JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

GenAI_ModelHub: Generative AI powered Data Science Automation platform.

Mr. S. D. Kale[1], Ajinkya Temgire[2], Digvijay Mane[3], Arpita Pol[4], Priyanka Patil[5]

[1,2,3,4,5] *Artificial Intelligence and Data Science Department, All India Shri Shivaji Memorial Society's Institute of Information Technology, Pune, Maharashtra, India*

***Abstract*— The "GenAI-ModelHub" project is intended to dramatically streamline and automate the major parts of the data science workflows, offering highly powerful tools that accelerate complex tasks using the best AI technologies available. It will primarily target issues related to querying, model building, and optimization while integrating the SQL and Pandas data manipulation processes. It simplifies hyperparameter tuning and optimizes deep learning architectures for both novice and expert users by automating the creation of baseline models. At the heart of GenAI-ModelHub's functionality are Large Language Models1 (LLMs) and Retrieval-Augmented Generation2 (RAG). These technologies power an intelligent, interactive chatbot that can assist users in real time, automate data preprocessing steps, and generate code for model training—all tailored to the specific needs and data of the user. This approach helps users quickly move from data querying to model training, saving time and reducing the technical expertise required for complex configurations.**

Index Terms* — *Large Language Models[1], Retrieval-Augmented Generation[2].

I. INTRODUCTION

In this data-driven environment, organizations as well as individuals are being flooded with information coming from various types and sources of data. With growth comes opportunities and challenges, and although data is essential in making informed decisions, the intricacies of large datasets pose enormous challenges. Manipulation, transformation, and building models of data are important in obtaining meaningful insights. However, these processes frequently require knowledge of several tools and programming languages along with considerable investments in time and expertise, mainly for activities like data querying, model generation, hyperparameter tuning, and optimizing deep learning models.

SQL and Pandas are two of the most frequently used frameworks that have been adopted for data manipulation, and these two frameworks hold different advantages. SQL is the language of powerful command and query structurally in relation to data base systems, and Pandas are libraries in python language that really shines in analysing data, specifically with the realm of machine learning. Even so, it presents an inconvenient challenge between using SQL and Pandas since there is an incompatibility if one doesn't know either one of these tools, hindering effective analysis. Moreover, even the expert in short, practitioners have a hard time navigating the maze of modern workflows in machine learning, which entail successive stages of experimenting, parameter fine-tuning, and optimizing models.

An integrated platform would be proposed through this address this challenge through the combination of four-stage solutions: bridging SQL and Pandas query, automatic baseline model generation, hyperparameter tuning, and deep learning optimization. This aims at delivering a seamless data science pipeline by leveraging state-of-the-art technologies, which range from large language models (LLMs) and Retrieval-Augmented Generation (RAG) to interactive user interfaces, to make the available essential tasks in the pipeline both streamlined and automated. Each module has been designed with an end user in mind, providing clear, intuitive tools that let users, be they novice or expert, perform sophisticated data transformations, model generation, and optimizations.

The first module, SQL and Pandas query bridging, allows users to input a query in either SQL or Pandas and receive the equivalent code in the other format. This feature is very helpful for people who may be fluent in SQL but not familiar with Pandas, or vice versa. By integrating a chatbot powered by LLMs with RAG, this module also allows the online support system for real-time questions and helps with query formulation and interpretation. Such support might make the difference between a longer learning curve about data manipulation in one framework, making it much easier to jump between frameworks.

II. LITERATURE SURVEY

The rapid advancements in artificial intelligence and machine learning have led to significant developments in areas such as natural language processing, code generation, hyperparameter optimization and query translation. In particular, transformer-based architectures, optimization algorithms, and neural machine translation models are crucial in addressing complex data science problems. This literature review explores recent studies that have made substantial contributions to these fields, focusing on methodologies, findings, and applications relevant to machine learning and automated processes.

One notable work, Application of Noise Filter Mechanism for T5-Based Text-to-SQL Generation (2023), by M.R. Aadhil Rushdy and Uthayasanker Thayasivam, presents a Seq2Seq-based transformer model using the T5 architecture. This model aims to address the challenge of generating SQL queries from textual input, which is mostly useful in environments where users without SQL expertise need to query databases. Noise filtering, is an addition that enhances the encoding and decoding stages, reaching an exact match of 72.7% and execution accuracy of 80.2% on the Spider dataset, used as a benchmark for text-to-SQL tasks. This research underlines the importance of robust encoding-decoding mechanisms in improving model performance in structured query generation.

Furthering the understanding of optimization techniques, A Survey on Hyperparameter Optimization of Machine Learning Models (2024), by Monica and Parul Agrawal, examines several approaches to hyperparameter tuning, including grid search, random search, Bayesian optimization, genetic algorithms, and evolutionary algorithms. Hyperparameter optimization is crucial for improving model accuracy and generalization. This study illustrates that Bayesian optimization along with more recent meta-heuristics may supersede the other methods by such a significant performance margin. With this background, this survey contributes to the ongoing need for efficiency and scalability of hyperparameter tuning strategies that increasingly support the execution of complex machine learning models.

In code generation, Rutuja Nikum, Vaishnavi Shinde, and Vijay Khadse designed Textual Query Translation into Python Source Code using Transformers (2022), which uses NMT to translate English queries into Python code. This research used a transformer-based architecture and reached a BLEU score of 0.78, indicating the effectiveness of its encoder-decoder model in translating queries into executable code. This model not only enhances its accuracy through data fusion and retraining but also incorporates a Flask-based user interface to improve accessibility and the user experience. This study is a prime example of how NMT and transformer models can bridge the gap between natural languages and programming languages and will help both novice and expert users in their code generation tasks.

Le et al. (2011) analyze the performance of several optimization algorithms in deep learning, focusing on the comparison between Limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) and Conjugate Gradient (CG) vs. Stochastic Gradient Descent (SGD). It is shown that L-BFGS and CG are superior to SGD in specific settings, specifically in low-dimensional problems like convolutional neural networks and sparsity-related tasks like sparse autoencoders.

Remarkably, L-BFGS was able to obtain a state-of-the-art error rate of 0.69% on the MNIST dataset without distortions or pretraining. This paper shows that the commonly used SGD should not be considered as the gold standard for all optimization methods without considering the context.

Sun et al. (2023) present an enhanced large language model called SQL-PaLM to improve Text-to-SQL accuracy. The model provides better performance for Spider and BIRD benchmarks due to diversified training data, use of relevant content in the databases, and reducing the size of the input via column selection, among other input sizes. Refining at test time also aids accuracy. This work, therefore, gives insights into the strengths of SQL-PaLM and its applicability to real-world scenarios. It demonstrates the potential to improve model adaptation when it comes to Text-to-SQL tasks.

Osman, Ghafari, and Nierstrasz (2020) present how hyperparameter tuning influences the accuracy of machine learning models in bug prediction. Their experiments show that hyperparameter tuning greatly improves the prediction accuracy of the Instance- Based k (IBK) algorithm and either improves or at least does not degrade Support Vector Machines (SVM). The study highlights the fact that default hyperparameters are usually suboptimal; it recommends that hyperparameter tuning be performed as an essential step in the machine learning process to improve bug prediction accuracy. Khadka et al. (2020) provide a combinatorial hyperparameter optimization (HPO) strategy in which an algorithm reduces the need for evaluating compared to traditional methods such as Grid Search, Random Search, and Bayesian Optimization. This method's execution gets similar or even superior results with fewer runs, making it worthwhile for large datasets and complicated models. The study shows that t-way testing is an efficient and effective alternative to resource-intensive HPO techniques, with significant benefits in terms of computational resources and time.

Lastly, An Empirical Study of Code Smells in Transformers-based Code Generation Techniques (2022), by Mohammad Latif Siddiq, Shafayat H. Mujumder, Maisha R. Mim, Sourav Jajodia, and Joanna C.S. Santos, studies the quality of code generated by transformers, particularly GPT-Neo and GitHub Copilot. The research assesses the occurrence of code smells and security bugs utilizing tools such as Pylint and Bandit by applying the method to transformer models and identifying how such problems have a tendency of arising within generated code from models. It demonstrates that these model-based datasets, at a higher frequency contain pre-existing errors that propagate during code generation. The need for quality control in refinement within the context of transformers code generation, particularly in light of the risks involved in deploying automatically generated code into production environments.

III. RESEARCH METHODOLOGY

A. System Architecture

The architecture of GenAI-ModelHub is structured as a modular and scalable platform designed to ensure security, performance, and usability across multiple data science automation tasks. The system comprises a Frontend Layer, a Flask Backend, and four specialized modules that provide distinct functionalities: Automated Baseline Model Generation, SQL ↔ Pandas Query Generation, Hyperparameter Tuning, and Deep Learning Optimization.

1 Frontend Layer

The frontend is bifurcated based on functional domains:

- HTML/CSS/JavaScript is used to implement the UI for the Baseline Model Generator and Query Module to provide a traditional web experience.
- Streamlit, a lightweight Python-based UI framework, is employed for the Hyperparameter Tuning and Deep Learning Optimization modules due to its suitability for rapid dashboarding and model interaction.

These frontend components interact with users through file uploads, task selections, and interactive elements without storing any sensitive data locally or remotely.

2 Flask Backend

A central Flask-based backend manages the entire request-response cycle. Its responsibilities include:

- Routing user inputs from the frontend to appropriate modules.
- Managing APIs and internal communications.
- Ensuring seamless integration of all four feature modules.
- Strictly following a no data storage policy to ensure user data confidentiality and eliminate risks of data leakage.

3 Functional Modules

3.1 Query Module (SQL ↔ Pandas Conversion)

This module serves as a dual-way query translator between SQL and Pandas. Users can upload one or more datasets and pose natural language questions about the data. The system uses the **Gemini Pro** LLM to convert the questions into both SQL and Pandas query forms, accompanied by descriptive explanations. This bridges the knowledge gap for professionals proficient in one domain but needing support in the other, streamlining data exploration across tools.

3.2 Baseline Model Generator

This module is designed to rapidly produce baseline machine learning models using a domain-specific logic-based system. Rather than relying on general-purpose generative models, conditional logic is applied to select preprocessing techniques and algorithms (e.g., Random Forest, Gradient Boosting) based on dataset properties. It also includes Pandas Profiling for data insights and visualization. Importantly, this module operates without storing data, enabling organizations to securely bootstrap models for exploration and benchmarking.

3.3 Hyperparameter Tuning Module

This no-code module enables users to upload pre-cleaned datasets, choose between classification or regression tasks, and experiment with hyperparameter tuning for a variety of algorithms. The UI, developed in Streamlit, allows for real-time selection and adjustment of parameters. Performance metrics are visualized graphically to assist in comparative analysis. This hands-on yet code-free environment reduces the technical barrier for optimal model configuration.

3.4 Deep Learning Optimization Module

Focused on computer vision tasks, this module accepts image datasets via Google Drive links. Data is downloaded to a temporary local directory and processed using Bayesian optimization via the Optuna library. Multiple CNN architectures are auto-generated and evaluated, and the best-performing architecture is provided along with complete exportable code. Visualizations of accuracy and model performance aid in selection. This facilitates optimal CNN design with reduced computational cost, offering a practical alternative to transformer-based models for image data.

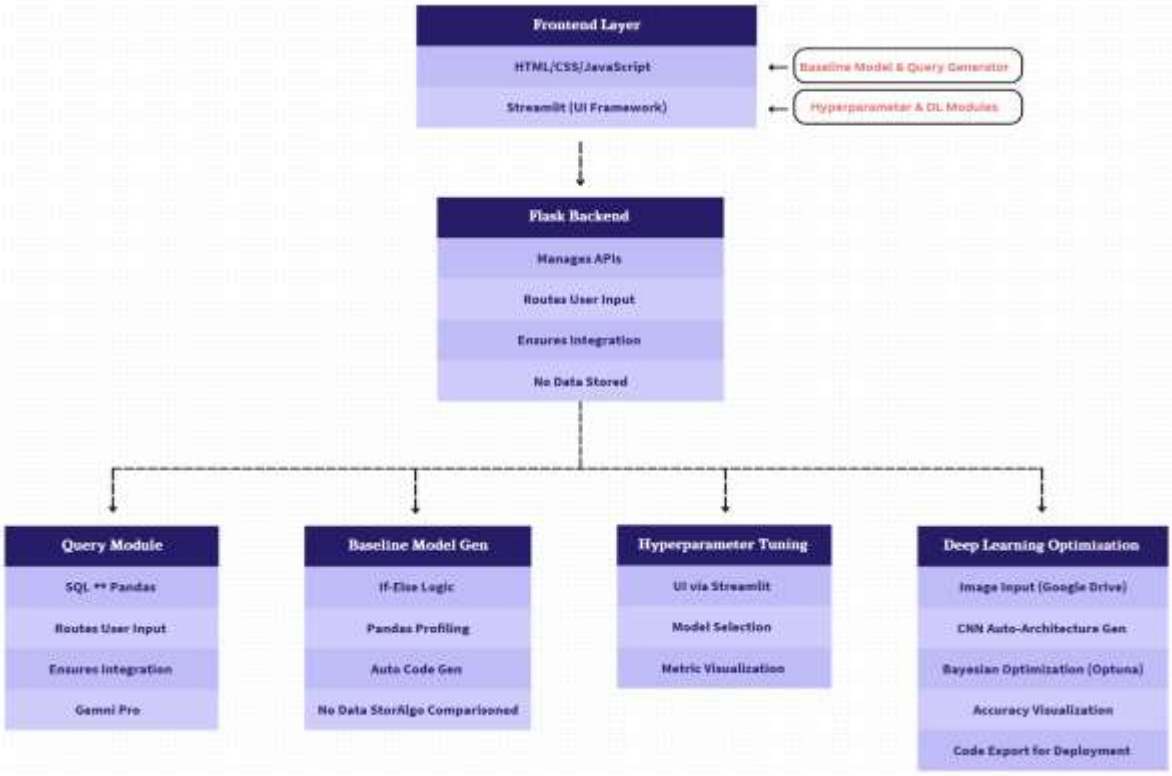

Figure 1: System Architecture

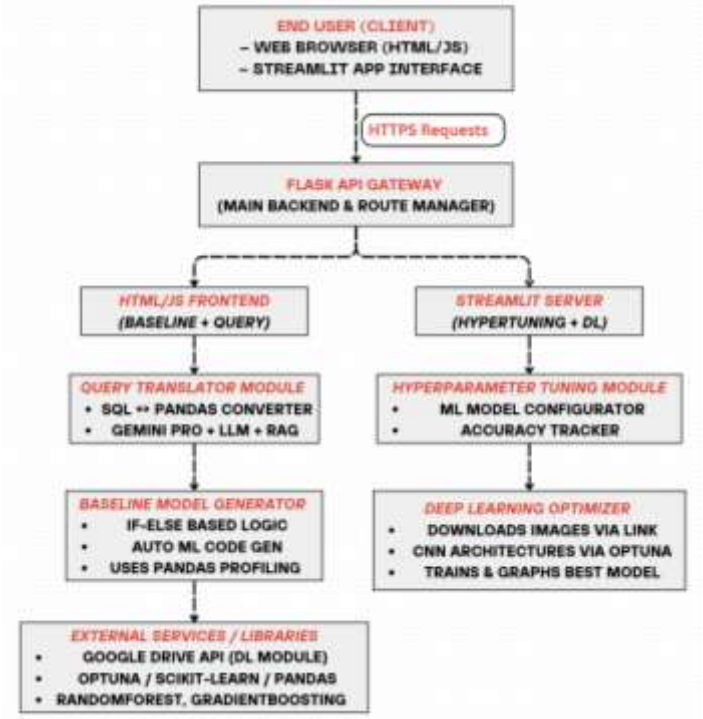

Figure 2: Network Communication Diagram

B. Data Flow Pipeline

1 Client Interaction and Routing

The pipeline begins with the user (client) selecting one of four major functionalities from the homepage interface:

1. Baseline Model Creation
2. Query Translator (SQL ↔ Pandas)
3. Hyperparameter Tuning
4. Deep Learning Optimization

This selection is routed via a centralized Flask-based controller that invokes the relevant user interface and backend workflow. The choice of UI framework depends on the module: HTML/CSS for the first two modules, and Streamlit for the latter two due to their dynamic data interaction needs.

2 Module-Specific Data Flow

2.1 Baseline Model Generator

- Input: Uncleaned tabular data uploaded via the HTML interface.
- Processing:
  - The Flask backend applies rule-based logic (if-else conditions) for data cleaning, preprocessing, and model code generation based on domain-informed rules.
  - Pandas Profiling is integrated to provide a summary of the dataset, including distributions and missing values.
  - Suitable models such as Random Forest or Gradient Boosting are automatically selected and trained.
- Output: Cleaned data statistics, code snippets, model summaries, and visualization results are rendered on the output HTML page.

2.2 Query Translator

- **Input**: One or more data files, accompanied by a natural language question submitted through the HTML interface.
- **Processing**:
  - The backend leverages the **Gemini Pro** LLM to interpret the question and generate equivalent SQL and Pandas queries.
  - It includes a brief natural language explanation of the logic.
- **Output**: Generated queries (SQL + Pandas) and their descriptions are displayed in the output HTML page, allowing users to execute or learn from the translations.

2.3 Hyperparameter Tuner

- **Input**: Cleaned dataset uploaded via the Streamlit UI.
- **Processing**:
  - Users select between regression or classification tasks.
  - Based on the selected task, multiple algorithms and their respective hyperparameters are exposed for tuning.
  - Real-time adjustments trigger backend model training using Flask.
- **Output**: Accuracy metrics and performance comparisons are presented on the Streamlit interface, allowing users to make informed tuning decisions without any coding.

2.4 Deep Learning Optimizer

- **Input**: Google Drive link to a folder containing structured image data, submitted via Streamlit.
- **Processing**:
  - Images are temporarily downloaded to the server.
  - **Bayesian optimization** via the **Optuna** library is employed to generate and evaluate multiple CNN architectures.
  - Models are trained for limited epochs, and their metrics are logged.
- **Output**:
  - Best-performing CNN model (with .h5 file and generated code) is returned.
  - Graphical metrics, including accuracy plots and optimization charts, are rendered for selection support.

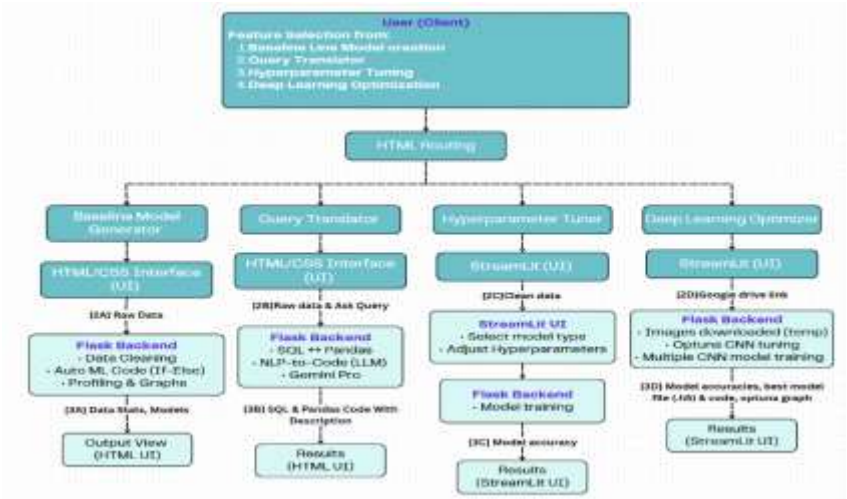

Figure 3: Data Flow Diagram

3 Security and Isolation

At every step in the pipeline, **data isolation** is maintained. No user data is persisted on the backend servers. Temporary data (such as image folders in the Deep Learning module) is deleted post-processing. All file uploads are processed in memory or transient storage.

4 Backend Integration

Despite the differing frontend technologies, all modules are tightly coupled to the **Flask backend**, which acts as a central orchestrator. This allows consistent routing, user session handling, and modular extension, providing the scalability required for further research or enterprise integration.

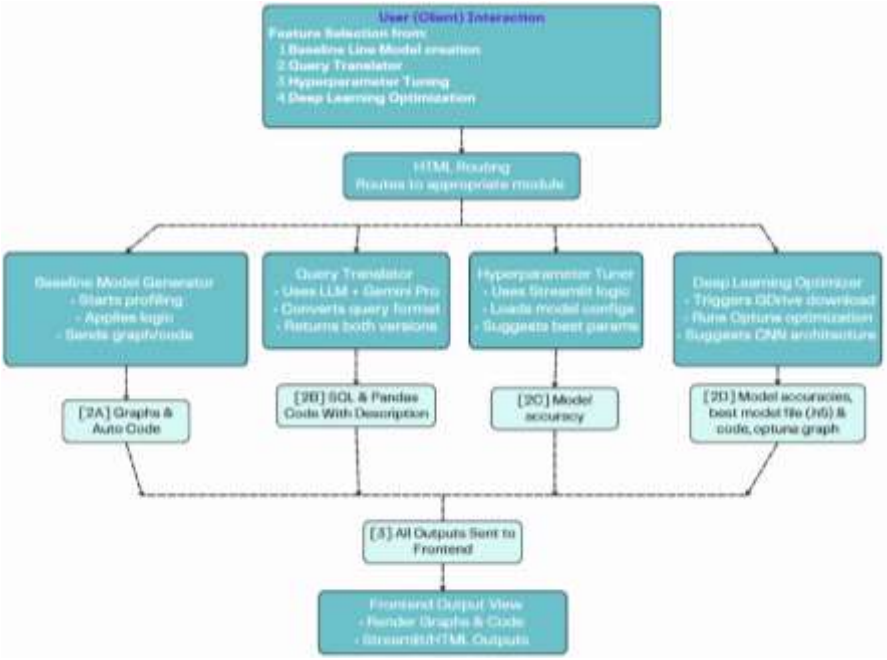

Figure 4: Real-Time Processing Diagram

C. *Tools and Models Used*

The development of **GenAI-Modelisub** required a deliberate selection of tools and machine learning frameworks that support both scalability and security, while ensuring ease of use for non-programmers. The platform is composed of four core functional modules, each tailored to a specific aspect of data science automation. The toolchain and modeling approaches are detailed below:

1 Backend Framework

The entire backend of the application is developed using the Flask micro web framework (Python), which serves as the central orchestrator for routing requests, handling file uploads, invoking logic modules, and rendering responses. Its lightweight and modular nature made it an ideal choice for supporting both traditional HTML/CSS frontends and modern Streamlit interfaces. Importantly, Flask also enables transient data handling, ensuring that no user data is stored, which enhances the platform's compliance with privacy and security standards.

2 Frontend Technologies

The project utilizes two parallel frontend stacks:

- HTML/CSS/JavaScript for the Baseline Model Generator and Query Translator modules, offering a conventional web experience and flexibility for visual outputs.
- Streamlit, an open-source Python-based frontend framework, is employed for the Hyperparameter Tuner and Deep Learning Optimizer modules. Streamlit allows rapid creation of interactive and reactive user interfaces without the need for HTML or JS knowledge, making it ideal for user-driven model adjustments.

3 Libraries and Techniques

- Pandas Profiling: Integrated into the Baseline Model Generator, this library performs automated exploratory data analysis (EDA), generating statistical summaries, correlations, and visualizations from uploaded tabular data.
- Rule-Based Code Generation (If-Else Logic): Instead of relying on generative AI models that may compromise sensitive enterprise data, custom if-else logic trees were crafted using deep domain knowledge to support automatic model building and preprocessing decisions.
- Popular Machine Learning Models:
    - For baseline modeling, algorithms such as Random Forest, Gradient Boosting, and other scikit-learn-supported classifiers/regressors are applied based on the data characteristics.
    - In the Hyperparameter Tuner, users can manually adjust hyperparameters of models like Logistic Regression, SVM, KNN, XGBoost, and more to identify the best fit using accuracy or other metrics.

4 Language Models and Query Generation

Gemini Pro (LLM): Integrated in the Query Translator, Gemini Pro enables natural language to code translation. It interprets user questions and generates syntactically correct and logically accurate queries in both SQL and Pandas, accompanied by brief explanatory notes. This dual-code output ensures accessibility for users proficient in either or both formats, addressing a common challenge in real-world data analysis environments.

5 Deep Learning and Optimization

Optuna Framework: Utilized in the Deep Learning Optimizer, Optuna performs Bayesian optimization for automated architecture search. By sampling combinations of hyperparameters and layer configurations (e.g., convolution filters, pooling strategies, stride lengths), it builds and evaluates multiple CNN models over a limited number of epochs.
Keras/TensorFlow: Used for defining and training CNN models. The final output includes the best model (in .h5 format), corresponding Python code, and training history graphs to support informed model adoption.

6 Cloud Integration

Google Drive Link Input: To facilitate large-scale image dataset input for CNN optimization, users are required to provide a link to a structured image folder in Google Drive. The system temporarily downloads the data, processes it securely, and deletes it post-processing to maintain data isolation.

Together, these tools and modeling techniques provide a comprehensive, secure, and user-friendly platform that empowers domain experts and data scientists alike to perform automated analysis, query generation, hyperparameter tuning, and model optimization—**without writing a single line of code**. The careful avoidance of persistent storage and minimal reliance on generative models (except where explainability is required) ensures that **GenAI-ModelHub** is both enterprise-ready and research-grade.

IV. Results and Discussion

A. Improved Query Translation: Using an advanced text-to-SQL and Pandas translation module, we should be able to create a smooth bridge between user inputs and SQL/Pandas queries. The model will then correctly interpret natural language questions and generate accurate, efficient code output in both SQL and Pandas formats. This chatbot will also make the tool more accessible to users who can learn and perfect their query-building process through interaction.

B. Automatic Baseline Model Generation This module aims at streamlining the initial model development phase by providing a clear and efficient baseline. Expected results include better starting points for model training with readily available accuracy benchmarks across several algorithms. Accompanying statistical inferences, graphical representations, and R code will offer useful insights, thus enabling users to make informed decisions in further model tuning and development.

C. Refined Hyperparameter Tuning Process: We would expect improved model accuracy and efficiency by using the hyperparameter tuning feature, which includes a user-friendly interface for parameter se-lection. It should help the users to find a good bias-variance trade-off and therefore improve performance through identification of appropriate hyperparameters that are based on detailed descriptions and recommendations.

D. Optimized Deep Learning Model Architectures: In terms of the optimization of deep learning model architecture, the system should be able to automatically propose the best settings in terms of number of layers, filter size, stride size, and any other parameter of the data used. This will be in more accurate and less computationally complex models. The users can further make adjustment and see effects in real-time, thus seeing the structure and model's performance

V. Conclusion

This project is an example of how automation in machine learning transforms the availability and efficiency of more advanced tools to users at any level of expertise. The integration of natural language query translation to SQL and Pandas, combined with real-time support through a chatbot, helps bridge technical knowledge gaps and makes it more usable. The automatic generation of the baseline model requires shorter time periods to establish a solid foundation for iterative improvement while the interactive module for hyperparameter tuning powers the user to optimize models easily and understand bias-variance trade-offs. Further, deep learning optimization shrinks the cycle of model building with effort by guiding users towards the optimal configuration thereby expediting easy selection since it diminishes trial and error, especially on complicated data. Together, these features contribute not only to more intuitive workflow for data science but also pave the way for more broad adoption of AutoML. These show how automation may democratize the capability to apply machine learning and simplify the process for industry, research, and education-again, opening interesting roads for further development. Future versions could take these features further by adding multi-language support and API integrations to extend the impact of the tool across various domains.

VI. Acknowledgment

We, Prof. Satish Kale, Ajinkya Temgire, Digvijay Mane, Arpita Pol and Priyanka Patil, would like to express our heartfelt gratitude to the Department of Artificial Intelligence and Data Science, AISSMS IOIT, Pune, for their continuous support, guidance, and for providing the infrastructure that made this research possible. We are also thankful to our friends, mentors, and families for their constant encouragement and motivation throughout the project. Additionally, we acknowledge the open-source communities

behind Gemini, Tensor Flow, Optuna and the Pandas (Bamboo LLM), whose contributions played a vital role in enabling the development of our real-time, GenAI_ModelHub: Generative AI powered Data Science Automation platform.

REFERENCES

[1] Application of Noise Filter Mechanism for T5-Based Text-to-SQL Generation by M.R. Aadhil Rushdy1, Uthayasanker Thayasivam2. 1Department of Computer Science and Engineering University of Moratuwa Katubedda, Sri Lanka, 2Department of Computer Science and Engineering University of Moratuwa Katubedda, Sri Lanka.

[2] A Survey on Hyperparameter Optimization of Machine Learning Models by Monica (Department of CSE&IT Jaypee Institute of Information Technology Noida, India monica.batra88@gmail.com), Parul Agrawal (Department of CSE&IT Jaypee Institute of Information Technology Noida, India parul.agarwal@ jiit.ac.in).S. Zhang, C. Zhu, J. K. O. Sin, and P. K. T. Mok, "A novel ultrathin elevated channel low-temperature poly-Si TFT," IEEE Electron Device Lett., vol. 20, pp. 569–571, Nov. 1999.

[3] Textual Query Translation into Python Source Code using Transformers by Rutuja Nikum1, Vaishnavi Shinde2, Vijay Khadse3. 1,2,3Computer Engineering College of Engineering Pune, India.

[4] An Empirical Study of Code Smells in Transformer- based Code Generation Techniques by Mohammed Latif Siddiq1 , Shafayat H. Majumder2 , Maisha R. Mim2 , Sourov Jajodia2 , Joanna C. S. Santos1, 1Department of Computer Science and Engineering, University of Notre Dame, USA, 2Department of Computer Science, Bangladesh University of Engineering and Technology, Dhaka, Bangladesh.

[5] A Combinatorial Approach to Hyperparameter Optimization by Krishna Khadka1, Jaganmohan Chandrasekaran2, Yu Lei3, Raghu N. Kacker4, D. Richard Kuhn5. 1,3 University of Texas at Arlington, Arlington, TX, USA. 2National Security Institute, Virginia Tech Arlington, VA, USA. 4,5Information Technology Laboratory, National Institute of Standards and Technology.

[6] Hyperparameter Optimization to Improve Bug Prediction Accuracy by Haidar Osman, Mohammad Ghafari, and Oscar Nierstrasz Software Composition Group, University of Bern, Bern, Switzerland.

[7] Conversion of Natural Language Query to SQL Query by Abhilasha Kate1, Satish Kamble2, Aishwarya Bodkhe3, Mrunal Joshi4. 1,2,3,4Dept. of IT Engineering PVG's COET, Pune, India.

[8] On Optimization Methods for Deep Learning by Quoc V. Le1, Jiquan Ngiam, Adam Coates, Abhik Lahiri, Bobby Prochnow, Andrew Y. Ng. Computer Science Department, Stanford University, Stanford, CA 94305, USA.