Volume: 13 Issue: V Month of publication: May 2025
DOI: https://doi.org/10.22214/ijraset.2025.70966

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue V May 2025- Available at www.ijraset.com

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.

Keywords: 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 analyzing 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. The second module addresses the usual problem of generation of baseline models. The main purpose of baseline modeling in machine learning is to establish a innovative model with which any more complex models can be compared and contrasted. However, this phase is usually overlooked by practitioners who only look for satisfactory accuracy at the very outset. By integrating our platform, one can rapidly and automatically preprocess raw data for generating baselines with multiple models, accuracy metrics, statistical inferences, and even visualizations.

Additionally, it generates R code for replicating the analysis, enhancing transparency and reproducibility.
Hyperparameter tuning, the third module, is an essential yet time-consuming process in model optimization. Selecting the right combination of hyperparameters can significantly impact model performance, yet finding the optimal settings typically requires multiple iterations and extensive domain knowledge. This streamlines the process by providing an interactive UI to select models and adjust hyperparameters, with descriptions explaining each parameter's effect on bias and variance. A chatbot is also available for support, answering questions related to hyperparameter selection and bias-variance trade-offs. The platform will also recommend the optimal hyperparameters based on dataset characteristics, making it easier to achieve better model accuracy with fewer trial and error attempts.
The last module deals with deep learning optimization, and particularly, the tuning of neural network architectures. Convolutional neural networks (CNNs) have been shown to be effective for deep learning, but careful parameter selection, including the number of convolutional layers, pooling layers, strides, and filters, can be very cumbersome. Usually, the optimal architecture is found by a process that is resource-consuming and cumbersome through extensive experimentation. Our platform provides an automatic solution, suggesting optimal configurations based on input data characteristics, and a user interface for the users to experiment with architectural changes. In sum, by automating much of the trial-and-error process of experimentation, this module helps the users build more efficient high-performing deep learning models.

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.

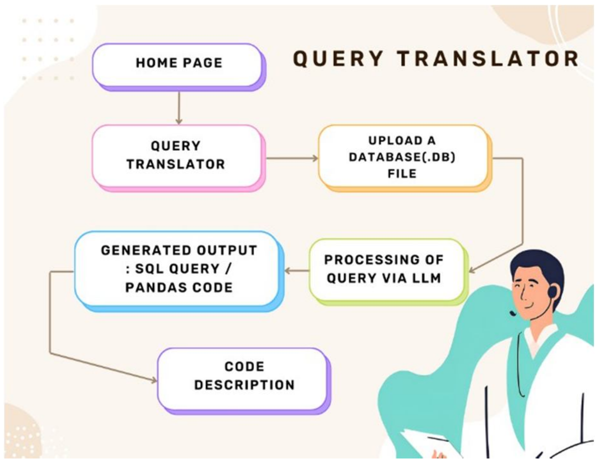
III. METHODOLOGY

Our platform consists of four key modules, each designed to address specific needs within the data science workflow. The architecture of this system leverages both generative AI and traditional machine learning methods, offering users a seamless experience for data manipulation, model training, and optimization.

A. SQL and Pandas Query Bridging

The first module connects the dots between SQL and Pandas as being among the most extensively used tools for data manipulation. A user can input a query, and our system will be able to produce the equivalent in both SQL and Pandas with explanations. A LLM and RAG-powered chatbot is also available to answer questions from users and advise on their way forward. This is a powerful capability for SQL-savvy users who aren't as comfortable with Pandas and vice versa; it is a good stepping point for people migrating between different data manipulation frameworks.

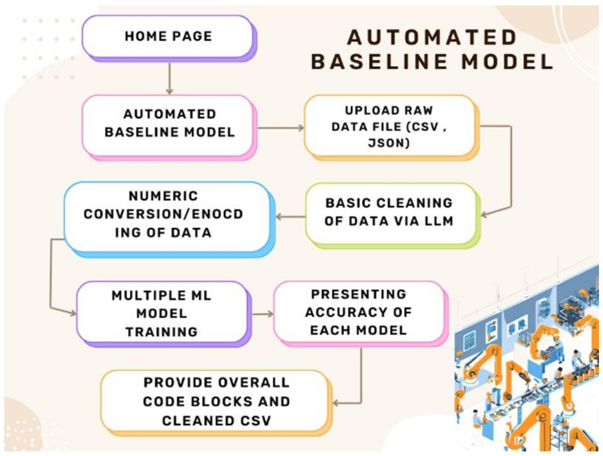
Fig 1: Flowchart for Query Translator Module


QUERY TRANSLATOR
HOME PAGE
QUERY TRANSLATOR
UPLOAD A DATABASE(.DB) FILE
PROCESSING OF QUERY VIA LLM
GENERATED OUTPUT : SQL QUERY / PANDAS CODE
CODE DESCRIPTION

B. Automatic Baseline Model Generation

This module is going to help the user in setting a baseline for models. Once it gets raw data from the user, the system pre-processes the data and transforms it into different numerical formats for training multiple models so that comparison can be made of their respective baseline performance. In addition to providing accuracy metrics, the platform also generates R code for the reproduction of analysis and visualizations of statistical summaries and data insights. This way, it allows users to be well grounded while building on solid and reproducible-benchmarks.

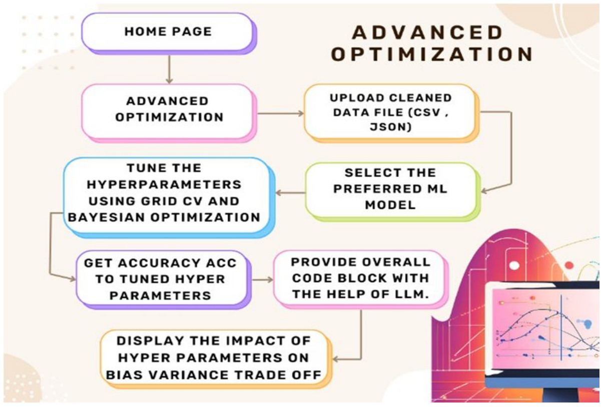
Fig 2: Flowchart for Automated Baseline Model


AUTOMATED BASELINE MODEL
HOME PAGE
AUTOMATED BASELINE MODEL
UPLOAD RAW DATA FILE (CSV , JSON)
BASIC CLEANING OF DATA VIA LLM
NUMERIC CONVERSION/ENOCD ING OF DATA
MULTIPLE ML MODEL TRAINING
PRESENTING ACCURACY OF EACH MODEL
PROVIDE OVERALL CODE BLOCKS AND CLEANED CSV

C. Hyperparameter Tuning

This tuning is often biased and variant, so there is a compromise between the bias and variance in our platform. Users can upload a cleaned dataset, select the model, and explore hyperparameter options through an interactive UI. Descriptions of each hyperparameter's impact on bias-variance trade-off are provided for users to help them make decisions. A chatbot is provided for further assistance, and the system recommends the optimal hyperparameter values based on the characteristics of the dataset. The platform also makes code snippets in order to understand and apply the tuning process in a self-supportive manner.

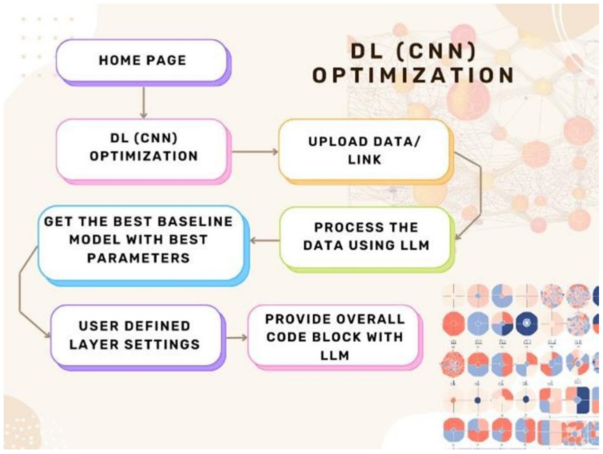
Fig 3: Flowchart for Hyperparameter Tuning Module


ADVANCED OPTIMIZATION
HOME PAGE
ADVANCED OPTIMIZATION
UPLOAD CLEANED DATA FILE (CSV , JSON)
SELECT THE PREFERRED ML MODEL
TUNE THE HYPERPARAMETERS USING GRID CV AND BAYESIAN OPTIMIZATION
GET ACCURACY ACC TO TUNED HYPER PARAMETERS
PROVIDE OVERALL CODE BLOCK WITH THE HELP OF LLM.
DISPLAY THE IMPACT OF HYPER PARAMETERS ON BIAS VARIANCE TRADE OFF

D. Deep Learning Optimization

For deep learning practitioners, one of the common challenges is determining the optimal number of convolutional layers, pooling layers, strides, and filters for a neural network. Our system provides an automated solution, allowing users to input their data as a link containing image data. The platform then suggests optimal parameters based on the dataset characteristics, and users can further refine the architecture using the UI. This module is designed to save the users' time by automating the trial-and-error process of architecture tuning, and it makes it possible to focus more on model interpretation and insights.

Fig 4: Flowchart for DL Optimization Module


DL (CNN) OPTIMIZATION
HOME PAGE
DL (CNN) OPTIMIZATION
UPLOAD DATA/ LINK
PROCESS THE DATA USING LLM
GET THE BEST BASELINE MODEL WITH BEST PARAMETERS
USER DEFINED LAYER SETTINGS
PROVIDE OVERALL CODE BLOCK WITH LLM

IV. EXPECTED RESULTS/ OUTCOMES

1) 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.
2) 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.
3) 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 selection. 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.
4) 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.

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).
[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
[9] Le1, Jiquan Ngiam, Adam Coates, Abhik Lahiri, Bobby Prochnow, Andrew Y. Ng. Computer Science Department, Stanford University, Stanford, CA 94305, USA