Automated Data Preparation using Semantics of Data Science Artifacts

Shubham Vashisth

A Thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Computer Science (Computer Science) at

Concordia University

Montréal, Québec, Canada

August 2023

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Shubham Vashisth**

Entitled: **Automated Data Preparation using Semantics of Data Science Artifacts**

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science (Computer Science)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

______________________ Chair
Dr. Jinqiu Yang

______________________ Examiner
Dr. Jinqiu Yang

______________________ Examiner
Dr. Abdelhak Bentaleb

______________________ Supervisor
Dr. Essam Mansour

Approved by ______________________
Dr. Joey Paquet, Chair
Department of Computer Science and Software Engineering

__________ August, 2023 ______________________
Dr. Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

Automated Data Preparation using Semantics of Data Science Artifacts

Shubham Vashisth

Data preparation is critical for improving model accuracy. However, data scientists often work independently, spending most of their time writing code to identify and select relevant features, enrich, clean, and transform their datasets to train predictive models for solving a machine learning problem. Working in isolation from each other, they lack support to learn from what other data scientists have performed on similar datasets. This thesis addresses these challenges by presenting a novel approach that automates data preparation using the semantics of data science artifacts. Therefore, this work proposes KGFarm [1], a holistic platform for automating data preparation based on machine learning models trained using the semantics of data science artifacts, captured as a knowledge graph (KG). These semantics comprise datasets and pipeline scripts. KGFarm seamlessly integrates with existing data science platforms, effectively enabling scientific communities to automatically discover and learn from each other's work. KGFarm's models were trained on top of a KG constructed from the top-rated 1000 Kaggle datasets and 13800 pipeline scripts with the highest number of votes. Our comprehensive evaluation uses 130 unseen datasets collected from different AutoML benchmarks to compare KGFarm against state-of-the-art systems in data cleaning, data transformation, feature selection, and feature engineering tasks. Our experiments show that KGFarm consumes significantly less time and memory compared to the state-of-the-art systems while achieving comparable or better accuracy. Hence, KGFarm effectively handles large-scale datasets and empowers data scientists to automate data preparation pipelines interactively.

[1] https://github.com/CoDS-GCS/kgfarm

Acknowledgments

I would like to express my deepest gratitude to my supervisor, Dr. Mansour, for his unwavering guidance, support, and mentorship throughout my research journey. His expertise and dedication have played a vital role in shaping me into a better researcher. I am truly fortunate to have had the opportunity to work under his supervision. I extend my sincere appreciation to the CoDS lab for providing an enriching environment and fostering a culture of academic excellence. I am grateful to Dr. Khaled Ammar and Dr. Antonio Cavalcante from BorealisAI for their invaluable contributions to my research, enabling me to develop practical solutions for real-world challenges. I would like to acknowledge the support and involvement of Niki Monjazeb, Mossad Helali, and Philippe Carrier, whose dedication and enthusiasm have enriched the quality of my research.

My deepest appreciation goes to my family and my best friend for their unwavering trust, belief, and support throughout my academic journey. I am grateful to my father, Dr. S.K Sharma, and my mother, Priti Sharma, for imparting in me the importance of education and perseverance. I also extend my gratitude to my brother, Sarthak Vashisth, for his constant encouragement. Additionally, I would like to thank my dear friend, Ishika Dhall for her unwavering support and belief in me. Finally, I would like to acknowledge my home country India for providing me with the educational opportunities to pursue my dreams and enriching my academic pursuits.

In conclusion, this thesis would not have been possible without the support and contributions of the individuals and institutions mentioned above. I am grateful for their guidance, encouragement, and belief in my abilities.

Contents

List of Figures

List of Tables

Chapter 1

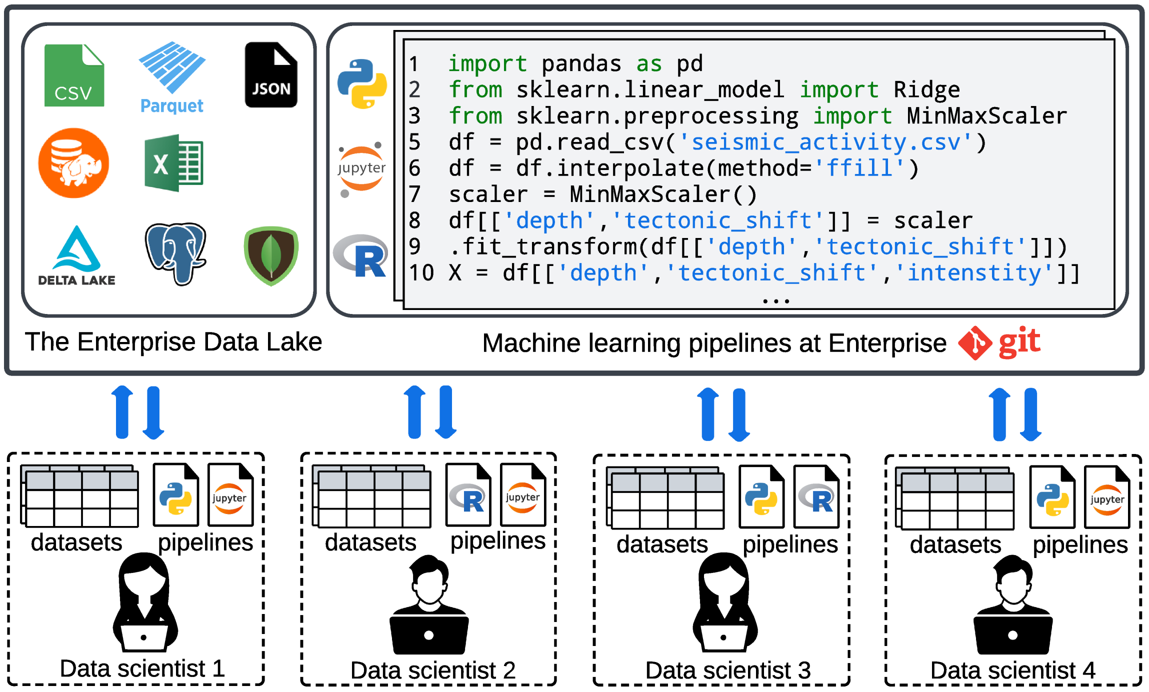
Introduction

1.1 Overview

Data preparation is a fundamental step for data science (Peng, Wu, Lockhart, & et al, 2021; Rezig, Bhandari, Fariha, & et al., 2021). It aims to construct the most informative version of a dataset to help train an accurate and reliable model. Data preparation is conducted to ensure data quality and quantity. Specifically for machine learning, data preparation involves operations such as feature identification, data enrichment, data cleaning, data transformation, and feature selection. In order to prepare data, data scientists devote a substantial amount of time and effort to meticulously code data preparation pipelines. They often work in isolation, with no support from previous work performed by fellow data scientists on similar datasets. The knowledge gained by data scientists from developing Data Science Pipelines (DSPs) is often not shared, which leads to a phenomenon called *"tribal knowledge"*, as illustrated in Figure 1.1. Therefore, there is a need to establish a culture of collaboration and knowledge-sharing to improve the efficiency and effectiveness of different aspects of DSPs through automation (Mansour, Srinivas, & Hose, 2021).

The DSPs created by data scientists in each enterprise are valuable assets for future projects. Furthermore, open ML portals, such as Kaggle [1] and OpenML [2] provide access to public repositories containing thousands of datasets and hundreds of thousands of DSPs. The wealth of knowledge

[1] https://www.kaggle.com
[2] https://www.openml.org



```
import pandas as pd
from sklearn.linear_model import Ridge
from sklearn.preprocessing import MinMaxScaler
df = pd.read_csv('seismic_activity.csv')
df = df.interpolate(method='ffill')
scaler = MinMaxScaler()
df[['depth','tectonic_shift']] = scaler
.fit_transform(df[['depth','tectonic_shift']])
X = df[['depth','tectonic_shift','intenstity']]
                    ...
```

Figure 1.1: Data scientists work in isolation to perform time-consuming, code-extensive, repetitive, and hard-to-scale data preparation. The experience gained during the data preparation process remains unshared.

available at these repositories can be leveraged to improve the efficiency and accuracy of DSPs and facilitate collaboration and knowledge-sharing among data scientists. By building upon the insights and learnings gained from existing DSPs, data scientists can avoid reinventing the wheel and instead focus on solving new and complex problems. However, these large repositories in data lakes contain heterogeneous datasets and pipelines, i.e., datasets in different formats (CSV, JSON, Parquet, Delta table, etc.) and pipeline scripts written in different languages and libraries. Hence, it is difficult to navigate this vast amount of knowledge to explore or automatically learn from relevant pipelines applied to similar datasets. Knowledge Graphs (KG) that represent datasets and DSPs (Helali, Vashisth, Carrier, & et al, 2021) can help enterprises stay at the forefront of data science innovation and drive more excellent value from the experience gained by their data scientists.

1.2 Contributions

This thesis proposes KGFarm, a comprehensive platform built on top of the KGLiDS system, aiming to automate data preparation holistically. KGFarm leverages task-specific trained models to automate various stages of the data preparation process, including data cleaning, data transformation, feature selection, and feature engineering. These models provide valuable assistance to data scientists when working with new datasets or encountering novel data science challenges by suggesting the most suitable operations for the task at hand. Moreover, to identify relevant features for a specific entity in a machine learning task, we employ ML for primary-key foreign-key (PK-FK) discovery (Rostin, Albrecht, Bauckmann, Naumann, & Leser, 2009), to estimate the physical representation of an abstract entity through column embeddings. This physical representation also serves as a basis for data enrichment, enabling the recommendation of join keys over schema-less data in data lakes (Helal, Helali, Ammar, & Mansour, 2021; Helali et al., 2021). In KGFarm data preparation operations such as data cleaning and data transformation are formalized as multiclass classification where the goal is to predict the most appropriate cleaning or transformation operations from a predefined set of operations. Additionally, feature selection is formalized as a binary classification task, where the model is used to infer the probability with which a feature should be selected to predict the given target variable.

Training and inference of these task-specific models is done using content embeddings of the datasets i.e. a table or columns instead of raw data values to improve generalization and scalability of this approach. These embeddings are computed using a deep learning approach presented in (Helali et al., 2021) inspired from (Mueller & Smola, 2019). The quality of these models depends on the performance of pipeline scripts and the diversity of datasets used in constructing the KG. In this work, we constructed a data science KG from top-rated 1000 Kaggle datasets and 13800 pipeline scripts with the highest number of votes. We trained our models based on the semantics captured in this KG. We conducted a comprehensive evaluation using 130 unseen datasets collected from different open data portals and AutoML benchmarks (Dua & Graff, 2017; Helali, Mansour, Abdelaziz, & et al, 2022). These datasets demand different kinds of data preparation, where each dataset is associated with an ML task i.e. binary classification, multiclass classification, or regression.

We use these datasets to compare KGFarm against the state-of-the-art (SOTA) systems in data cleaning (Biessmann, Rukat, Schmidt, & et al, 2019; Rekatsinas, Chu, Ilyas, & et al, 2017), data transformation (Nargesian, Samulowitz, Khurana, & et al., 2017), feature selection and feature engineering (Kaul, Maheshwary, & Pudi, 2017) tasks. Our experiments show that KGFarm achieves comparable accuracy to the SOTA while significantly outperforming them in terms of processing time and memory usage. In addition, we compared KGFarm's PK-FK discovery approach which makes use of light-weight column embeddings to computationally expensive inclusion dependencies (Rostin et al., 2009). Furthermore, we present a real-world use case of KGFarm deployed in preparing data for a mechanical engineering research team working on a smart city project that aims to address stability challenges in hybrid power systems. Thereby, illustrating KGFarm's capability in automating different aspects of data preparation as a holistic platform while dealing with large-scale diverse datasets.

In summary, the contributions of this thesis are:

- a fully-fledged platform (KGFarm [3]) automating different aspects of data preparation in an interactive and scalable manner.
- a novel formalization of data cleaning, transformation, and feature selection as classification tasks based on the semantics of data science artifacts and column embeddings. This formalization enables KGFarm to scale to large datasets.
- a novel approach for feature identification by estimating abstract entities over physical columns. This estimation leverages machine learning to efficiently discover PK-FK relationships in schema-less data sources.
- a real-world use case of KGFarm deployed in the mechanical engineering sector of a smart city project to address stability challenges in hybrid power systems, showcasing KGFarm's capability in automating data preparation with respect to simulated data.
- a comprehensive evaluation using 130 unseen datasets from AutoML benchmarks and SOTA, such as HoloClean (Rekatsinas et al., 2017) and DataWig (Biessmann et al., 2019) for data

[3]https://github.com/CoDS-GCS/kgfarm

cleaning, LFE (Nargesian et al., 2017) for data transformation and AutoLearn (Kaul et al., 2017) for feature engineering. Our experiments show the superiority of KGFarm over SOTA in terms of time and memory performance while achieving comparable or better accuracy.

1.3 Outline

This thesis consists of six chapters that delve into the details behind the approach followed to automate the problem of data preparation using the KGFarm platform. Chapter 2 provides an overview of related literature, focusing on the concept of human-in-the-loop for data science and various systems employed in the space of automated data preparation. Chapter 3 establishes the necessary background by discussing the semantics of data science artifacts, the concept of linked data science, and the motivation behind automated data preparation techniques leveraging linked data science. In Chapter 4, the KGFarm platform is presented, highlighting its system architecture, and supported data preparation operations including feature identification, data enrichment, data cleaning, data transformation, feature selection, and feature engineering. This is followed by an exploration of the key characteristics of KGFarm such as taylor-stitched recommendations, scalability, and integration with existing data science workflows. Chapter 5 showcases a use case that demonstrates KGFarm's seamless integration into a typical data science pipeline followed by a thorough evaluation of the KGFarm versus state-of-the-art systems in automating data preparation for machine learning tasks. Finally, in Chapter 6, the thesis concludes with a discussion on future work and its potential impact on data science innovation, and knowledge sharing for enterprises.

Chapter 2

Literature Review

2.1 Human-in-the-loop for Data Science

Existing data discovery systems and data science platforms do not automate data preparation and feature engineering based on the knowledge extracted from the data science repositories (Mansour et al., 2021). There are several systems that utilize human-in-the-loop to accelerate data science workflow. These systems can be bifurcated into three categories based on the environment they support, mainly spreadsheet environments, workflow environments, and notebook environments.

For example, Trifacta (Trifacta, 2023) is a software that assists data scientists to prepare data by enabling them to visually explore and interact with their data by providing profiled data statistics in a spreadsheet environment. In the category of workflow environment, there are systems and platforms such as Einblick (Einblick, 2023) and Alteryx (Alteryx, 2023) that provide a canvas GUI with drag-and-drop functionality to add several configurable data preparation operations such as join, visualization, etc. without writing explicit code. Finally, the notebook environment includes libraries and packages such DataPrep (Peng et al., 2021) that are easy to integrate into existing machine learning and data science pipelines. DataPrep is an open-source library that aims to accelerate the data preparation process by automating Exploratory Data Analysis (EDA) through simple APIs that offer profiled data statistics and visualizations for data preparation in a scalable fashion.

As these complex machine learning pipelines are often developed by data scientists and machine learning practitioners, a notebook environment is highly preferred in these scenarios due to the fact that 1) ML pipeline generation process is highly iterative in nature (Peng et al., 2021). 2) ML pipelines involve the application of several other libraries like PyTorch (Paszke et al., 2017), TensorFlow (*TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*, n.d.), and Theano (Al-Rfou et al., 2016) while modeling. 3) Notebook environment allows more freedom to the data scientists in designing their custom pipelines, unlike spreadsheet and workflow environments.

2.2 Related Systems

Systems automating the EDA process such as DataPrep (Peng et al., 2021) do try to accelerate the pipeline generation process by automating the data profiling and visualizations in data preparation however, the core problems described in 1.1 including 1) High-time consumption, 2) Code-intensiveness 3) Lack of scalability 4) Repetitiveness and 5) Isolated development are minimized to some extent but not addressed completely.

Numerous systems have been developed to address specific aspects of data preparation, including data cleaning, data transformation, and feature engineering. Among these, HoloClean (Rekatsinas et al., 2017) stands out as a comprehensive framework that unifies qualitative and quantitative data repairing approaches. It integrates integrity constraints, external data sources, and statistical properties of the input data to automatically generate a probabilistic program for data repair. The core idea of HoloClean is to consider the input dataset as a noisy version of an underlying clean dataset and leverage all available signals to propose data repairs. The framework employs statistical learning and probabilistic inference techniques to perform data repairs effectively.

Another promising system within the scope of data cleaning is DataWig (Biessmann et al., 2019) which is an open-source Python library that specializes in missing value imputation for tables with heterogeneous data types. It offers a wide range of feature extractors, including advanced deep learning techniques, and adopts an end-to-end learning approach using the symbolic API of Apache mxnet (Chen et al., 2015), ensuring efficient execution on both CPUs and GPUs environments. The imputation model in DataWig is inspired by well-established methods such as MICE (Multivariate Imputation by Chained Equations) (van Buuren & Groothuis-Oudshoorn, 2011), allowing it to perform imputations in a chained manner while considering the relationships between variables.

For data transformations and feature engineering, there has been extensive research over the past decade. Work by (Waring, Lindvall, & Umeton, 2020) discusses several systems that make use of unique learning methods for automating data transformation and feature engineering. Frameworks such as (Kanter & Veeramachaneni, 2015; Katz, Shin, & Song, 2016; Kaul et al., 2017; Lam et al., 2017) make use of an expand-reduce learning approach. ExploreKit (Katz et al., 2016) is one of the feature engineering frameworks designed for automated feature generation and selection in machine learning tasks. It leverages an expand-reduce learning approach, wherein it generates a large set of candidate features by combining information from the original features to maximize predictive performance based on user-selected criteria. To handle the exponential growth of the feature space, ExploreKit introduces a learning-based feature selection approach. This method efficiently predicts the utility of new candidate features, allowing for the identification of relevant features while outperforming traditional feature selection techniques. ExploreKit's iterative process involves candidate feature generation, ranking, evaluation, and selection, allowing it to choose a subset of features that minimize the learning algorithm's error.

Subsequently in data transformation and feature engineering, AutoLearn (Kaul et al., 2017) demonstrated superior performance as compared to ExploreKit. AutoLearn uses a regression-based approach to automate feature engineering. AutoLearn exhibited enhanced capabilities in generating highly predictive and domain-generalizable features. A key feature of AutoLearn is its utilization of distance correlation to mine significant pairwise associations between features, both linear and non-linear, without relying on domain-specific heuristics. Moreover, it incorporates a measure to assess the extent of forecast deviation concerning the independent variable. To enhance model generalization and prevent overfitting, AutoLearn employs a two-step process comprising stability selection and information gain. Through these steps, AutoLearn ensures the creation of robust and informative features, contributing to improved performance of machine learning models.

Concurrently, techniques like Nargesian et al. (2017) were also proposed that used meta-learning to automate data transformation and feature engineering process. Concurrently, techniques like LFE Nargesian et al. (2017) were also proposed, utilizing a meta-learning approach to automate the data transformation and feature engineering process. LFE employs past feature engineering experiences to learn the effectiveness of applying transformations on numerical features. Unlike traditional approaches, LFE does not rely on model evaluation or explicit feature expansion and selection. It recommends a set of useful transformations for features, considering ten unary and four binary transformations, such as log, square-root, frequency, square, round, tanh, sigmoid, isotonic regression, zscore, normalization, sum, subtraction, multiplication, and division. To represent numerical feature characteristics, LFE uses Quantile Sketch Array (QSA), a non-parametric representation that summarizes data into a small number of buckets, effectively capturing the approximate Probability Distribution Function of values. LFE has been evaluated over several open datasets from UCI and OpenML, showcasing its effective and efficient performance in optimizing the data transformation and feature engineering process.

Feature stores are being widely adopted for storing and managing reusable features by serving as centralized repositories. However, most existing feature stores, such as Feast (Feast: Feature Store for Machine Learning, 2022; Kakantousis, Kouzoupis, Buso, & et al., 2019), require manual maintenance. Other data preparation systems such as Auto-Suggest (Yan & He, 2020) learns from a collection of data science notebooks to recommend operations, such as Join, Pivot, Unpivot, and GroupBy but lacks the crucial data preparation operations. Additionally, AutoML systems, such as KGpip(Helali et al., 2022) have been proposed that learn to predict mainly a classifier and perform hyperparameter tuning. They may include data prepossessing by analyzing the raw data. Unlike these systems, our proposed approach automates data preparation pipelines for operations, such as data cleaning, transformation, and feature selection, based on the semantics of data science artifacts.

At the same time, KGLiDS (Helali et al., 2021) is a scalable platform we developed to construct a data science KG, called LiDS graph Helali et al. (2021) as shown in Figure 3.1. This system does not provide models to automate data preparation and feature engineering pipelines. Hence, there is a need for systems that automate data preparation and feature engineering based on the this highly interconnected linked data science knowledge graph, which contains the semantics of vast pipelines and datasets available at the public or enterprise level.

Chapter 3

Semantics of Data Science Artifacts

In the realm of data science, the notion of the semantics of data science artifacts holds paramount importance in facilitating effective collaboration, reproducibility, and knowledge exchange. Data scientists, as they engage in the analysis of vast amounts of data, generate a plethora of artifacts, ranging from datasets to pipeline scripts, which form the foundation of their work.

However, the intricate nature of these artifacts often poses challenges for individuals who are not intimately involved in their creation. It is at this juncture that understanding the semantics of these artifacts becomes pivotal. By comprehending the underlying meaning, relationships, and functionalities encoded within these artifacts, data scientists can transcend the barriers of complexity and ensure their work is accessible and comprehensible to a wider audience. By establishing a shared understanding of the semantics, data scientists foster a collaborative environment wherein the exchange of ideas, methodologies, and insights becomes seamless. This, in turn, spurs innovation and promotes the efficient utilization of existing work. For instance, when faced with similar problems, teams can leverage pre-existing artifacts' semantics to expedite their problem-solving process. This not only saves time and effort but also facilitates the accumulation of knowledge within data science.

In the context of machine learning, the interplay between data science artifacts directly impacts the performance and effectiveness of the models developed. The semantics of these artifacts, such as the data preparation steps, feature engineering techniques, and model configuration, dictate the behavior and outcomes of the machine learning models. Understanding these semantics is essential for interpreting and reproducing the results obtained, thereby ensuring transparency and reliability in the field. Moreover, the semantics of data science artifacts serve as the bedrock for reproducibility. By documenting and communicating the semantics of their artifacts, data scientists enable others to validate and replicate their findings. This not only strengthens the scientific rigor of the field but also encourages the building of trust and confidence among researchers and practitioners.

This chapter introduces an innovative approach to effectively utilize the semantics of data science artifacts in a systematic fashion. The sections within this chapter delve into a comprehensive exploration of the adopted approach, which aims to facilitate the concept of *"linked data science"* and automation of data preparation as one of its evident applications.

3.1 Linked Data Science & KGLiDS

In the pursuit of advancing data science practices, the concept of linked data science (Helali et al., 2021) emerges as a visionary approach (Mansour et al., 2021) that seeks to establish a comprehensive navigational structure, seamlessly connecting diverse data science artifacts. Linked data science represents a paradigm shift, wherein datasets, pipeline scripts, and code libraries are not seen as isolated entities but rather as interconnected components of a larger ecosystem. By capturing the semantics of these artifacts, linked data science unlocks a deeper understanding of their interdependencies and establishes a cohesive network that transcends disciplinary boundaries.

One of the key benefits of linked data science lies in its ability to foster cross-domain collaborations. By bridging the gaps between disparate fields and domains, this approach facilitates the exchange of knowledge, methodologies, and best practices. Data scientists from different disciplines can seamlessly navigate through the linked structure, discovering relevant artifacts, and leveraging their insights to tackle complex challenges that require multidisciplinary expertise. The interconnected nature of linked data science empowers researchers to explore artifacts, uncover novel insights, and drive innovation at the intersection of various domains.

Furthermore, linked data science acts as a catalyst for knowledge sharing and dissemination across platforms, enterprises, and institutions. Through the establishment of standardized semantic relationships, data scientists can effectively communicate the purpose, structure, and applicability of their artifacts. This enables practitioners to discover, understand, and build upon existing work, fueling the collective intelligence of the data science community.

Another significant advantage of linked data science lies in its potential to automate and streamline data science workflows. By harnessing the interconnectivity and semantic richness of the artifacts, automated systems can intelligently navigate through the linked structure, retrieving relevant datasets, discovering relevant pipeline scripts, and leveraging code libraries. This automation not only enhances efficiency but also ensures consistency and reproducibility in data science endeavors, reducing manual errors and enabling faster iterations in the development and deployment of models.

Creating and maintaining a scalable structure that enables linked data science poses significant challenges in the field. The complexity arises from the need to capture, organize, and interconnect a diverse range of data science artifacts, including datasets, pipeline scripts, and code libraries while ensuring the coherence and integrity of the underlying semantics. Establishing meaningful relationships and interdependencies among these artifacts, especially in large-scale and rapidly evolving data science environments, requires careful design and robust infrastructure.

The challenges lie not only in the technical aspects of building and managing the structure but also in fostering a collaborative and sustainable ecosystem. Data science artifacts are constantly evolving, and new artifacts are continuously being created. Thus, ensuring the longevity and adaptability of the linked data science structure becomes a crucial concern. Additionally, accommodating heterogeneous data sources (Helal et al., 2021), addressing data quality and consistency issues, and handling the evolving nature of data science methodologies further compound the challenges.

In response to these challenges, we have developed KGLiDS [1] (Knowledge Graph-based Linked Data Science) (Helali et al., 2021), a pioneering platform that addresses the complexities of enabling linked data science in a scalable manner. KGLiDS employs knowledge graph technologies to capture and represent the semantics of data science artifacts in a structured manner. By leveraging machine learning techniques, it extracts valuable insights from these artifacts and populates the knowledge graph, forming the backbone of the linked data science framework.

Two fundamental components in KGLiDS are Data Profiling and Pipeline Abstraction. The Data Profiling component focuses on analyzing datasets, generating insights about their characteristics, and enriching the knowledge graph with detailed information about the datasets and their attributes.

[1] https://github.com/CoDS-GCS/kglids

It employs scalable deep learning-based data profiling techniques and fine-grained type inference methods to construct a global representation of datasets, including column embeddings that capture similarities between column values. The Pipeline Abstraction component analyzes data science pipelines, capturing their structure and flow, and enriches the knowledge graph with insights about the data science workflow. It helps in understanding the structure and flow of data processing in pipelines, which is essential for constructing the LiDS graph. By extracting relevant information from pipelines, this component enhances the knowledge graph's representation of the data science artifacts and their interconnections.

These two components, Data Profiling, and Pipeline Abstraction, play vital roles in enhancing the knowledge graph's richness and comprehensiveness. They provide valuable insights into datasets and pipelines, enabling data scientists to explore and navigate interconnected resources effectively. By incorporating these insights into the knowledge graph, KGLiDS empowers data scientists to uncover meaningful relationships, automate tasks, and unlock the full potential of linked data science in a scalable and efficient manner. Through the seamless integration of Data Profiling and Pipeline Abstraction, KGLiDS enables data scientists to explore, analyze, and share knowledge within the data science community. By leveraging the capabilities of these components, data scientists can gain deeper insights into datasets, understand the flow of data processing in pipelines, and harness the power of linked data science for improved decision-making and innovation.

3.2 Capturing Data Semantics with Column Embeddings

Capturing the semantics of data is a fundamental aspect of the linked data science approach. In linked data science, one of the key challenges is to capture the similarities between column values and leverage them to enhance data understanding and automate various data science tasks. To address this challenge, the concept of column embeddings was adopted in KGLiDS, providing a powerful mechanism to capture data semantics and facilitate efficient data profiling. The primary motivation behind utilizing column embeddings in KGLiDS is to capture column similarity. By representing columns as embeddings, we can quantify the similarity between their values and establish meaningful connections between related columns. This enables more accurate predictions of column content similarities and facilitates data discovery without exposing the raw content of datasets. This is particularly valuable in enterprise settings where access to raw data may be restricted, but insights derived from column-level similarities can still be leveraged.

To generate column embeddings, KGLiDS incorporates a technique known as the column learned representation (CoLR). The CoLR approach builds upon the concept of deep embeddings of distributions, inspired by (Mueller & Smola, 2019) which aims to identify high-confidence variable matches by leveraging learned vector embeddings of datasets. By adaptively accounting for natural forms of data variation encountered in practice, CoLR produces fixed-size models capable of generating column embeddings. The data profiling component in KGLiDS plays a crucial role in generating the CoLR embeddings for each column. It takes into account the fine-grained type information and raw column values, leveraging pre-trained embedding models to capture the underlying semantics. By considering both the type and values, the CoLR approach ensures a more comprehensive representation of the column, enabling a more accurate capture of column similarities.

Notably, capturing data semantics using column embeddings provides several benefits. Firstly, it offers a compact representation of fixed-size embeddings, reducing storage requirements while maintaining the essential information about column similarities. Additionally, KGLiDS decomposes datasets into independent tables and further decomposes each table into columns. This decomposition allows for scalable profiling and provides deeper insights into the data structure, facilitating more granular analysis.

In conclusion, the utilization of column embeddings in KGLiDS to capture the underlying semantics of data plays a crucial role. These embeddings provide a solid foundation for automating several applications in data science by leveraging the LiDS graph and KGLiDS platform.

3.3 Knowledge Graph for Linked Data Science

KGLiDS utilizes graph technology to generate the LiDS (Linked Data Science) graph, which captures the semantics of data science artifacts and represents them in a structured and navigable

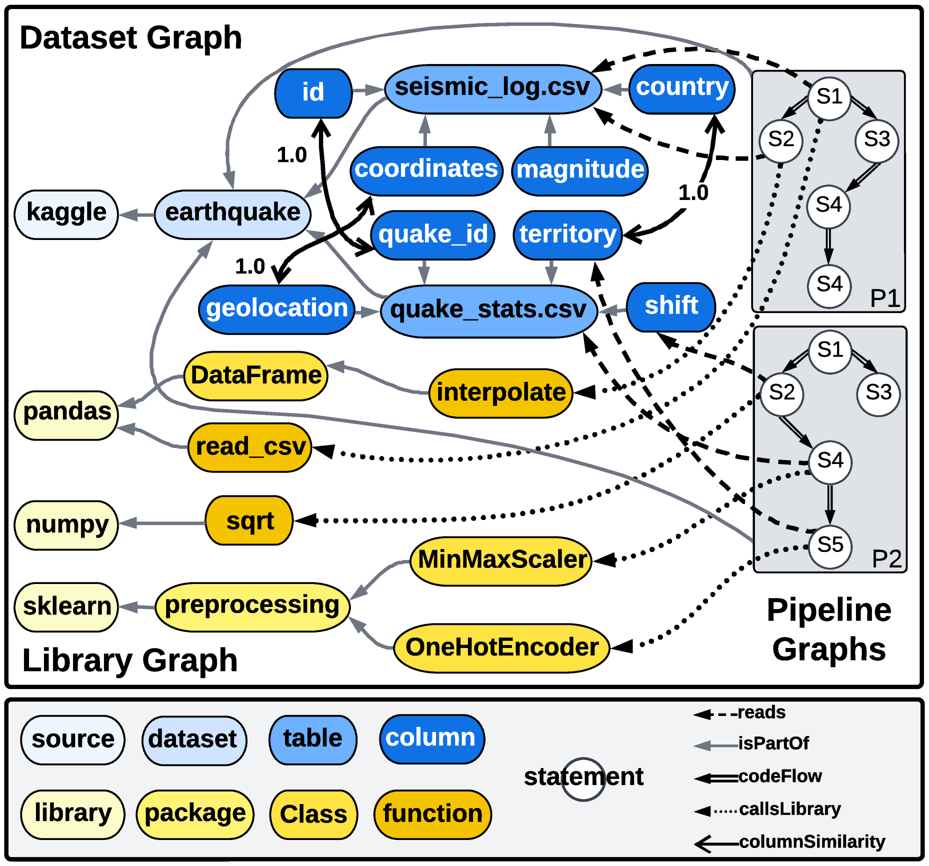


Figure 3.1: A summary of the LiDS graph, comprising the physical data science artifacts such as dataset, library, and pipeline graphs.

manner. This graph serves as the foundation for automating various data science tasks and streamlining workflows, saving valuable time and effort for data scientists.

To ensure standardized and well-structured storage of the knowledge graph, KGLiDS utilizes the LiDS ontology specifically designed for Linked Data Science. This ontology defines the entities involved in data science platforms, such as data, pipelines, and libraries, along with their relationships. It encompasses 13 classes, 19 object properties, and 10 data properties, leveraging the Web Ontology Language (OWL 2) and Uniform Resource Identifiers (URIs) for easy publication and sharing of the LiDS graph on the Web. Each entity within the ontology is associated with an RDF label and RDF type, facilitating RDF reasoning on top of the LiDS graph.

In the LiDS knowledge graph, physical data science artifacts, such as datasets, tables, columns, libraries, and pipeline scripts, are represented as nodes. These nodes are interconnected by edges that depict various relationships, including the *codeFlow* between pipeline statements, *columnSimilarity* between distinct columns, *isPartOf* relations between artifacts, and more. Figure 3.1 provides an example of a LiDS graph, illustrating the holistic view of different data science artifacts and their interactions. For instance, it showcases the *columnSimilarity* between the *country* and *territory* columns belonging to different tables. The graph also captures the hierarchical relationships among artifacts, such as the *country* column being *isPartOf* the *seismic_log.csv* table, which in turn is *isPartOf* the *earthquake* dataset, and so on. Furthermore, the graph captures the application of various data science operations to columns and datasets, such as the *sqrt* operation from the *numpy* library applied to the *shift* column. Overall, the LiDS graph provides an effective navigational structure and establishes meaningful connections among physical data science artifacts, enabling data scientists to explore, exchange, and learn from them more effectively.

3.4 An Augmented LiDS Graph for Data Preparation

Linked Data Science enabled by KGLiDS system opens the scope of automation over typical data science and machine learning workflows. It also creates opportunities to address several non-trivial problems in data science. For example, KGpip is the state-of-the-art (Helali et al., 2022) meta-learning system that leverages graph neural networks (Zhou et al., 2020) and column embeddings on top of pipeline graphs to optimize AutoML tasks. In a similar fashion, the work in this thesis exploits the KGLiDS system and LiDS graph to address the time-consuming, hard-to-scale, code-intensive and isolated problem of data preparation efficiently and effectively.

In machine learning, the problem of feature identification over large data lakes for modeling (Hai, Kang, Koutras, Ionescu, & Katsifodimos, 2022) is addressed by augmenting the LiDS knowledge graph. This augmentation includes the addition of abstract concepts like entity and feature view beside the physical data science artifacts like datasets and pipeline scripts. An entity (Feast: Feature Store for Machine Learning, 2022) here is a distinct object or concept which is physically represented by a column. Identifying the physical representation of an entity is the key to discovering features. Figure 3.2 illustrates an instance of such an augmented LiDS graph where an entity *quake* is physically *representedBy* column *quake_id* and *feature view 01* is physically represented

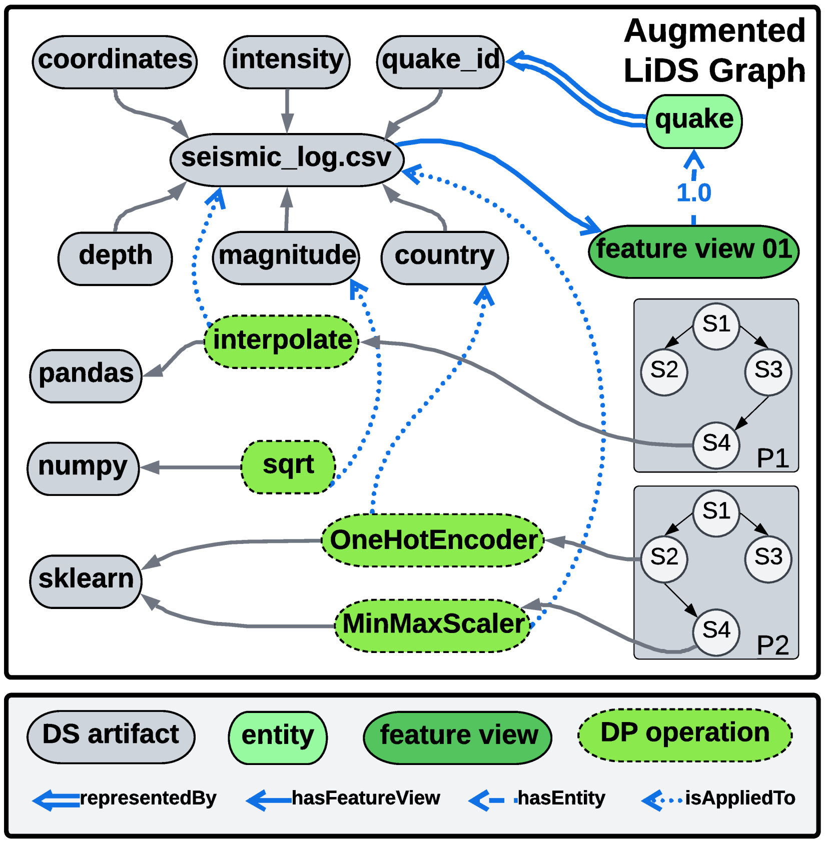


Figure 3.2: An augmented LiDS graph, which highlights the abstraction over physical data science artifacts (column abstracted as entity and table abstracted as feature view) along with data preparation operations.

by the table *seismic_log.csv*. A feature view maintains the information about the entity and the set of features that define that particular entity. For example in figure 3.2 table *seismic_log.csv hasFeatureView feature view 01* which *hasDefaultEntity* as *quake*. From here, *feature view 01* can be inferred to contain columns such as *coordinates, intensity, depth, magnitude*, etc. which serve as the features defining the entity *quake*.

The LiDS graph contains several types of nodes for representing distinct elements of a pipeline script such as a package, Class, function, etc. However, there are certain sets of packages, Classes, and functions that specifically contribute towards data preparation for machine learning. The augmented LiDS graph illustrated in 3.2 highlights a few of such data preparation operations. For example the *interpolate* operation which *isAppliedTo* the *seismic_log.csv* table and the *sqrt* operation which *isAppliedTo* the *magnitude* column. This direct relationship between the operation and features creates a lucrative opportunity to train machine learning models that are capable of predicting data preparation operations while dealing with ad-hoc datasets by benefiting from the knowledge previously accumulated by other data scientists.

Therefore to incorporate the above-discussed methodologies so as to solve the problem of data preparation and feature discovery effectively, this thesis proposes the KGFarm system which augments the LiDS knowledge graph with abstractions that are critical for data preparation. KGFarm uses machine learning to predict primary-key foreign-key PK-FK column relationships that are capable of uniquely representing and integrating data and therefore serve as the physical representation of an entity. Moreover, KGFarm exploits the CoLR embeddings and uses the augmented LiDS graph to generate labeled data to train task-specific recommendation models to automate data cleaning, data transformation, and feature selection tasks. Chapter 4 presents the in-depth architecture and methodology adopted in the KGFarm system to prepare data automatically by augmenting the LiDS graph with high accuracy and efficiency.

Chapter 4

KGFarm: A Holistic Platform for Automating Data Preparation

The previous chapter provided an overview of linked data science and introduced the motivation behind leveraging the KGLiDS system and the LiDS graph to address the problem of automated data preparation. Building upon these ideas, this chapter presents the KGFarm system, a comprehensive platform designed specifically for automated data preparation using the semantics of the data science artifacts captured in the LiDS graph. KGFarm further enhances the LiDS graph by augmenting abstract concepts such as entities and feature views, which play a crucial role in the process of feature identification and data enrichment (Feast: Feature Store for Machine Learning, 2022; Kakantousis et al., 2019). This augmentation enables KGFarm to create a more comprehensive and enriched knowledge graph tailor-made for automating data preparation in machine learning tasks. By utilizing this augmented knowledge graph, KGFarm generates labeled data that can be used to train machine learning models capable of recommending task-specific data preparation operations on ad-hoc datasets. One of the key advantages of KGFarm is its seamless integration within conventional data science or machine learning workflows. It assists data scientists in automating the tedious process of data preparation, eliminating the need for time-consuming exploratory data analysis. With KGFarm, data scientists can focus on higher-level tasks and leverage the platform's recommendations for efficient and effective data preparation.

In this chapter, we will delve into the architecture of the KGFarm system, providing a thorough understanding of its different components. This chapter will also explore the underlying methodology behind several data preparation operations supported by KGFarm. By examining each operation in detail, we aim to provide comprehensive insights into how KGFarm automates these critical tasks by exploiting the accumulated knowledge of data scientists in the past. Finally, the chapter concludes by highlighting the core characteristics of the KGFarm system. These characteristics encompass KGFarm's ability to provide taylor-stitched recommendations, ensuring accurate and tailored suggestions for data preparation. Additionally, KGFarm boasts scalability, allowing it to handle large-scale datasets efficiently. It also incorporates leakage-aware data transformation techniques, ensuring data integrity throughout the process. Lastly, we will emphasize how KGFarm seamlessly integrates with existing data science workflows, making it a valuable tool for data scientists seeking to automate the data preparation process.

4.1 KGFarm Overview

The architecture of the KGFarm system is depicted in Figure 4.1. The system consists of four core components that collectively offer comprehensive data preparation capabilities. These components include the (A) KG Augmentor, (B) Training Manager, (C) Inference Manager, and (D) APIs & Interface Library. Each component plays a vital role in enabling the end-to-end automation of the data preparation phase in the data science workflow.

First and foremost, the KG Augmentor component is fundamental to the system as it augments the LiDS graph by introducing abstract entities and feature views that complement the physical columns and tables. This augmentation enhances the graph's knowledge representation, making it a valuable resource for identifying relevant features within the data. By incorporating these abstractions, the augmented graph becomes an invaluable tool for discovering meaningful insights crucial for data preparation. Secondly, The Training Manager component focuses on generating labeled data using the augmented LiDS graph to train task-specific data preparation recommendation models. These models leverage the rich information present in the graph to provide targeted recommendations for data preparation operations on ad-hoc datasets. By learning from the accumulated

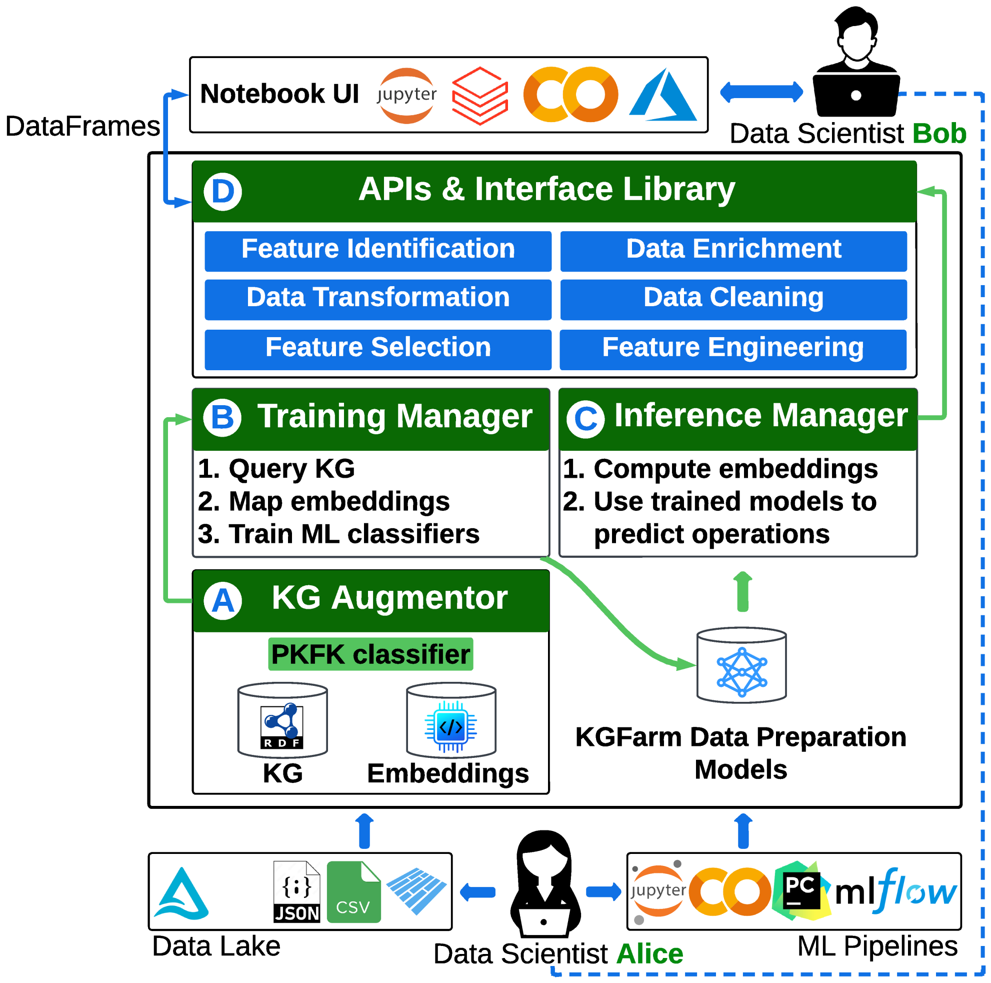


Figure 4.1: An overview of KGFarm's main components

knowledge within the graph, the models can assist data scientists in performing data preparation tasks more effectively and efficiently. Thirdly, the Inference Manager component is responsible for real-time inference, leveraging the trained models to make predictions. It uses the recommendations generated by the models to guide data scientists throughout the data preparation process, offering actionable recommendations. By harnessing the power of the trained models, the Inference Manager empowers data scientists to make informed decisions and streamline their data preparation workflows efficiently. Lastly, to ensure seamless integration with existing data science workflows, KGFarm offers an extensive set of APIs and an Interface Library. These APIs provide data scientists with the means to interact with the system, access its functionalities, and seamlessly incorporate it into their existing tooling and processes. This integration simplifies the adoption of KGFarm within established data science environments, allowing data scientists to leverage its automated data preparation capabilities without disruption. The upcoming sections will delve deeper into each of these components, providing a comprehensive understanding of their roles and contributions to the automated data preparation process.

4.2 KG Augmentation

The KG Augmentor is a crucial component of the KGFarm system that is responsible for augmenting the LiDS knowledge graph. Its primary function is to identify entities within the data and create corresponding feature views that encapsulate the characteristics of these entities across physical columns and tables. Enriching the LiDS graph with additional nodes and edges (as shown in Figure 3.2), expands its knowledge representation. The KG Augmentor adheres to the principles of the LiDS ontology and utilizes the Web Ontology Language (OWL) to preserve interoperability and data sharing. This augmentation process enhances the graph's ability to capture complex relationships and seamlessly integrates new abstracted information into the existing structure.

To identify the physical representation of an entity, the KG Augmentor leverages a PK-FK (primary-key foreign-key) classifier inspired by (Rostin et al., 2009) which can predict column pairs with a primary-key foreign-key relation for schema-less datasets in data lakes (Bogatu, Fernandes, Paton, & Konstantinou, 2020; Khatiwada et al., 2023). By identifying such columns that uniquely represent entities and serve as join keys, the KG Augmentor establishes connections between abstract entities and their corresponding physical artifact i.e. column in the graph.

In addition to enriching the LiDS graph with abstract entities and feature views, the KG Augmentor incorporates data preparation operations from the pipeline graph into the LiDS knowledge graph. This incorporation addresses three key motivations. Firstly, data preparation operations in pipelines can be expressed as classes or methods, making their relationship with columns and tables ambiguous. By establishing a direct link between the operations and the columns or tables to which they are applied, the KG Augmentor accurately captures and represents these operations within the augmented graph. Secondly, not all methods in the pipeline script contribute to data preparation.

Some methods, such as *read_csv*, are used for other purposes such as reading data rather than performing any data preparation operation. By selectively including relevant operations that contribute to data preparation, the KG Augmentor focuses on meaningful data preparation steps within the augmented graph. Thirdly, there is no direct link between data preparation operations and columns or tables in the LiDS graph. The absence of this connection hinders the overall understanding of the data and the relationships between operations and specific data elements. By establishing a relationship between operations and corresponding columns and tables, the KG Augmentor bridges this gap and improves the graph's representation concerning the data preparation workflow.

By augmenting the LiDS graph with abstract entities, feature views, and data preparation operations, the KG Augmentor enhances the system's understanding of the underlying data preparation workflow. This information empowers subsequent components, such as the Training Manager, to leverage these relationships when building task-specific data preparation recommendation models. The approach used by the KG Augmentor to 1) discover PK-FK for identifying the physical representation of an entity and 2) augmenting the LiDS graph are described in the subsection below.

4.2.1 PK-FK Discovery

To estimate the physical representation of an entity, the KG Augmentor incorporates a PK-FK classifier. This classifier aims to identify PK-FK column pairs within the dataset, which is valuable in scenarios where columns have low cardinality or when multiple columns exhibit high uniqueness. The PK-FK classifier is motivated by the observation that a table's primary key or foreign key is the most probable candidate to represent the entity when no column in the table is unique. For example, the augmented instance of the LiDS graph illustrated in figure 3.2 shows that the entity *quake* is represented by the physical column *quake_id*. In cases where there are multiple columns with high uniqueness, the frequency with which a column is referenced as a primary key or a foreign key in other datasets serves as a tiebreaker to identify the column representing the entity physically.

The task of PK-FK classifier discovery is formalized as a binary classification problem. A set of column pairs with high column similarity provided by the CoLR embeddings is compiled by querying the LiDS graph. Column similarity enabled by embeddings is preferred over Inclusion

Table 4.1: Features used in the PK-FK classifier for estimating Primary-key (C_1) and Foreign-key (C_2) relationships.

Feature #	Feature	Relationship used for feature implementation
F1	Content Similarity (C_1, C_2)	data:hasContentSimilarity
F2	Semantic Similarity (C_1, C_2)	data:hasSemanticSimilarity
F3	Range Discrepancy (C_1, C_2)	data:hasMinValue, data:hasMaxValue
F4	Table size ratio (C_1, C_2)	data:hasTotalValueCount
F5	Cardinality (C_1)	data:hasDistinctValueCount
F6	Cardinality (C_2)	data:hasDistinctValueCount
F7	Name Suffix (C_1)	schema:name
F8	Name Suffix (C_2)	schema:name

Dependencies (Rostin et al., 2009) as it demands less computation in comparison to inclusion dependencies which require analyzing the raw column content.

The classifier used in this approach is random forest, which employs 5-fold cross-validation for training and evaluation. The PK-FK Classifier utilizes eight distinct features which are similar to several hand-crafted features proposed by (Rostin et al., 2009). In this work, we modified a few of these features so that they can be easily queried using the LiDS graph and computed on the fly in order to identify PK-FK relationships in an efficient manner. All eight employed features along with the relationship (predicate) used to query information concerning that feature are summarized in table 4.1. Each feature plays a crucial role in assessing the relationship between the candidate's Primary Key (C_1) and Foreign Key (C_2). Here's a breakdown of each feature and its significance in the classification process:

F1: Content Similarity (C_1, C_2): This feature measures the content similarity between columns (C_1) and (C_2) using the <*data:hasContentSimilarity*> relationship in the LiDS graph [1]. The content similarity is computed by calculating the cosine distance between the CoLR embedding representations of the two columns. By evaluating the content similarity, this feature provides insights into the potential relationship and resemblance in the data patterns represented by the columns. Specifically, when identifying PK-FK relationships, capturing the content similarity between candidate columns is essential as it offers valuable information

[1]Prefix data: <*http://kglids.org/ontology/data/*>

about their data content alignment and potential connection. The high content similarity suggests a higher likelihood of a PK-FK relationship between the columns.

F2: **Semantic Similarity** (C_1, C_2): This feature evaluates the semantic similarity between columns (C_1) and (C_2) using the <*data:hasSemanticSimilarity*> relationship in the LiDS graph. The semantic similarity is measured in a similar fashion to content similarity, however, it calculates the cosine distance between the word embedding (Goikoetxea, Agirre, & Soroa, 2016) representation of the two columns. By capturing the semantic resemblance, this feature provides insights into the potential semantic relationship and correspondence in the data patterns represented by the columns. Specifically, in the context of identifying PK-FK relationships, capturing the semantic similarity between candidate columns can be particularly useful, as PK-FK column names often follow similar semantic patterns.

F3: **Range Discrepancy** (C_1, C_2): This feature assesses the range discrepancy between columns (C_1) and (C_2) using the <*data:hasMinValue*> and <*data:hasMaxValue*> relationships in the LiDS graph. It calculates whether the maximum and minimum values of the foreign key column (C_2) fall within the range of the maximum and minimum values of the primary key column (C_1) to produce a binary output of 0 or 1. In PK-FK relationships, it is expected that the dependent values in the foreign key column are distributed more or less evenly over the referenced values in the primary key column. By comparing the range of the foreign key column with the range of the primary key column, this feature provides insights into the distribution patterns and alignment between the referenced and dependent values.

F4: **Table size ratio** (C_1, C_2): This feature calculates the ratio of the number of tuples in the foreign key column (C_2) to the number of tuples in the primary key column (C_1) using the <*data:hasTotalValueCount*> relationship in the LiDS graph. The feature examines the proportion of tuples in (C_2) relative to (C_1) and captures how dependent attributes typically do not reference only a very small subset of their primary keys. In the context of identifying PK-FK relationships, this feature provides insights into the relative sizes of the primary key column and the foreign key column. In a PK-FK relationship, the foreign key column is expected to have a comparable or larger number of tuples compared to the primary key column.

Algorithm 1 Classification of Primary-key Foreign-key pairs

```
Input: LiDS Knowledge Graph $\mathcal{KG}$,
       Primary-key foreign-key classifier $\mathcal{PC}$
Output: Primary-key Foreign-key pairs $\mathcal{P}$
1: Query $\mathcal{KG}$ to retrieve column pairs $\mathcal{S} = \{(C_1, C_2) \mid C_1, C_2 \in \mathcal{KG} \wedge sim(C_1, C_2) \geq \theta\}$ where $sim(C_1, C_2)$
   is the content similarity measure between columns $C_1$ and $C_2$, and $\theta$ is the content similarity threshold.
2: Calculate Primary-key Foreign-key feature set $\mathcal{F} = \{(f_{1_{C_1C_2}}, f_{2_{C_1C_2}}, ..., f_{8_{C_1C_2}}) \mid C_1, C_2 \in \mathcal{S}\}$ where
   $f_{1_{C_1C_2}}, f_{2_{C_1C_2}}, ..., f_{8_{C_1C_2}}$ are the features for candidate columns $C_1$ and $C_2$.
3: Classify Primary-key Foreign-key column pairs $\mathcal{P}$ by applying $\mathcal{PC}$ on $\mathcal{F}$ such that $\mathcal{P} \subset \mathcal{S}$.
```

F5: **Cardinality** (C_1): This feature captures the cardinality of the primary key column (C_1) using the <*data:hasDistinctValueCount*> relationship in the LiDS graph. It counts the number of distinct values present in (C_1). In the context of identifying PK-FK relationships, the cardinality of the primary key column is an important consideration. Higher cardinality indicates a larger number of unique values in (C_1), which suggests a stronger likelihood of it being a primary key. PK-FK relationships typically involve a primary key column with higher cardinality, as it serves as a reference for multiple foreign key values.

F6: **Cardinality** (C_2): This feature captures the cardinality of the foreign key column (C_2) similarly to F5 by using the <*data:hasDistinctValueCount*> relationship in the LiDS graph. It counts the number of distinct values present in (C_2). In the context of identifying PK-FK relationships, foreign key columns typically exhibit some level of diversity in their values, as they reference primary key values from other tables.

F7: **Name Suffix** (C_1): This feature examines the suffix of the column name in the candidate primary key column (C_1). It returns 1 if the column name ends with specific sub-strings that are commonly associated with join keys, such as *'id'*, *'num'*, *'key'*, *'ref'*, or *'code'* else 0. The presence of these suffixes can suggest in favor of the column representing a primary key.

F8: **Name Suffix** (C_2): This feature examines the suffix of the column name in the candidate foreign key column (C_2). Similarly to Feature 7, it returns 1 if the column name ends with specific sub-strings else 0.

Algorithm 1 outlines the process for classifying PK-FK column pairs using the PK-FK classifier

Algorithm 2 Abstracting Entities and Feature Views in LiDS Knowledge Graph

```
Input: LiDS Knowledge Graph $\mathcal{KG}$,
       Primary-key foreign-key pairs $\mathcal{P}$
Output: Augmented LiDS Graph $\mathcal{KG}'$
 1: for table $\in \mathcal{KG}$ do
 2:     Create a feature view node                                  ▷ Abstract feature view
 3:     Link the feature view node to its physical table: add_edge (table.uri, farm:hasFeatureView, featureView.uri)
 4:     for column $\in$ table do
 5:         if column $\in \mathcal{P}$ then
 6:             Create an entity node                                ▷ Abstract entity
 7:             Link the entity node to its physical column: add_edge (entity.uri, farm:representedBy, column.uri)
 8:             Link the entity node to its feature view: add_edge (featureView.uri, farm:hasEntity, entity.uri, column.uniqueness)
 9:         end if
10:     end for
11: end for
```

and the computed features. Given a LiDS Knowledge Graph ($\mathcal{KG}$) and the Primary-key foreign-key classifier ($\mathcal{PC}$), the algorithm proceeds as follows. First, the $\mathcal{KG}$ is queried to retrieve column pairs (C_1, C_2) where the content similarity measure $sim(C_1, C_2)$ between the columns exceeds a specified threshold θ. These column pairs form the set $\mathcal{S}$. Next, the feature set $\mathcal{F}$ is computed for each candidate column pair (C_1, C_2) in $\mathcal{S}$, consisting of the eight features $f_{1_{C_1C_2}}, f_{2_{C_1C_2}}, ..., f_{8_{C_1C_2}}$. Finally, the PK-FK classifier ($\mathcal{PC}$) is applied to classify the primary-key foreign-key column pairs $\mathcal{P}$ using the feature set $\mathcal{F}$, resulting in $\mathcal{P}$ being a subset of $\mathcal{S}$. By utilizing this classifier and computing these features dynamically, PK-FK column pairs can be efficiently classified, aiding in the identification of physical representations of entities.

4.2.2 LiDS Graph Augmentation

LiDS graph augmentation aims to expand its knowledge representation and improve its ability to capture complex relationships concerning data preparation. By enriching the graph with 1) abstract entities, feature views, and 2) labeled data preparation operations, the LiDS graph Augmentation facilitates a more comprehensive understanding of the underlying data preparation workflow and enables the generation of precise and context-aware recommendations for data scientists.

One of the key steps in the LiDS Graph augmentation is the abstraction of entities and feature views, as outlined in Algorithm 2. This algorithm leverages the insights provided by the PK-FK classifier (explained in Algorithm 1) to estimate the physical representation of entities. Algorithm 2 starts by iterating through each table in the LiDS knowledge graph. For each table, a feature view node is created, representing an abstract view of the features associated with the table. This feature view node is then linked to its corresponding physical table, establishing the relationship between the abstract and physical representations. Next, the algorithm iterates through each column in the table and checks if the column is part of the identified primary-key foreign-key pairs ($\mathcal{P}$). If it is, an abstract entity node is created to represent the column. This entity node is then linked to its corresponding physical column, indicating the relationship between the abstract and physical representations. Additionally, the entity node is linked to the feature view, capturing the association between the entity and the abstract view of features.

One of the key highlights of the abstraction approach, which leverages the PK-FK classification, is its ability to capture join keys. Unlike relying solely on factors like column uniqueness, the approach considers the relationships between columns in PK-FK pairs. For instance, in a *rides* dataset with columns such as *'ride_amount'*, *'ride_duration'*, and *'driver_id'*, the approach correctly identifies the *driver* as an entity, even when the same driver has multiple trips. Conversely, columns like *'ride_duration'* or *'ride_amount'*, despite potentially having high uniqueness, are not considered as they do not serve as join keys. Another advantage of this approach is its ability to handle datasets with multiple entities. By extracting both primary and foreign keys, the approach captures multiple entities within a dataset. In the *rides* dataset with columns such as *'driver_id'* and *'passenger_id'*, both the *passenger* and *driver* are captured as foreign keys, with *'passenger_id'* and *'driver_id'* serving as their respective physical representations. Furthermore, the abstraction approach effectively handles situations where there may be multiple potential physical representations for an entity. In a *students* dataset with columns like *'student_id'*, *'student_social_insurance_number'*, and *'student_email'*, multiple candidates may arise for representing the entity *student*. The approach determines the most appropriate representation by considering the frequency of appearance in the list of identified PK-FK pairs ($\mathcal{P}$). Typically, the column that appears more frequently, such as *'student_id'*, is selected as the physical representation of the entity over other potential representations.

Labeling data preparation operations is a critical aspect of the LiDS graph augmentation process. As LiDS graph utilizes static code analysis to capture various classes, packages, and functions

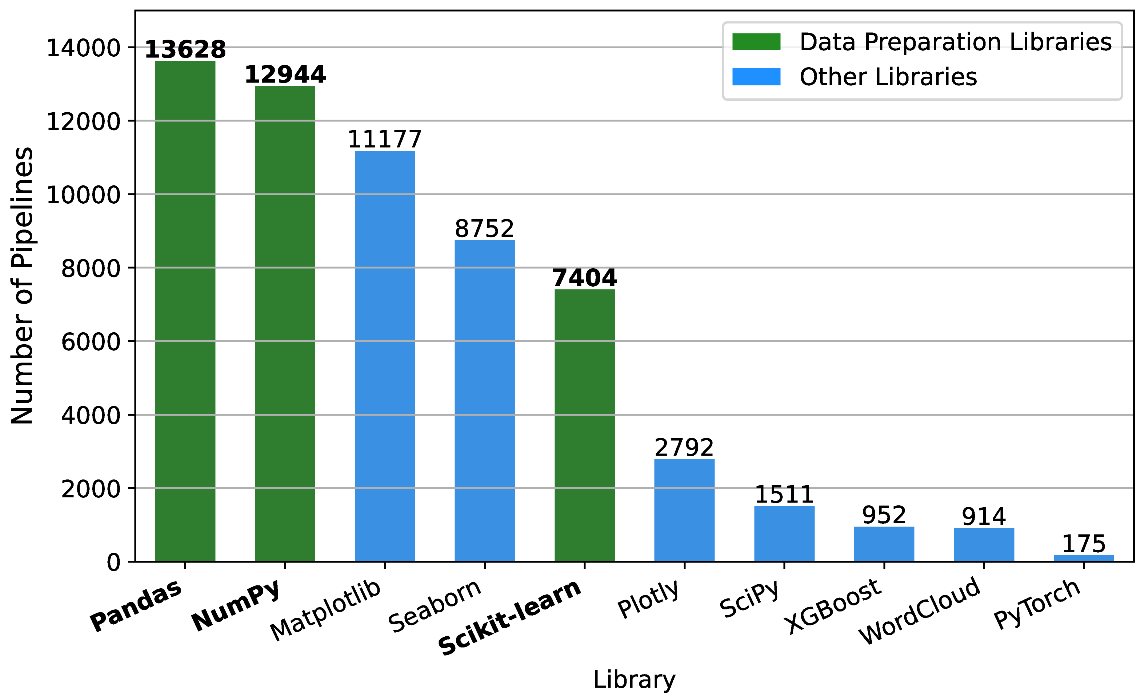


Figure 4.2: Top-k Libraries ($k = 10$) used in $\sim$ 13.8k Kaggle Pipelines, emphasizing the ones that contribute towards Data Preparation.

within pipeline scripts, it becomes essential to identify the specific set of classes and functions that contribute to data preparation. To accomplish this, we queried the LiDS to determine the top 10 most-used libraries in approximately 13.8k pipelines. Figure 4.2 showcases these libraries, with a specific focus on the ones that are dedicated to data preparation tasks. Notably, libraries such as *Pandas*, *NumPy*, and *Scikit-learn* emerge as the most prominent ones for data preparation. While other libraries like *Matplotlib*, *Seaborn*, and *Plotly* are primarily employed for data visualization, which aids exploratory data analysis (EDA) and insights gathering during data preparation (Peng et al., 2021). It is worth mentioning that these visualization libraries do not directly transform the data but instead enable data scientists to inspect the data through visualizations. Therefore, by focusing on *Pandas*, *NumPy*, and *Scikit-learn* as the key libraries for data preparation, we can selectively extract a specific set of operations provided by each of these libraries. These operations are typically available in the form of functions or classes as shown in figure 3.2. For example, *Pandas* provides various cleaning operations through functions such as *interpolate*, *fillna*, etc. *Scikit-learn* offers transformation operations through classes like *StandardScaler* and *OneHotEncoder*, etc. Additionally, *NumPy* provides useful functions like *Sqrt*, *log*, etc. Using the LiDS ontology, we define data preparation operations $\mathcal{DP}$, which encompass these classes and functions contributing to data preparation. By querying the LiDS graph, we can identify the operation nodes within $\mathcal{DP}$ and by using predicates like *pipeline:callsClass* and *pipeline:callsFunction* [2], as well as the affected columns (features) using the *pipeline:readsColumn* predicate. These nodes are then linked using the *farm:isAppliedTo* [3] relationship, as depicted in Figure 3.2 of the augmented LiDS graph and labeled by a node representing the data preparation operation type like *data transformation* or *data cleaning*.

The augmented LiDS graph (Figure 3.2) incorporates four additional classes: *entity*, *feature view*, *data transformation*, and *data cleaning*, along with four object properties that play vital roles in the graph's structure and semantics:

- The relationship between a physical table and an abstract feature view is denoted by the predicate *http://kgfarm.com/ontology/hasFeatureView*.
- The relationship between an abstract entity and a physical column is represented by the predicate *http://kgfarm.com/ontology/representedBy*.
- The relationship between an abstract feature view and an abstract entity is captured by the predicate *http://kgfarm.com/ontology/hasEntity*.
- The relationship between a pipeline class/function and a physical column is expressed through the predicate *http://kgfarm.com/ontology/isAppliedTo*.

The augmented LiDS knowledge graph serves as a valuable resource for discovering features and enriching the profiled datasets by estimating the primary-key foreign-key (PK-FK) column pairs. Furthermore, it labels operations such as data cleaning and transformation within the pipelines that contribute to the data preparation phase. This graph can be efficiently utilized by employing SPARQL queries on profiled datasets, enabling users to retrieve valuable insights and perform data-driven analyses. Additionally, the augmented LiDS graph is leveraged by the Training Manager

[2]Prefix pipeline: <*http://kglids.org/ontology/pipeline/*>
[3]Prefix farm: <*http://kgfarm.com/ontology/*>

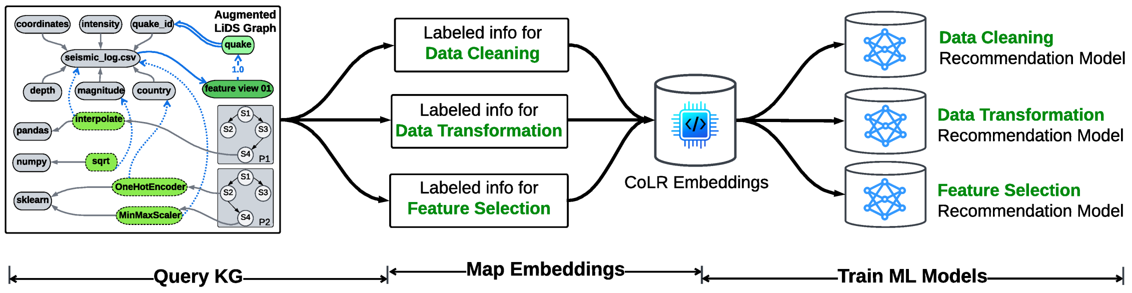


Figure 4.3: KGFarm's Training Manager which 1) queries the Augmented LiDS Knowledge Graph 2) Maps column embeddings and 3) Trains task-specific ML models for automating Data Preparation.

(Section 4.3) to train machine learning models and generate recommendations for data preparation operations, including data cleaning, data transformation, and feature selection, for ad-hoc datasets.

4.3 Training Manager & Data Preparation Models

The Training Manager is a critical component within the KGFarm framework, playing a pivotal role in the automation and streamlining of the data preparation process in data science workflows. As the second component in the architecture (as depicted in Figure 4.1), the Training Manager leverages the power of machine learning and harnesses the extensive knowledge encapsulated within the LiDS graph to develop sophisticated recommendation models. These models are designed to provide accurate and context-aware recommendations for various data preparation tasks, leveraging the insights gained from data scientists who have previously worked with similar datasets and encountered similar data preparation challenges.

Figure 4.3 illustrates the three fundamental steps involved in the Training Manager's workflow. One of the primary objectives of the Training Manager is to train models that are specifically tailored for three essential data preparation operations: data cleaning, data transformation, and feature selection. These operations are critical for ensuring data quality, transforming data into suitable representations, and selecting informative features, respectively. By providing intelligent recommendations for these tasks, the Training Manager empowers data scientists to make informed decisions and efficiently navigate the complexities of the data preparation process, ultimately leading to improved or comparable accuracy in machine learning tasks in less time.

To derive valuable insights and knowledge, the Training Manager performs queries on the LiDS graph, extracting relevant details about specific types of data preparation operations. For example, when training data transformation recommendation models, the Training Manager searches for frequently used transformation techniques within the augmented LiDS graph. This approach enables the identification of popular operations, such as the *OneHotEncoder* for categorical feature transformation or the *Sqrt* function for numerical feature transformation. By exploring historical patterns and trends, the Training Manager gains a deep understanding of how these operations have been applied to features in various datasets.

Once the relevant information is gathered, the Training Manager proceeds to map the extracted features, along with their corresponding CoLR embeddings, to numerical representations. This mapping process ensures that the features can be effectively processed and analyzed by machine learning algorithms. By transforming the features into a numerical format, the Training Manager facilitates the subsequent stages of recommendation model development, enabling them to leverage the extracted insights and patterns to provide accurate and meaningful recommendations. To train the recommendation models, the Training Manager employs various machine learning techniques, including deep neural network classifiers. These models are trained using the processed information, treating the task of recommending data preparation operations as a multiclass classification problem. By leveraging the historical patterns and correlations captured in the labeled datasets, the Training Manager enables the models to learn and generalize from the training data, empowering them to provide task-specific recommendations for data cleaning, data transformation, and feature selection.

A notable advantage of the recommendation models generated by the Training Manager is their ability to handle unseen datasets. This adaptability is achieved by drawing upon the collective knowledge and expertise of data scientists who have previously worked with similar datasets. By leveraging the experiences and insights gained from past datasets, the recommendation models can generalize and provide accurate recommendations for new and diverse datasets. This capability is particularly valuable in dynamic data science environments, where data scientists encounter a wide range of datasets with varying characteristics.

The trained recommendation models developed by the Training Manager are utilized by other components within the KGFarm system, namely the Inference Manager and the API and Interface

Library. These components leverage the models to provide real-time, task-specific recommendations for data preparation operations on the fly. By seamlessly integrating the recommendation models into the data science workflow, the Training Manager enhances the efficiency and effectiveness of the entire data preparation process. This automation, efficiency, and accuracy in the data preparation phase empower data scientists to make informed decisions and achieve high-quality data preparation outcomes.

In the field of data science, encountering unseen or ad-hoc datasets is a common occurrence. These datasets have not been previously profiled or analyzed within the LiDS graph, making it hard to directly retrieve information about the data preparation operations performed on them. To address this challenge, KGFarm introduces a solution for recommending data preparation operations, such as data cleaning, data transformations, feature selection, and feature engineering, for these unprofiled datasets using machine learning techniques.

The approach taken by KGFarm is rooted in the principle that similar data often requires similar data preparation operations. By leveraging the vast repository of data science artifacts, including datasets and pipelines, KGFarm's Training Manager builds task-specific machine learning models that efficiently recommend the most suitable data preparation operations. These operations encompass a range of tasks, from cleaning the data and transforming it into suitable representations to selecting informative features. The recommendation models formalize these tasks as a classification problem, allowing for accurate and context-aware recommendations.

In the following subsections, we delve into the detailed methodology employed by KGFarm's Training Manager to generate these task-specific recommendation models. By understanding the underlying approach and techniques, we can gain insights into how KGFarm effectively addresses the challenge of recommending data preparation operations for unseen or ad-hoc datasets, empowering data scientists and ML practitioners to make informed decisions in their data science workflows.

4.3.1 Data Cleaning

Data cleaning is a critical step in ensuring the quality and reliability of datasets used for machine learning tasks (Xu et al., 2015). In KGFarm, the task of data cleaning recommendation is formalized as a multiclass classification problem. The goal is to predict the most suitable cleaning operation

Table 4.2: Supported Data Cleaning Techniques for Handling Missing Values in KGFarm

Operation	Library	Application
Drop	Pandas	Remove rows and columns with missing values.
Fillna	Pandas	Fill missing values with statistics such as mean, median, or mode.
Imputation	Scikit-learn	Replacing missing values using machine learning.
Interpolation	Pandas	Estimating missing values based on existing values within a range.

for a given dataset with missing values, in order to improve the accuracy of the associated machine learning task. Data cleaning in KGFarm is specifically concentrated towards handling missing values and outliers, as these are common challenges that can adversely impact the performance and reliability of machine learning models. By leveraging the knowledge and insights captured in the LiDS graph, KGFarm equips data scientists with robust recommendation models for effectively addressing missing values and identifying outliers in their datasets.

Problem Definition

Given a dataset $\mathcal{D}$ with missing values $\mathcal{M} = \{m_1, m_2, ..., m_n\}$ and a machine learning task $\mathcal{L}$, the recommendation task in KGFarm is to predict the most suitable cleaning operation $c_r \in \mathcal{C}$ that can handle $\mathcal{M}$, resulting in improved accuracy for $\mathcal{L}$ based on the cleaning techniques applied by data scientists on other datasets similar to $\mathcal{D}$. Additionally, KGFarm incorporates outlier detection as an integral part of its data cleaning process. To handle the missing values $\mathcal{M}$, KGFarm considers a range of commonly used cleaning techniques $\mathcal{C}$ provided in *Pandas* and *NumPy*, which are summarized in Table 4.2 including operations such as *Drop*, *Fillna*, *Imputation*, and *Interpolation*.

Training

The training process for data cleaning recommendation is performed using the Training Manager component of KGFarm. It begins by querying the LiDS graph to retrieve datasets that contain missing values and the corresponding applied data cleaning operations. CoLR embeddings are generated and mapped to represent the independent variables, while the cleaning operation is treated as the target variable. To train the recommendation model, a deep neural network is employed to classify the most suitable data cleaning operation among *Drop*, *Fillna*, *Imputation*, and *Interpolation* techniques, enabling effective handling of missing values. Additionally, an independent outlier detection model is trained using a similar approach. The LiDS graph is queried to identify datasets on which outlier detection has been performed. By incorporating these datasets along with those where outlier detection is not applied, a balanced classification model is created. The average CoLR embeddings for all datasets in the training set are computed, and a random forest classifier is trained to determine whether there is a need for outlier detection in a given dataset. This training process equips KGFarm with the capability to detect the presence of outliers, further enhancing the data cleaning functionality of the system.

Inference

The Inference Manager component of KGFarm utilizes the trained models during the inference phase to provide data cleaning recommendations. The first step in the inference process is to check for the presence of outliers in the dataset. If outliers are detected, the local outlier factor technique (Alghushairy, Alsini, Soule, & Ma, 2021) is applied to convert the outlier values to missing values (represented as NaN). This ensures that the subsequent data cleaning operations are performed on a dataset that is free from outlier influences. Once the dataset is prepared by handling outliers, the pre-trained data cleaning model comes into play. For a given dataset $\mathcal{D}$ with missing values $\mathcal{M} = \{m_1, m_2, ..., m_n\}$, the input to the data cleaning model is the averaged embeddings of the features that contain missing values $\mathcal{M}$. The model then predicts the recommended data cleaning operation $c_r \in \mathcal{C}$. This prediction can be expressed using the following logic:

$$\text{cleaning operation for } \mathcal{D} : \begin{cases} c_r, & \text{if } p_r > \alpha_1 \\ \text{none}, & \text{otherwise} \end{cases} \tag{1}$$

In Equation 1, p_r represents the probability score assigned by the data cleaning model to recommend the cleaning operation c_r, and α_1 is the confidence threshold, which is determined empirically. The threshold ensures that only recommendations with high confidence scores are applied to the dataset.

Table 4.3: Supported Data Transformations Techniques in KGFarm

Transformation type	Feature type	Transformation	Library	Application
Scaling	Numerical	StandardScaler	Scikit-learn	Removes mean and scales to unit variance.
		MinMaxScaler	Scikit-learn	Normalizes feature values.
		RobustScaler	Scikit-learn	Scales respecting outliers.
		MaxAbsScaler	Scikit-learn	Scales by maximum absolute value.
Unary	Numerical	Log	NumPy	Applies natural logarithm.
		Sqrt	NumPy	Computes square root.
		Square	NumPy	Computes square.
		Tanh	NumPy	Applies hyperbolic tangent.
		OrdinalEncoder	Scikit-learn	Encodes ordinal feature.
		OneHotEncoder	Scikit-learn	Encodes nominal features.
	Categorical	OrdinalEncoder	Scikit-learn	Encodes ordinal feature.
		OneHotEncoder	Scikit-learn	Encodes nominal features.

Furthermore, it is important to consider the type of feature, whether categorical or numerical, when applying the recommended cleaning operation. For example, if the recommended operation is *Fillna* and the feature is categorical, it is filled with the mode value. On the other hand, if the feature is numerical, it is filled with the mean value. Similar considerations are made for other cleaning methods based on the nature of the feature.

4.3.2 Data Transformation

Data transformation plays a crucial role in preparing datasets for machine learning tasks (V. G. Raju, Lakshmi, Jain, Kalidindi, & Padma, 2020). In KGFarm, the task of data transformation recommendation is formalized as a classification problem. The goal is to predict a set of transformations that can enhance the accuracy of the machine learning task based on the transformation techniques applied by data scientists on similar datasets and features in the past.

Problem Statement

Given a dataset $\mathcal{D}$ with features $\mathcal{F} = \{f_1, f_2, ..., f_n\}$ and a machine learning task $\mathcal{L}$, the recommendation task is to predict a set of transformations $\mathcal{T} = \{t_1, t_2, ..., t_m\}$ that can improve the accuracy of $\mathcal{L}$ based on the transformation techniques applied by data scientists on other datasets and features similar to $\mathcal{D}$ and $\mathcal{F}$, respectively. The problem of data transformation recommendation in KGFarm is subdivided into two primary steps: 1) recommending scaling transformations applied to the entire dataset and recommending 2) unary feature transformations applied to individual features. To transform the features $\mathcal{F}$, KGFarm considers a wide range of frequently used scaling and unary transformation techniques $\mathcal{T}$ that are provided in the *Scikit-learn* and *NumPy* libraries, which are summarised in Table 4.3. These transformations include several scaling techniques such as *StandardScaler*, *MinMaxScaler*, etc., and unary transformations such as *Log*, *Sqrt*, *OneHotEncoder*, etc.

Unary transformations are valuable for manipulating data in machine learning (Nargesian et al., 2017). However, to maximize their effectiveness, applying scaling transformation beforehand can improve the performance of unary transformations, leading to an overall improved model effectiveness. The primary motivation behind this lies in addressing the challenges posed by varying data magnitudes. For instance, algorithms like KNN and SVM rely on measuring the distance between data points. Without proper scaling, features with larger values tend to dominate the feature similarity measurement, leading to suboptimal or inadequate model performance (V. N. G. Raju, Lakshmi, Jain, Kalidindi, & Padma, 2020). By scaling the data, we ensure that all features are treated on an equal footing which further prevents certain features from overshadowing others due to their inherent scale, thus facilitating a balanced and unbiased unary transformation. Another compelling reason for utilizing scaling transformation is its efficacy in handling outliers. Outliers, characterized by extreme values, can significantly impact unary transformations by causing distortion in the transformation process and hindering accurate modeling. By scaling the data, we effectively reduce the influence of outliers, enabling the unary transformations to exhibit greater robustness and resilience. Thus incorporating scaling transformations prior to unary transformations offers a range of benefits that positively impact the performance.

Training

The training process for data transformation recommendation is facilitated by the Training Manager, which orchestrates the necessary steps to train the models. It begins by querying the LiDS graph to retrieve the features and the applied data transformation operations. This process is repeated three times, once for scaling transformations and twice for unary feature transformations (numerical and categorical). After querying the graph, the Training Manager maps the retrieved features and transformations to their respective CoLR embeddings. This mapping enables the models to capture the inherent relationships and patterns between the features and transformations by serving as the independent variables and the transformation technique serving as the target variable.

For scaling transformations, the CoLR embeddings of numerical features are averaged to generate a holistic representation of the dataset. In the case of unary transformations, the CoLR embeddings of individual transformed features are retained, preserving their distinct characteristics. Once the features and transformations are mapped to their embeddings, the training process commences. For scaling transformations, a deep neural network model is trained using a multiclass classification approach. The model is trained on four classes: *StandardScaler*, *MinMaxScaler*, *RobustScaler*, and *MaxAbsScaler*, representing the most commonly used scaling transformation techniques.

Similarly, for unary feature transformations, two separate deep neural network models are trained. The numerical unary transformation model is trained on six classes: *Log*, *Sqrt*, *Square*, *Tanh*, *OrdinalEncoder*, and *OneHotEncoder*, covering a wide range of numerical unary transformation techniques. The categorical unary transformation model, on the other hand, is trained on two classes: *OrdinalEncoder* and *OneHotEncoder*, encompassing the commonly used categorical unary transformation methods.

Inference

During the inference phase, the trained models are utilized by the Inference Manager to provide data transformation recommendations. The inference process begins by checking for the presence of categorical columns in the dataset. If categorical columns exist, the categorical unary transformation model is first employed to recommend the appropriate transformation techniques. Next, scaling transformations are applied to the dataset, followed by unary transformations if recommended.

For a given dataset $\mathcal{D}$ with features $\mathcal{F} = \{f_1, f_2, ..., f_n\}$, the scaling transformation model predicts the recommended scaling technique $s_r \in \mathcal{T}$. This can be expressed as follows:

$$\text{scaling transformation for } \mathcal{D} : \begin{cases} s_r, & \text{if } p_r > \alpha_2 \\ \textit{none}, & \text{otherwise} \end{cases} \tag{2}$$

Similarly, for numerical and categorical unary transformations, the input is the embedding of the individual numerical and categorical features, respectively. The output is the recommended unary transformation technique $u_r \in \mathcal{T}$. This can be expressed as follows:

$$\text{Unary transformation for } f \in \mathcal{F} : \begin{cases} u_r, & \text{if } p_r > \alpha_3 \\ \textit{none}, & \text{otherwise} \end{cases} \tag{3}$$

In equations 2 and 3, p_r represents the confidence score of the classifier to recommend the scaling transformation s_r and the unary transformation u_r, respectively. The thresholds for confidence, denoted as α_2 and α_3, are determined empirically for each of the three data transformation recommendation models.

4.3.3 Feature Selection

Feature selection plays a crucial role in machine learning by identifying the most relevant and predictive features for a given task (Kumar & Minz, 2014). In KGFarm, the task of feature selection is formulated as a binary classification problem, where the goal is to predict a subset of features from a dataset that have high relevance or predictive power in predicting a target class for a specific machine learning task. This selection process leverages the similarity to previously chosen features that have demonstrated effectiveness in predicting similar target classes.

Problem Statement

Given a dataset $\mathcal{D}$ with features $\mathcal{F} = \{f_1, f_2, ..., f_n\}$, a target class k, and a machine learning task $\mathcal{D}$, the problem of feature selection in KGFarm is to predict a subset of features $\mathcal{F}'$ that have high relevance in predicting target class k for task $\mathcal{L}$. The selection of features in $\mathcal{F}'$ is based on the similarity to previously selected features that have shown effectiveness in predicting similar targets.

The LiDS graph provides information about the selected and discarded features associated with the target variable. Techniques such as *train_test_split* and *SelectKBest* are employed to mine the target, selected, and discarded variables from the dataset. The LiDS ontology offers predicates such as *pipeline:hasTarget*, *pipeline:hasSelectedFeature*, and *pipeline:hasNotSelectedFeature* to efficiently query these relationships directly from the graph.

Training

The training process for feature selection is facilitated by the Training Manager, which orchestrates the necessary steps to train the models. Similar to the approach followed in data cleaning and data transformation, the LiDS graph is queried to retrieve the features $\mathcal{F}'$ that were selected for a given target class k in each pipeline. The selected features are labeled as *selected*, while the discarded features $\mathcal{F} - \mathcal{F}'$ are labeled as *discarded*. CoLR embeddings are then computed for the selected and discarded features along with their target variables. This results in labeled modeling data, where the feature embeddings of the selected or discarded features represent the independent variables, and the target variable is represented by the binary classes *selected* or *discarded*. These steps are repeated for three different machine learning tasks: 1) binary classification, 2) multiclass classification, and 3) regression, to accommodate the nature of the target variable. Three independent deep neural network classifiers are trained per machine learning task to recommend the subset of features that should be selected for predicting the target class k.

Inference

During the inference phase, the trained models are utilized by the Inference Manager to recommend the subset of features to be selected. Given a dataset $\mathcal{D}$ with features $\mathcal{F} = f_1, f_2, ..., f_n$, a target class k, and a machine learning task $\mathcal{L}$, the input to these models is the feature embeddings of the features in $\mathcal{F}$ and the target class k. The output is the probability p_r indicating the predicting

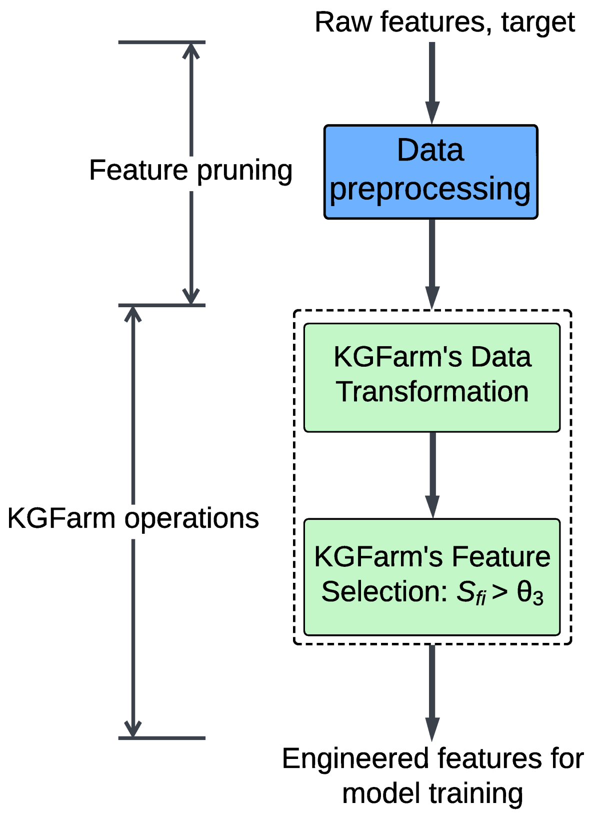

Raw features, target
Data preprocessing
Feature pruning
KGFarm's Data Transformation
KGFarm operations
KGFarm's Feature Selection: $S_{fi} > \theta_3$
Engineered features for model training

Figure 4.4: Feature Engineering Pipeline in KGFarm

power or relevance for predicting the class k. This can be expressed as follows:

$$\text{Selection of feature } f \text{ for predicting target class } k : \begin{cases} \textit{selected}, & \text{if } p_r > \alpha_4 \\ \textit{discarded}, & \text{otherwise} \end{cases} \tag{4}$$

In equation 4, p_r represents the probability with which feature $f \in \mathcal{F}$ should be selected for predicting the target class k, and α_4 is the threshold for the feature selection relevance score, which is determined empirically for each of the three models per machine learning task.

4.3.4 Feature Engineering

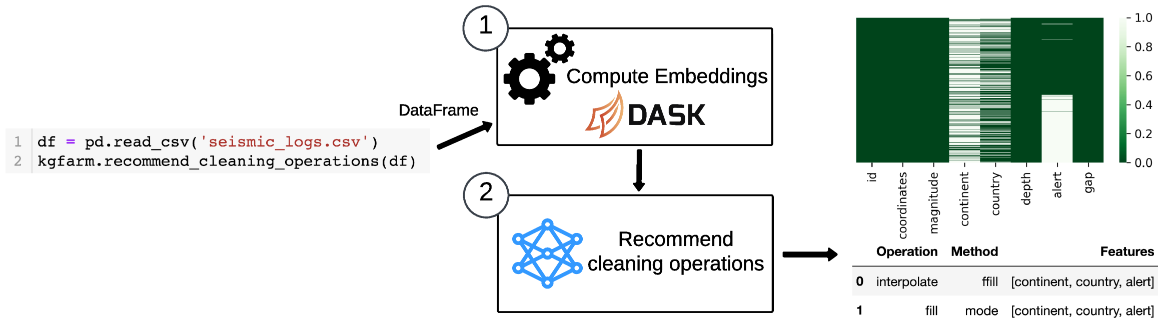
Feature engineering is a crucial and essential step in machine learning, involving the application of best practices to engineer an optimal set of features for solving specific tasks (Kaul et al., 2017; Nargesian et al., 2017). In KGFarm, feature engineering is implemented as a streamlined pipeline illustrated by figure 4.4 that integrates statistical data preprocessing and sophisticated predefined data preparation operations such as data transformation and feature selection. This comprehensive approach aims to maximize the accuracy and efficiency of machine learning tasks, particularly when dealing with diverse and ad-hoc datasets. By combining these operations, KGFarm empowers data scientists to leverage the full potential of feature engineering in an efficient way.

Data Preprocessing

In the data preprocessing step, various lightweight statistical techniques are employed to handle the challenge of evaluating all candidate features in the original feature space, especially in the case of high-dimensional datasets. KGFarm leverages efficient statistical methods, such as information gain and feature correlation, to prune the set of candidate features and improve the computational efficiency and scalability of the overall feature engineering pipeline (Kaul et al., 2017). The information gain technique is used to estimate the relevance and influence of each candidate feature in predicting the target class. For datasets with a small number of features (less than 500), all features with an information gain value greater than 0 are selected. However, for datasets with a larger number of features (500 or more), only the top 10% of features, ranked by their information gain, are selected. This approach allows for a more focused analysis of the most informative features while effectively managing the computational complexity. Additionally, Pearson correlation is employed to address the issue of multicollinearity among features. When highly correlated features are present, which can lead to overfitting, KGFarm eliminates the feature with the lower information gain to mitigate the negative effects of redundancy and improve the overall feature distribution within the feature set.

Data Transformation and Feature Selection

After the data preprocessing step, the pruned set of features is passed to KGFarm for data transformation and feature selection. The data transformation phase involves applying appropriate techniques, as described in subsection 4.3.2, to enhance the quality and representation of the features. This step enables the features to capture more meaningful patterns and relationships in the data, improving the overall predictive performance. Following the data transformation, KGFarm applies



```
df = pd.read_csv('seismic_logs.csv')
kgfarm.recommend_cleaning_operations(df)
```

	Operation	Method	Features
0	interpolate	ffill	[continent, country, alert]
1	fill	mode	[continent, country, alert]

Figure 4.5: KGFarm's Inference Manager that exploits 1) Fixed-sized column embeddings, 2) Dask, and 3) Pre-trained data preparation models to recommend data preparation operations in real-time.

its feature selection operation, as explained in subsection 4.3.3, to identify the most influential and relevant transformed features for predicting the target variable. By selecting a subset of features that have demonstrated high predictive power in similar tasks, KGFarm aims to further optimize the feature set and enhance the performance of machine learning models which might be missed by the statistical data preprocessing phase.

The proposed feature engineering pipeline in KGFarm is particularly well-suited for binary and multiclass classification tasks. However, for regression problems with continuous target variables, the pipeline is modified to incorporate a more suitable statistical filter method, such as the Anova-test, which is able to handle continuous targets. This modification ensures that the feature engineering process is tailored to the nature of the target variable, enabling more accurate predictions. By integrating data preprocessing, data transformation, and feature selection, the feature engineering pipeline in KGFarm optimizes the feature engineering process and enhances the predictive performance of machine learning models. It automates key steps in feature engineering, enabling data scientists to focus on higher-level tasks and empowering them to derive the most informative and relevant features for their machine learning tasks.

4.4 APIs & Inference Manager

The Interface Manager, as the third component within the KGFarm system, plays a critical role in delivering real-time data preparation functionalities with accuracy and efficiency. Its primary objective is to seamlessly integrate data scientists with the recommendation models generated by the

Training Manager, enabling them to efficiently perform data preparation tasks. Figure 4.5 provides an illustration of the Interface Manager in action.

When an unseen dataset is provided as a DataFrame, the Interface Manager initiates the data preparation process by calculating the Column Learned Representation (CoLR) embedding for each column in the dataset. These embeddings capture essential information about the columns, facilitating subsequent data preparation decisions. CoLR embeddings represent a compact and informative representation of the columns, enabling efficient storage, processing, and analysis. The next crucial step for the Interface Manager is to leverage the task-specific pre-trained models generated by the Training Manager. These models encapsulate the knowledge and insights acquired during the training process, enabling them to provide accurate and context-aware recommendations for data cleaning, data transformation, and feature selection tasks. By utilizing the CoLR embeddings of the columns, the Interface Manager employs these pre-trained models to predict the most relevant operation to be applied to the unseen dataset. This recommendation process empowers data scientists to efficiently perform data preparation tasks on ad-hoc or previously unseen datasets without the need for manual exploration or analysis (EDA), saving valuable time and effort.

Scalability and efficiency are key strengths of the Interface Manager. To achieve scalability, the Interface Manager leverages Dask (Dask Development Team, 2016), a parallel computing framework that efficiently handles datasets with a significant number of features. By distributing the computation across multiple cores or machines, the Interface Manager can process large-scale datasets in a highly efficient manner. This parallelization ensures quick and responsive recommendations, enabling the system to handle the computational demands of data-intensive tasks. Additionally, the use of CoLR embeddings further contributes to scalability by providing a concise and informative representation of the columns. The fixed-size embeddings compress the essential characteristics of the columns, enabling efficient storage and processing even for datasets with a large number of rows. The Interface Manager operates through a two-step process. Firstly, it computes the CoLR embeddings of the ad-hoc dataset using Dask to parallelize the computation and efficiently handle datasets with a high number of features. Once the embeddings are obtained, the Interface Manager applies the recommendation models developed by the Training Manager to provide precise and task-specific data preparation recommendations. These recommendations are delivered in real-time, allowing data scientists to streamline their workflow and make informed decisions regarding data cleaning, data transformation, and feature selection.

The interface Manager seamlessly couples the output of recommended data preparation operations with the API and Interface Library, facilitating intuitive and user-friendly interactions between data scientists and the recommendation models. Serving as a vital bridge between data scientists and the recommendation models generated by the Training Manager, the API and Interface Library provide a convenient and efficient means for data scientists to apply the recommended data preparation operations and obtain the desired outcomes. This integration streamlines the data scientists' workflow and empowers them to make informed decisions, achieving high-quality data preparation outcomes. Leveraging CoLR embeddings and scalable computing techniques, the Interface Manager enables efficient data preparation on ad-hoc datasets, enhancing the productivity and effectiveness of data scientists. By seamlessly integrating with the API and Interface Library, the Interface Manager ensures a smooth and intuitive user experience, enabling data scientists to navigate the data preparation process with confidence and achieve their data science goals effectively.

The APIs and Interface Library within KGFarm serve as a fundamental component, facilitating seamless integration with popular Python-based data science platforms such as Jupyter Notebook or Google Colab. This integration empowers users to leverage the powerful capabilities of KGFarm interactively or programmatically while writing their pipeline scripts. The APIs are designed to provide a smooth workflow experience, allowing users to input a Pandas dataframe and obtain a Pandas dataframe as output, ensuring compatibility with widely-used data manipulation tools.

The APIs and Interface Library play a crucial role in assisting users with both seen and unseen datasets. For unseen datasets, the Interface Manager is utilized to provide recommendations for data preparation operations. By capitalizing on the knowledge captured by the recommendation models, the APIs enable efficient and precise application of the recommended operations, eliminating the need for users to possess explicit knowledge of data preparation techniques. This seamless integration streamlines the data preparation process, enabling users to quickly apply cherry-picked recommendations from the Inference Manager to their datasets, saving valuable time and effort.

Moreover, these APIs can be used for real-time querying of the LiDS knowledge graph when dealing with profiled (seen) data. Each API encapsulates a specific data preparation task and utilizes SPARQL queries to retrieve up-to-date and accurate information from the LiDS graph. By leveraging the comprehensive knowledge captured in the graph, the APIs empower users with recommendations based on the experiences of other data scientists in the past, guiding them through the data preparation process effectively.

Designed as a Python package, the KGFarm APIs seamlessly integrate into typical data science workflows, accelerating the overall development and execution of data science pipelines. The user-friendly and efficient interface of the APIs and Interface Library enhances the usability and accessibility of KGFarm, making it a valuable tool for data scientists striving for efficient and effective data preparation in their endeavors. To illustrate the practical use of the KGFarm APIs, we highlight their application in solving a machine learning task related to *earthquake magnitude* prediction.

Feature Identification

Identifying the right set of features for a machine learning task can be a challenging and time-consuming process. KGFarm addresses this problem by providing an API for feature identification:

```
identify_features(entity='earthquake',target='magnitude')
```

This API takes the entity and target names associated with the machine learning task as input. It queries the LiDS graph to fetch the feature view linked to the entity ('earthquake') that includes the target ('magnitude') as a feature. The API then returns all the columns, excluding those representing the entity and the target, as the potential features describing the entity.

Data Enrichment

Data enrichment is crucial for enhancing the base modeling data by augmenting the initial set of features with information from different data sources. KGFarm provides the following APIs to efficiently address this problem:

```
enrichment_info = search_enrichment_options(df)
```

This API requires the base dataframe as input and returns a dataframe containing information about all possible tables that can be used to enrich the base dataframe. KGFarm leverages the physical representation of entities to recommend the most accurate set of joinable tables. The obtained information can then be directly used to perform enrichment using the following API:

```
enrich(df, enrichment_info.iloc[0])
```

The *enrich* API takes the base dataframe and the enrichment information obtained from the *search_enrichment_options* API as input. KGFarm returns the enriched dataframe by applying the provided enrichment information. When working with time-series data, the presence of stale features can impact the predictiveness of the model. To address this issue, KGFarm offers an additional parameter to handle stale features:

```
enrich(df, enrichment_info.iloc[0], freshness=365)
```

This API accepts the base dataframe, enrichment information from the *search_enrichment_options* API, and a freshness parameter that defines the time window of the features (365 days in the provided example). KGFarm performs the join using the provided information and eliminates records that do not fall within the freshness time window. This ensures that outdated records are filtered out, preventing the introduction of noise into the model.

Data Cleaning

Data cleaning in KGFarm focuses on visualizing, fixing, and removing incomplete or corrupted data. To support this task, KGFarm provides the following API:

```
cleaning_info = recommend_cleaning_operations(df)
```

The *recommend_cleaning_operations* API takes a raw dataframe as input and uses the data cleaning recommendation model to return the most suitable cleaning operations as a dataframe. Users can then directly apply the recommended cleaning techniques using the following API:

```
clean(df, cleaning_info.iloc[0])
```

The *clean* API accepts the cleaning operation obtained from the *recommend_cleaning_operations* API, along with the raw dataframe. It returns the cleaned dataframe after applying the provided cleaning operation.

Data Transformation

KGFarm simplifies the transformation of numerical and categorical features in a dataset. Users can leverage the following API to perform data transformation:

```
transform_info = recommend_data_transformations(X_train)
```

This API takes the untransformed dataframe as input and uses the data transformation recommendation model to return the most probable transformation techniques as a dataframe. Similarly, users can apply the recommended transformation of their choice without writing dedicated code using the following API:

```
apply_transformation(X_train,transform_info.iloc[0])
```

The *apply_transformation* API accepts the transformation technique obtained from the *recommend_data_transformations* API, along with the untransformed dataframe. It returns the transformed dataframe along with the transformation model after applying the provided transformation technique.

Feature Selection

After preparing the data, it is essential to select the most relevant features for predicting the target variable. This helps avoid overfitting and reduces training time by creating a more concise dataset for modeling. KGFarm offers the following API for feature selection:

`recommend_top_k_features(``X_train,y_train)`

The *recommend_top_k_features* API expects the independent variables (features) as a dataframe and the target as a Pandas series object as input. It returns a dataframe with the features and their corresponding relevance scores with respect to the target variable. This enables data scientists to sub-select features based on their specific requirements.

The KGFarm APIs and Interface Library provide a comprehensive set of tools to assist data scientists in various data preparation tasks. By seamlessly integrating with existing Python-based data science platforms and offering intuitive and user-friendly interfaces, these APIs empower data scientists to efficiently explore, clean, transform, enrich, and select features from their datasets. Through their versatile functionality and smooth integration, the KGFarm APIs enhance the overall data preparation process, enabling data scientists to make informed decisions and achieve high-quality outcomes in their machine learning tasks.

4.5 KGFarm Characteristics

In this section, we delve into the core characteristics of KGFarm that establish it as a powerful and efficient tool for automated data preparation. We explore key aspects such as taylor stitched recommendations, scalability, leakage-aware data transformation, and integration with existing data science workflows. These characteristics exemplify the unique capabilities of KGFarm and underscore its potential to enhance the efficiency and effectiveness of the data preparation process for machine learning tasks. By leveraging these features, data science practitioners can tap into the collective wisdom and experience of their peers, enabling a high level of automation and driving optimal outcomes in their machine learning endeavors.

4.5.1 Taylor Stitched Recommendations

In KGFarm, the KG Augmentor can be extended to label specific data preparation operations (referred to as $\mathcal{DP}$ in subsection 4.2.2), enabling the training of predictive models for recommending tailored operations. This augmentation process requires two essential components: 1) the implementation of operation(s) of interest, such as the *CountEncoder* transformation offered in the *category_encoders* library [4], provided as a class/function, and 2) pipelines that incorporate these operation(s). By fulfilling these conditions, enterprises can generate and build custom ML models using the Training Manager, which can predict data preparation operations inspired by their own code repositories and pipelines.

These recommendations leverage machine learning to analyze the implementations of data preparation operations within ML pipelines. By training models on labeled data that capture the relationships between data transformations, feature selections, and other preparatory operations, KGFarm can provide tailored recommendations to data science practitioners.

Recognizing the significance of similar data demanding similar operations, KGFarm has been developed to generate custom models capable of recommending operations inspired by previously seen code repositories and pipelines. By harnessing the collective intelligence within an organization, KGFarm enables data scientists and practitioners to extract valuable insights from their existing ML pipelines. By examining the historical usage patterns and performance of data preparation operations, KGFarm identifies effective and efficient combinations of operations for specific machine learning tasks, empowering data scientists to make informed decisions and leverage the best practices established within their organization.

By incorporating Taylor Stitched Recommendations into their data science workflow, enterprises can enhance the efficiency and effectiveness of their data preparation process. The ability to generate custom ML models that predict relevant operations based on their own code repositories and pipelines provides a powerful tool for automating and optimizing data preparation. This feature streamlines the overall machine learning pipeline and promotes the adoption of best practices within the organization, leading to improved model performance and accelerated pipeline development.

4.5.2 Scalability

Scalability is a crucial characteristic of KGFarm that enables accelerated and efficient data preparation for machine learning tasks, even with large and complex datasets. KGFarm achieves scalability through various components and techniques integrated into its framework.

As discussed in section 4.4, KGFarm addresses the challenge of handling large datasets during data

[4] https://github.com/scikit-learn-contrib/category_encoders

cleaning, data transformations, and feature selections by utilizing CoLR embeddings instead of raw column values. CoLR embeddings compress and represent column information in a more compact form, facilitating efficient processing and analysis. This approach proves particularly effective when dealing with a large number of rows, as it reduces memory requirements and speeds up computations.

To parallelize computations on the level of features, KGFarm leverages the Dask framework. Dask enables distributed computing and parallel execution of tasks, significantly enhancing the scalability of KGFarm. By dividing the computation across multiple cores or machines, Dask enables faster processing of large datasets with numerous columns.

During the feature engineering phase, KGFarm incorporates statistical pruning techniques to achieve greater scalability. These techniques employ lightweight statistical methods, such as information gain and feature correlation, to prune candidate features. By eliminating irrelevant or redundant features early in the process, KGFarm reduces the computational burden and enhances the scalability of subsequent operations.

In terms of identifying physical representations of entities, KGFarm adopts a scalable approach by utilizing content similarity, which is a superset of inclusion dependencies (Fernandez et al., 2018; Helal et al., 2021). Classification is performed on top of content similarity to identify PK-FK relationships (as described in subsection 4.2.1). This approach enables the efficient handling of large and complex datasets with numerous entities.

By incorporating these scalability-enhancing techniques, KGFarm empowers data scientists and practitioners to address data preparation challenges associated with large-scale datasets. The ability to efficiently and effectively clean and transform data at different levels, from rows to columns, ensures that KGFarm can meet the demands of real-world enterprise-scale data scenarios.

4.5.3 Leakage-aware Data Transformation

Data leakage refers to the inadvertent inclusion of information from the test set or the target class throughout the data preparation phase (specifically during the data transformation process), leading to overfitting, overestimated performance, and potentially biased models (Samala, Chan, Hadjiiski, & Koneru, 2020). KGFarm has been designed with a focus on mitigating the problem of leakage during data transformations, ensuring accurate model evaluation and unbiased predictions.

In many machine learning tasks, scaling the data is a common preprocessing step. However, if scaling transformations, such as *StandardScaler* or *MinMaxScaler*, are naively applied on the entire dataset without considering the issue of leakage, it can introduce data leakage. These scaling transformations utilize statistical properties of the feature space, such as mean, standard deviation, minimum, and maximum values, including information from the test set. As a result, the transformed feature may contain direct information about the test set, leading to an overestimation of the model's performance during evaluation.

KGFarm recognizes the potential for leakage in certain types of transformations, while others are considered relatively safer. The scaling transformations described in table 4.3 are prone to leakage whereas unary transformations such as encoding techniques: one-hot encoding or ordinal encoding, typically do not introduce leakage as they perform a mapping of categories to numerical representations.

To address the issue of leakage during scaling transformations, KGFarm adopts a leakage-aware approach. When scaling transformations are applied using KGFarm's APIs and Library Interface (presented in section 4.4), it returns the transformed training dataframe along with the transformation model fitted on the training data. This transformation model can then be utilized to transform the test data in an isolated manner, preventing leakage. For example, the following API can be used to transform data:

```
X_train, scaler = apply_transformation(X_train, transform_info.iloc[0])
```

Here, along with the transformed train set, the user is also returned with the *scaler* model, which can be used to transform the test set isolatedly as follows:

```
X_test = scaler.transform(X_test)
```

By adopting this leakage-aware strategy, KGFarm ensures that the accuracy and performance of the model are not overpromised during evaluation. It promotes a more realistic and unbiased assessment of the model's generalization capabilities, allowing for more reliable predictions on unseen data. Incorporating leakage-aware data transformation techniques in KGFarm provides data scientists with greater confidence in the performance and reliability of their machine learning models.

4.5.4 Integration with Existing Data Science Workflow

KGFarm is designed to seamlessly integrate with the existing data science workflow, providing data scientists with a higher level of automation in the data preparation process. It eliminates the need for time-consuming exploratory data analysis (EDA) and offers a comprehensive solution for efficient data preparation.

As an open-source Python library [5], KGFarm is easily accessible and can be integrated into various data science environments. Its APIs are designed to work on top of dataframes, enabling smooth integration with popular Python frameworks such as Pandas, Dask, PySpark, etc. This compatibility ensures that data scientists can leverage their existing tools while benefiting from the advanced capabilities of KGFarm.

Not only does KGFarm recommend the most appropriate data preparation operations for ad-hoc datasets, but it also provides APIs that allow data scientists to apply those operations without writing explicit code.

[5] https://github.com/CoDS-GCS/kgfarm

This simplifies the implementation process and reduces the time and effort required for data preparation tasks.

By offering a holistic platform for data preparation, KGFarm empowers data scientists to streamline their workflow and focus on the core aspects of their machine learning projects. The automation and integration capabilities of KGFarm contribute to a more efficient and productive data science process, ultimately leading to improved and accelerated model performance. In summary, KGFarm serves as a valuable addition to the data science ecosystem, providing data scientists with a comprehensive solution for automated data preparation. Its open-source nature, dataframe-based APIs, and seamless integration with existing frameworks make it a powerful tool for enhancing the efficiency and effectiveness of data science workflows.

Chapter 5

Use Case & Evaluation

This chapter presents the practical application and evaluation [1] of KGFarm in real-world scenarios, demonstrating its efficiency and effectiveness in addressing data preparation challenges and enhancing the performance of machine learning tasks.

The chapter begins with a use case study showcasing the utilization of KGFarm by a mechanical engineering research team working on a smart city project. The team's focus is on the stability of hybrid power systems, with the objective of identifying instability, designing controllers, and developing recommender systems to effectively address disruptions. KGFarm plays a crucial role in assisting the team by recommending data transformation operations, eliminating the need for manual code writing and iterative exploratory data analysis (EDA), thereby accelerating the machine learning pipeline curation process.

Furthermore, the chapter includes an experimental evaluation that compares KGFarm with state-of-the-art systems. The evaluation encompasses data cleaning, data transformation, feature selection, and feature engineering tasks, using 130 unseen datasets obtained from AutoML benchmarks. It assesses the effectiveness and efficiency of KGFarm by evaluating the impact of the recommended data preparation operations on machine learning tasks. Additionally, the evaluation includes a comparison of KGFarm's approach for determining primary key-foreign key PK-FK relationships using content similarity as an alternative to inclusion dependencies, which serves as the basis for feature discovery and data enrichment.

Through the use case study and experimental evaluation, this chapter provides valuable insights into the practical utilization and evaluation of KGFarm in real-world workloads. It highlights the benefits of employing KGFarm in various scenarios and emphasizes its ability to automate the data preparation phase, enhancing the efficiency and effectiveness of data science workflows.

[1] All experiments have been executed on a machine running Ubuntu 20.04.4 LTS, with a 2.40 GHz Intel Core Processor CPU with 16 cores and 88.5 GB of RAM within a time budget of 3 hours.

5.1 Use Case

This section presents a use case study that demonstrates the application of KGFarm in the mechanical engineering sector of a smart city project. The use case focuses on addressing the stability challenges in hybrid power systems for developing data-driven controllers to effectively manage disruptions.

Ensuring stability in hybrid power systems is crucial as we transition to more sustainable energy sources. Conventional power systems with fossil fuel-based generators exhibit stability due to the electrical inertia of their components. However, hybrid systems integrating renewable energy sources lack this inertia, making it difficult to maintain a consistent frequency and prevent disruptions. Therefore, this use case leverages machine learning techniques to detect component faults and disruptions in hybrid power systems.

In the initial step of the project, the mechanical engineering research team focuses on performing fault detection in hybrid power systems. Simulations are conducted using PowerFactory [2], a power system simulation software, to replicate the behavior of an IEEE nine-bus power system when a fault (short-circuit event) occurs in one of the system elements for a specific duration. The data generated from these simulations is utilized to train deep learning models for predicting the stability of hybrid power systems. The objective is to identify the faulty component out of the 24 components present in the power system, formulating the task as a multiclass classification problem.

The mechanical engineering research team has developed their own ML pipeline in two weeks' time specifically tailored to address the fault detection problem in hybrid power systems. Within their pipeline, they incorporate various manual data transformation techniques guided by exploratory data analysis (EDA) processes and trial and error. For modeling, the team implements a deep learning architecture comprising 8 hidden layers, each containing 256 nodes. The activation function utilized in these hidden layers is the Leaky ReLU (Rectified Linear Unit), and the optimization of the model's performance is achieved using the Adam optimizer. The architecture also includes a dropout layer with a rate of 0.5 to prevent overfitting. The output layer consists of 24 nodes, representing the different classes or labels corresponding to the components. To ensure robust model performance and mitigate overfitting, the team employs k-fold cross-validation, splitting the dataset into training and validation sets. During training, the model minimizes the categorical cross-entropy loss for the multiclass classification tasks of classifying the faulty component in the hybrid power systems.

The ML pipeline incorporates three unique, hand-curated data transformation techniques including:

(1) Z-score Normalization

$$X_{zscore} = \frac{X - \mu_X}{\sigma_X} \tag{5}$$

[2]https://www.digsilent.de/en/

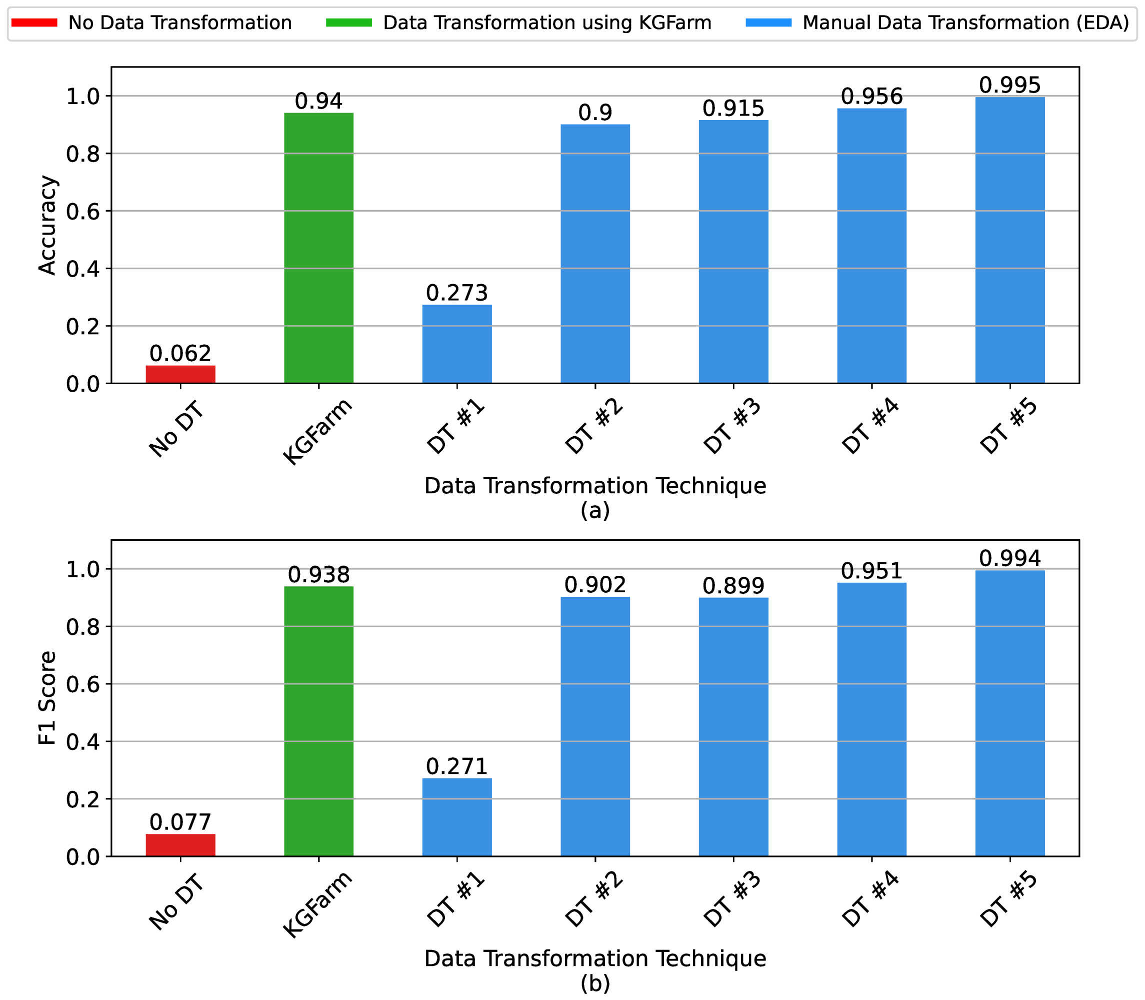


Figure 5.1: Comparative (a) Accuracy and (b) F1 Score of Data Transformation Methods in the ML Pipeline

In equation 5, X_{zscore} represents the standardized value, X represents the original value (simulated voltage with faults), μ_X represents the mean and σ_X represents the standard deviation of X.

(2) Min-Max Normalization

$$X_{min-max} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{6}$$

In equation 6, $X_{min-max}$ represents the normalized value, X represents the original value (simulated voltage with faults), X_{min} and X_{max} represents the minimum and maximum values of X respectively.

(3) Differential Shift

$$X_{differential} = X - X_{default} \tag{7}$$

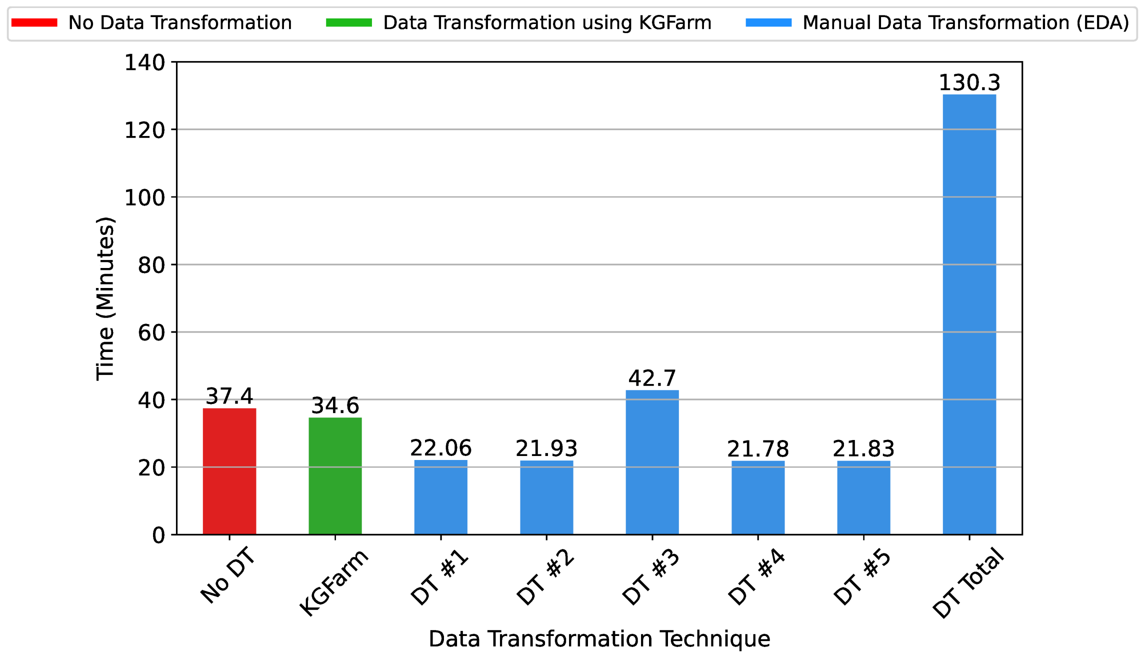


Figure 5.2: Comparative Time Consumption (in minutes) for Data Transformation Methods in the ML Pipeline

In equation 7, $X_{differential}$ represents the differential shift, X represents the original value (simulated voltage with faults) and $X_{default}$ represents the fault-free voltage i.e. constant or default of X. This transformation uses the domain expertise of the mechanical engineering team. In the ML pipeline, $X_{default}$ is the simulated voltage without fault and contains no anomaly, and X is the simulated voltage with faults. Hence subtracting the default values from the original value (with faults) highlights the deviation or shift caused by the presence of the faults.

Using the combination of these three transformation methods described by equation 5, 6 and 7 the pipeline implemented five transformation techniques including:

- DT #1: Min-Max Normalization
- DT #2: Min-Max Normalization + Differential Shift
- DT #3: Z-score Normalization
- DT #4: Z-score Normalization + Differential Shift
- DT #5: Differential Shift

Integrating KGFarm in this use case provided significant advantages for the mechanical engineering research team. KGFarm offers a seamless and efficient solution for automating the data preparation phase. By leveraging KGFarm, the team can streamline the data transformation process without the need for manual EDA or writing explicit code. This automation enables the team to accelerate the data preparation stage and focus on modeling the fault detection problem. KGFarm serves as a valuable tool, facilitating the data preparation tasks required for their machine learning models.

To measure the value that KGFarm adds to this problem, we implemented the same pipeline with the exact deep learning model, but instead of manual EDA-based data transformations, we used KGFarm to recommend and apply data transformations on this unseen data for modeling. KGFarm recommended the following transformations: StandardScaler, which corresponds to the Z-score Normalization technique shown in equation 5; followed by Sqrt and Log transformations on specific features. In this study, we compared the accuracy, F1 score, and time consumption of the manual data transformation techniques versus those recommended by KGFarm. Figure 5.1 (a) shows the comparative accuracy, and Figure 5.2 (b) shows the F1 score of the different techniques. It can be observed that KGFarm outperforms DT #1, DT #2, and DT #3, achieving an accuracy of 0.94 and an F1 score of 0.938, which is comparable to DT #4. DT #5, which relies solely on differential shift, achieves the highest effectiveness with an accuracy of 0.995 and an F1 score of 0.994. Applying no transformation or using Min-Max Normalization leads to the lowest effectiveness. However while observing the time consumption (shown in Figure 5.2), running the pipeline with KGFarm for recommending and applying the most suitable data transformations took 34.6 minutes, while applying all five data transformation techniques guided by EDA took 130.3 minutes, more than 3 times the time taken by the pipeline using KGFarm.

Therefore, this use case proves KGFarm's capability in handling diverse datasets like the simulated dataset. By automating the data preparation, KGFarm enables users to efficiently transform their data while maintaining comparable accuracy. It offers a valuable solution for streamlining the data preparation phase by eliminating the need for manual exploratory data analysis, trial and error, and explicit coding.

5.2 Experimental Evaluation

We evaluate KGFarm to several related systems including the SOTA data preparation systems. For data cleaning, we compare KGFarm against HoloClean (Rekatsinas et al., 2017), a data cleaning system that uses statistical learning and inference to unify a range of data repairing methods, and DataWig (Biessmann et al., 2019), an open-source library that uses deep learning feature extractors to perform missing value imputations. For data transformation and feature engineering, we compare KGFarm against AutoLearn (Kaul et al., 2017),

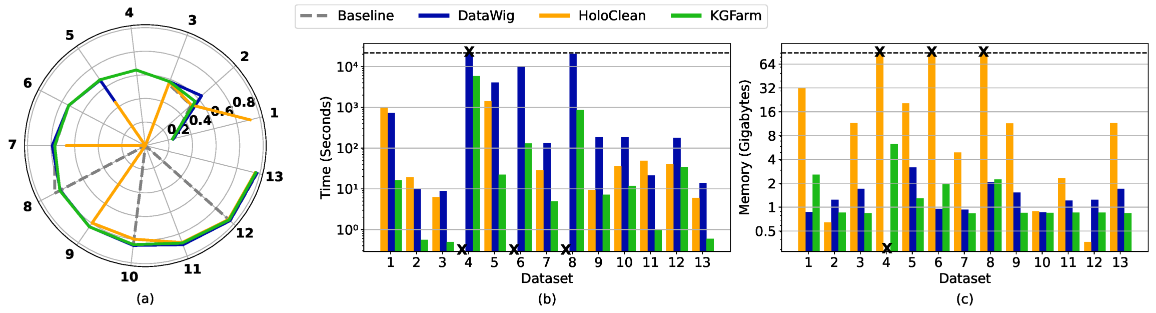


Figure 5.3: Data Cleaning: the performance of KGFarm vs. existing systems on multiple ML tasks on 13 datasets. (a) in the radar diagram, the outer numbers indicate different dataset IDs and the ticks inside the figure denote performance ranges of respective metrics; e.g., 0.2, 0.4, ..., etc. for F1 in each ML task per dataset. For any dataset, the system with the out most curve has the best performance. (b) and (c) the time and memory consumed per system to perform data cleaning on the dataset and train the model. HoloClean and DataWig timeout or exceed the memory budget in several cases. KGFarm outperforms HoloClean and DataWig in most cases with better scalability in terms of time and memory.

a regression-based feature learning algorithm that automates feature engineering, and LFE (Nargesian et al., 2017), a meta-learning approach that automates data transformation for classification problems. For feature selection, we compare KGFarm to the most commonly used feature selection techniques, such as Filter, Wrapper, and Embedded method. Additionally, we also compare KGFarm's PK-FK discovery approach using content similarity versus inclusion dependencies (Rostin et al., 2009).

For training the KGFarm's data preparation models, we used the top-rated 1000 Kaggle datasets comprising $\sim$ 3.7k tables, $\sim$ 140k columns, and $\sim$ 13.8k pipelines so as to control the quality of our recommendations. For testing the data preparation recommendations for data cleaning, data transformation, feature selection and feature engineering to the above-mentioned systems and techniques, we used a total of 130 unseen datasets, comprising datasets [3] collected by KGpip (Helali et al., 2022) from different AutoML benchmarks and others datasets from UCI repository (Dua & Graff, 2017). Here, each dataset is associated with an ML task. We perform data preparation using KGFarm and the above-mentioned systems followed by modeling. We consider the model performance as our main metric to measure the accuracy of each system. We also consider the time and memory consumed by the system. A complete list of the 130 datasets including their ID and statistics is provided in the appendix A.1 of this thesis.

5.2.1 Data Cleaning

In this set of experiments, we used the eight datasets from the AutoML benchmarks (Helali et al., 2022) that contained missing values and supplemented them with five datasets from the UCI repository (Dua & Graff, 2017). Our evaluation consisted of cleaning these datasets using KGFarm, DataWig, and HoloClean. Our evaluation metric includes the F1 scores of a random forest classifier trained using cross-validation over ten folds on the cleaned datasets.

In Figure 5.3 (a), it can be observed that KGFarm consistently occupies the outer edge of the radar graph, which serves as a visual representation of its consistently high F1 scores. Regarding efficiency, KGFarm is outperforming DataWig and HoloClean in 85% and 77% of datasets concerning time and in 69% and 62% of datasets w.t.o the memory usage respectively.

HoloClean faced memory constraints while attempting to clean datasets 4, 6, and 8, resulting in an inability to complete the cleaning process for these datasets. These datasets are given an F1 score of 0 in Figure 5.3 (a) and marked with an 'x' on the x-axis in Figure 5.3 (b), indicating that no time values were obtained for these datasets and an 'x' on top of the bar in Figure 5.3 (c), indicating that the memory capacity of the machine used was exceeded. DataWig exceeded the time budget of 3 hours while cleaning dataset 4, resulting in an 'x' on top of the bar in Figure 5.3 (b) and an 'x' on the x-axis in Figure 5.3 (c). We also consider a baseline method that performs modeling without data cleaning. The baseline method faced challenges with datasets, whose IDs are 9, 11, and 13. One notable advantage of KGFarm is its ability to successfully complete the cleaning process on larger datasets, such as datasets whose IDs are 4, 6, and 8, where HoloClean encountered memory limitations. As a result, KGFarm utilizes comparatively less memory compared to HoloClean, which generates multiple tables containing dataset information throughout its cleaning process. KGFarm balances between efficiency and cleaning effectiveness, allowing users to clean their datasets effectively with little time and memory constraints.

5.2.2 Data Transformation

This set of experiments compared KGFarm to LFE and AutoLearn using the open datasets used in LFE's experimental evaluation (Nargesian et al., 2017) to facilitate the direct comparison between these systems. Our evaluation procedure involved applying the data transformation recommended by KGFarm, LFE, and AutoLearn to each dataset. Subsequently, we trained a random forest classifier using a 10-fold cross-validation approach. Finally, we computed the F1 score for each system, as depicted in Figure 5.4 (a). In Figure 5.4 (b), KGFarm outperformed LFE on approximately 69% and AutoLearn on 62% of the datasets in the F1 score

[3] `https://github.com/CoDS-GCS/kgpip-public/tree/master/benchmark_datasets`

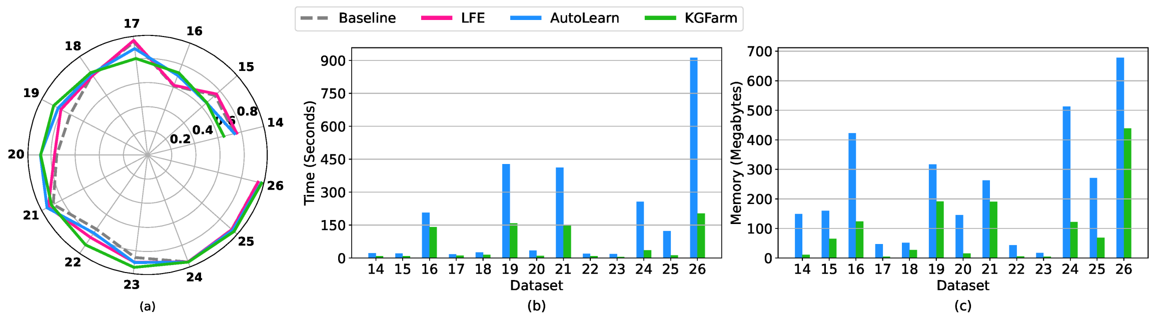


Figure 5.4: Data Transformation: the performance of KGFarm vs. existing systems on multiple ML tasks on 13 datasets. (a) in the radar diagram, the outer numbers indicate different dataset IDs and the ticks inside the figure denote performance ranges of respective metrics, e.g., 0.2, 0.4, ..., etc. for F1 in each ML task per dataset. For any dataset, the system with the out most curve has the best performance. KGFarm outperforms LFE and AutoLearn in most cases. (b) and (c) the time and memory consumed per system to perform data transformation on the dataset and train the model. F1 scores of LFE are reported by the authors, however, the code is not available. KGFarm outperforms AutoLearn in terms of time and memory performance.

metric while consuming less time and less memory than AutoLearn consistently.

5.2.3 Feature Selection

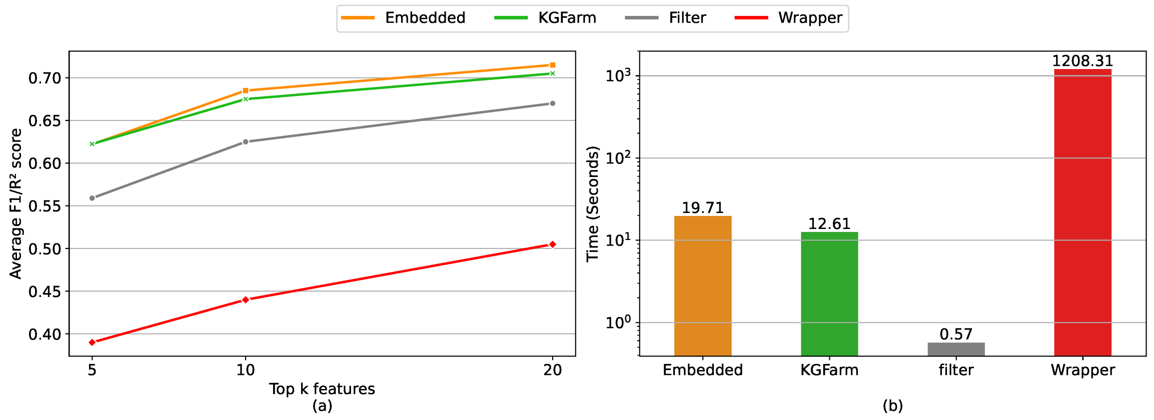


Figure 5.5: Average (a) F1/R^2 of a random forest over different values of k, which represents the top-recommended features for the classification task. (b) Time taken by each feature selection technique to determine feature importance.

KGFarm is compared to the most commonly used feature selection methods, such as Filter, Wrapper, and Embedding (Kumar & Minz, 2014) over thirty-four datasets that have no less than 30 features in (Helali

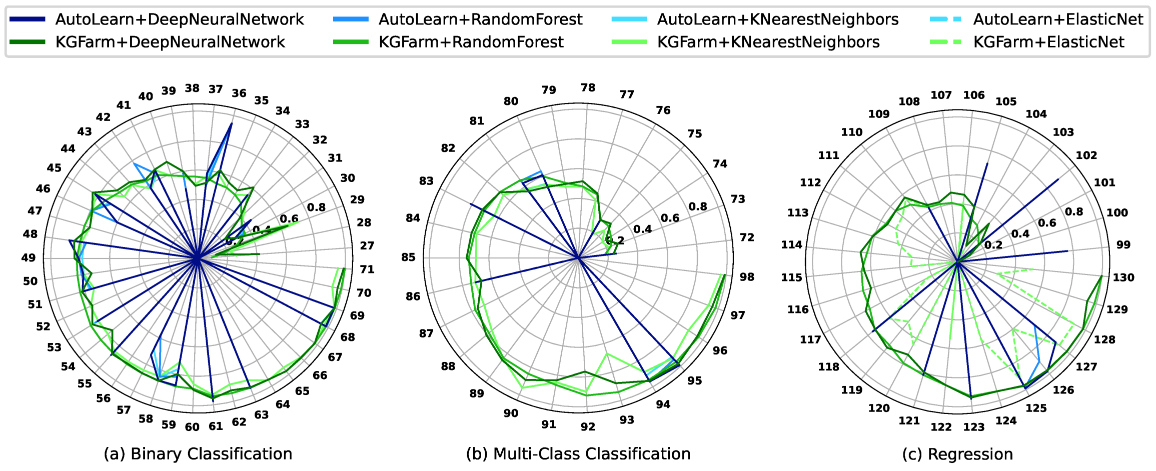


Figure 5.6: Feature Engineering: the performance of KGFarm vs. AutoLearn on multiple ML tasks on 104 datasets categorized into (a) Binary Classification Tasks, (b) Multi-Classification Tasks, and (c) Regression Tasks. After feature engineering, we trained using different ML methods indicated with variant colors. In the radar diagram, the outer numbers indicate different dataset IDs and the ticks inside the figure denote the performance ranges of respective metrics. For any dataset, the system with the out most curve has the best performance. KGFarm outperforms AutoLearn in most cases.

et al., 2022) benchmark. One feature selection technique was selected per method, namely: Anova-test for the Filter, Decision tree for the Embedded, and finally, Recursive Feature Elimination (RFE) for the Wrapper method. All these techniques were used alongside KGFarm to determine the top-k features (k $\in \{5, 10, 20\}$) of most relevance for a particular target variable. We used a random forest classifier with 10-fold cross-validation over these top-k features and F1 for classification, and R^2 scores for the regression task were measured. Figure 5.5 (a) shows the effectiveness of the KGFarm in feature selection where KGFarm and Embedded technique has the highest efficiency over k $= 5$. The embedded technique outperforms other methods over higher values of k followed by KGFarm and Filter. For efficiency, as shown in Figure 5.5 (b), the Filter method using statistical Anova-test, was significantly faster than techniques followed by KGFarm, Embedded, and finally, the Wrapper method being the least efficient. Overall, KGFarm provides a respectable balance between efficiency and effectiveness.

5.2.4 Feature Engineering

We compared KGFarm to AutoLearn, a commonly used feature engineering system, using a total of 104 datasets designed for binary and multi-class classification and regression tasks. Both KGFarm and AutoLearn were used to engineer i.e. prune and transform the most relevant features for each dataset. Furthermore, to

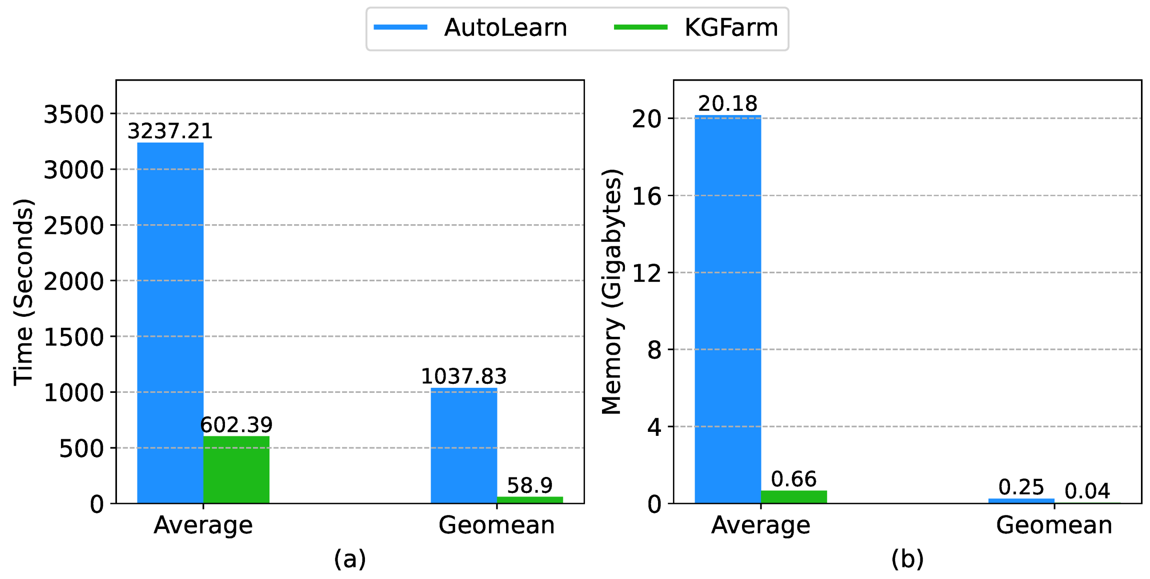


Figure 5.7: (a) Time and (b) Memory consumed for Feature Engineering. KGFarm consumes significantly less time and memory on average than AutoLearn.

Table 5.1: Comparison of Performance Metrics: Content Similarity vs. Inclusion Dependency

Metric	Content Similarity	Inclusion Dependency
Accuracy	0.89	0.90
Precision	0.87	0.86
Recall	0.88	0.89
F1	0.87	0.87

ensure unbiased evaluation, these engineered features were combined with multiple machine learning algorithms, including a Deep neural network, a Random forest, KNN, and ElasticNet. AutoLearn struggles while handling larger datasets, resulting in timeout issues (40%) or out-of-memory errors (12%). Additionally, AutoLearn's pipeline does not generalize well while pruning features, leading to excessive pruning of datasets until no features remained to process (8%) as well as with handling datasets consisting solely of categorical features (1%). KGFarm has outstanding performance as illustrated in Figures 5.6 and 5.7.

5.2.5 PK-FK Discovery

In the experiment, we conducted a comparative analysis between our novel PK-FK discovery approach, utilizing computationally efficient content similarity, and the traditional inclusion dependency method (Rostin et al., 2009) using the spider algorithm (Bauckmann, Leser, Naumann, & Tietz, 2007). To evaluate the performance of the PK-FK discovery, we trained an ensemble model on eight features as explained in table 4.1, using data from the LiDS graph that captures content similarity through CoLR embeddings. We employed a leave-one-out evaluation approach on six databases from diverse domains, including life sciences (SCOP, MSD, UniProt), movies (Movielens, Filmdienst), and TPC-H benchmark. During the evaluation, we recorded various metrics such as accuracy, precision, recall, and F1 score. This entire process was then repeated using inclusion dependency column pairs instead of column similarity.

The results, as shown in Table 5.1, demonstrate the efficacy of the content similarity approach in PK-FK discovery. The model utilizing content similarity achieved comparable scores to the computationally heavy inclusion dependency method. Specifically, both approaches resulted in the same F1 score of 0.87 and comparable accuracy, precision, and recall. These findings highlight the efficiency and effectiveness of our PK-FK discovery approach, showcasing its potential to replace the more computationally expensive inclusion dependency method without compromising on accuracy.

Chapter 6

Conclusion & Future Work

This thesis presents KGFarm, a fully-fledged platform for automating data preparation using the semantics of data science artifacts captured as a knowledge graph. To construct this knowledge graph, we utilized an existing system, which collected data from top-rated 1000 Kaggle datasets and 13800 pipeline scripts with the highest number of votes. Within KGFarm, data cleaning, data transformation, and feature selection tasks are formulated as machine learning classification tasks, trained on the knowledge graph. Additionally, KGFarm augments this knowledge graph to abstract physical data science artifacts by discovering PK-FK column relations using machine learning.

Additonally, this thesis presents a practical use case demonstrating the effectiveness of KGFarm in solving real-world challenges faced by data scientists. Furthermore, we conducted a comprehensive evaluation using 130 ad-hoc datasets, which were not part of the training data for our models. To assess the performance of KGFarm, we compared it with state-of-the-art data preparation systems. The experimental results reveal that KGFarm outperforms other systems, achieving significant reductions in time and memory usage, up to two orders of magnitude, while simultaneously improving accuracy, particularly with large-scale datasets.

Looking ahead, KGFarm holds several promising future directions to further enhance its capabilities and impact in the field of data science automation. One significant direction involves exploring the use of Graph Neural Network (GNN) (Abdallah, Nguyen, Nguyen, & Mansour, 2021; Zhou et al., 2020) models directly on the LiDS knowledge graph. Currently, KGFarm extracts labeled information from the graph to train machine learning models for data preparation tasks. By leveraging GNNs, which are specifically designed to handle graph-structured data, KGFarm can benefit from the inherent graph structure and consider information propagated through neighboring nodes and various relationships within the graph. This integration of GNNs could potentially improve KGFarm's ability to capture complex patterns and dependencies within data science artifacts, leading to more accurate data preparation recommendations.

Addressing feature bias detection is another crucial aspect for the future development of KGFarm. Feature bias refers to the presence of biased features in datasets, which can have significant implications in sensitive domains, such as shortlisting job applications or insurance (Bellamy et al., 2018). To tackle this issue, KGFarm aims to develop models that can identify biased features based on the context of a specific data science task. By detecting and mitigating biased features, KGFarm can contribute to the development of fair and ethical data science pipelines (Nargesian, Asudeh, & Jagadish, 2021), ensuring that data-driven decisions are free from unfair biases.

Furthermore, KGFarm envisions automating the creation of data science pipelines from a concise description of a data science task (Einblick, 2023). This ambitious direction involves developing models capable of understanding the given task description in terms of dataset topics and related pipelines similar to large language models (Kasneci et al., 2023). These models would then link the data topics to actual datasets, tables, or columns in the data science knowledge graph. By effectively identifying relevant datasets, KGFarm can use them as input for an Automated Machine Learning (AutoML) system to generate a full data science pipeline automatically. This automation would enable users to express their data science tasks in natural language and receive fully automated and optimized pipelines as output, streamlining the entire data science workflow and minimizing manual efforts.

Appendix A

Benchmark

Table A.1: 130 Open Datasets used for Evaluating KGFarm against State-Of-The-Art Systems for Data Preparation

Dataset ID	Dataset	Size (MB)	# Rows	# Columns	# Categorical Features	# Numerical Features	# Missing Columns	Target	ML Task	Source
1	jm1	1.754	10885	22	5	17	5	defects	binary	OpenML
2	hepatitis	0.008	155	20	0	20	0	Class	binary	UCI
3	cleveland_heart_disease	0.012	303	15	0	15	2	condition	binary	UCI
4	albert	256.302	425240	79	25	54	45	class	binary	AutoML
5	adult	5.59	48842	15	9	6	2	class	binary	AutoML
6	higgs	21.694	98050	29	9	20	9	class	binary	AutoML
7	titanic	0.082	891	12	5	7	4	Survived	binary	Kaggle
8	APSFailure	99.152	76000	171	170	1	169	class	binary	AutoML
9	credit-g	0.16	1000	21	14	7	3	class	binary	AutoML
10	credit	0.031	690	16	9	7	7	A16	binary	UCI
11	horsecolic	0.019	300	28	0	28	21	surgery	binary	UCI
12	housevotes84	0.017	435	17	17	0	16	Class	binary	UCI
13	breastcancerwisconsin	0.02	699	11	0	11	1	diagnosis	binary	UCI
14	Pima-indians-subset	0.023	768	9	0	9	0	Outcome	binary	UCI
15	Diabetes	0.023	768	9	0	9	0	Outcome	binary	UCI
16	Madelon	4.969	2600	501	0	501	0	Class	binary	UCI
17	Spectf-heart	0.01	80	45	0	45	0	OVERALL-_DIAGNOSIS	binary	UCI
18	Sonar	0.082	208	61	0	61	0	Class	binary	UCI
19	AP-omentum-ovary	19.148	275	10936	0	10936	0	Tissue	binary	UCI
20	Credit-a	0.044	690	6	0	6	0	Class	binary	UCI
21	AP-omentum-lung	14.171	203	10936	0	10936	0	Tissue	binary	UCI
22	Hepatitis	0.008	155	20	0	20	0	Class	binary	UCI
23	Labor	0.003	60	18	0	18	0	"class"	binary	UCI
24	Spambase	0.671	4601	58	0	58	0	Class	binary	UCI
25	Ionosphere	0.078	351	35	0	35	0	column_ai	binary	UCI
26	Gisette	121.912	13500	5000	0	5000	0	labels	binary	UCI
27	ozone-level-8hr	1.411	2534	73	0	73	0	Class	binary	OpenML
28	eeg-eye-state	1.714	14980	15	0	15	0	Class	binary	OpenML
29	guillermo	655.67	20000	4297	0	4297	0	class	binary	AutoML
30	kc1	0.34	2109	22	0	22	0	defects	binary	AutoML

31	blood-transfusion-service-center	0.029	748	5	0	5	0	Class	binary	AutoML
32	mc1	2.754	9466	39	0	39	0	c	binary	OpenML
33	pc4	0.413	1458	38	0	38	0	c	binary	OpenML
34	pollen_871	0.176	3848	6	1	5	0	binaryClass	binary	OpenML
35	numerai28.6	16.167	96320	22	0	22	0	attribute_21	binary	AutoML
36	delta_ailerons	0.326	7129	6	1	5	0	binaryClass	binary	OpenML
37	Hill_Valley_with_noise	0.934	1212	101	0	101	0	target	binary	PMLB
38	airlines	32.921	539383	8	3	5	2	Delay	binary	AutoML
39	Hill_Valley_without_noise	0.934	1212	101	0	101	0	target	binary	PMLB
40	quake	0.067	2178	4	1	3	0	binaryClass	binary	OpenML
41	mammography	0.597	11183	7	1	6	0	class	binary	OpenML
42	puma32H_752	2.063	8192	33	1	32	0	binaryClass	binary	OpenML
43	christine	67.667	5418	1637	0	1637	0	class	binary	AutoML
44	ailerons	4.301	13750	41	1	40	0	binaryClass	binary	OpenML
45	sylvine	0.821	5124	21	0	21	0	class	binary	AutoML
46	space_ga_737	0.166	3107	7	1	6	0	binaryClass	binary	OpenML
47	MiniBooNE	49.74	130064	51	0	51	0	signal	binary	AutoML
48	kin8nm_807	0.563	8192	9	1	8	0	binaryClass	binary	OpenML
49	puma8NH_816	0.563	8192	9	1	8	0	binaryClass	binary	OpenML
50	phoneme	0.247	5404	6	0	6	0	Class	binary	AutoML
51	waveform-5000	1.564	5000	41	1	40	0	binaryClass	binary	OpenML
52	jasmine	3.301	2984	145	0	145	0	class	binary	AutoML
53	Australian	0.079	690	15	0	15	0	A15	binary	AutoML
54	electricity	3.111	45312	9	1	8	0	class	binary	OpenML
55	delta_elevators	0.508	9517	7	1	6	0	binaryClass	binary	OpenML
56	bank32nh_833	2.063	8192	33	1	32	0	binaryClass	binary	OpenML
57	cpu_small_735	0.813	8192	13	1	12	0	binaryClass	binary	OpenML
58	wind_847	0.752	6574	15	1	14	0	binaryClass	binary	OpenML
59	fri_c1_1000_25	0.198	1000	26	1	25	0	binaryClass	binary	OpenML
60	cpu_act_761	1.375	8192	22	1	21	0	binaryClass	binary	OpenML
61	breast_cancer_wisconsin	0.135	569	31	0	31	0	target	binary	PMLB
62	spambase	2.036	4601	58	0	58	0	target	binary	PMLB
63	ionosphere	0.094	351	35	0	35	0	target	binary	PMLB
64	nomao	31.291	34465	119	0	119	0	Class	binary	AutoML
65	kr-vs-kp	0.902	3196	37	37	0	0	class	binary	AutoML
66	Amazon_employee-_access	2.5	32769	10	0	10	0	target	binary	AutoML
67	OVA_Breast	128.919	1545	10937	1	10936	0	Tissue	binary	OpenML
68	analcatdata_supreme	0.247	4052	8	1	7	0	binaryClass	binary	OpenML
69	page-blocks	0.459	5473	11	1	10	0	binaryClass	binary	OpenML
70	riccardo	655.67	20000	4297	0	4297	0	class	binary	AutoML
71	MagicTelescope	1.741	19020	12	1	11	0	class:	binary	OpenML
72	abalone	0.287	4177	9	1	8	0	Class_number-_of_rings	multiclass	OpenML
73	fabert	50.338	8237	801	0	801	0	class	multiclass	AutoML
74	robert	549.393	10000	7201	0	7201	0	class	multiclass	AutoML
75	helena	13.927	65196	28	0	28	0	class	multiclass	AutoML
76	kropt	1.498	28056	7	4	3	0	game	multiclass	OpenML
77	volkert	80.522	58310	181	0	181	0	class	multiclass	AutoML
78	dilbert	152.664	10000	2001	0	2001	0	class	multiclass	AutoML
79	Fashion-MNIST	419.235	70000	785	0	785	0	class	multiclass	AutoML
80	wine_quality_white	0.449	4898	12	0	12	0	target	multiclass	PMLB
81	wine_quality_red	0.147	1599	12	0	12	0	target	multiclass	PMLB
82	jannis	35.136	83733	55	0	55	0	class	multiclass	AutoML

83	vehicle	0.123	846	19	1	18	0	Class	multiclass	AutoML
84	connect-4	22.163	67557	43	0	43	0	class	multiclass	AutoML
85	cnae-9	7.062	1080	857	0	857	0	Class	multiclass	AutoML
86	glass	0.016	205	10	0	10	0	target	multiclass	PMLB
87	mnist_784	419.235	70000	785	0	785	0	class	multiclass	OpenML
88	jungle_chess_2pcs_raw-endgame_complete	2.394	44819	7	1	6	0	class	multiclass	AutoML
89	satimage	1.815	6430	37	0	37	0	class	multiclass	OpenML
90	shuttle	4.425	58000	10	0	10	0	class	multiclass	AutoML
91	dionis	193.691	416188	61	0	61	0	class	multiclass	AutoML
92	covertype	243.802	581012	55	0	55	0	class	multiclass	AutoML
93	splice	1.509	3190	62	62	0	1	Class	multiclass	OpenML
94	segment	0.353	2310	20	1	19	0	class	multiclass	AutoML
95	car_evaluation	0.29	1728	22	0	22	0	target	multiclass	PMLB
96	mfeat-factors	3.311	2000	217	0	217	0	class	multiclass	AutoML
97	optdigits	2.787	5620	65	0	65	0	class	multiclass	OpenML
98	pendigits	1.426	10992	17	0	17	0	class	multiclass	OpenML
99	debutanizer	0.146	2394	8	0	8	0	y	regression	OpenML
100	witmer_census_1980	0.002	50	6	1	5	1	HSPerc	regression	OpenML
101	bng_breastTumor	8.899	116640	10	0	10	0	target	regression	PMLB
102	puma32H_308	2.063	8192	33	0	33	0	thetadd6	regression	OpenML
103	kin8nm_189	0.563	8192	9	0	9	0	y	regression	OpenML
104	poker	86.022	1025010	11	0	11	0	target	regression	PMLB
105	pollen_529	0.176	3848	6	0	6	0	DENSITY	regression	OpenML
106	bng_lowbwt	2.373	31104	10	0	10	0	target	regression	PMLB
107	bank32nh_558	2.063	8192	33	0	33	0	rej	regression	OpenML
108	space_ga_507	0.166	3107	7	0	7	0	ln(VOTES-/POP)	regression	OpenML
109	bng_echomonths	1.335	17496	10	0	10	0	target	regression	PMLB
110	house_16H	2.955	22784	17	0	17	0	target	regression	PMLB
111	mercedes-benz-greener-manufacturing	12.139	4209	378	8	370	0	y	regression	Kaggle
112	bng_pbc	144.959	1000000	19	0	19	0	target	regression	PMLB
113	bng_pwLinear	14.867	177147	11	0	11	0	target	regression	PMLB
114	puma8NH_225	0.563	8192	9	0	9	0	thetadd3	regression	OpenML
115	house_8L	1.565	22784	9	0	9	0	target	regression	PMLB
116	rainfall_bangladesh	0.511	16755	4	2	2	1	Rainfall	regression	OpenML
117	houses	1.417	20640	9	0	9	0	target	regression	PMLB
118	wind_503	0.752	6574	15	0	15	0	MAL	regression	OpenML
119	socmob	0.053	1156	6	4	2	2	counts_for-_sons_current-_occupation	regression	OpenML
120	bng_pharynx	83.923	1000000	11	0	11	0	target	regression	PMLB
121	sulfur	0.539	10081	7	0	7	0	y1	regression	OpenML
122	fried	3.422	40768	11	0	11	0	target	regression	PMLB
123	bank8FM	0.563	8192	9	0	9	0	rej	regression	OpenML
124	2dplanes	3.422	40768	11	0	11	0	target	regression	PMLB
125	weather_izmir	0.112	1461	10	0	10	0	Mean_temp-erature	regression	OpenML
126	stock	0.073	950	10	0	10	0	company10	regression	OpenML
127	cpu_small_227	0.813	8192	13	0	13	0	usr	regression	OpenML
128	cpu_act_573	1.375	8192	22	0	22	0	usr	regression	OpenML
129	pol	5.608	15000	49	0	49	0	target	regression	PMLB
130	mv	3.733	40768	12	0	12	0	target	regression	PMLB

Appendix B

Master's Coursework and Contributions

B.1 Master Coursework

Course	Course Code	Semester	Grade
DISTRIBUTED SYSTEM DESIGN	COMP 6231	Winter 2021	A
ADV. PROG. PRACTICES	SOEN 6441	Winter 2021	A-
FOUNDATIONS/SEMANTIC WEB	COMP 6531	Fall 2021	A+
BIG DATA ANALYTICS	SOEN 6111	Winter 2022	A+

B.2 Awards and Contributions

- Best Poster - A Feature Discovery Platform for Data Science Across the Enterprise, DSDS Workshop, 2022, Montreal, Canada.
- Best Poster - KGFarm: Automated Data Preparation using Semantics of Data Science Artifacts, VLDB Summer School, 2023, Cluj-Napoca, Romania.
- Helali, M., **Vashisth, S.**, Carrier, P., Hose, K. and **Mansour, E.**, 2023. Linked Data Science Powered by Knowledge Graphs* (to be submitted at SIGMOD 2024).

References

Abdallah, H., Nguyen, D., Nguyen, K., & Mansour, E. (2021). Demonstration of kgnet: a cognitive knowledge graph platform. In O. Seneviratne, C. Pesquita, J. Sequeda, & L. Etcheverry (Eds.), *Proceedings of the ISWC 2021 posters, demos and industry tracks: From novel ideas to industrial practice co-located with 20th international semantic web conference (ISWC 2021), virtual conference, october 24-28, 2021* (Vol. 2980). CEUR-WS.org. Retrieved from `https://ceur-ws.org/Vol-2980/paper311.pdf`

Alghushairy, O., Alsini, R., Soule, T., & Ma, X. (2021). A review of local outlier factor algorithms for outlier detection in big data streams. *Big Data and Cognitive Computing*, *5*(1). Retrieved from `https://www.mdpi.com/2504-2289/5/1/1` doi: 10.3390/bdcc5010001

Al-Rfou, R., Alain, G., Almahairi, A., Angermueller, C., Bahdanau, D., Ballas, N., ... others (2016). Theano: A python framework for fast computation of mathematical expressions. *arXiv e-prints*, arXiv–1605.

Alteryx. (2023). Retrieved from `https://www.alteryx.com/`

Bauckmann, J., Leser, U., Naumann, F., & Tietz, V. (2007). Efficiently detecting inclusion dependencies. In *2007 ieee 23rd international conference on data engineering* (p. 1448-1450). doi: 10.1109/ICDE.2007.369032

Bellamy, R. K. E., Dey, K., Hind, M., Hoffman, S. C., Houde, S., Kannan, K., ... Zhang, Y. (2018, October). *AI Fairness 360: An extensible toolkit for detecting, understanding, and mitigating unwanted algorithmic bias.* Retrieved from `https://arxiv.org/abs/1810.01943`

Biessmann, F., Rukat, T., Schmidt, P., & et al. (2019). Datawig: Missing value imputation for tables. *J. Mach. Learn. Res.*, *20*(175), 1–6.

Bogatu, A., Fernandes, A. A. A., Paton, N. W., & Konstantinou, N. (2020). Dataset discovery in data lakes. In *2020 ieee 36th international conference on data engineering (icde)* (p. 709-720). doi: 10.1109/ICDE48307.2020.00067

Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., … Zhang, Z. (2015). Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274*.

Dask Development Team. (2016). Dask: Library for dynamic task scheduling [Computer software manual]. Retrieved from https://dask.org

Dua, D., & Graff, C. (2017). *UCI machine learning repository.*

Einblick. (2023). Retrieved from https://www.einblick.ai/

Feast: Feature Store for Machine Learning. (2022). Retrieved from https://feast.dev/

Fernandez, R. C., Abedjan, Z., Koko, F., Yuan, G., Madden, S., & Stonebraker, M. (2018). Aurum: A data discovery system. In *34th IEEE international conference on data engineering, ICDE 2018, paris, france, april 16-19, 2018* (pp. 1001–1012). IEEE Computer Society. Retrieved from https://doi.org/10.1109/ICDE.2018.00094 doi: 10.1109/ICDE.2018.00094

Goikoetxea, J., Agirre, E., & Soroa, A. (2016). Single or multiple? combining word representations independently learned from text and wordnet. In *Proceedings of the thirtieth conference on artificial intelligence (AAAI)* (pp. 2608–2614). Retrieved from http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/11777

Hai, R., Kang, Y., Koutras, C., Ionescu, A., & Katsifodimos, A. (2022). Bridging the gap between data integration and ml systems. *arXiv preprint arXiv:2205.09681*.

Helal, A., Helali, M., Ammar, K., & Mansour, E. (2021). A demonstration of kglac: A data discovery and enrichment platform for data science. *Proceedings of the VLDB Endowment*, *14*(12), 2675–2678.

Helali, M., Mansour, E., Abdelaziz, I., & et al. (2022). A scalable AutoML approach based on graph neural networks. *PVLDB*, *15*(11).

Helali, M., Vashisth, S., Carrier, P., & et al. (2021). Linked data science powered by knowledge graphs. *CoRR*, *abs/2303.02204*.

Kakantousis, T., Kouzoupis, A., Buso, F., & et al. (2019). Horizontally scalable ml pipelines with a feature store. In *Sysml*.

Kanter, J. M., & Veeramachaneni, K. (2015). Deep feature synthesis: Towards automating data science endeavors. In *2015 ieee international conference on data science and advanced analytics (dsaa)* (p. 1-10). doi: 10.1109/DSAA.2015.7344858

Kasneci, E., Sessler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., ... Kasneci, G. (2023). Chatgpt for good? on opportunities and challenges of large language models for education. *Learning and Individual Differences*, *103*, 102274. Retrieved from https://www.sciencedirect.com/science/article/pii/S1041608023000195 doi: https://doi.org/10.1016/j.lindif.2023.102274

Katz, G., Shin, E. C. R., & Song, D. (2016). Explorekit: Automatic feature generation and selection. In *2016 ieee 16th international conference on data mining (icdm)* (p. 979-984). doi: 10.1109/ICDM.2016.0123

Kaul, A., Maheshwary, S., & Pudi, V. (2017). Autolearn - automated feature generation and selection. In *Icdm* (pp. 217–226).

Khatiwada, A., Fan, G., Shraga, R., Chen, Z., Gatterbauer, W., Miller, R. J., & Riedewald, M. (2023, may). Santos: Relationship-based semantic table union search. *Proc. ACM Manag. Data*, *1*(1). Retrieved from https://doi.org/10.1145/3588689 doi: 10.1145/3588689

Kumar, V., & Minz, S. (2014). Feature selection: a literature review. *SmartCR*, *4*(3), 211–229.

Lam, H. T., Thiebaut, J.-M., Sinn, M., Chen, B., Mai, T., & Alkan, O. (2017). One button machine for automating feature engineering in relational databases. *arXiv preprint arXiv:1706.00327*.

Mansour, E., Srinivas, K., & Hose, K. (2021). Federated data science to break down silos [vision]. *SIGMOD Rec.*, *50*(4).

Mueller, J., & Smola, A. (2019). Recognizing variables from their data via deep embeddings of distributions. In *International conference on data mining (ICDM)* (pp. 1264–1269).

Nargesian, F., Asudeh, A., & Jagadish, H. V. (2021, jul). Tailoring data source distributions for fairness-aware data integration. *Proc. VLDB Endow.*, *14*(11), 2519–2532. Retrieved from https://doi.org/10.14778/3476249.3476299 doi: 10.14778/3476249.3476299

Nargesian, F., Samulowitz, H., Khurana, U., & et al. (2017). Learning feature engineering for classification. In *Ijcai.*

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., … Lerer, A. (2017). Automatic differentiation in pytorch.

Peng, J., Wu, W., Lockhart, B., & et al. (2021). Dataprep.eda: Task-centric exploratory data analysis for statistical modeling in python. In *Sigmod* (pp. 2271–2280).

Raju, V. G., Lakshmi, K. P., Jain, V. M., Kalidindi, A., & Padma, V. (2020). Study the influence of normalization/transformation process on the accuracy of supervised classification. In *2020 third international conference on smart systems and inventive technology (icssit)* (pp. 729–735).

Raju, V. N. G., Lakshmi, K. P., Jain, V. M., Kalidindi, A., & Padma, V. (2020). Study the influence of normalization/transformation process on the accuracy of supervised classification. In *Icssit* (p. 729-735).

Rekatsinas, T., Chu, X., Ilyas, I. F., & et al. (2017). Holoclean: Holistic data repairs with probabilistic inference. *PVLDB*, *10*(11).

Rezig, E. K., Bhandari, A., Fariha, A., & et al. (2021). DICE: data discovery by example. *PVLDB*, *14*(12).

Rostin, A., Albrecht, O., Bauckmann, J., Naumann, F., & Leser, U. (2009). A machine learning approach to foreign key discovery. In *12th international workshop on the web and databases, webdb 2009, providence, rhode island, usa, june 28, 2009.*

Samala, R. K., Chan, H.-P., Hadjiiski, L., & Koneru, S. (2020). Hazards of data leakage in machine learning: a study on classification of breast cancer using deep neural networks. In *Medical imaging 2020: Computer-aided diagnosis* (Vol. 11314, pp. 279–284).

Tensorflow: Large-scale machine learning on heterogeneous systems. (n.d.). Retrieved from https://www.tensorflow.org/ (Software available from tensorflow.org)

Trifacta. (2023). Retrieved from https://www.trifacta.com/

van Buuren, S., & Groothuis-Oudshoorn, K. (2011). mice: Multivariate imputation by chained equations in r. *Journal of Statistical Software*, *45*(3), 1–67. Retrieved from https://www.jstatsoft.org/index.php/jss/article/view/v045i03 doi: 10.18637/jss

.v045.i03

Waring, J., Lindvall, C., & Umeton, R. (2020). Automated machine learning: Review of the state-of-the-art and opportunities for healthcare. *Artificial Intelligence in Medicine*, *104*, 101822. Retrieved from https://www.sciencedirect.com/science/article/pii/S0933365719310437 doi: https://doi.org/10.1016/j.artmed.2020.101822

Xu, S., Lu, B., Baldea, M., Edgar, T. F., Wojsznis, W., Blevins, T., & Nixon, M. (2015). Data cleaning in the process industries. *Reviews in Chemical Engineering*, *31*(5), 453–490. Retrieved 2023-07-11, from https://doi.org/10.1515/revce-2015-0022 doi: doi:10.1515/revce-2015-0022

Yan, C., & He, Y. (2020). Auto-Suggest: Learning-to-recommend data preparation steps using data science notebooks. In *SIGMOD* (pp. 1539–1554).

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., ... Sun, M. (2020). Graph neural networks: A review of methods and applications. *AI Open*, *1*, 57-81. Retrieved from https://www.sciencedirect.com/science/article/pii/S2666651021000012 doi: https://doi.org/10.1016/j.aiopen.2021.01.001