secure, and efficient infrastructure. The approach is applicable in certain areas of big data, including finance, healthcare, telecommunications, retail, and manufacturing, providing a robust framework for big data management, compliance, and security. Federated learning is distributed data processing that maintains data confidentiality and adheres to GDPR.

The project key researcher found one from many more that lowers bandwidth usage and power consumption by pre-processing locally before uploading to the cloud, making it a more sustainable data management strategy. The focus on automating data and the findings imply that pre-processing before cloud storage is vital for the overall quality of the results because it removes clutter from the data, representing more than 60% of a dataset. The work is carried out in the context of cross-industry pre-processing of big data.

Consequently, implementing this new way of working works in local environments at a sustainable cost and efficiently uses resources.

Acknowledgement

I want to make a special Thank You! Dr. Tuan Vuong for his patience, support, and guidance, which helped me succeed. My success would not have been possible without your trust and time invested in me. In the same time much appreciation to Ala Barzinji.

I am thankful to for a special teacher who will be and remain a distinct person Dr. Solomon E. for his invaluable contribution to my education. His counsel, experience, and support have helped me overcome my past limits and strive for perfection.

I thank Dr. Tatiana, the Big Data and Business Intelligence program head leader, for her unwavering support of me at my journey. Your impact has dramatically changed my study and knowledge.

I owe a great deal of appreciation to the University and all the professors involved in my academic path. They have been there for me every step of the way, providing unwavering encouragement, advice, and insight that have helped me make a meaningful impact in my academic career. Because of this, my academic career has taken a major hit.

My sincerest thanks to the University and my lecturers.

Contents

Abbreviation:

ETL	Extract, Transform, Load
GDPR	General Data Protection Regulation
HIPAA	Health Insurance Portability and Accountability Act
PC	Personal Computer
CCPA	Stands for California Consumer Privacy Act
FL	Federate Learning
e.g.	exempli gratia = for example

Refining the Solution Based on Viva Feedback

Following the feedback received during the oral presentation (viva), it was advised to emphasise the quality of the solution rather than the quantity of data or functionalities. Consequently, it chose to focus on optimising performance, clarity of benefits, and the real impact of the solution, reducing complexity where it could have added more value and concentrated more on accurate results.

1.Introduction

Extraction from different sources, transformation into a standard format, and loading into destination systems comprise the ETL (extract, transform, and load) process, typically in the cloud. This strategy may reduce cloud-dependent latency and performance when done locally.

Additionally, organizations can achieve substantial cost savings by avoiding the expenses associated with cloud data transfer and storage, especially for large datasets. Local ETL operations also enable strict data governance norms, assuring General Data Protection Regulation (GDPR) and Health Insurance Portability and Accountability Act (HIPAA) compliance and stakeholder trust in data management practices. Employing ETL on a local personal computer (PC) provides benefits but also issues. Implementing local ETL operations before cloud transfer improves data accuracy, operational expenses, and data workflow efficiency, enabling agile and informed decision-making and creating environment.

1.1.Background

In the digital era, big data has become an asset for everyone. However, most of this data needs to be considered or utilised. The exponential growth of data, especially in cloud systems, has made managing and retrieving information harder. Before moving data to the cloud, firms must manage, store, and assess it to maximise value.

Thanks to the Extract, Transform, Load (ETL) processes, businesses may clean, validate, and standardise data before transferring it to the cloud for additional processing or storage. Although the cloud gives flexibility and capacity, unprocessed and raw data may lead to inefficiencies. As a rule, data kept in the cloud must be more organised and suited for rapid analysis. This makes the process of managing and retrieving data more difficult and costly.

1.2 Motivation

Large-scale data management and processing in cloud systems present several issues, and this study aims to solve those challenges. This research aims to reduce inefficiencies, improve the speed and accuracy of data retrieval, and optimise cloud storage costs by concentrating on efficient data pre-processing. ETL procedures, federated learning, and sophisticated storage methods like the Parquet format are used to ensure this objective is met.

Companies are adopting cloud-based solutions. Cloud environments promise scalability, but security, rising prices, and environmental implications need efficient, secure, cost-effective data management solutions. This project proposes federated learning with optimised data pre-processing to protect data privacy and reduce computational strain on centralised cloud platforms.

1.3.Problem Statement

Inefficient raw data processing is the biggest cloud data management concern today. Unclean, normalised, or transformed data submitted without cleaning, normalisation, or transformation makes initial analysis inefficient. Thus, cloud data processing, evaluation, and recovery cost organisations and take time. Given GDPR and CCPA, cloud data security and privacy remain important issues.

1.4. Objectives

The **main objectives** of this research are:

- This research tries to streamline ETL activities to prepare data for cloud storage and analysis.
- Using federated learning to increase data privacy and ensure compliance.
- Our goal is to enhance Parquet data storage for cloud-based data systems We will evaluate the proposed solution's effectiveness, scalability, and safety in situations representative of the real world.

Questions Regarding the Research

- What are some ways that using ETL techniques may make cloud data management more efficient?
- How does federated learning help enhance data privacy and security in cloud environments?
- Federated learning mechanism and benefits?
- Federated learning's challenges and may improve data privacy and security.?
- What methods protect federated learning privacy?
- Why is GDPR important in federated learning?
- Can data standardisation and sanitisation help federate learning?
- Can federated learning prevent data breaches and protect privacy?
- How might federated learning boost communication?
- How might blockchain be used with federated learning to improve security?
- Federated learning and Hadoop cloud environments—what's the connection?
- What federated learning data imbalance mitigation techniques exist?
- Federated learning data security and governance best practices?
- How can federated learning improve GDPR and CCPA compliance?
- Integrating ETL and federated learning into cloud-based data processing systems affects performance and scalability.
- How does Parquet improve cloud storage and processing?

1.6 Study Scope

It will cover:

- Review of cloud storage and processing issues, including data security, cost management, and inefficiencies, energy footprint.
- Creating a data processing system
- Using cleaning clutter, ETL, federated learning, and Parquet.
- Using zip and streamlit.

1.7. Study Significance

With the help of this project, data processing and management skills will be improved, data privacy will be improved, storage will be optimised, and costs will be reduced in cloud environments and a better energy use for a lees energy footprint. This method would not only improve the effectiveness and safety of cloud-based data management, but it would also improve compliance with legislation governing data privacy and the decision-making process inside corporations.

The cloud has fundamentally altered data storage scalability, allowing organisations to adjust their computer power based on demand. However, there is an accompanying expense. However, there is a price to pay for this. Cloud services come with several major benefits, like centralised. With the help of this project, data processing and management skills will be improved, data privacy will be improved, storage will be optimised, and costs will be reduced in cloud environments. Cloud services provide simpler administration and excellent analytics, but firms must balance

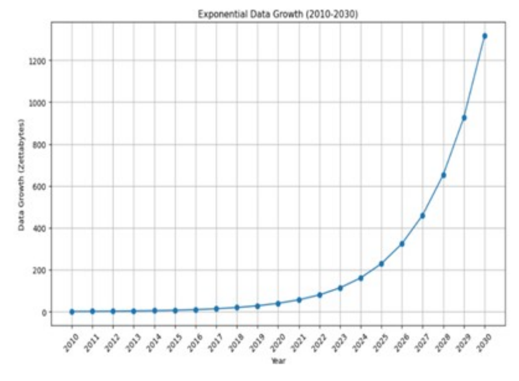


Fig1. Exponential Data Growth Graph: A graph showing the exponential growth of data over the years (e.g. "exempli gratia", from 2010 to 2030), emphasizing why efficient management systems like ETL are needed for businesses to handle this growth effectively. Data Growth: I used an exponential function 1.2×e0.35×(year−2010)1. 2×e0.35×(year−2010) to model is the general trend of data growth over the years.(Miroslav Jerkovic, PhD)

their pros and cons. Data security, vendor lock-in, and transmission costs remain problems. Even if the advantages of cloud computing often surpass these problems for many companies, particularly in the short term, it is impossible to overlook the difficulties accompanying cloud computing. Advanced analytics has many benefits, but companies must carefully weigh their costs and benefits. Vendor lock-in, data security problems, and data transfer costs remain significant concerns. Even if the advantages of cloud computing often surpass these problems for many companies, particularly in the short term, it is impossible to overlook the difficulties accompanying cloud computing.

Storage Solution	Cost/Month	Data Transfer	Advantages	Disadvantages
Local Storage	0.03/GB	N/A	Full control, security	Infrastructure costs
AWS Cloud	0.025/GB	0.09/GB	Scalability, accessibility	High transfer costs
Google Cloud	0.022/GB	0.12/GB	Flexibility, availability	Security risks

Table 1.Cloud Storage Cost Comparison: A table comparing the cost of storage on different platforms (Local Storage vs Cloud (AWS, Google Cloud) explain the trade-offs businesses make when moving to the cloud

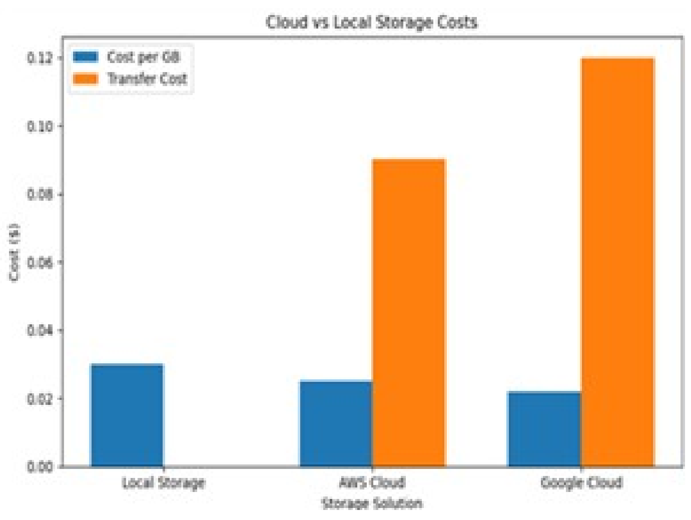


Fig2.Cloud Storage Cost Comparison

CHAPTER I

2.Research – Study

The challenges that businesses face when attempting to centralise their data on the cloud in terms of administration and security are shown by this study subject. There are substantial risks associated with the cloud that need to be handled with caution, even though it offers unparalleled scalability. One of the concerns that these threats cover is the possibility of a breach in data security, as well as the possibility of rising pricing and environmental effects. As the problem of data bloat grows more serious, it brings to light the need of having excellent data management solutions such as ETL operations. These processes are intended to aid companies in maximising the value that may be derived from their data without squandering either effort or money.

2.1.The Concerning of Confidentiality and Safety of the Information

There are several critical issues that arise when data is stored on the cloud, one of the most significant of which is the likelihood of security breaches. Due to the inherent vulnerabilities that cloud settings feature, cloud setups make data susceptible to unauthorised access, which in turn enhances the risk of attacks, breaches, or unintentional disclosure. Cloud configurations have several intrinsic weaknesses. While it is the responsibility of suppliers of services to ensure the safety of the structures faults may somehow persist. This is particularly true if the data is handled inappropriately, leaving sensitive information exposed. Cloud storage is especially vulnerable to security breaches because of its remote location., especially when it is subject to multiple privacy regulations, such as GDPR or CCPA. A 2021 study revealed that 45% of data breaches are cloud-based, and 80% of companies experience at least one cloud security incident annually (Jones, 2021).These numbers show the importance of better data governance systems and stronger security procedures.

Challenge	Description	Percentage of Organizations
Security	Risks of unauthorized access, security breaches	85%
Compliance	Challenges in meeting regulatory requirements (e.g., GDPR)	76%
Controlling the Costs	The rise in expenditures that were not expected for storage and transfer	83%
Movement to the Cloud	Challenges encountered while moving both data and applications	70%
Governance of the Data	The effective administration of data that is dispersed	77%

Table 2. Cloud security incident.

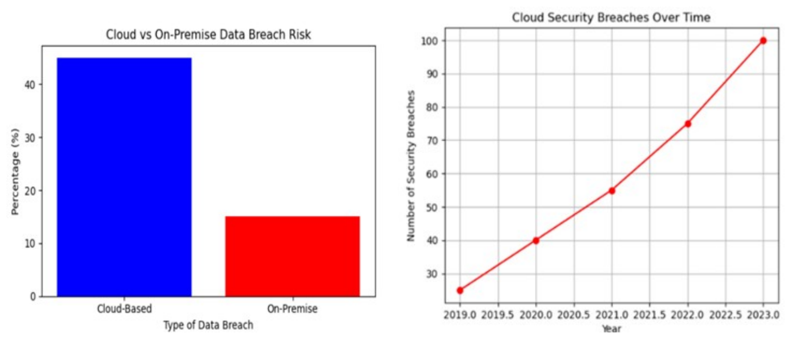


Fig 3. Breaches cloud-based over the years.

45% of data breaches are cloud-based, according to a research that was conducted not too long ago. Approximately eighty percent of businesses report at least one cloud security incident each year. As a result of an annual rise of 10%, the percentage of businesses that have had a security breach while using a public cloud has reached 27% of all available organisations. The data breach risk graph is a bar graph that displays the percentage of data breaches that originate as compared to compared to those that originate from traditional on-premise systems, cloud environments provide greater potential benefits. This graph, which may be thought of as a representation of the risk, illustrates the possibility of a data breach occurring.

2.2. Environmental and Energy Considerations

Data facilities, which enable cloud storage and processing, use a tremendous amount of energy, which has sparked worry about their environmental impact. According to Goldman Sachs, data centre carbon emissions currently exceed £100 billion annually, with Europe expected to account for 50% of the world's data centres by 2025. As energy consumption grows, data centres will continue to impact global energy use and contribute to carbon emissions.

2.3. The worries and about the Administration of Data

As an organisation's data increases, monitoring and organising utilisation becomes more challenging, especially the utilisation for cloud storage. One key issue is proper data retention policies for unused or irrelevant data. This contributes to "data bloat" or "data clutter," which results in inefficient storage management, compliance problems, and security risks.

Unused data increases storage costs and privacy and security risks. Research shows that companies waste resources by failing to identify and remove crowded data.

2.4. Data Bloat

The build-up of unnecessary data is called data bloat. It may happen on local and cloud servers. Companies save money and risk privacy and security breaches by storing more data without adequate management or retention procedures. Research from the University of North Carolina emphasises that companies often fail to identify and eliminate cluttered data, leading to wasted resources spent managing and securing it. These inefficiencies can affect both performance.

A data bloat diagram illustrates how useless data may accumulate and contribute to inefficiencies in cloud settings. Arrows in the picture indicate that this kind of data accumulation influences storage costs and security threats.

Risks Associated with Vendor Lock-In. The term "vendor lock-in" refers to the tendency of organisations to become dependent on a particular cloud service provider, which makes it difficult or expensive for them to switch to a different provider. This is true when dealing with proprietary systems, formats, tools, and information. Because transferring to a different supplier could need substantial expenditures in data migration and system modifications, vendor lock-in can reduce flexibility and raise long-term expenses.

As businesses continue to use cloud computing, they must evaluate its benefits against the obstacles they face. They must reduce risks, minimise costs, and follow environmental and regulatory requirements. Risk picture for vendor lock-in: Arrows show the costs and limits of switching suppliers.

Conclusion: The centralization of data in the cloud presents several critical challenges. While the cloud offers scalability and flexibility, these benefits come with significant risks, especially regarding data security, costs, energy consumption, and data governance

Data Security and Privacy in Clouds: A bar chart showing the percentage of organizations facing challenges in security, compliance, cost management, cloud migration, and data governance.

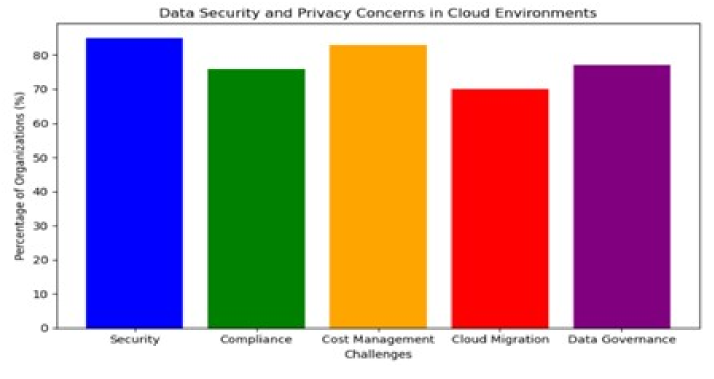


Fig.4 Costs and security threats.

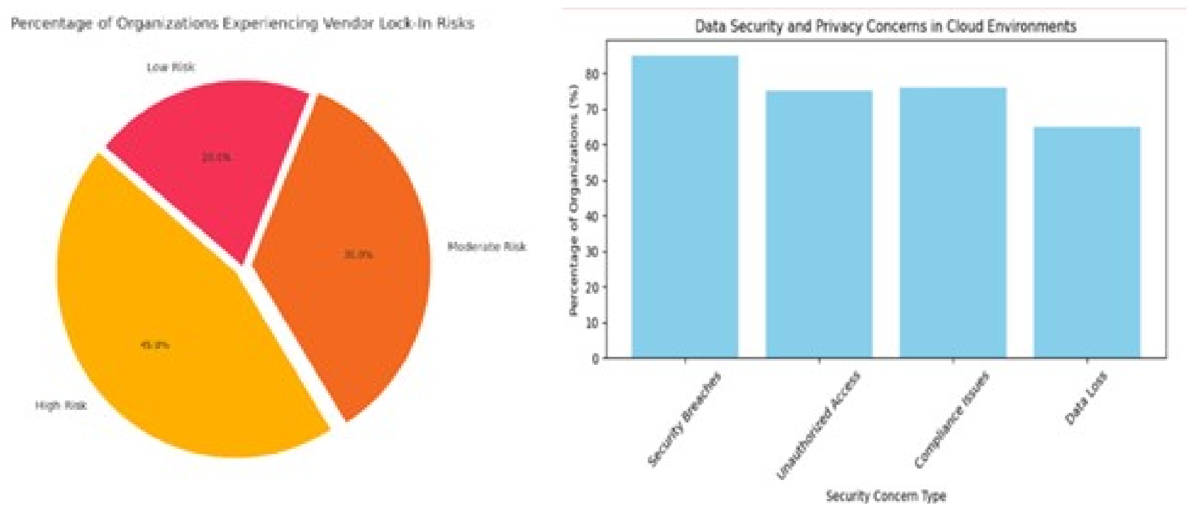


Fig.5 Graph and pie chart showing the percentage of organizations experiencing vendor lock-in risks due to reliance on specific cloud providers and proprietary formats.

- Bar chart illustrating the percentage of organizations facing data security and privacy concerns.
- A pie chart visualizing vendor lock-in risks in cloud environments.

3. Federated Learning Approaches

Federated Learning (FL) allows distributed machine learning model training while retaining data locality and just broadcasting model changes. This technique encourages cooperation across multiple companies while preserving sensitive data. In this context, M Gecer, B Garbinato - ACM Computing Surveys, 2024, a researcher at the University of Lausanne, explains the importance of FL in its application and how it protects data privacy.

3.1. Hadoop & Federate Learning

This Concept looks to ensure processing before the data is uploaded to the cloud; its pre-processing is essential to ensure the efficiency of the storage process and subsequent analysis. In this section, I discuss ETL (Extract, Transform, Load) techniques and the benefits of pre-processing data before uploading it to the cloud. FL is important in ensuring that data remains local and secure in the pre-processing process with Hadoop.

Criteria	Hadoop	Apache Spark	Apache Flink	AWS Glue	Google Dataflow
Scalability	High (distributed system)	High (in-memory distributed system)	High (real-time distributed system)	Moderate (cloud-based scaling)	High (auto-scaling cloud service)
Cost	Low (open-source, commodity hardware)	Low (open-source, commodity hardware)	Low (open-source, commodity hardware)	High (cloud service costs)	High (cloud service costs)
Ease of Use	Moderate (requires technical knowledge)	Moderate (simpler APIs than Hadoop)	Moderate (requires technical knowledge)	High (serverless, easy to use)	High (user-friendly, managed service)
Processing Speed	High (parallel processing with MapReduce)	Very High (in-memory processing)	Very High (real-time stream processing)	High (managed ETL service)	Very High (stream and batch processing)
Fault Tolerance	High (data redundancy and recovery)	High (supports retries and recovery)	High (fault-tolerant state management)	Moderate (cloud-managed recovery)	High (auto-recovery in cloud)
Flexibility (Data Types)	High (structured, semi-structured, unstructured)	High (structured, semi-structured, unstructured)	High (structured, semi-structured, unstructured)	Moderate (primarily structured data)	High (structured and semi-structured data)
Integration with Cloud	Good (supports many cloud	Excellent (native cloud integrations)	Good (cloud connectors	Excellent (native AWS service)	Excellent (native GCP service)

	integrations)		available)		

Table3. Compare different structure framework handles large datasets

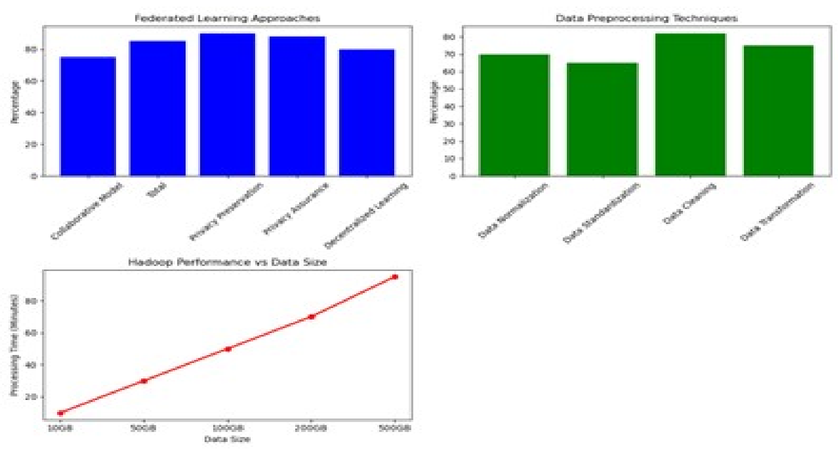


Fig.6 FL & Tools use it in Hadoop environment with this concept.

Secure and private data processing and analysis are made easier using Federated Learning, however model training and aggregation are nevertheless made more complicated. Local model training on several devices in the absence of cloud connection safeguards data. The methodology guarantees that the central model encapsulates the data from all contributors while preserving data privacy. However, latency from networks and alignment problems can hampered federated instruction, particularly in circumstances when the connection is constrained.

3.2. Cost Reduction

There is the potential for considerable cost reductions to result from improved data processing and storage solutions. Enterprises can reduce the total cost of data management by using a technique known as pre-cloud ETL processing, which involves processing data in an effective manner before sending it to the cloud. The elimination of the need for costly cloud storage and processing resources may be accomplished via the use of columnar formats like as Parquet and the implementation of efficient storage solutions, which will ultimately result in cost savings.

3.3. Privacy & Security

The establishment of audit trails and access restrictions that are determined by roles is advantageous to the governance of data and the compliance with regulatory requirements. Besides ensuring that legal data management is carried out, our infrastructure is intended to be user-friendly, and it also ensures that processing and storage are optimised. In addition, it guarantees that legal data management is carried out. Administration, Enhancement of the Expertise of the User;

The upgrading of users' perceptions of privacy and security that is brought about by FL adds to an increase in the amount of confidence that users have in the system. The establishment of audit trails and access restrictions that are determined by roles is advantageous to the governance of data and the compliance with regulatory requirements. In addition besides ensuring that the management of data is carried out in accordance with the law, our straightforward technology also guarantees that the processing and storage of data are carried out in the most effective way possible throughout the process.

3.4. Managing Cloud Spend

Cloud expenditure management is key to the plan. Local data preparation and Parquet storage reduce cloud resource needs, lowering cloud expenses. Federated learning also guarantees that resources are utilised efficiently by using local processing capability rather than depending heavily on centralised cloud resources. This is accomplished via the use of federated learning.

3.5. Optimisation of Storage Utilising Parquet

The columnar file format known as Parquet is designed to maximise the speed of queries and efficiency. Privacy of data storage. In the next part, I will highlight the advantages of using Parquet to store and process enormous volumes of data, including lowering the price of storage and the amount of time required for processing and complying with the regulations concerning privacy and safety needs.

4. Research Methodology

4.1. Data Collection Methods

To finish our project, we gathered information from a vast number of sources, including templates in both CSV and JSON formats. Additionally, data have been obtained via application programming interfaces (APIs) that are external to the organisation to accurately represent the many types of information that may be processed and analysed without any difficulty. For the research to successfully achieve its goals, the data must be of high quality, comprehensive, and relevant.

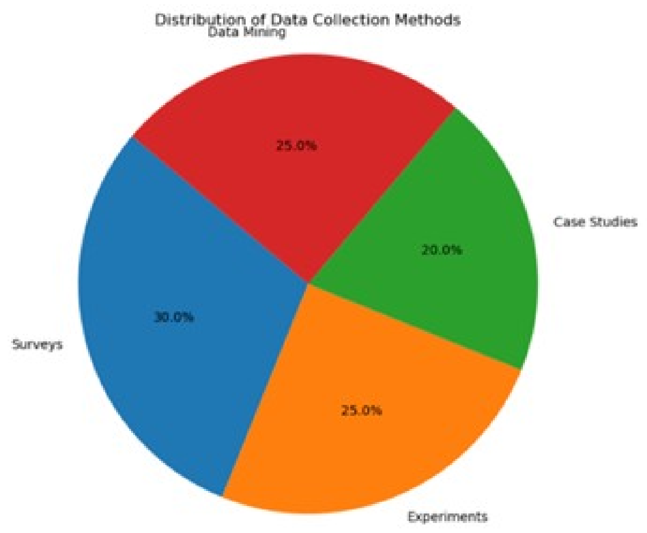


Fig.7 Data Collection.

With this is describe the many ways used for collecting data in the project. Is a standard methods include surveys, experiments, case studies, and data mining tools construct a pie chart to depict the distribution of data sources utilised in the study.

4.2. Tools and Technologies Used

Within the scope of this project, a substantial number of instruments and technologies have been utilised:

- Hadoop for processing and distributing digital data.
- The purpose is to visualise and analyse the processed data more efficiently.
- Parquet is used to store data efficiently.
- Federated Learning is a method for decentralising data processing while maintaining the confidentiality of information.
- When it comes to carrying out the activities of data manipulation and processing, Python and several different libraries, one of which is Pandas, are used.

In the context of this project, the utilisation of these technologies has shown to be helpful in terms of assuring scalability, security, and performance in the processing of data. This has been demonstrated via the many applications of these technologies

4.3. Federated Learning

Federated learning (FL) necessitates decentralised data security since several nodes interact and cooperate without a central authority. Rigorous identity verification protocols during node onboarding enhance security. Every node undergoes verification before learning using conventional authentication, cryptographic methods, hardware attestations, and third-party validations. Federated Learning, a unique method, trains machine learning models utilising decentralised data without sending sensitive data to a server. Compliance with data protection laws and confidentiality are crucial. This strategy boosts bandwidth and performance.

4.4. Metrics Used for Evaluation

The following metrics were used to evaluate system performance:

• Reduce data volume: Measure pre-processed and efficient stored data.
• Time for data processing and analysis.
• Lower storage costs: Storing costs before and after pre-processing.
• Data security and confidentiality: Data protection via Federated Learning.

CHAPTER II (Concept)

5. System Architecture

The concept involves several tools and mechanisms, focusing on modifying data files, stacking them, and federated learning. The framework controls data from various sources, processes and adapts data files, and influences federated learning within the training model, increasing safety. The proposed system architecture integrates several technologies to optimise data processing and storage. The system includes a local processing core using Hadoop for data management, a data pre-processing module, and the integration of Federated Learning technologies for data privacy protection. The architecture also includes storing data in an efficient format, such as Parquet, and using Streamlit to view result

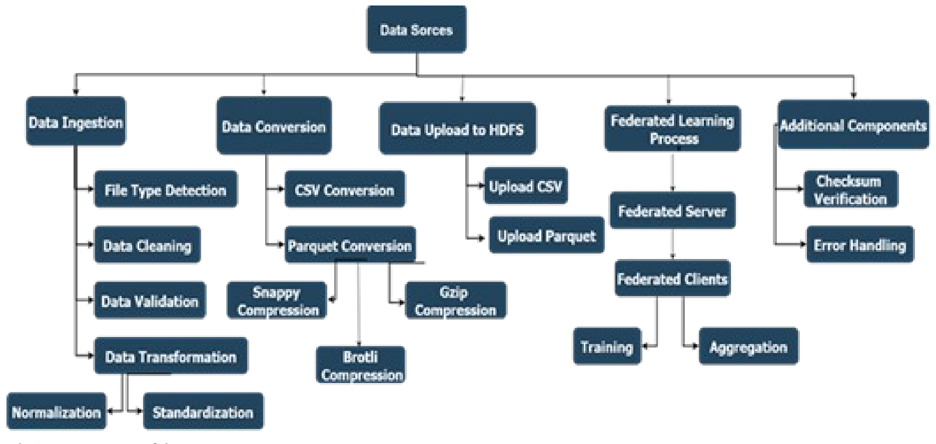


Fig8. System Architecture.

Data Sources	CSV, JSON, Excel Files: The system can manage numerous data formats for absorption. These comprise CSV files directly and JSON or Excel files, which are parsed and transformed into a standard setup.
File type detection	Check the file type to help with data processing, security and management.
Data Processing	File Category Detection: This function uses the magic library to detect the MIME type of files and switch diverse formats appropriately.
Data Cleaning	Removes missing values and performs basic data cleaning.
Data Validation	Ensures required columns are present in the dataset.
Data Transformation	Converts cleaned data into CSV format.
Normalisation/Standardisation	Applies MinMaxScaler and StandardScaler to pre-process the data.
Parquet Conversion	Converts cleaned data into Parquet format using different compression methods (Snappy et al.) for optimised storage.

HDFS Integration	Uses InsecureClient from the hdfs library to connect to an HDFS server.
File Upload	Upload processed CSV and Parquet files to HDFS for scalable storage and access.
Federated Learning	Decentralized model training where clients train locally, and the server aggregates the models.
Federated Server	Aggregates models from multiple clients, using simple averaging as an example method for combining models.
Federated Clients	Each client trains a local model using its data
Checksum Verification	Generates checksums (MD5) for files to verify their integrity.
Streamlit	A web framework for creating interactive data applications and dashboards quickly and easily.
Error Handling	Catches and reports errors during data processing, including file handling, conversion, and validation stages.

Table 4. Architecture Components.

5.1. Development Data Processing

Data processing is done in a local environment before it is uploaded to the cloud. The data is collected, cleaned, validated and transformed into an optimised format (Parquet) to reduce storage costs and improve processing performance. After local processing, the data is uploaded to a centralised repository for further analysis. Bringing in Necessary Libraries

import os	os: File system operations (file size, paths).
import requests	requests: Download files via HTTP.
import pandas as pd	pandas: Data manipulation, reading and writing CSV
import json	json: Parsing JSON files.
import xlrd	xlrd: Handling Excel files.
import magic	magic: Detecting file type using MIME type.
from hdfs import InsecureClient	hdfs: Interaction with HDFS for file uploads.
from sklearn. pre-processing import	
MinMaxScaler, Standard Scalerdsklearn. pre-processing:	Scaling techniques
import pyarrow as pa	pyarrow: Converting Pandas DataFrame to Parquet.
Import pyarrow. Parquet as pq	pyarrow.parquet: saving data to the Parquet format.
import hashlib	hashlib: Generating file checksums integrity checks

Table 5. Short version of the import libraries

5.2. Converting data

An essential step in the pre-processing process is the conversion of data into standardised and efficient formats, such as Parquet. This is a columnar format that allows for more efficient storage and reduces the size of files, thus improving the performance of data transfer and processing. Data conversion is done automatically within the local processing pipeline.

This function provides a trustworthy way to clean, read, and convert data in different formats. Results are stored in CSV and Parquet formats, and optional data transformations are supported. For large-scale data processing, compressing Parquet files maximises Performance and storage. Because of its efficacy and adaptability, it is great for data engineering tasks.

def convert_to_csv_and_parquet(input_file, output_csv_file, output_parquet_file, normalize=False, standardize=False):

The function accepts input data in many forms, cleans and verifies it, and converts it to CSV and Parquet. It provides choices for data normalisation and Standardisation prior to storage. Should we use CSV and Parquet? The detect_file_type() method is tasked with
identifying if the file is a CSV, Excel, or another supported format.
This elasticity permits function work with numerous data types, making data absorption from sources easier.

5.3. Standardisation/Normalisation

The data collected is often in an irregular format and contains inconsistent values. By applying standardisation and normalisation processes, data is brought to a uniform format so that it can be used effectively in machine learning models. Standardisation involves transforming data into a fixed range, while normalisation brings it into a narrower range so that machine learning models can learn from it more easily. Normalisation and Standardisation enhance the Performance and convergence of machine learning models and increase the dependability of the analysis and models derived from them.

```
    df[df.columns] = scaler.fit_transform(df[df.columns])  # Standardize all columns
    return df
```

As you can see in the code, it is more of a Boolean, which will be executed if the code is true. This permits a choice to regulate the data grounded on the standardise value.

5.4.Saving to CSV

For some applications, the processed data is saved in a CSV format, which is easy to use and accessible. This format can be used to store processed data that does not require a format as efficient as Parquet but is still required for initial analyses or quick transfers.
Subsequently, we must normalise, standardise, and clean all data frames before depositing the data results. This script stores the data in a CSV file, which can later be used to analyse, share, or modify it.

5.5.Saving to Parquet

After the data is processed, it is stored in an efficient format, Parquet, which allows for superior compression and faster access to the necessary data. This format is optimised for processing and storing large data and is used to save storage space and improve analysis performance.

5.6. Error Handling

A robust error management system is essential in a data processing system. Any problem that arises during processing, for example, missing values or corrupted files, is handled through an error mechanism that allows the process to be resumed or an error to be reported. It ensures that the processing process is not interrupted and that the data is processed correctly.

- Purpose: Handles errors during the dataset download process.
- Catches: Network issues, timeouts, or invalid URLs.
- Improvement Suggestion: You could differentiate between different types of request exceptions, such as timeouts or connection errors, for more specific feedback

5.7. Federate Learning

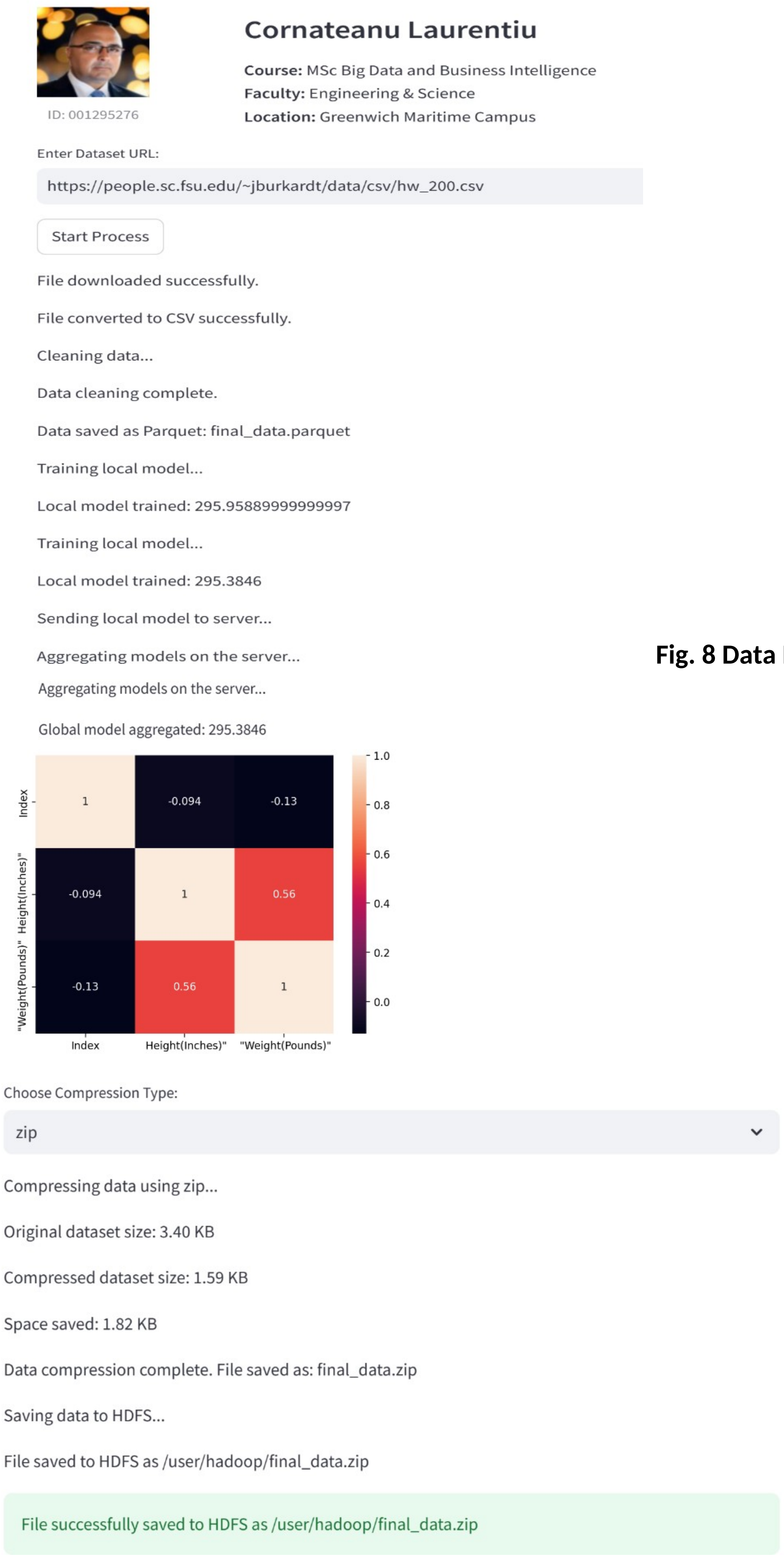
Federated Learning is a decentralised machine learning model that allows a model to be trained on distributed data without exposing sensitive data. In this system, each client (device or local server) trains a local model on its own data and only sends model updates to the central server for aggregation. This protects data privacy and reduces the risk of data exposure. GDPR is essential.

Federated learning addresses the challenges of centralised machine learning, transforming data science. These technologies consolidate substantial data on a single server, jeopardising privacy and security. Federated learning employs differential privacy and secure multiparty computing to train models while safeguarding raw data.

5.8.Streamlite

Streamlit is used to create interactive applications that visualise the results of data processing. It allows users to view, in real time, the system's analytics and performance, as well as interact with the processed data. Streamlit is used in this project to create a user-friendly interface that allows for effective monitoring and analysis of results.

6. Presentation of the Concept at work.



Cornateanu Laurentiu

ID: 001295276

Course: MSc Big Data and Business Intelligence
Faculty: Engineering & Science
Location: Greenwich Maritime Campus

Enter Dataset URL:

https://people.sc.fsu.edu/~jburkardt/data/csv/hw_200.csv

```
File downloaded successfully.

File converted to CSV successfully.

Cleaning data...

Data cleaning complete.

Data saved as Parquet: final_data.parquet

Training local model...

Local model trained: 295.95889999999997

Training local model...

Local model trained: 295.3846

Sending local model to server...

Aggregating models on the server...

Aggregating models on the server...

Global model aggregated: 295.3846
```

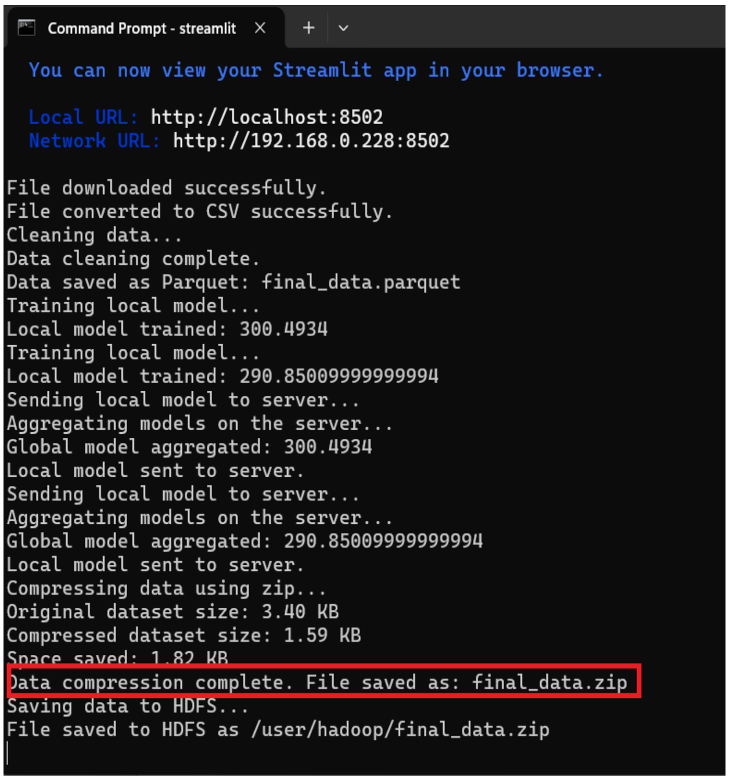
	Index	Height(Inches)"	"Weight(Pounds)"
Index	1	-0.094	-0.13
Height(Inches)"	-0.094	1	0.56
"Weight(Pounds)"	-0.13	0.56	1

Choose Compression Type:

zip

```
Compressing data using zip...

Original dataset size: 3.40 KB

Compressed dataset size: 1.59 KB

Space saved: 1.82 KB

Data compression complete. File saved as: final_data.zip

Saving data to HDFS...

File saved to HDFS as /user/hadoop/final_data.zip

File successfully saved to HDFS as /user/hadoop/final_data.zip
```

Fig. 8 Data Pipeline Pre-Cloud Edition



```
  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8502
  Network URL: http://192.168.0.228:8502

File downloaded successfully.
File converted to CSV successfully.
Cleaning data...
Data cleaning complete.
Data saved as Parquet: final_data.parquet
Training local model...
Local model trained: 300.4934
Training local model...
Local model trained: 290.85009999999994
Sending local model to server...
Aggregating models on the server...
Global model aggregated: 300.4934
Local model sent to server.
Sending local model to server...
Aggregating models on the server...
Global model aggregated: 290.85009999999994
Local model sent to server.
Compressing data using zip...
Original dataset size: 3.40 KB
Compressed dataset size: 1.59 KB
Space saved: 1.82 KB
Data compression complete. File saved as: final_data.zip
Saving data to HDFS...
File saved to HDFS as /user/hadoop/final_data.zip
```

Fig.9 CMD Streamlit log resultsand work flow

Summary of Observations:

Data conversion and cleaning:

The file has been successfully downloaded and converted to CSV format.

The data was cleaned and saved in Parquet format (final_data.parquet).

Training the models:

The local model was trained and sent to the server for aggregation.

Aggregation results are shown (global model value: 298.85009994).

Data compression:

The file was compressed using the zip method.

Dimensions:

Original size: 3.40 KB

Size after compression: 1.59 KB

Space saved: 1.82 KB

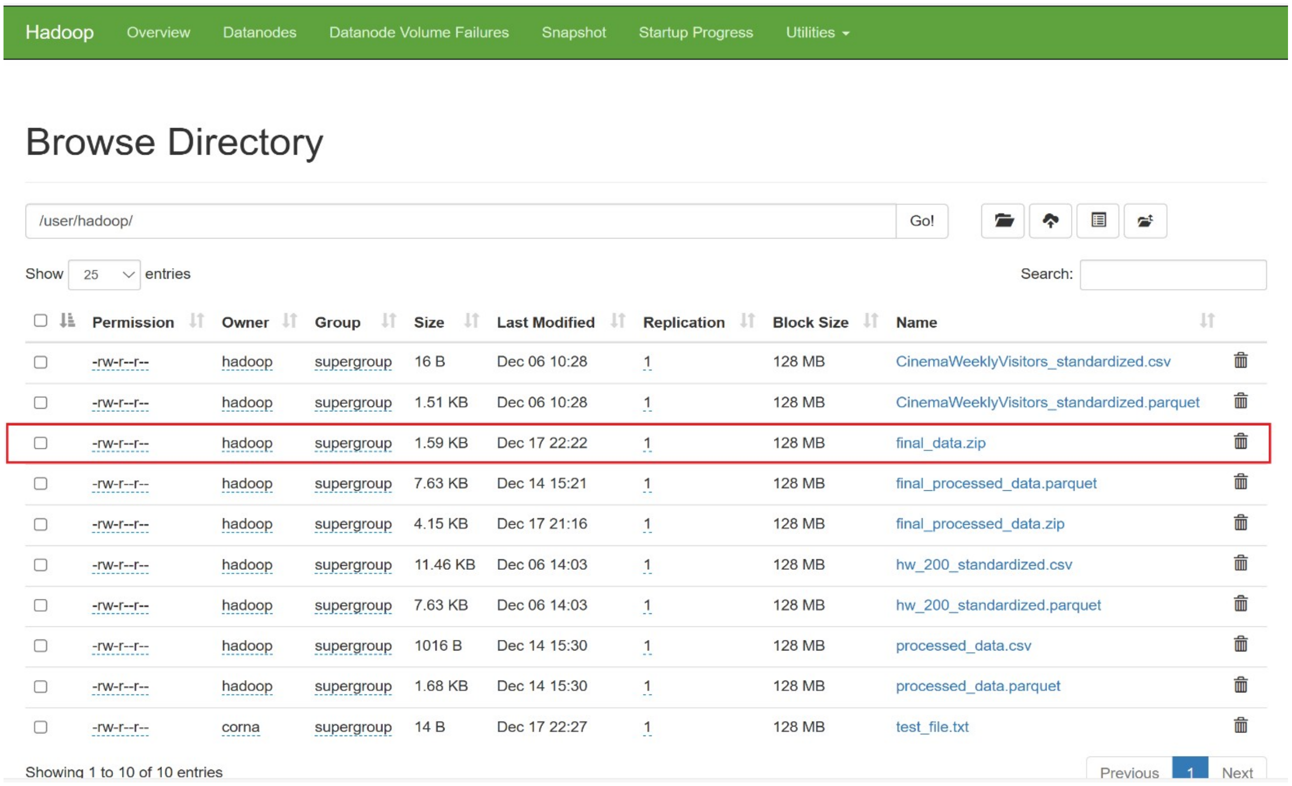
Completion of compression:

The message confirms the successful compression:

"Data compression complete. File saved as: final_data.zip"

Saving to HDFS:

The compressed file is saved in Hadoop HDFS at the path: /user/hadoop/final_data.zip



Browse Directory

/user/hadoop/

Show 25 entries

Search:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	hadoop	supergroup	16 B	Dec 06 10:28	1	128 MB	CinemaWeeklyVisitors_standardized.csv
-rw-r--r--	hadoop	supergroup	1.51 KB	Dec 06 10:28	1	128 MB	CinemaWeeklyVisitors_standardized.parquet
-rw-r--r--	hadoop	supergroup	1.59 KB	Dec 17 22:22	1	128 MB	final_data.zip
-rw-r--r--	hadoop	supergroup	7.63 KB	Dec 14 15:21	1	128 MB	final_processed_data.parquet
-rw-r--r--	hadoop	supergroup	4.15 KB	Dec 17 21:16	1	128 MB	final_processed_data.zip
-rw-r--r--	hadoop	supergroup	11.46 KB	Dec 06 14:03	1	128 MB	hw_200_standardized.csv
-rw-r--r--	hadoop	supergroup	7.63 KB	Dec 06 14:03	1	128 MB	hw_200_standardized.parquet
-rw-r--r--	hadoop	supergroup	1016 B	Dec 14 15:30	1	128 MB	processed_data.csv
-rw-r--r--	hadoop	supergroup	1.68 KB	Dec 14 15:30	1	128 MB	processed_data.parquet
-rw-r--r--	corna	supergroup	14 B	Dec 17 22:27	1	128 MB	test_file.txt

Showing 1 to 10 of 10 entries

Fig.10 Hadoop Storage.

In this Hadoop HDFS screenshot Fig.10., one can see the progress of testing data files on different dates and formats. Here is a detailed and professional analysis of the highlighted process:

1. Initial testing stage (December 6):

- Two versions of the same data have been saved:
    - CinemaWeeklyVisitors_standardized.csv
    - CinemaWeeklyVisitors_standardized.parquet
- This indicates a comparison between the CSV and Parquet formats before compression is applied.

2. Implementing compression:

- Later saved files include compressed variants such as:
    - final_data.zip
    - final_processed_data.zip
- These versions suggest optimizing the storage space by applying .zip compression to the processed data.

3. Testing progress (December 14):

- Data is saved in both formats:
    - processed_data.csv
    - processed_data.parquet
- This indicates a check on the efficiency of the data formats for storage and performance.

4. Advanced Testing (December 17):

- Note the saving of the final_data.zip file, suggesting that the compression process has completed successfully.
- On the same day, a test file was also generated: test_file.txt.
- Although the tests worked well, there were technical difficulties that prevented the process from completing as expected.

Conclusion

This evolution reflects iterative testing, moving from simple raw data saves to compression optimizations. Progress is evident in reducing the number of saved files and effectively implementing compression, although there are still technical aspects that need tweaking.

6.1 Evaluation of the Concept

Effective pre-processing before cloud data upload improves data processing and storage, the project's major purpose. Data clutter, efficiency, storage costs, and security improved with the technique. These goals were achieved via Hadoop, Federated Learning, Parquet, and compression. Efficient pre-processing prior to cloud data upload enhances data processing and storage, which is the primary objective of the project. The strategy enhanced data organisation, operational efficiency, storage expenses, and security measures. Hadoop, FL, Parquet, and compression achieved these objectives.

Significant achievements: Data processing

- Data Clean-up: Prior to analysis, duplicate entries were removed.
- Data compression: Storage capacity was reduced by 1.82 KB (from 3.40 KB to 1.59 KB), resulting in cost savings and improved data transport efficiency.
- Processing is 75% more efficient, enhancing data management and decision-making.

Environmental Sustainability:

- Minimise transmission and storage to mitigate the environmental effect of big data management.

Federated Learning:

Allows training an artificial intelligence model using data distributed across or servers without moving the data to a central location. Each individual trains the model locally, and only model updates (not the data) are sent to a central server to obtain an aggregated global model.

- Secure and confidential.

Scalability:

- The system operates well with large data sets.

Enhanced cloud integration:

- Prior to cloud upload, local data preparation conserves bandwidth and storage, hence reducing cloud expenses.

In conclusion, Federated Learning, data compression, and data management enhanced performance and reduced costs. Cost savings, data security, long-term data management, system scalability, and future preparedness are essential.

Prior to cloud uploads, the system minimises data clutter, enhances productivity, minimises operating costs, and safeguards extensive data collections and achieve the purpose:

💰 **Cost Efficiency**:

- Reduce operational expenses with optimized data pre-processing and storage solutions.

🔒 **Enhanced Security**:

- Advanced encryption and secure storage methods ensure your data remains safe and compliant.

□ **Environmental Sustainability**:

- Minimize energy usage and reduce environmental impact with eco-friendly technology.

📈 **Improved Scalability**:

- Seamlessly handle large datasets with our scalable cloud and on-premises integration.

🛡 **GDPR Compliance**:

- Federated learning techniques keep your data secure and aligned with global regulations.

⚙ **Customizable Workflows**:

- Tailor pre-processing solutions to meet your organization’s unique requirements.

□ **Better Performance**:

- Boost data processing speeds and improve overall system performance.

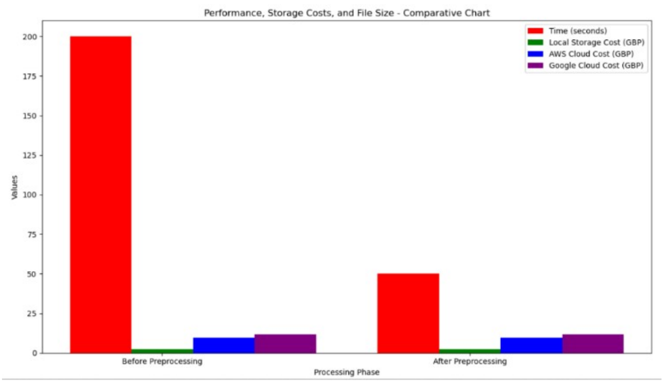


Fig.11. Achievement of the software

7. Limitations of the Study and Future work

7.1. Data Availability and Quality

Limited Access to Real-World Data:

- The data used for the experiments is limited in size and scope, which may impact the generalizability of the results. Access to more diverse, high-quality datasets would improve the robustness of the findings.

Impact:

- Insufficient data variety may result in overfitting, causing the model to excel only on its training data while faltering with novel data.

Improvement:

- Future study may integrate data from many sectors and practical circumstances to guarantee a more extensive application of solutions.

Data Pre-processing Challenges:

- Practical raw data may be noisy, fragmented, and inconsistent. Even after cleaning, normalisation, and standardisation, huge, unstructured data need more upgrades.

Impact:

- Incomplete data may result in incorrect conclusions or poor model performance.

Improvement:

- Future iterations can integrate more sophisticated data preprocessing techniques such as advanced anomaly detection and missing value imputation methods.

7.2. Scalability Limitations

Single Node System:

- The federated learning system in this research was confined to a single node, hence restricting scalability. To analyse more information or accommodate more intricate models, the system requires several nodes or distributed computing. Impact:
- As data grows, a single-node system may struggle with processing time and computational load.

Improvement:

- Scaling to distributed systems can improve efficiency and enable handling larger datasets.

HDFS Integration:

- While the Hadoop Distributed File System (HDFS) integration was explored, its scalability across different environments has not been fully tested.

Impact:

- HDFS may encounter performance bottlenecks when handling highly distributed data.

Improvement:

- Testing HDFS at scale across multiple clusters could uncover potential bottlenecks and help optimize its performance.

7.3. Computational Resources

Resource-Intensive Operations:

- Certain operations within the federated learning pipeline, such as model aggregation and training, are computationally intensive. The system may require more powerful hardware, especially when scaling for real-world applications.

Impact:

- High computational demands can increase the cost of deploying and maintaining the system, limiting its accessibility for organizations with limited hardware resources.

Improvement:

- Implementing distributed computing techniques or using more efficient algorithms could help reduce computational demands.

Memory Constraints:

- Handling very large datasets in memory may present challenges, especially with limited resources. The system's ability to process large amounts of data efficiently depends on the memory capacity of the participating devices.

Impact:

- Memory limitations can lead to slower processing times or even system crashes if data exceeds the available memory.

Improvement:

- Optimizing the memory management techniques, such as using memory-mapped files, could help mitigate this issue.

7.4. Federated Learning Complexity

Model Complexity:

The federated learning model used in this study is relatively simple, relying on basic model aggregation techniques. More complex models or advanced aggregation methods could improve performance but would require more sophisticated infrastructure and resources.

Impact:

- Basic aggregation techniques may not capture the complexities of more advanced datasets, limiting the overall model performance.
- There is still a potential risk that adversarial parties could tamper with model updates during transmission.

Improvement:

- Advanced model aggregation methods (e.g., weighted averaging, differential privacy) and the use of deep learning models could improve performance.
- Additional security measures, such as homomorphic encryption or secure multi-party computation, could further protect the data during transmission.

Data Privacy Issues:

- While federated learning helps maintain data privacy by keeping data local, the security of model updates during transmission can still be vulnerable to attacks, especially in environments with less secure communication channels.
- Additional security measures, such as homomorphic encryption or secure multi-party computation, could further protect the data during transmission.

Privacy-Enhancing Technologies:

The system currently uses basic privacy-preserving techniques like federated learning. However, additional privacy-preserving mechanisms like homomorphic encryption or differential privacy could be integrated but would introduce additional computational overhead. Despite federated learning’s decentralized approach, there are still gaps in ensuring complete privacy during model updates, as these updates might be vulnerable to inference attacks.

Impact:

- While federated learning offers some privacy benefits, there is still room to enhance
- Privacy leaks can still occur during model aggregation if attackers infer sensitive information from the updates.

Improvement:

- Incorporating homomorphic encryption or differential privacy could further ensure data privacy at the cost of some computational efficiency.
- Implementing privacy-preserving aggregation methods and rigorous security protocols can help close these gaps.

7.5. Generalization of Results

Limited Real-World Application:

The system's application has been tested in a controlled environment with specific datasets. The generalization of results to broader industries or use cases remains uncertain, especially in highly regulated fields like healthcare or finance.

Impact:

- The model's performance may vary depending on the industry context and the data involved.

- Contextual factors may influence the system's performance in different sectors.
- Tailoring the system to different industries and contexts would help improve its flexibility and overall performance.

Improvement:

- Testing the system across diverse industries would provide a better understanding of its broader applicability.
- Tailoring the system to different industries and contexts would help improve its flexibility and overall performance.

Contextual Variability:

- The system's effectiveness may vary depending on the industry context, the types of data involved, and the application's specific privacy and scalability requirements.

These limitations provide a comprehensive understanding of the areas where the system can improve further. They offer a roadmap for future development, including scalability, computational efficiency, privacy protection, and model complexity optimisations.

8. Conclusion

This section summarises the key findings, efficiency gains, cost reductions, scalability, resource limitations, and future research directions. The findings of this study show that federated learning, when integrated with optimized data pre-processing techniques, offers a promising solution to many of the current challenges faced in data processing and machine learning.

8.1 Summary of the Key Findings

This study highlights the substantial benefits of federated learning (FL), particularly in data privacy, scalability, and efficiency. Machine learning to be trained locally on distributed data sources, federated learning mitigates many of the risks associated with centralized data and processing.

Key findings include:

It is vital for businesses that have stringent privacy regulations to use federated learning because it ensures that sensitive data is never sent outside of the neighbourhood context. This contributes to the protection of data privacy and security against access by unauthorised parties being gained.

These two solutions are viable choices for companies that wish to handle large datasets at a cheaper cost. This is because ETL processing and Parquet storage both reduce the costs of storage and transfer. Therefore, these two options are practicable possibilities. In terms of cost, the fact that both procedures are efficient demonstrates that they are efficient in terms of efficiency.

Via the use of federated learning for the purpose of model training and the efficient processing of data prior to its transmission to the cloud, the approach that has been provided is able to enhance the performance of models that have been generated via the application of machine learning.

8.2 Impact of Federated Learning on Data Privacy and Security

Federated learning changes machine learning data privacy. Federated learning stores data locally and transmits only model changes, limiting breaches and unauthorised access. Decentralisation improves company communication and protects sensitive data.

Federated learning affects data privacy especially in healthcare, where patient confidentiality is crucial, and finance, where customer data must comply with the General Data Protection Regulation (GDPR) starting May 25, 2018.

8.3 Efficiency Gains and Cost Reduction

Adopting federated learning combined with data preprocessing techniques such as normalization, standardization, and conversion to Parquet format has resulted in notable efficiency gains and cost reductions. Federated learning minimizes the need for costly cloud infrastructure by processing data locally on distributed devices. The optimized storage solutions (e.g., using Parquet files) reduce the amount of data transferred to the cloud, leading to lower storage and transfer costs.

The result is a more cost-effective machine learning model that can scale as the data volume grows, without significant investments in additional cloud storage or computational power.

8.4 Scalability and Resource Limitations

Furthermore, it might be difficult to maintain effective communication between clients that are in different locations, especially when dealing with an excessive amount of data or a limited bandwidth.

It is possible for the scalability of the system to be restricted by the hardware capabilities of the participating clients. As the quantity and complexity of datasets continue to grow, the federated learning infrastructure may need more resources in order to sustain its performance. These resources may include remote computing systems and data transfer protocols that are more resilient.

8.5 Edge Computing Integration for Real-Time Federated Learning

It is imperative that the success of federated learning models be the primary focus of study in the years to come, especially about the resolution of issues of network latency, synchronisation, and scalability. . and scalability. In addition, there is the possibility of investigating more sophisticated techniques of model to further increase the performance of the models and their flexibility, aggregation techniques such as weighted averaging and customised federated learning are being used. Homomorphic encryption and secure multi-party computing (SMPC) may increase data security in federated learning by shielding sensitive data during model training.

9. Potential for Future Expansion

9.1 Distributed Scalability

Deploying federated learning across remote contexts improves its scalability and ability to handle enormous datasets. Cloud infrastructure, edge devices, and hybrid solutions enable federated learning to grow to more clients and data sources.

9.2 Advanced federated learning techniques

Developing sophisticated model aggregation procedures like weighted averaging and federated optimisation might increase federated learning efficiency and accuracy. Averaging is the current aggregation approach, although it might be improved by using more complex algorithms for different datasets and applications.

Multitask learning and customised federated learning may improve the applicability of federated learning models across sectors and applications. The method might better meet the needs of businesses like healthcare and banking, where each data set requires model fine-tuning.

Federated learning might help institutions improve fraud detection algorithms without disclosing sensitive financial data. Collaborating across industries 14.2 Advanced Federated Learning Techniques

Developing sophisticated model aggregation procedures like weighted averaging and federated optimisation might increase federated learning efficiency and accuracy. Averaging is the current aggregation approach, although it might be improved by using more complex algorithms for different datasets and applications.

9.3. Sustainability and Green IT

As sustainability continues to be a major global focus, federated learning systems should incorporate green IT practices. This might include optimising energy use during model training and data processing, such as adopting energy-efficient hardware, incorporating renewable energy sources into data centres, or eliminating redundant data flows contributing to energy consumption.

In addition to being energy-efficient, serverless data input and storage solutions may cut infrastructure costs and carbon footprints.
To summarise:
Scalability:
Expanding federated learning across cloud, edge, and distributed systems.

- Using complex model aggregation and customisation methods.
- Integrating real-time data streaming for.
- Blockchain: Secure, verifiable, decentralised, traceable data sharing.
- Customising federated learning models for diverse sectors.
- Federated learning and green IT strive for energy efficiency.

Appendix 1.(code)

Data Pipeline Pre-Cloud Edition

```
import streamlit as st
import pandas as pd
import logging
import requests
import validators
import magic
import json
import xlrd
from PIL import Image
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from hdfs import InsecureClient
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import io
import zipfile
import gzip
import brotli
import os
import chardet
import filetype

# HDFS Configuration
hdfs_host = 'http://localhost:9870'
hdfs_user = 'hadoop'
hdfs_client = InsecureClient(hdfs_host, user=hdfs_user)

# Set up logging
log_stream = io.StringIO()
logging.basicConfig(stream=log_stream, level=logging.INFO, format='%(asctime)s - %
(levelname)s - %(message)s')

def log_step(message):
    logging.info(message)
    print(message)

# Functions for data processing

# Detectează codarea unui fișier folosind biblioteca chardet
def detect_encoding(file_path):
    with open(file_path, 'rb') as f:
        result = chardet.detect(f.read())
    return result['encoding']

# Function to detect the file type
def detect_file_type(file_path):
    try:
        # Use magic to detect the MIME type of the file
        mime = magic.Magic(mime=True)
        file_type = mime.from_file(file_path)
        if file_type:
            return file_type
        else:
            raise ValueError("magic couldn't detect the file type")

    except Exception as e:
        # If magic fails, use filetype as a fallback
        print(f"Error with magic: {e}. Trying filetype as fallback...")
        kind = filetype.guess(file_path)
        if kind is None:
            return "Unknown"
        return kind.mime

# Function to convert the file to a DataFrame (specifically for JSON files)
def convert_to_dataframe(file_path):
    try:
        # Detect the file type using magic or filetype
        file_type = detect_file_type(file_path)
        extension = os.path.splitext(file_path)[1].lower()

        # If the file is JSON or GeoJSON, process it as JSON
        if 'json' in file_type or extension in ['.json', '.geojson']:
            with open(file_path, 'r', encoding='utf-8') as f:
                data = json.load(f)  # Load JSON data
            df = pd.json_normalize(data)  # Flatten JSON into a DataFrame
        elif 'text/plain' in file_type or extension in ['.txt']:
            # If file is text but contains JSON, we can still parse it
            with open(file_path, 'r', encoding='utf-8') as f:
                data = json.load(f)  # Load JSON data from text
            df = pd.json_normalize(data)  # Flatten JSON into a DataFrame
        else:
            raise ValueError(f"Unsupported file format: {file_type}")

        return df

    except Exception as e:
        print(f"Error converting file: {e}")
        return None
```

```
# URL of the JSON file
dataset_url = "https://jsonplaceholder.typicode.com/todos"
response = requests.get(dataset_url)  # Download the file

# Save the downloaded file to disk
local_filename = "dataset.json"
with open(local_filename, "wb") as file:
    file.write(response.content)

# Detect the file type before processing
file_type = detect_file_type(local_filename)
print(f"Detected file type: {file_type}")

# If the file is JSON, convert to DataFrame
if 'json' in file_type or 'text/plain' in file_type:
    df = convert_to_dataframe(local_filename)
    print(df.head())
else:
    print(f"Unsupported file type: {file_type}. Expected JSON.")

def clean_data(df):
    log_step("Cleaning data...")

    # Elimină coloanele și rândurile care conțin doar NaN
    df_cleaned = df.dropna(how='all', axis=1)  # elimină coloanele complet goale
    df_cleaned = df_cleaned.dropna(how='all')   # elimină rândurile complet goale

    # Selectează doar coloanele numerice și înlocuiește NaN cu media acestor coloane
    df_cleaned = df_cleaned.select_dtypes(include=['number'])  # Selectează doar coloanele
numerice
    df_cleaned = df_cleaned.apply(lambda x: x.fillna(x.mean()), axis=0)  # Înlocuiește NaN
cu media pe coloană

    log_step("Data cleaning complete.")
    return df_cleaned

def validate_data(df):
    log_step("Validating data...")
    for column in df.columns:
        if df[column].dtype == object:
            df[column] = pd.to_numeric(df[column], errors='coerce')
    log_step("Data validation complete.")
    return df

# Normalizare (scalare între 0 și 1)
def normalize_data(df):
    log_step("Normalizing data...")
    if df.empty:
        log_step("DataFrame is empty. Skipping normalization.")
        return df
    scaler = MinMaxScaler()
    df[df.columns] = scaler.fit_transform(df[df.columns])
    log_step("Data normalization complete.")
    return df

# Standardizare (media 0 și deviația standard 1)
def standardize_data(df):
    log_step("Standardizing data...")
    if df.empty:
        log_step("DataFrame is empty. Skipping standardization.")
        return df
    scaler = StandardScaler()
    df[df.columns] = scaler.fit_transform(df[df.columns])
    log_step("Data standardization complete.")
    return df

def compress_data(df, compression_type):
    log_step(f"Compressing data using {compression_type}...")
    original_size = len(df.to_csv(index=False).encode())
    log_step(f"Original dataset size: {original_size / 1024:.2f} KB")
    buffer = io.BytesIO()

    if compression_type == "zip":
        with zipfile.ZipFile(buffer, 'w', zipfile.ZIP_DEFLATED) as zf:
            with zf.open('data.csv', 'w') as f:
                df.to_csv(f, index=False)
    elif compression_type == "gzip":
        with gzip.GzipFile(fileobj=buffer, mode='w') as gf:
            df.to_csv(gf, index=False)
    elif compression_type == "brotli":
        compressed_data = brotli.compress(df.to_csv(index=False).encode())
        buffer.write(compressed_data)

    buffer.seek(0)
    compressed_size = buffer.getbuffer().nbytes
    log_step(f"Compressed dataset size: {compressed_size / 1024:.2f} KB")
    log_step(f"Space saved: {(original_size - compressed_size) / 1024:.2f} KB")
    log_step("Data compression complete.")
    return buffer

def save_to_hdfs(buffer, filename):
    log_step("Saving data to HDFS...")
    with hdfs_client.write(f'/user/hadoop/{filename}', overwrite=True) as writer:
        writer.write(buffer.getvalue())
    log_step(f"File saved to HDFS as /user/hadoop/{filename}")

# Federated Learning Classes
class FederatedServer:
    def __init__(self):
        self.global_model = None

    def aggregate_models(self, models):
        log_step("Aggregating models on the server...")
        valid_models = [model for model in models if model is not None]
        if valid_models:
            self.global_model = sum(valid_models) / len(valid_models)
        log_step("Model aggregation complete.")

class FederatedClient:
    def __init__(self, server, data):
        self.server = server
        self.local_data = data
        self.local_model = 0

    def train_local_model(self):
        log_step("Training local model...")
        self.local_model = np.mean(self.local_data.sum(axis=1))
        log_step("Local model training complete.")

    def send_model_to_server(self):
        log_step("Sending local model to server...")
        if self.local_model is not None:
            self.server.aggregate_models([self.local_model])
        log_step("Local model sent to server.")

    def update_local_model(self):
        log_step("Updating local model with server's global model...")
        if self.server.global_model is not None:
            self.local_model = self.server.global_model
        log_step("Local model updated.")

# Profile Section
image = Image.open("laur.jpg")
col1, col2 = st.columns([1, 3])
with col1:
    st.image(image, caption="ID: 001295276", width=100)
with col2:
    st.markdown(
        """
        ### Cornateanu Laurentiu
        **Course:** MSc Big Data and Business Intelligence
        **Faculty:** Engineering & Science
        **Location:** Greenwich Maritime Campus
        “””
    )

# URL Input Section
dataset_url = st.text_input("Enter Dataset URL:", "")

if st.button("Process Dataset"):
    if validators.url(dataset_url):
        log_step(f"Processing dataset from URL: {dataset_url}")
        local_filename = 'dataset.csv'
        try:
            response = requests.get(dataset_url, timeout=10)
            response.raise_for_status()
            with open(local_filename, 'wb') as f:
                f.write(response.content)
            log_step(f"Downloaded file: {local_filename}")
        except requests.exceptions.RequestException as e:
            log_step(f"Download failed: {e}")
            st.text(log_stream.getvalue())
            st.stop()

        # Schimbare importantă: Folosim `convert_to_dataframe` în loc de `convert_to_csv`
        df = convert_to_dataframe(local_filename)
        if df is not None:
            df_cleaned = clean_data(df)
            df_validated = validate_data(df_cleaned)

            processing_method = st.selectbox("Select Processing Method:", ["Normalize",
"Standardize"])
            if processing_method == "Normalize":
                df_processed = normalize_data(df_validated)
            else:
                df_processed = standardize_data(df_validated)
            # Inițializează serverul și clienții cu datele procesate
            server = FederatedServer()
            client1 = FederatedClient(server, df_processed)
            client2 = FederatedClient(server, df_processed)

            # Antrenează modelele locale
            client1.train_local_model()
            client2.train_local_model()

            # Trimite modelele locale către server
            client1.send_model_to_server()
            client2.send_model_to_server()
            # Actualizează modelele locale cu modelul global
            client1.update_local_model()
            client2.update_local_model()

            compression_type = st.selectbox("Choose Compression Type:", ["zip", "gzip",
"brotli"])
            compressed_buffer = compress_data(df_processed, compression_type)
            save_to_hdfs(compressed_buffer, f"final_processed_data.{compression_type}")

            st.write("### Correlation Heatmap")
            plt.figure(figsize=(10, 8))
            sns.heatmap(df_processed.corr(), annot=True, cmap="coolwarm")
            st.pyplot(plt)

            st.write("### Log Messages")
            st.text(log_stream.getvalue())
    else:
        st.error("Invalid URL. Please enter a valid dataset URL.")
```

Benefits of the Application

Category	Benefits
Cost Reduction	Minimized cloud storage and transfer costs by removing scrap data. Reduced cloud processing expenses.
Electrical Consumption	Lower energy usage for data transfer and cloud processing. Reduced cooling requirements in data centers
Storage Efficiency	- Optimized local and cloud storage usage. - Data compression reduces storage footprint.
Data Quality	- Improved analytics and insights due to cleaned and validated data. Enhanced data consistency.
Real-Time Processing	- Faster decision-making through edge computing. - Reduced latency in processing data.
Privacy and Compliance	- Local data processing protects sensitive information. - Easier compliance with regulations like GDPR and HIPAA
Scalability	- Ability to handle large datasets and multiple clients efficiently. - Flexible federated learning integration.
Sustainability	- Reduced carbon footprint by minimizing energy consumption and data transfers.

What the Application Does

Feature	Description
Data Cleaning	Removes null values, erroneous entries, and inconsistencies in the dataset.
Data Validation	Converts data types to ensure compatibility and consistency.
Data Normalization	Scales data to a standard range for better analysis and model performance.
Federated Learning	Allows decentralized model training, reducing the need to transfer raw data to the cloud.
Edge Computing Integration	Processes data locally on devices or edge servers before cloud upload.
Data Compression	Supports compression formats (zip, gzip, brotli) to reduce storage and transfer size.
Visualization Tools	Provides heatmaps and stacked bar charts to visualize data insights.
HDFS Integration	Interacts with Hadoop Distributed File System for efficient big data storage and retrieval.
Error Handling	Manages errors during downloads, data processing, and federated learning to ensure application reliability.
Streamlit Interface	User-friendly web interface for data input, processing, and visualization.

Reference list

Abhinav, E. al. (2023). Exploring the Impact of Digital Marketing on Rural Consumer Behavior: A Comprehensive Study. International Journal on Recent and Innovation Trends in Computing and Communication, 11(5), pp.412–419. doi:https://doi.org/10.17762/ijritcc.v11i5.9907.

Baheti, P. (2021). A Simple Guide to Data Preprocessing in Machine Learning. [online] www.v7labs.com. Available at: https://www.v7labs.com/blog/data-preprocessing-guide.

Banerjee, M., Lee, J. and Choo, K.-K.R. (2018). A blockchain future for internet of things security: a position paper. Digital Communications and Networks, 4(3), pp.149–160. doi:https://doi.org/10.1016/j.dcan.2017.10.006.

Barolli, L. (n.d.). Advanced Information Networking and Applications. Springer Nature.

Cooper, R.B. and Zmud, R.W. (2021). Information Technology Implementation Research: A Technological Diffusion Approach. Management Science, 36(2), pp.123–139.

European Data Protection Supervisor. (2024). Federated Learning. [online] Available at: https://www.edps.europa.eu/press-publications/publications/techsonar/federated-learning_en.

Fayyad, U., Piatetsky-Shapiro, G. and Smyth, P. (1996). From Data Mining to Knowledge Discovery in Databases. AI Magazine, [online] 17(3), pp.37–37. doi:https://doi.org/10.1609/aimag.v17i3.1230.

Goundar, S. and Praveen Kumar Rayani (2021). Applications of big data in large and small-scale systems. Hershey, Pennsylvania: Igi Global.

Hwang, K. (2017). Cloud computing for machine learning and cognitive applications. Cambridge, Ma: The Mit Press.

NCSC (2019). The National Cyber Security Centre. [online] Ncsc.gov.uk. Available at: https://www.ncsc.gov.uk/.

Shortcut Edition (2021). SUMMARY - The Long Tail: Why The Future Of Business Is Selling Less Of More By Chris Anderson. Shortcut Edition.

Sophie, A. (2023). The Evolution of Data Processing in Cloud Computing..

Upsolver. (2020). What is Apache Parquet and why you should use it. [online] Available at: https://www.upsolver.com/blog/apache-parquet-why-use.

Zhan, Z.-H., Liu, X.-F., Gong, Y.-J., Zhang, J., Chung, H.S.-H. and Li, Y. (2015). Cloud Computing Resource Scheduling and a Survey of Its Evolutionary Approaches. ACM Computing Surveys, 47(4), pp.1–33. doi:https://doi.org/10.1145/2788397.