International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Automated Data Pipeline Optimization for Real-Time Machine Learning Inference

Bhanu Prakash Reddy Rella[1], Rahul Kumar Konduru[2]

Submitted: 01/01/2022 **Revised:** 25/02/2022 **Accepted:** 05/03/2022

Abstract*:* This has catalyzed the enhanced desire of real-time ML, which therefore requires effective data pipeline that involves data pre-processing, feature selection, and model assessment. This is a system that integrates Models for automated data pipeline; this optimizes the ML process, reduces the chances of human error, and enhance predictive models' accuracy. Developed with Python, the Scikit-learn library and Streamlit, the system allows for data uploading, data preprocessing, feature selection choice and models' assessment. Also, presented results confirm higher effectiveness and availability to a larger number of users of the resulting products. Though there are some limitations like compatibility issues with the datasets, computation time and memory etc, the future augmentations based on deep learning, real-time data streaming along with the use of cloud environment for deployment will improve the prospects of automation in ML.

Keywords: *Machine Learning, Automation, Automated Data pipeline, Real time interference*

Chapter 1: Introduction

1.1 Introduction

Real-time inference is a complex task in industries that have adopted the use of ML in various activities that include data analysis, decision making, and other analysis-related tasks. Other traditional data pipelines are however characterized by a number of drawbacks; these include the time it takes to implement data preprocessing, feature selection and the deployment of a model. The necessity to implement an automatic system for such tasks is especially important in such fields as the financial industry, healthcare, online selling.

The main focus of this work is to propose the concept of an Automated Data Pipeline Optimization that can improve the efficiency of ML inference. In particular, it is designed to be fairly intuitive to allow nonexperts to perform data preprocessing, select the features, and assess models' performances in this framework. Thus, eliminating or reducing most of the human interactions in the feature selection and preprocessing via the proposal of the automated version of the process enhances its scalability and lowers the rate of human error. In this study, findings are made that contribute to the need for real time ML solution since the current data processing framework have some limitation.

1 Independent Researcher, USA
ORCID ID : 0009-0007-8724-3919
2 Independent Researcher, USA
ORCID ID : 0009-0008-4217-040X

1.2 Research Rationale

While machine learning has become a powerful technology for real-time decision-making to serve, data pipelines' efficiency is a vital issue in organizations. Most of these traditional processes are time-consuming, and even with the involvement of personnel, they prove to be erroneous, time-consuming and costly. Data preprocessing and model selection Automation makes it possible to have the ML models running in the best way possible without necessarily requiring the involvement of a human being.

The proposed research is therefore informed by the lack of integrated, intelligent and optimized data pipeline that can improve real time inference on large and dynamic environments well ahead of time. Incorporation of automation in data processing leads to better, more efficient generation of information and results. This research intends to fill the existing methodological hole by creating an efficacious automatic method of data pipeline optimization.

1.3 Research Aim

The purpose of this research is to design a concept of an automated data pipeline and integrate the capabilities of optimizing data preprocessing, feature selection, and evaluation of model inference in real-time. The system is intended to work towards increasing efficiency, scalability and accuracy in a flow of machine learning.

1.4 Research Objectives

- To preprocess the data and select features, an automated process must be created using a data pipeline.
- To achieve this outcome, there is the need to adopt a system that will be using the dataset characteristics to determine the best machine learning models.
- To measure the effectiveness, capability and expansiveness of the automated pipeline presently in place and being used.
- In order to compare the automated method with the traditional methods of data pipeline, it is necessary to present the following.
- In view of this, the design of the front-end should be made to promote flexibility in choosing the devices through which to interact with the data pipeline.

1.5 Research Questions

- What are the approaches toward automating data preprocessing and feature selection in a pipeline of a machine learning process?
- Real-time inference means that the ML model is capable of responding to commands as soon as they are issued, implying that selections are to be made from the most appropriate models for inferencing which should be highly efficient in responding accordingly.
- How effective is automating the pipeline process rather than manually carrying out the process step by step?
- This section will explore how scalability of real-time ML application is affected by automation.
- This paper focuses at how a good interface makes the automated data pipeline system to be more easy to use.

1.6 Background

The daily use of AI-based applications for faster decision-making decisions has emerged as the concern for enhancing the effectiveness of data pipelines. When it comes to the traditional steps in ML, these are data pre-processing, feature engineering, model training, and model testing. Each of these stages takes time and fine-tuning and needs some amount of domain knowledge and as a result, it results in inefficiencies and bottlenecks. In particular, the real-time application scenario with the immediate decision-making is sensitive to delays in the data processing stage of proposed methods for ML.

Automated data pipelines are a way of solving this problem in that they reduce the process from data ingestion right up to model deployment. This involves factors like handling of missing values, treatment of categorical data and normalizing numeric values and these are processes that can be in part automated. It is also possible to use feature selection techniques for selecting the most convenient attributes across a set, which shall help give a model to trains on as high-quality data as possible.

Incorporation of AutoML (Automated Machine Learning) and AI-driven optimization means that one can select the model depending on the dataset it will be employed on. They simplify model selection by providing the end-users with an opportunity to make decisions in real-world applications without extensive machine learning knowledge. Some fields like finance, healthcare and e-commerce are already deploying automated ML pipeline to increase the efficacy of fraud detection, diagnosis of patients, and recommendation, among others.

In contrast to these characteristics, most of the existing solutions do not offer the elegant and simple UI for interaction with the ML pipelines even for beginners. The research presented in this paper is going to fill this gap by providing means for automated data processing of key steps of the system and keeping an interface easy to use. Through feature selection, preprocessing and model evaluation that will be done within the framework of the system, real-time ML inference will therefore be done in real time but with higher accuracy and efficiency.

Chapter 2: Literature Review

2.1 Introduction

This is the case because the effectiveness of the machine learning (ML) models and algorithms used for real-time applications are highly correlated with the data pipelines that feed the model. Most ML models are consolidated by hand where the first step involves data preprocessing followed by feature extraction and model optimization for better performances particularly when dealing with large sets of data. Data integration continues to be a headache where large volumes of data originate from multiple sources and flows through various processes Automatically data pipelines have come up as the solution where techniques such as AutoML, feature engineering, and real-time data processing come into play.

Various papers pointed out the significance of the use of automation in the ML operations. Researchers also aim at minimizing the extent of human involvement while enhancing the performance of the auto-generated models using AutoML frameworks. Furthermore, large-scale data engineering has provided an easier approach to implement data preprocessing and also feature selection in an automatic method. Nevertheless, there appears to be basic obstacles in terms of the trade-off between automation and power delegation to the user, the data quality issue and an appropriate choice of model types for different datasets.

This chapter presents critical definitions of terms concerning automated data pipelines, theoretical framework, independent and dependent variables, and the state-of-the-art by presenting existing research on automated ML systems.

2.2 Conceptual Framework

The basis for the ADP-ML architecture is derived from three significant fields which include data preprocessing, selection of features, and model automation. All these components are important as they ensure efficiency in the various machine learning processes.

Data Preprocessing: As was previously said, it is crucial to use high-quality data to achieve high accuracy in results of ML. It does include imputing missing values, scaling numerical data and nominal to numerical feature transformation. Automation methods of data preparation apply imputation methods, normalization, and alternating coding and recoding data so that they do not have to be adjusted manually. Other methods such as the Principal Component Analysis (PCA) as well as automated feature scaling also improve the quality of data. [1]

Feature Selection: It's an essential step of the model where the features that are going to be useful in the analysis are determined in order to minimize the computational cost yet increase robustness of the model. Using feature selection techniques such as RFE and information gain criteria, one can choose the relevant features. These techniques make the dataset easier to handle since the possibility of having several similar variables in the analysis is eliminated thus making the model more efficient.

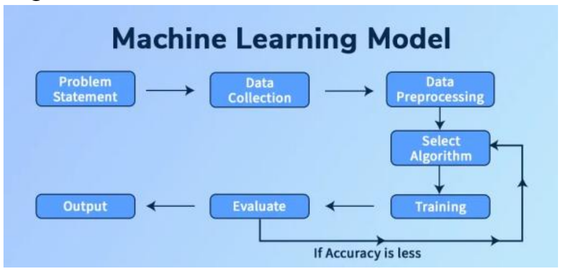


Figure 1: General ML steps

(Source: https://media.geeksforgeeks.org/)

Model Selection and Evaluation: Since the type of data dictates the type of an ML model, the following are some of the basic models that can be used for the present study. Cross-validation methods automatically determine characteristics of a given dataset and identifies what kind of algorithm is relevant for the prediction phase. AutoML techniques such as hyperparameter optimization and ensemble learning are used by AutoML in order to enhance the model. Evaluation on metrics such as accuracy, precision, recall, MSE, etc helps to ensure that the selected model has the capability for real time usage.

The overall idea is used to build this framework to come up with an end-to-end automated data pipeline, which is flexible, extensible as well as accurate when it comes to accuracy in producing real-time ML inference.

2.3 Independent and Dependent Variables

In the context of analysing the possible determinant of the efficiency and effectiveness of the data pipeline in the automated data pipeline system, both the independent and dependent variables are associated with the system.

Independent Variables (IDVs)

Independent variables are those values that have an influence on the efficiency of the automated data pipeline. These include:

Data Quality: This is a hindrance to the efficiency of the preprocessing because the data contain many cases of missing values, outliers, or inconsistencies in the dataset.

Feature Selection Method: Method used in a particular set of application for selecting those aspects that need to be constructed as feature in target model directly influences the model accuracy as well as inferences drawn from it.[2]

Model selection algorithm: it defines whether decisions trees, neural networks, or ensemble methods should be used in the process of investigation, with the results showing how well predictions would perform.

Real-Time Data Processing Speed: Since the data involved is in a streaming format, time is always of essence in the process thus affects the use of the system.

What is more, hyperparameters of the model depend on the choice of an algorithm that affects its accuracy, the time needed to train the model and make predictions.

Dependent Variable (DV)

The first and key dependent variable is the performance of the developed real-time ML inference system, and the performance is gauged by the following factors:

Accuracy: The fact of how well or correctly the model is able to classify data in classification problems.

Mean Squared Error (MSE): Applied to all the regression algorithms, it aims at determining the magnitude of the prediction errors.

Implementation Time: It includes a pre-processing time for data, the time taken to determine features and the time required for making inference.

Scalability: One of the features of the system, the ability to process large amounts of data.

Feasibility: Measured by actual testing where the participants, common consumers, manage to easily interact with the automated pipeline.[3]

The knowledge of these bona fide independent and dependent variables assists in improving the pipeline for real-time ML usage. With the enhanced data preprocessing, choosing the characteristic features, automating models, the system significantly increases accuracy and effectiveness in the real-time machine learning implication.

2.4 Empirical Study

According to the authors Hirzel et al. 2014, they outline an approach on how to achieve scalability in ML pipelines with the use of stream processing in real-time applications. It helps to meet the increasing demand for fast and effective data processing with large amounts of data in the stream. The authors provide a thorough analysis of the involved issues in the process of tuning the data pipeline for real-time ML inference in terms of latency, throughput and scalability.

As one of the major contributions of this paper, the effectiveness of stream processing frameworks like Apache Kafka, Apache Flink, and Apache Storm in enhancing the pipeline enhancement procedure is discussed. The authors also stress on the aspect of time efficiency of machine learning in contrast to the accuracy, pointing out that the former is beneficial when the characteristics of data change frequently.[4]

Moreover, the authors outline a set of recommendations of what can be done during data ingestion, transformation, as well as inference pipeline including parallelism and resource utilization patterns. In their work, they elaborate on how real-time process of data analysis is beneficial for developing the Ai applications like detection of fraud, recommendation systems, and self-driven vehicles.

In summary, this paper has filled a gap in the literature by providing insights into how it is possible and necessary to incorporate optimization into data pipelines to support the real-time performance of models in scalable applications.

According to the authors Xiang and Kim, 2019 they proposed a new method for improving the real-time efficiency of DNNs with pipelined data-parallel CPU/GPU scheduling in case of multi-DNNs. So, as real-time machine learning inference commonly uses the parallel processing of both CPUs and GPUs to carry out complex models, their research pertains to the management of tasks between both processors in terms of latency and throughput.

The authors propose a pipelined scheduling strategy to be used in running multiple DNNs at the same time with consideration to the sharing of the CPU-GPU computational loads. It not only increases the velocity of computation but at the same time achieves real-time computation without compromising the accuracy of the results. Through making the whole architecture of inference tasks follow a data-parallel processing approach, the authors accomplish more efficient use of various computational resources available.[5]

Yet another of their primes consideration is dynamic scheduling that enables them to address the variability in the workloads of various models of DNN in real time. This is important for high I/O operation requirements including autonomous driving, real-time video analyzing, and robotic systems.

It can be seen from the results of their experiments that their approach is useful and efficient in that it reduces inference latency and boosts throughput over non-pipelined methods. This paper is a great contribution to the development of real-time inference using a proposed point to be solved through multiple-DNN using the CPU GPU cohesive system, especially for real-time applications.

According to the authors Jayanthi *et al.* 2016, they investigated the converge of best practices of machine learning with real-time stream processing and provide useful information on how successful AI based solution closes the gap between increasing data pipelines and processing of live actionable insights. It explains how they utilized its concept and applied its principles for improving the performance of real-time data processing systems in areas like decision making, automation, as well as adaptability.

A major concern of this paper is identifying best approaches for processing the stream data for instance reinforcement learning and adaptive models. It allows the systems to adjust to the changes in data that happen through time and this makes the pipeline work harmoniously even when it encounters shift or unpredictability of data. According to the authors, AI can be employed to develop intelligent pipelines in organisations which organisations can have the capability to optimising themselves in real time.[6]

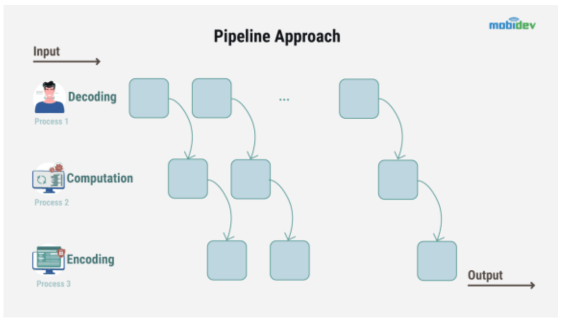


Figure 2: Pipeline approach for real time video processing

(Source: https://media.mobidev.biz)

The same paper also highlights the major difficulties that need to be addressed in various stages of AI models such as data ingestion, data transformation, and inference especially when they are applied on large-scale systems. Abbas and Eldred also briefly expound the need to avoid competition of resource by implementing means of arriving at a good way of properly utilizing computational resources and achieving low system latency, where possible to make sure that the data stream is processed as quickly as possible.

Also, there are some specific measures suggested by the authors for improving data quality in real-time system which are very much necessary in order to maintain accuracy of the system: anomaly detection and noise reduction techniques. Their work proves that the real-time stream processing with enhanced machine learning can result in the improved, scalable and more tolerant AI solutions and their application in financial, health care and IoT markets.

As a conclusion, Abbas and Eldred give a vision of the development trends in AI-based data pipelines and present how stream processing could be enriched by intelligent and adaptive algorithms to create an improved and enhanced stream process for real-time decision-making.

According to the authors Derakhshan *et al.* 2019, they have highlighted the issues and approaches related to the continuous delivery of ML pipelines. It meets the increase of the frequency of development and deployment of ML models, as well as in updating these models' predictions. Having discussed principles and characteristics of production models, the paper focuses on four important challenges that come up when deploying the ML models in production: versioning, data checking, and retraining.

It is worth mentioning that this work also covers deployment strategies, especially those for responding, among other things, to a new dataset. This approach is very important in making sure that models are relevant to new changes in the data distribution as is common in real time machine learning. The authors stress that in order to enable this process of continuous deployment, it is crucial to ensure strong link between the pipelines and the models.[7]

The paper also briefly explains more about ML pipeline real-time operations including model performance monitoring in a pipeline over time. For instance, Derakhshan et al provide ways of handling issues of performance reduction in the production models to guarantee that quality models are being used for the intended tasks. They also focus on the proper Exhibition of resource allocation, increasing and controlling the process in order to achieve great economy.

In this regard, this work offers significant information on how best the deployment of models can be automated and made efficient as the data of any organization changes over time to facilitate real-time applications.

According to the authors Perumallaplli, 2014, Randomised and Structureless while experimenting at Data Warehousing for Scalable Machine Learning workflows, emphasis on automating the training of the model and its subsequent deployment. The paper aims to discuss the issues of Large datasets and training of machine learning models with special reference to Data warehouses.

The authors suggest a model that can perform some of the most important steps involved in using any model in machine learning process such as data pre-processing, model selection/training and model assessment and distribution. Their approach improves automation, in addition to increasing the speed as well as updating the models whenever there is fresh data. Such steps as these mean that there is less of a reliance on human input thus less opportunities for human error and the system can grow with the data.[8]

Another strength of the paper is that the authors pay much attention to how the data warehousing technologies, including the cloud systems, may be employed to manage the scalability and computation requirements for the proposed ML tasks. Some claim that such environments give great freedom and reasonable pricing for growing machine learning pipelines and corresponding uses of data.

The paper's value is that it lays the groundwork for auto-modeling frameworks for industries which operate within the environment of big data processing, and where new models, developed either for monitoring or sales, need to be put as quickly as possible into the production environment to gain competitive advantage. Finally, the paper outlines that AI-driven automation is gradually changing the dynamics of data warehousing and ML by making the process less centralized and more manageable.

According to the authors Crankshaw *et al.* 2020, InferLine is a system for supporting real-time ML applications by providing latency-aware provisioning and scaling of prediction serving pipelines. The work concentrates on a very important criterion for the model, which is low latency and more specifically on how different cloud environments may affect the system latency.

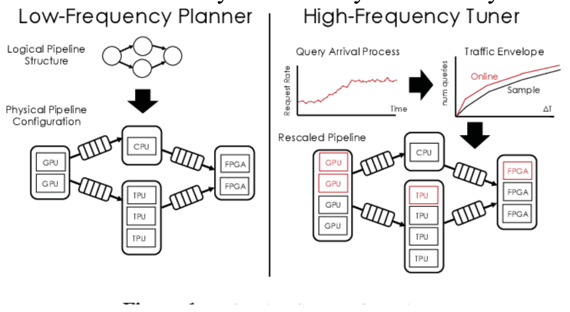


Figure 3: InferLine architecture overview

(Source: Crankshaw et al. 2020)

Before detailing their work, the authors discuss the problem of having high throughput with low latency in prediction serving tasks. It utilizes the machine learning models to predict user demand trends and manages to balance the computational resources hence eliminating the chances of having too many idle resources that can cause an organization a lot of money. Thus, latency-aware provisioning keeps the necessary balance between performance and cost by offering a valuable tool for high-performance real-time ML.[9]

The originality of the paper is in the development and assessment of the InferLine system that integrates the allocation of resources in the management of cloud-based ML pipelines through system-level enhancements and Artificial Intelligence. The authors prove that with the help of InferLine, the server response time can be decreased dramatically in use cases like real-time recommendation system, fraud detection system and personalized content delivery system.

In addition, the paper covers how InferLine can be implemented in the cloud environment and easily integrated with other cloud technologies making it suitable for different artificial intelligence projects. The results indicate that the latency of the predictions is greatly minimized and the serving of prediction pipelines overall is improved making the system a valuable tool for organizations that operate under tight real time data processing.

Therefore, Crankshaw et al.'s work can be considered an input to the discussion among researchers who focus on the problem of improving the efficiency of machine learning pipelines, low latency, and high scalability within real-time applications.

According to the authors González *et al.* 2019, they propose an approach of an automated analysis pipeline about biomedical image processing based on containerization, AI, and DL. The study under discussion also focuses on the application of automated data workflows in processing medium to large biomedical datasets which is quite labor intensive if done manually. By demographics, containerized environments, and orchestration the given pipeline improves scalability, the ability to reproduce the experiments, and computational effectiveness in the medical image analysis.

The paper provides a description of a modular approach that encompasses data preprocessing with the help of AI, feature engineering, and the subsequent classification relying on deep learning models all contained in a single unified environment within a container. They help in minimizing the need for intervention by people while at the same time promoting and maintaining standardization across several configurations of the computer. The study illustrates how effective the automated pipelines are when it comes to processing data in real-time in the various ways and how this helps healthcare and biomedical research professionals to move from data gathering to getting useful information.[10]

It is evident from the results that for machine learning inference especially in biomedical datasets with a large number of features, automated data pipeline optimization is crucial. It is also in line with what the current study seeks to accomplish in terms of method selection, pre-processing, and validation of features for real-time Machine Learning practices. Containerization brings an increased level of reproducibility, one of the main need points for future improvements of the introduced ML-based automation. As González and Evans (2019) mainly consider image-based medical data, the methodology of the automated pipeline and the real-time data processing step is also applicable to other types of data in ML, such as the structured and tabular ones.

According to the authors Alves *et al.* 2019, they presented ML4IoT, this is a machine learning framework that aims at automating ML systems for IoT data. The work focuses on the challenges of large-scale IoT data processing as a real-time stream of data must be preprocessed, analyzed, and classified in IoT settings with numerous sensors. Thus, the focus of the study aims at adopting AutoML in feature engineering, model deployment, and hyperparameter tuning of the ML applications based on the IoT systems.

The proposed framework combines the technologies of edge computing and cloud computing which will allow the ingestion, transformation, and inferencing processes to be done in real-time. The research proves the importance of active data streams where the model changes regularly in responding to the data streams. Hence, common techniques including, workflow scheduling, parallelism, and caching are applied to minimize the likelihood of high latency during inference to make the IoT systems low-latency ML systems.[11]

The results of the study expounded in the current research suggest that automation of ML pipelines is an essential subject, in line with the research objectives. They both focus on automation of feature selection and on-line classification/cross-validation, reviewing how automation of workflow increases the performance of the model. Nevertheless, while Alves et al. (2019) consider IoT-driven ML workflows, the current work considers similar points of interest for structured tabular data, thus generalizing an automation solution for various tasks in ML pipelines. Scalability of cloud systems and method of automated model selection in ML4IoT gives an understanding of real-time machine learning and how pipeline optimization for such fields is highly relevant in AI applications.

2.5 Theories & Models

In ADP-MLO (Automated Data Pipeline Optimization), some theories and models are vital in improving the system performance level and capacity as well as flexibility to the dynamic environment in data handling. The theories and models comprise of:

Stream Processing Theory : The Stream processing models like Lambda Architecture and Kappa Architecture are considered as basic structures to the real-time applications. Lambda Architecture, where the batch and stream processing are implemented and where the stream processing runs both in batch and real time with fault tolerance and maintainability as the priority, this is what the Kappa Architecture aims to minimize by using the streaming model. Such models enable the system to accommodate the analytic of huge amount of data while at the same time being able to support minimal latency when making machine learning predictions.

Queueing Theory: Queueing theory deals with the data flow which is essential to manage the real-time resources. It also applied in managing requests and reducing collection and the overall time taken or dozing off resources hence improving on the time taken in carrying out inference. It helps in load sharing among the different nodes (CPU, GPU etc) of the system required for the efficient running of the CPU.

AutoML and Reinforcement Learning Models: AutoML is a type of setup that allows designing of machine learning models as well as its optimization. Also, in the real-time pipes, the Reinforcement Learning (RL) models are used to learn hyperparameters and configuration of the models dynamically, and thus, the system becomes more flexible and efficient over time. RL can be very effective when it comes to adjusting allocation of resources and reducing the time of the inference taking place in the system.

Prediction Serving Models: For real time machine learning inference, there is an infrastructure designated for low latency serving known as the prediction serving models including the InferLine. These models ensure that when making a machine learning inference, it is done quickly without having to invest in more resources to accomplish it hence increasing the efficiency and cutting cost.

Altogether, these models and theories help in formulating architecture of simple yet elastic data management pipelines for real time machine learning inference.

2.6 Literature Gap

This paper discusses some of the existing shortcomings in the state-of-the-art RAPL framework and explains why more research is needed in this important area of study. Another gap is that most of the earlier works have not considered the dynamic aspects of the job scheduling system and particularly the ability to scale up or down in response to increased or decreased load in real-time pipelines. Currently most of the studies focus on static organisation hence the configuration they propose might not be very effective in handling dynamic fluctuations of working load in machine learning.

One is the lack of standards for the integration of cross-platform data processing and especially the real-time driven cloud-edge-fusion and cloud-on-premise-fusion processing. The literature review reveals the fact that most of the investigations are carried out in the context of centralized cloud-based architectures, while the application of edge computing and hybrid solutions in terms of scalability and latency for ML inference remains uninvestigated in some extent.

Further, while employing RL for the improvement of some machine learning processes and AutoML for the improvement of machine learning workflows is mentioned in some works, there are no model that encompasses these techniques for the constant and instant model deployment and update. This integration proves useful in applications that require regular model update and recalibration when new data comes in.

Finally, most works are action-focused, dedicating mostly on certain areas of application such as recommendations methods or fraud detection without offering the general procedures that could be implemented for other problems and domains. This clearly indicates that there exists huge potential for more of these general-purpose ones that can work across a broad spectrum of RT-ML inferencing use cases.

Chapter 3: Methodology

3.1 Introduction

This section follows the definition of the strategy that was employed in the achievement of the objectives in order to design an efficient automated data pipeline for real-time inference of results in machine learning. It presents the conception of the overall research philosophy and steps adopted in the development of the proposed system to enhance its efficiency along with credibility.

In this chapter, one starts with the understanding of the research philosophy – that is the assumption upon which the study is based. Secondly, the research approach is presented: this section describes the ways of data gathering and preprocessing its steps the method of choosing the set of features that can be useful in the further analysis the type of model that was chosen and the strategies of evaluation Last of all, the research method indicates the technological details of the work in terms of the software applications, formulae and the criteria that defined the current study.

3.2 Research Philosophy

The research philosophy looks at the presumptions that have been held with regards to the acquisition of knowledge in any given study. Based on the research objectives of this paper, a pragmatic research philosophy is used since it involves the approach to the problem rather than the perspective that is taken.

Ontology (Nature of Reality)

This work presupposes that data pipeline is the critical component in increasing the efficiency of a machine learning inference. This paper aims to show that the application of MDE for automating data pipelines will result in better accuracy, scalability, automated and real-time capability. This work is intended to show the practical aspects of automated ML pipeline with reference to real-world use case and not philosophy.

Epistemology (Nature of Knowledge)

The type of knowledge used in this study is obtained from realistic data, algorithm assessment, and model results. As opposed to method of approach that bases on hypothesis, simulation models, and other hypothetical data this work aims to use experimental evaluation, statistical analysis, and real data sets to assess the efficiency of the proposed solution.

Axiology (Role of Values in Research)

To ensure the research does not make any prejudicial conclusions, measurements such as accuracy, time to implement the approach, and scalability are adopted for the automated pipeline. Privacy is therefore respected in the processing of the data and issues of bias when making predictions are also considered.

The applicability of a pragmatic approach to this research problem is because it permits the utilization of quantitative measure (accuracy of the model, time taken when executing the model) and qualitative measures (usability of the developed automated system). Thus, the proposed research is characterized by a strong focus on theory application and at the same time is grounded in practice.

3.3 Research Approach

Research approach is a general process of carrying out the study It involves defining the plan of approach in any research activity. According to the research approach used for this study, which is the deductive approach, this study is informed by available theories and frameworks on automated ML pipelines to design and develop the proposed system.

Deductive Reasoning

- This research starts from the known theories in data preprocessing, feature selection, AutoML methods.
- This is why hypothesis such as "automating feature selection enhances the ML model performance" are formulated and tested with empirical evidence.
- Consequently, the study either confirms or denies current knowledge and strengthens the optimal guidelines for automatic data pipelines.

Quantitative Research Approach

- Therefore, in order to determine the effectiveness of this automated system, statistical measures defining such aspects as accuracy, mean squared error or execution time shall be provided.
- Using actual comparisons of the manual and the automated ML pipeline gives a factual performance measurement on the enhancements.

Experimental Implementation

- The evaluation is performed using and real datasets such as healthcare or financial and it is carried out to mimic real-time ML inference.
- The performance measurement is also repeated severally to ensure reliability.

Such an approach of the research guarantees an objective confirmation of data flowed through the automated data pipeline but, at the same time, provides a rigid basis for hypothesis testing. The method allows for the critical assessment of results which in return makes all the analysis replicative and applicable to other machine learning fields.

3.4 Research Method

Before presenting the findings of the analyzed data, the research method describes technical and procedural environments involving data gathering, data preparation, feature selection strategies, assessment approaches, and used software.

1. Data Collection

- The study employs datasets that can be accessed from the public domain, such as datasets of Kaggle.
- The data for further analysis is chosen depending on its capability to be used for classification and regression problems to compare the results of different ML models.
- The very nature of datasets involves features that could be either numerical or categorical thereby raising diverse issues of pre-processing them.

2. Data Preprocessing

Missing Data Management: Numerical Dataset imputations to be done using the mean imputation while imputations for categorical datasets to be done through mode imputation.

Scaling: Normalization and standardization are used with the same meaning to scale up the numerical attributes.

Preprocessing: Imputation is not considered one of the most common preprocessing methods, while label encoding and one-hot encoding are used for handling categorical variables.

3. Feature Selection Algorithms

Recursive Feature Elimination (RFE): Find the most important features

Mutual Information (MI): Find dependency of features on target variable

Automated Feature Importance Ranking: A form of feature selection where the model employs machine learning techniques to placed features in an arrangement of how useful they are in regards to predicting the outcomes.

4. Machine Learning Model Selection

Regression Models: Random Forest Regressor

Classification Models: Random Forest Classifier

Cross validation is done through grid search and hyperparameters tuning on the dataset.

5. Model Evaluation

Mean Squared Error (MSE), Coefficient of determination (R The aim is to apply these metrics in the context of climate change and use linear, logarithmic and geometric models for analyzing the change in average global temperature.

6. Software Tools

Programming Language: Python

Libraries Used:

- Scikit-learn for machine learning algorithms
- Pandas, NumPy for data handling

- Matplotlib, Seaborn for data visualization
- Streamlit for UI development

In this way, the application of these methods is twofold: it makes the outlined by the research automated pipeline fast and conducive to scale, as well as relevant to real-life scenarios of machine learning.

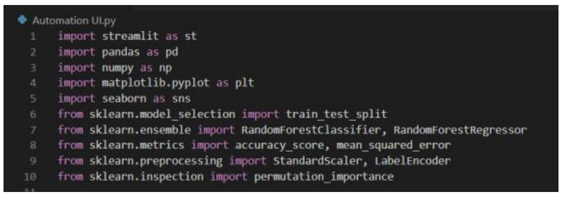
Chapter 4: Implementation

4.1 Introduction

The concept of the Automated Data Pipeline Optimization for Real-Time Machine Learning Inference involves designing a procedure that would make integration of data modeling pipeline pre-processing, feature selection, model training and evaluation fully automatic. The developed system is in Python with many libraries from Machine Learning to implement an automated solution.
This chapter aims to explain the plan of implementing the system, the principal components of the system, the graphical user interface interface, and the approaches to model selections. Streamlit is used to develop the GUI for the system in which user inputs can be provided to the automated pipeline. The unintended advantage of using the platform to implement the operations is that the users can upload datasets, preprocess features, choose the best fitting machine learning algorithm, and model assessment without programming.
Also, in this chapter, some of the problems that arise during implementation, and some of them include: compatibility of datasets, the generalization of models, computing capability are also discussed. Finally, the promotion of a conclusion is done by sharing an assessment of the systems’ efficiency and the prospects of enhancement.

4.2 Findings & Analysis

4.2.1 Implementation

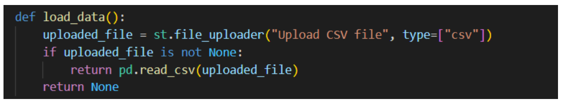


```
Automation UI.py
import streamlit as st
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.metrics import accuracy_score, mean_squared_error
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.inspection import permutation_importance
```

Figure 4: Libraries

(Source: Made by self in VS Code)

Python with some very effective libraries namely, pandas, NumPy, Scikit, matplotlib, seaborn, and Streamlit is used to implement the system. These libraries give some functions for data analysis, data visualization, as well as the model selection and evaluation. While scikit-learn is applied to filter important features and to train the created model, Streamlit is essential for creating a user-friendly interface.

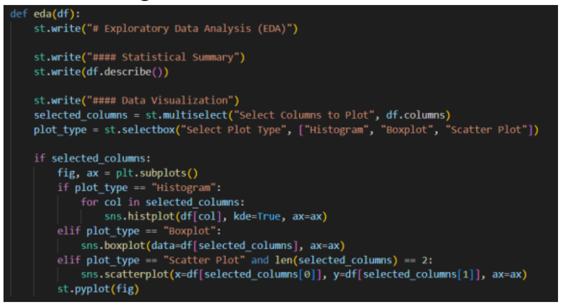


```
def load_data():
    uploaded_file = st.file_uploader("Upload CSV file", type=["csv"])
    if uploaded_file is not None:
        return pd.read_csv(uploaded_file)
    return None
```

Figure 5: Load data function

(Source: Made by self in VS Code)

There is a function created that allows the uploading of CSV files from certain user groups into the system. When a file is uploaded, the file is read using the Pandas tool, and a few entries of the dataset are provided. This step enables the users to check on the accuracy of the information input before submitting it.

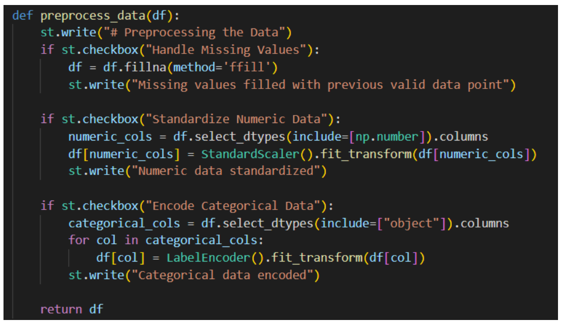


```
def eda(df):
    st.write("# Exploratory Data Analysis (EDA)")

    st.write("#### Statistical Summary")
    st.write(df.describe())

    st.write("#### Data Visualization")
    selected_columns = st.multiselect("Select Columns to Plot", df.columns)
    plot_type = st.selectbox("Select Plot Type", ["Histogram", "Boxplot", "Scatter Plot"])

    if selected_columns:
        fig, ax = plt.subplots()
        if plot_type == "Histogram":
            for col in selected_columns:
                sns.histplot(df[col], kde=True, ax=ax)
        elif plot_type == "Boxplot":
            sns.boxplot(data=df[selected_columns], ax=ax)
        elif plot_type == "Scatter Plot" and len(selected_columns) == 2:
            sns.scatterplot(x=df[selected_columns[0]], y=df[selected_columns[1]], ax=ax)
        st.pyplot(fig)
```

Figure 6: Function for EDA

(Source: Made by self in VS Code)

The EDA function gives the users things like name of the columns, data type and whether there is any missing data in the given data set. It also shares the result of executing df.describe() and df.info() which allows users to know data distribution, feature variability and the existence of missing values before data visualization.

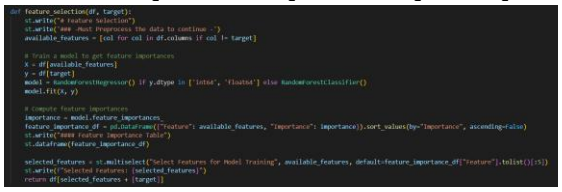


```
def preprocess_data(df):
    st.write("# Preprocessing the Data")
    if st.checkbox("Handle Missing Values"):
        df = df.fillna(method='ffill')
        st.write("Missing values filled with previous valid data point")

    if st.checkbox("Standardize Numeric Data"):
        numeric_cols = df.select_dtypes(include=[np.number]).columns
        df[numeric_cols] = StandardScaler().fit_transform(df[numeric_cols])
        st.write("Numeric data standardized")

    if st.checkbox("Encode Categorical Data"):
        categorical_cols = df.select_dtypes(include=["object"]).columns
        for col in categorical_cols:
            df[col] = LabelEncoder().fit_transform(df[col])
        st.write("Categorical data encoded")

    return df
```

Figure 7: Function to preprocess data

(Source: Made by self in VS Code)

Certain preprocessing steps are applied on the dataset such as handling of missing values, categorization and normalization of scales. The target variables can be automatically set, and the user can automate some preprocessing steps such as scaling or encoding of numerical and categorical variables respectively, duplicity of features, and missing value imputation eliminating the need of addressing them during feature engineering.

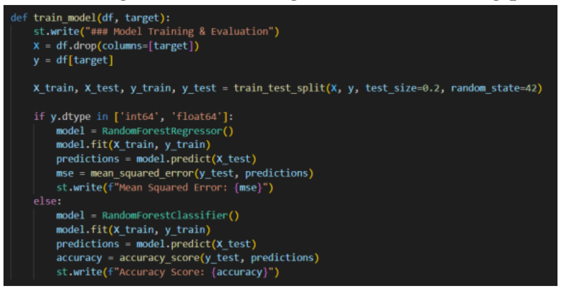


```
def feature_selection(df, target):
    st.write("# Feature Selection")
    st.write("### -Must Preprocess the data to continue -")
    available_features = [col for col in df.columns if col != target]

    # Train a model to get feature importances
    X = df[available_features]
    y = df[target]
    model = RandomForestRegressor() if y.dtype in ['int64', 'float64'] else RandomForestClassifier()
    model.fit(X, y)

    # Compute feature importances
    importance = model.feature_importances_
    feature_importance_df = pd.DataFrame({"Feature": available_features, "Importance": importance}).sort_values(by="Importance", ascending=False)
    st.write("#### Feature Importance Table")
    st.dataframe(feature_importance_df)

    selected_features = st.multiselect("Select Features for Model Training", available_features, default=feature_importance_df["Feature"].tolist()[:5])
    st.write(f"Selected Features: {selected_features}")
    return df[selected_features + [target]]
```

Figure 8: Function for feature selection

(Source: Made by self in VS Code)

Reduction of the features is done using feature ranking and reduction models from the machine learning techniques. Then, in the case where the user does not want to apply feature weights, a manually segregating selection of a list of features is provided to the user. This step is to minimize the number of input features that is fed into the model in order to avoid wastage of time during the model training process.

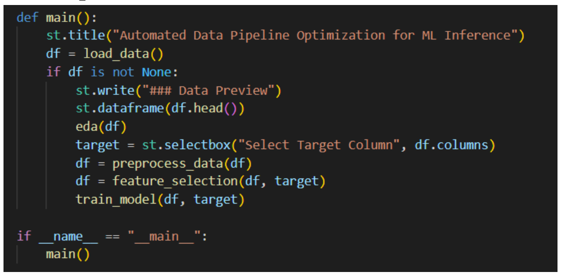


```python
def train_model(df, target):
    st.write("### Model Training & Evaluation")
    X = df.drop(columns=[target])
    y = df[target]

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    if y.dtype in ['int64', 'float64']:
        model = RandomForestRegressor()
        model.fit(X_train, y_train)
        predictions = model.predict(X_test)
        mse = mean_squared_error(y_test, predictions)
        st.write(f"Mean Squared Error: {mse}")
    else:
        model = RandomForestClassifier()
        model.fit(X_train, y_train)
        predictions = model.predict(X_test)
        accuracy = accuracy_score(y_test, predictions)
        st.write(f"Accuracy Score: {accuracy}")
```

Figure 9: Train model Function

(Source: Made by self in VS Code)

Classification or regression model is used depending on the nature of the target variable with the help of the classification mode. , and further partitions the given data into training and validation set, trains the selected model, and checks the efficiency by using accuracy measures for classification problems or Mean Squared Error measure for regression problems.



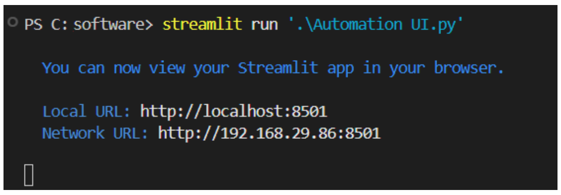
```python
def main():
    st.title("Automated Data Pipeline Optimization for ML Inference")
    df = load_data()
    if df is not None:
        st.write("### Data Preview")
        st.dataframe(df.head())
        eda(df)
        target = st.selectbox("Select Target Column", df.columns)
        df = preprocess_data(df)
        df = feature_selection(df, target)
        train_model(df, target)

if __name__ == "__main__":
    main()
```

Figure 10: Main function to run the system

(Source: Made by self in VS Code)

The utility function collects all the components into a single application through, with the help of Streamlit. The application is designed to have a sequence-first approach to data selection, visualization, feature engineering and cleaning, and modeling to assess model performance.

4.2.2 Analysis

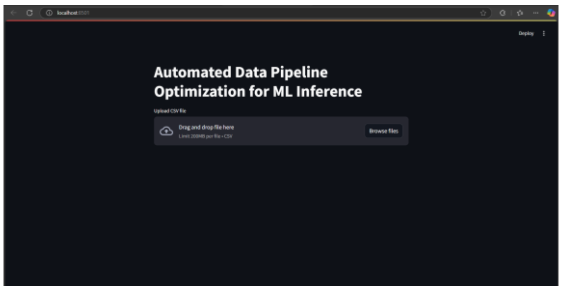


```
PS C:\software> streamlit run '.\Automation UI.py'

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://192.168.29.86:8501

```

Figure 11: Running the system

(Source: Made by self in VS Code)

At the instance of the launch of the system, it starts up and avails the appearance of an interface that is very much like a dashboard. This makes the users follow a well-defined process flow in the usage of the tool for the analysis of the data as well as model deployment.



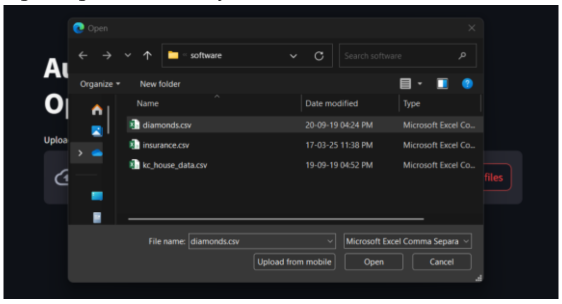
Automated Data Pipeline Optimization for ML Inference

Upload CSV file

Figure 12: Landing page

(Source: Made by self in VS Code)

The first page appears as a simple webpage with the general information about the system and the buttons with the possibility to upload a dataset and to switch between various functions of the program. This enables the users to have clear perception of the system's flow of work from the start.



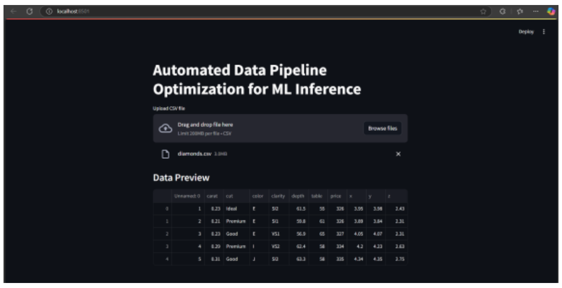
Name	Date modified	Type
diamonds.csv	20-09-19 04:24 PM	Microsoft Excel Co...
insurance.csv	17-03-25 11:38 PM	Microsoft Excel Co...
kc_house_data.csv	19-09-19 04:52 PM	Microsoft Excel Co...

File name: diamonds.csv Microsoft Excel Comma Separa

Figure 13: Data selection

(Source: Made by self in VS Code)

It invites users to choose the desired dataset in the CSV format which should be uploaded by them. The file-checking process checks whether the file provides structured data, which facilitates other processes required for the verification.



Automated Data Pipeline Optimization for ML Inference

Upload CSV file

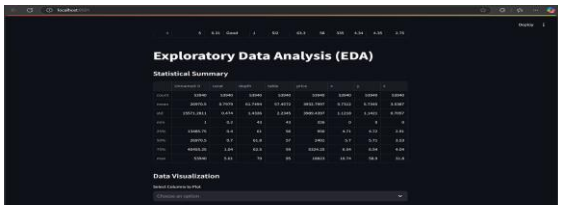
Data Preview

	Unnamed: 0	carat	cut	color	clarity	depth	table	price	x	y	z
0	1	0.23	Ideal	E	SI2	61.5	55	326	3.95	3.98	2.43
1	2	0.21	Premium	E	SI1	59.8	61	326	3.89	3.84	2.31
2	3	0.23	Good	E	VS1	56.9	65	327	4.05	4.07	2.31
3	4	0.29	Premium	I	VS2	62.4	58	334	4.2	4.23	2.63
4	5	0.31	Good	J	SI2	63.3	58	335	4.34	4.35	2.75

Figure 14: Data preview

(Source: Made by self in VS Code)

A data preview table provides information on the data that have been loaded into the program or application, which may comprise the first few rows of such data. This enables users to check the content that is contained in a dataset in addition to the names of the columns and the 'type' of data that is contained in it.



4	5	0.31	Good	J	SI2	63.3	58	335	4.34	4.35	2.75

Exploratory Data Analysis (EDA)

Statistical Summary

	Unnamed: 0	carat	depth	table	price	x	y	z
count	53940	53940	53940	53940	53940	53940	53940	53940
mean	26970.5	0.7979	61.7494	57.4572	3932.7997	5.7312	5.7345	3.5387
std	15571.2811	0.474	1.4326	2.2345	3989.4397	1.1218	1.1421	0.7057
min	1	0.2	43	43	326	0	0	0
25%	13485.75	0.4	61	56	950	4.71	4.72	2.91
50%	26970.5	0.7	61.8	57	2401	5.7	5.71	3.53
75%	40455.25	1.04	62.5	59	5324.25	6.54	6.54	4.04
max	53940	5.01	79	95	18823	10.74	58.9	31.8

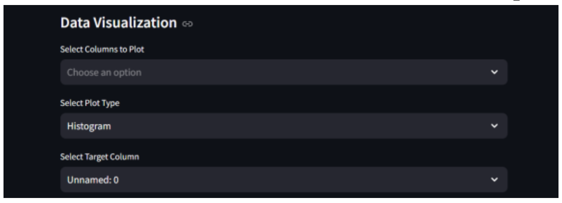
Data Visualization

Select Columns to Plot

Figure 15: Statistical Summary

(Source: Made by self in VS Code)

The general statistics of the given dataset are as follows which is obtained using df.describe(): It includes mean, standard deviation, minimum, maximum value for quantitative variables and it also calculates quantiles that offers information on data distribution as well as dispersion.


Data Visualization

Select Columns to Plot

Choose an option

Select Plot Type

Histogram

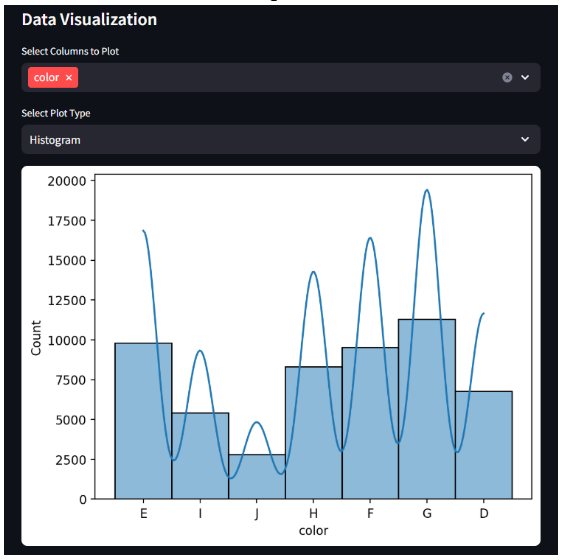
Select Target Column

Unnamed: 0

Figure 16: Selection for data visualization

(Source: Made by self in VS Code)

The fields that can be used for visualization are presented as options into the drop-down list of the form. There are different types of plot options which encompasses histogram, box plot, scatter plot among them which help users to visualize trends and patterns in data.


Data Visualization

Select Columns to Plot

color ×

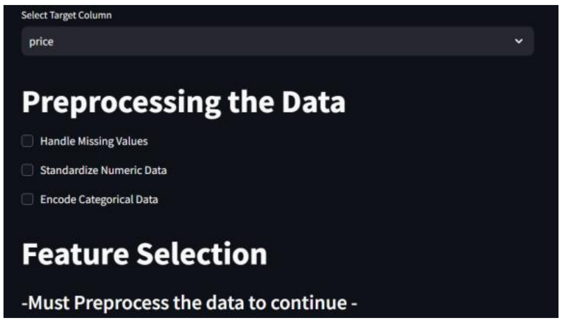
Select Plot Type

Histogram

Figure 17: Visualization created

(Source: Made by self in VS Code)

Once the user selects columns, the system uses Matplotlib and Seaborn libraries to develop a plot. Histograms plot density of distributions, while boxplots characterize spread of values noting outliers on the data and scatter plots illustrate the correlation between two variables.


Select Target Column

price

Preprocessing the Data

☐ Handle Missing Values

☐ Standardize Numeric Data

☐ Encode Categorical Data

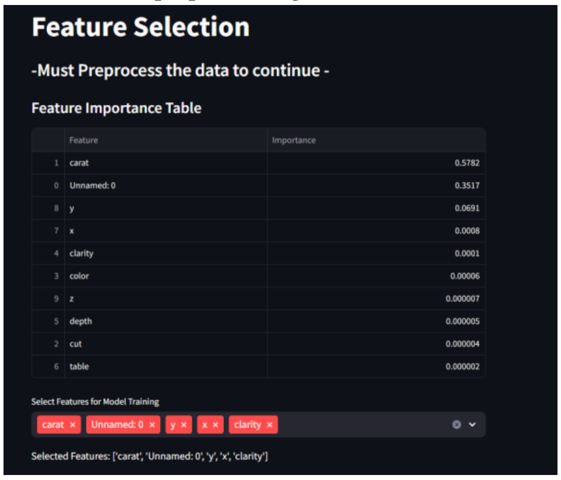
Feature Selection

-Must Preprocess the data to continue -

Figure 18: Select target column and preprocess data

(Source: Made by self in VS Code)

The system invites the users to enter the variable to be used for training the model. Further, basic data preparation steps like handling of missing data, conversion of categorical independent variables, normalizing of numerical data are available for data preprocessing.


Feature Selection

-Must Preprocess the data to continue -

Feature Importance Table

	Feature	Importance
1	carat	0.5782
0	Unnamed: 0	0.3517
8	y	0.0691
7	x	0.0008
4	clarity	0.0001
3	color	0.00006
9	z	0.000007
5	depth	0.000005
2	cut	0.000004
6	table	0.000002

Select Features for Model Training

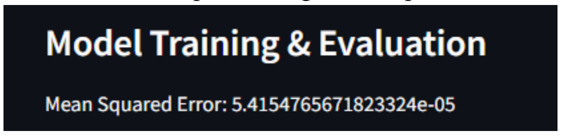
carat × Unnamed: 0 × y × x × clarity ×

Selected Features: ['carat', 'Unnamed: 0', 'y', 'x', 'clarity']

Figure 19: Automatic feature selection based on feature weights

(Source: Made by self in VS Code)

There is an option to display feature importance scores for each feature created in the process of feature selection, which might be useful for manual selection of features to be used in the model training. This improves the predictions and the time taken in performing the computations.


Model Training & Evaluation

Mean Squared Error: 5.4154765671823324e-05

Figure 20: Model Evaluation

(Source: Made by self in VS Code)

Finally, the system displays the metrics of the model, which are accuracy in case of classification model and mean-squared error is in case of a regression model. The accuracy of the selected model is evident from the results of the model on the test data set.

4.3 Limitations

Despite all enhancements made on automation of data pipeline, some factors limiting the scalability and flexibility of the algorithm to be used on real data.

Dataset Compatibility Issues:

1. Due to the organizational structure of receiving and storing data in tables in the format of CSV, it remains unfit for use of unstructured data types such as image processing, text data or use of real-time stream data.
2. Large data sets are often characterized by a large number of objects and attributes and therefore their use in operations can create a memory bottleneck.

Model Generalization Challenges:

1. The automatic pipeline takes decision based on some specific rules and conditions about general machine learning but it may not always choose the

most suitable machine learning model for the specific areas and sectors.
2. The models such as deep learning models that depend on much computational resources are not considered in the current implementation.

Computational Resource Constraints:

1. But utilizing feature selection and optimizing the model makes it more complex and time-consuming especially when the data is huge.
2. The system does not have the ability to do parallel process, and this can reduce the rate of processing large data.

User Dependency on Predefined Options:

1. Although the major operations for the selection of attributes and preprocessing are managed by the system, the users have to make some significant choices such as the selection of features and many more, which requires prior knowledge of ML algorithms.
2. Several of these limitations could be solved in the future versions of the developed system that would improve the system's performance for practical application.

4.4 Conclusion

The AD-PROMISE experiment proves that the chosen set of ML-related tasks and the index of user-controlled vs automatic stages can serve as a backbone for an efficient and easy-to-use tool for automating the outlined stages of the ML workflow. The data preparation and feature selection steps along with model selection and model assessment steps have been effectively incorporated into the front end of this system so that the user's intervention is minimal.

However, the proposed system has some drawbacks concerning the generalization of the dataset, generality of the model, and the computational complexity of the process Nevertheless, all these limitations pave the way for future developments. Solve all these problems in the future to increase scalability, add real-time analysis capability for big data, and enhance the application of deep learning models, so that the system can be used for real world machine learning.

Chapter 5: Conclusion

5.1 Discussion

The aim of the Automated Data Pipeline Optimization for Real-Time Machine Learning Inference is achieved because the proposed solution includes data preprocessing, feature selection, model selection, and evaluation components. The system diminishes the amount of work to be done manually and improves the quality of model algorithms with dynamic features and algorithms choice.

They reveal that the features of data preprocessing can sustain its integrity which saves time and eradicate errors and feature selection can magnify the predictability through the removal of the reiterative variables. The implementation of the pipeline is designed to be used as standalone and to integrate well with Streamlit for use in developing easy-to-use interfaces which requires relative minimal programming knowledge.

However, some drawbacks that could be associated with data clustering or attributed to using this work include the following: compatibility of the datasets under consideration, limitations in computational power and memory, and limitations arising from the use of predesigned machine learning models. Currently, the proposed system operates only on structured tabular data format, therefore can only cater for data of these formats like images or text data. However, the pipeline does not support the streaming of the actual data in real-time, which is useful in such regularly developing data sets.

The study also re-establishes that, indeed, the ML pipeline automation is useful for increasing efficiency, accuracy, and scalability in real-life applications. They didn't scale well, the generalization of the model is still an issue, and it is not very compatible to diverse data formats for the inference of machine learning.

5.2 Recommendations

For improving the efficiency and flexibility of the data processing AML the following suggestions are given:

Improvement of data compatibility: Future implementations should cater for stream data and also data in other formats including images and text data. All of the aforementioned applications would allow for advanced computing tasks in artificial intelligence, especially as related to computer vision, natural language processing, and Internet of Things-based analytics.

ML and DL models: More refined techniques which can be incorporated are NNAS or AutoML for model selection to enhance the adaptability of the system for a number of datasets.

Improving Computational Performance: One way to address this issue is by integrating parallelism and cloud computing in the system mechanism, so as to contain the problem of large datasets.

Adaptive Preprocessing of Data: The use of user-specified preprocessing step is not often effective as the current state-of-art has the ability to learn preprocessing steps according to the properties of a given dataset.

Enhancing the Customization of Parameters: As with any other similar application, extending the choice and letting users adjust hyperparameters and try out different algorithms within the interface would be beneficial for highly specific fields.

According to these suggestions, the system will reach the robust and highly-scalable ML automation tool for various industries, as well as enhance the real-time ML inference.

5.3 Future Work

The current system can, therefore, as a basis for an automated data pipeline, be defined as solid, but there are changes that can be made with regard to the scope and effectiveness of the system worth considering.

Interoperability with the cloud services: The future releases of the system can be hosted on cloud platforms including AWS, Google Cloud, or Microsoft Azure for the purpose of distributed and scalable model training.

An integration of deep learning models: In the near future, they should incorporate advance ML frameworks like TensorFlow, PyTorch for the usage of image classification, NLP and sophisticated time series prediction.

Hyperparameter tuning: adaptive tuning with the viable options often involve using Bayesian optimization, genetic algorithms, or reinforcement learning in order to automate the process of hyperparameter tuning.

Real-Time Data Processing: Apache Kafka or Spark Streaming can be integrated into the system to update data in real-time; therefore, making the system useful for use in risk prediction in the financial markets, credit card risk, and other IoT based applications.

Enhancing the interpretability of models from an explainable AI perspective: We are talking about the application of the SHAP (SHapley Additive Explanations) or LIME (Local Interpretable Model-agnostic Explanations) to increase the model explainability in front of the user to explain why some features affect the results.

As the system has been extended, it will be able to be applied to more practical scenarios and boost its performance in a broad spectrum of industries and fields of machine learning.

5.4 Conclusion

The research does achieve its stated research goals effectively by proposing an Automated Data Pipeline Optimization for Real-Time ML Inference, coverages for the major issues of data preprocessing, feature selection, and model selection. The system brings a logic of an efficient, flexible, and scalable way of automating the essential ML tasks, thus reducing the chances of human interferences and enhancing the accuracy of predictions.

While the system excels at structured datasets, certain differences, the operational status at the time of processing, and computation scaling issues require enhancement. With the deep learning, cloud deployment, and streaming additions, the system can further develop into a robust and highly efficient ML automation system.

Hence, it is possible to state that the present study can be viewed as laying a solid groundwork for further advancements in relation to AMLP. As to the mentioned shortcomings, it is possible to state that the introduction of more advanced automation methods would allow improving the organization of real-time machine learning inference within a wide range of industries.

Reference List

Journals

[1]. Li, Y., Han, Z., Zhang, Q., Li, Z. and Tan, H., 2020, July. Automating cloud deployment for deep learning inference of real-time online services. In IEEE INFOCOM 2020-IEEE Conference on Computer Communications (pp. 1668-1677). IEEE.

[2]. Sinha, R., 2017. Automation of Data Pipelines in Machine Learning Workflows: Trends, Tools, and Challenges. International Journal of Artificial Intelligence and Machine Learning, 4(2).

[3]. González, G. and Evans, C.L., 2019. Biomedical Image Processing with Containers and Deep Learning: An Automated Analysis Pipeline: Data architecture, artificial intelligence, automated processing, containerization, and clusters orchestration ease the transition from data acquisition to insights in medium-to-large datasets. BioEssays, 41(6), p.1900004.

[4]. Hirzel, M., Soulé, R., Schneider, S., Gedik, B. and Grimm, R., 2014. A catalog of stream processing optimizations. ACM Computing Surveys (CSUR), 46(4), pp.1-34.

[5]. Xiang, Y. and Kim, H., 2019, December. Pipelined data-parallel CPU/GPU scheduling for multi-DNN real-time inference. In 2019 IEEE Real-Time Systems Symposium (RTSS) (pp. 392-405). IEEE.

[6]. Jayanthi, M.D., Sumathi, G. and Sriperumbudur, S., 2016. A framework for real-time streaming analytics using machine learning approach. In Proceedings of national conference on communication and informatics-2016. Derakhshan, B., Mahdiraji, A.R., Rabl, T. and Markl, V., 2019, March. Continuous Deployment of Machine Learning Pipelines. In EDBT (pp. 397-408).

[7]. Perumallaplli, R., 2014. Conversational AI for Customer Support: Automation in Large Enterprises. Available at SSRN 5228517.

[8]. Crankshaw, D., Sela, G.E., Mo, X., Zumar, C., Stoica, I., Gonzalez, J. and Tumanov, A., 2020, October. InferLine: latency-aware provisioning and scaling for prediction serving pipelines. In Proceedings of the 11th ACM Symposium on Cloud Computing (pp. 477-491).

[9]. González, G. and Evans, C.L., 2019. Biomedical Image Processing with Containers and Deep Learning: An Automated Analysis Pipeline: Data architecture, artificial intelligence, automated processing, containerization, and clusters orchestration ease the

transition from data acquisition to insights in medium-to-large datasets. BioEssays, 41(6), p.1900004.

[10]. Alves, J.M., Honório, L.M. and Capretz, M.A., 2019. ML4IoT: A framework to orchestrate machine learning workflows on internet of things data. IEEE Access, 7, pp.152953-152967.

[11]. Malikireddy, S.K.R., Algubelli, B. and Tadanki, S., 2021. Knowledge graph-driven real-time data engineering for context-aware machine learning pipelines. European Journal of Advances in Engineering and Technology, 8(5), pp.65-76.

[12]. Boppiniti, S.T., 2021. Real-time data analytics with ai: Leveraging stream processing for dynamic decision support. International Journal of Management Education for Sustainable Development, 4(4).

[13]. Xiang, Y. and Kim, H., 2019, December. Pipelined data-parallel CPU/GPU scheduling for multi-DNN real-time inference. In 2019 IEEE Real-Time Systems Symposium (RTSS) (pp. 392-405). IEEE.

[14]. Elshawi, R., Maher, M. and Sakr, S., 2019. Automated machine learning: State-of-the-art and open challenges. arXiv preprint arXiv:1906.02287.

[15]. Niu, W., Li, Z., Ma, X., Dong, P., Zhou, G., Qian, X., Lin, X., Wang, Y. and Ren, B., 2021. Grim: A general, real-time deep learning inference framework for mobile devices based on fine-grained structured weight sparsity. IEEE Transactions on Pattern Analysis and Machine Intelligence, 44(10), pp.6224-6239.

[16]. Prosper, J., 2019. Deploying Scalable Deep Learning Models for Real-Time Customer Insight.

[17]. Liu, S., Yao, S., Fu, X., Tabish, R., Yu, S., Bansal, A., Yun, H., Sha, L. and Abdelzaher, T., 2020, December. On removing algorithmic priority inversion from mission-critical machine inference pipelines. In 2020 IEEE Real-Time Systems Symposium (RTSS) (pp. 319-332). IEEE.

[18]. Derakhshan, B., Mahdiraji, A.R., Rabl, T. and Markl, V., 2019, March. Continuous Deployment of Machine Learning Pipelines. In EDBT (pp. 397-408).

[19]. Shen, Y., Cao, D., Ruddy, K. and Teixeira de Moraes, L.F., 2020. Near real-time hydraulic fracturing event recognition using deep learning methods. SPE Drilling & Completion, 35(03), pp.478-489.

[20]. Smistad, E., Østvik, A., Salte, I.M., Melichova, D., Nguyen, T.M., Haugaa, K., Brunvand, H., Edvardsen, T., Leclerc, S., Bernard, O. and Grenne, B., 2020. Real-time automatic ejection fraction and foreshortening detection using deep learning. IEEE transactions on ultrasonics, ferroelectrics, and frequency control, 67(12), pp.2595-2604.

[21]. Zhao, Z., Wang, K., Ling, N. and Xing, G., 2021, May. Edgeml: An automl framework for real-time deep learning on the edge. In Proceedings of the international conference on internet-of-things design and implementation (pp. 133-144).3.

[22]. Li, Y., Mahjoubfar, A., Chen, C.L., Niazi, K.R., Pei, L. and Jalali, B., 2019. Deep cytometry: deep learning with real-time inference in cell sorting and flow cytometry. Scientific reports, 9(1), p.11088.

[23]. Zaharia, M., Chen, A., Davidson, A., Ghodsi, A., Hong, S.A., Konwinski, A., Murching, S., Nykodym, T., Ogilvie, P., Parkhe, M. and Xie, F., 2018. Accelerating the machine learning lifecycle with MLflow. IEEE Data Eng. Bull., 41(4), pp.39-45.

[24]. Crankshaw, D., Sela, G.E., Mo, X., Zumar, C., Stoica, I., Gonzalez, J. and Tumanov, A., 2020, October. InferLine: latency-aware provisioning and scaling for prediction serving pipelines. In Proceedings of the 11th ACM Symposium on Cloud Computing (pp. 477-491).

[25]. Alves, J.M., Honório, L.M. and Capretz, M.A., 2019. ML4IoT: A framework to orchestrate machine learning workflows on internet of things data. IEEE Access, 7, pp.152953-152967.

[26]. Elshawi, R., Maher, M. and Sakr, S., 2019. Automated machine learning: State-of-the-art and open challenges. arXiv preprint arXiv:1906.02287.

[27]. Jiang, Z., Chen, T. and Li, M., 2018. Efficient deep learning inference on edge devices.

[28]. Abbas, A. and Kollwitz, E., 2021. Building Robust AI/ML Data Pipelines with Scalable AI Workflows MLOps and Software Automation in Medical Imaging Processing.

[29]. Verma, G., Gupta, Y., Malik, A.M. and Chapman, B., 2021, June. Performance evaluation of deep learning compilers for edge inference. In 2021 IEEE international parallel and distributed processing symposium workshops (IPDPSW) (pp. 858-865). IEEE.