International Research Journal of Education and Technology

Peer Reviewed Journal, **ISSN 2581-7795**

SCALABLE DATA CLEANING AND PREPROCESSING FOR BATCH ML MODELS USING PYSPARK

O. Srinivas[1] , M. Madhusri[2] , P. Sathwika[3], D. Sai Anusha[4],N. Praveen Naik[5]

[1]Post Graduate, Assistant Professor of Dept of Data Science Engineering, GNITC, Hyderabad

[2]Under Graduate, Dept of Data Science Engineering, Guru Nanak Institutions Technical Campus, Hyderabad

[3]Under Graduate, Dept of Data Science Engineering, Guru Nanak Institutions Technical Campus, Hyderabad

[4]Under Graduate, Dept of Data Science Engineering, Guru Nanak Institute of Technical Campus, Hyderabad

[5]Under Graduate, Dept of Data Science Engineering, Guru Nanak Institutions of Technical Campus, Hyderabad

---***---

Abstract - *This project focuses on building a robust and scalable data cleaning and preprocessing framework using PySpark on Azure Databricks. The solution is designed to handle large-scale datasets efficiently, ensuring high-quality input data for batch machine learning models. By automating data wrangling tasks such as handling missing values, outlier detection, normalization, and feature encoding, the pipeline improves the accuracy and reliability of downstream ML models.*

Key Features:

- ***High-Performance Data Processing:*** *Utilizes PySpark on Azure Databricks to clean and preprocess massive datasets efficiently.*
- ***Automated Handling of Missing Data:*** *Implements imputation techniques (mean,median,mode, KNN imputation) to ensure data completeness.*
- ***Outlier Detection & Treatment:*** *Uses statistical methods (Z-score, IQR) and machine learning- based anomaly detection for data consistency.*
- ***Feature Engineering& Transformation:*** *Supports one-hot encoding, label encoding, scaling, and PCA for optimized ML input features.*
- ***Batch Processing for ML Pipelines:*** *Enables scalable preprocessing work flows for large datasets used in batch ML training.*
- ***ML Flow Integration for Data Versioning:*** *Tracks preprocessed datasets and transformations for model reproducibility.*

This solution streamlines data preparation for machine learning, ensuring high-quality, structured input for better predictive performance.

1. INTRODUCTION

In many modern applications—such as customer analytics, fraud detection, recommendation systems, and predictive maintenance—data is generated in large volumes from multiple sources. Processing such high-volume datasets using traditional tools (e.g., pandas or single-node systems) becomes inefficient, slow, and often infeasible due to memory and compute limitations.

To address these challenges, Apache Spark has emerged as a powerful distributed computing framework designed for big data processing. PySpark, the Python interface for Apache Spark, allows developers and data scientists to scale their data processing workflows using Python's simplicity and Spark's performance. It enables the execution of complex data cleaning and preprocessing tasks across large datasets in a distributed and parallelized fashion.

This project aims to leverage PySpark for building a scalable, efficient, and modular data preprocessing pipeline tailored for batch machine learning models. The pipeline is designed to support standardized data cleaning operations, accommodate varying data schemas, and integrate smoothly into machine learning workflows that require periodic or scheduled retraining with new data batches.

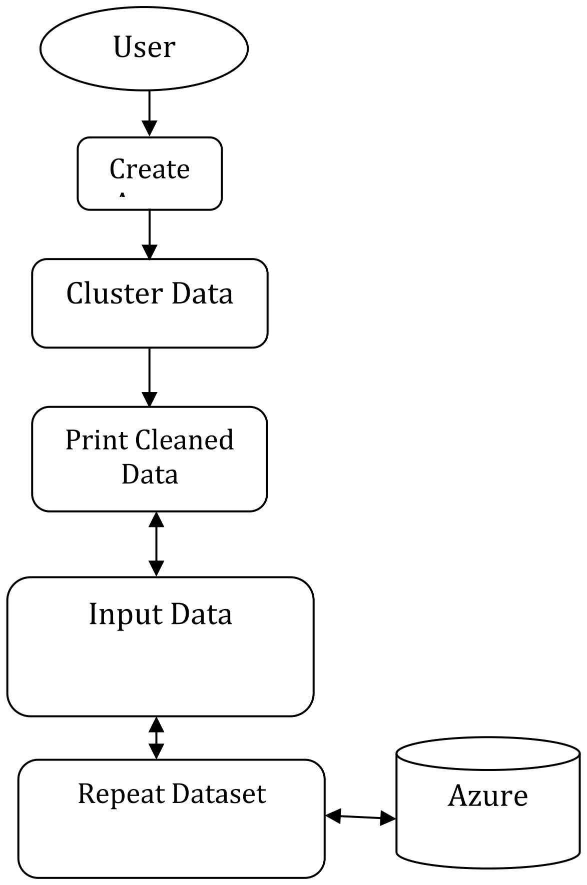
By implementing this solution, organizations can significantly reduce the time and computational resources needed to

prepare data, while improving the

performance and maintainability of their ML models. The scalable nature of PySpark also ensures that the solution remains effective as data volumes grow over time.

Method of Analysis:

We propose a scalable data cleaning and preprocessing methodology using PySpark for batch ML model pipelines. The method includes:



- Data Ingestion: Structured data is loaded from distributed sources with schema enforcement using StructType to ensure consistency.
- Missing Value Handling: Nulls in numerical columns are imputed with median values (via approxQuantile()), and categorical columns use mode-based imputation.
- Deduplication & Anomaly Detection: Duplicates are removed using hashed keys; statistical outliers are detected via z-score and IQR methods.

- Feature Transformation: Includes standardization (StandardScaler), categorical encoding (StringIndexer, OneHotEncoder), and date parsing.
- Balancing & Sampling: Stratified sampling using sampleBy ensures class balance in training/validation splits.
- Pipeline Integration: All steps are organized into a Pipeline; intermediate results are cached for performance.
- Scalability Evaluation: Performance is benchmarked on datasets up to 1B rows, measuring execution time, resource utilization, and data quality metrics.

1. METHODOLOGY

The proposed methodology leverages Apache Spark's distributed computing capabilities through its Python API (PySpark) to develop a scalable, modular pipeline for data cleaning and preprocessing. This pipeline is designed to process large-scale, heterogeneous datasets efficiently, making them suitable for downstream batch machine learning model training and inference.

1. Data Ingestion
2. Data Cleaning
3. Feature Engineering
4. Feature Selection (Optional)
5. Data Set Assembly
6. Model Training and Inference

Design Engineering deals with the various diagrams for the implementation of project. Design is a meaningful engineering representation of a thing that is to be built. Software design is a process through which the requirements are translated into representation of the software. Design is the place where quality is rendered in software engineering. Design is the means to accurately translate customer requirements into finished product.

1.CONCLUSIONS

This project successfully implemented a system for handling large volumes of data from a cloud-based SQL database using native services on Azure. By utilizing a combination of Azure SQL Database, Azure Databricks, Azure Data Lake, and other Azure services, we established a scalable and efficient data pipeline that ensures seamless integration, transformation, and analysis of cloud-based data. The system was designed to handle a variety of data formats, from structured relational data to semi structured and unstructured data, offering flexibility for diverse business and analytical needs.

Key strengths of the system include its ability to collect and store data securely in the cloud, its efficient data preprocessing and transformation capabilities, and its ability to provide insightful analysis through Azure's integrated tools. Azure's cloud-native services ensure high availability, disaster recovery, and strong data security protocols, critical for any enterprise-grade system. Furthermore, leveraging Azure Databricks provided the computational power needed for handling large datasets and performing complex data operations with ease. The system not only streamlines the process of ingesting, storing, and analyzing data but also ensures that data handling is optimized for performance and security, crucial for managing large-scale data in cloud environments. This project is a significant step toward building data infrastructure that can support future analytical, reporting, and machine learning tasks, thus enabling data-driven decision-making. By combining the best of Azure's cloud services, we've created a platform that provides the foundation for scaling and extending the capabilities of data analytics in a secure and efficient manner.

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to O. Srinivas for guiding me throughout this project on the Scalable data sharing and preprocessing for batch ML models using pyspark.

REFERENCES

1. **Chen, T., & Guestrin, C. (2016). XGBoost:** "A scalable tree boosting system". Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785-794. ACM.

2. **De Montis, A., Chessa, A., Campagna, M., Caschili, S., & Deplano, G. (2010)**. "Modeling commuting systems through a complex network analysis: A study of the Italian islands of Sardinia and Sicily". Journal of Transport and Land Use, 2(3).

3. **Zhang, L., & Li, H. (2017). "Cloud-based data storage and management:** A review of Microsoft Azure". International Journal of Cloud Computing and Services Science, 5(2), 30-45.

4. **Chung, R., & Tran, S. (2018).** "Optimizing cloud data integration workflows with Azure Data Factory". Cloud Computing Technologies, 3(4), 85-97.