SR_Report.docx

Vishwakarma Group of Institutions

Document Details

Submission ID	trn:oid:::3618:93345988
Submission Date	Apr 28, 2025, 8:51 PM GMT+5:30
Download Date	Apr 28, 2025, 8:53 PM GMT+5:30
File Name	SR_Report.docx
File Size	549.5 KB

12 Pages

1,930 Words

12,309 Characters

6% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography

Match Groups

10 Not Cited or Quoted 6%
Matches with neither in-text citation nor quotation marks

0 Missing Quotations 0%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

5%	Internet sources
2%	Publications
4%	Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Match Groups

- **10** Not Cited or Quoted 6%
  Matches with neither in-text citation nor quotation marks
- **0** Missing Quotations 0%
  Matches that are still very similar to source material
- **0** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

5%	Internet sources
2%	Publications
4%	Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Internet	**www.coursehero.com**	**2%**
2	Internet	**arxiv.org**	**1%**
3	Internet	**curin.chitkara.edu.in**	**<1%**
4	Internet	**ijaidsml.org**	**<1%**
5	Submitted works	**Concordia University on 2025-01-20**	**<1%**
6	Publication	**Tasneem Ahmed, Shrish Bajpai, Mohammad Faisal, Suman Lata Tripathi. "Advanc...**	**<1%**

A PRELIMENERY REPORT ON SOFTWARE ROBOTICS

AUTO ML - HYPERPARAMETER OPTIMIZATION

SUBMITTED TO THE VISHWAKARMA INSTITUTE OF INFORMATION TECHNOLOGY, PUNE
IN THE PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE

1

BACHELOR OF TECHNOLOGY (COMPUTER ENGINEERING)

SUBMITTED BY :

Sr. No.	Name	Division	PRN No.	Roll No.
1	Pravin Kumawat	C	22210292	323042
2	Gaurav Pawar	C	22210838	323057
3	Ambarish Satbhai	C	22211511	323061
4	Rugved Vaidya	C	22211330	323073

3

DEPARTMENT OF COMPUTER ENGINEERING

BRACT'S VISHWAKARMA INSTITUTE OF INFORMATION TECHNOLOGY

SURVEY NO. 3/4, KONDHWA (BUDRUK), PUNE – 411048, MAHARASHTRA (INDIA).

1. Introduction

1.1 Overview

The goal of the AI-Based AutoML System for Hyperparameter Optimisation is to automate the process of choosing the best machine learning model and its ideal hyperparameters. Users can preprocess data, optimise model selection using automated methods, and upload datasets to the system. It is possible to download the finished trained model for later use. Users with little experience in data science can now choose machine learning models more easily thanks to this system.Users can upload datasets, choose the target column, and have the data automatically preprocessed through the system's Flask-powered web-based interface. Numerous machine learning models, including Random Forest, XGBoost, LightGBM, Logistic Regression, and K-Nearest Neighbours, are supported by the system. It carries out missing value imputation, categorical variable encoding, and automatic feature scaling. To determine the ideal model configuration, the system also employs hyperparameter tuning through RandomizedSearchCV. Users can effortlessly incorporate the trained models into their applications because they are stored for future predictions.

1.2 Motivation

The goal of the AI-Based AutoML System for Hyperparameter Optimisation is to automate the process of choosing the best machine learning model and its ideal hyperparameters. Users can preprocess data, optimise model selection using automated methods, and upload datasets to the system. It is possible to download the finished trained model for later use. Users with little experience in data science can now choose machine learning models more easily thanks to this system.Users can upload datasets, choose the target column, and have the data automatically preprocessed through the system's Flask-powered web-based interface. Numerous machine learning models, including Random Forest, XGBoost, LightGBM, Logistic Regression, and K-Nearest Neighbours, are supported by the system. It carries out missing value imputation, categorical variable encoding, and automatic feature scaling. To determine the ideal model configuration, the system also employs hyperparameter tuning through RandomizedSearchCV. Users can effortlessly incorporate the trained models into their applications because they are stored for future predictions.

1.3 Problem Definition and Objectives

1.3.1 Problem Definiton : Manual hyperparameter tuning is inefficient and inconsistent.

1.3.2 Objective :
- Build an end-to-end system for automatic hyperparameter tuning.
- Allow users to upload any dataset.
- Automatically suggest the best model with optimized parameters.

1.4 Project Scope & Limitations

Scope : Supports classification and regression tasks.
Limitations : No support for extremely large datasets (> 500MB).
Focused on tabular data (CSV input).

1.5 Methodologies of problem solving

1.5.1 Problem Understanding and Requirement Analysis

Objective: Gain an understanding of the challenges associated with traditional machine learning model building, particularly with regard to hyperparameter tuning.

Identification of scope: Restricted to regression and classification on structured/tabular datasets.

1.5.2 Collecting requirements:

1.Upload the dataset.

2.Let the user choose the desired column.

3.automatically prepare and clean the data.

4.Select the optimal model automatically and adjust its hyperparameters.

5.Show evaluation visualisations and metrics.

1.5.3 Dataset Handling

1.Data Input Module: Using the system interface, users upload a CSV dataset.
2.Target Column Selection: The user can choose the prediction target (dependent variable) by selecting any column from the list that the system displays after the file has been uploaded.

1.5.4 Data Preprocessing

For the best model performance, data preprocessing makes sure the dataset is cleaned and organised.

Managing Missing Values:

1.Enter the mean or median values in the numerical columns.
2.Enter "Unknown" or "Mode" in the categorical columns.
3.Identification of Data Type:
4.Differentiate between text, boolean, categorical, and numerical data types.
5.Categorical variable encoding:
a)Depending on the needs of the model, use either label encoding or one-hot encoding.

6.Scaling Features:

a)When required, StandardScaler or MinMaxScaler are used,particularly for distance-based algorithms.

7.Selecting features:
a)To lessen multicollinearity, eliminate highly correlated features (

1.5.5 Model Selection

For training, several machine learning models are taken into consideration:

1.Regression using Logistic
2.Random Forest Regressor/Classifier
3.XGBoost
4.Trees of Decisions
5.SVM, or support vector machine

The system uses cross-validation scores to compare performance after performing an initial training with default parameters.
For hyperparameter optimisation, the top two or three models are shortlisted.

1.5.6 Hyperparamter Optimization

To get the best accuracy, the hyperparameters must be adjusted
We use three main tactics:

1.A Search at Random:
Hyperparameter combinations are sampled at random.
Ideal for search spaces with many dimensions.
Quicker than the Grid Look for more expansive parameter spaces.

2.Search Grid:
a thorough search across a manually chosen subset of hyperparameters.
Ideal for small and controllable parameter spaces.

3.The Optuna Framework's Bayesian Optimisation:
makes an informed decision about which set of hyperparameters to try next by using the results of previous evaluations.
Because it concentrates on promising regions of the search space, it is more efficient.

For every model:

There is a defined hyperparameter space (for example, the number of Random Forest trees or the XGBoost learning rate).
To determine the optimal configuration, the optimiser performs a predetermined number of trials (for example, 50–100).

1.5.7 Model Evaluation

Following model optimisation and training:
Tasks for Classification:

1.Precision
2.Accuracy
3,Remember
4.F1-Score
5.Matrix of Confusion
6.AUC Score and ROC Curve
7.Sensitivity and Specificity

Regression Exercises:
1 1.MAE, or mean absolute error
2.RMSE, or root mean squared error

Coefficient of Determination, or R2 Score
1.It generates visualisations such as precision-recall curves, confusion matrices, and bar graphs.

1.5.8 Best Model Recommandation

The system suggests the optimal model based on the evaluation metrics.
provides the name of the model, the final accuracy (rounded to two decimals), and the optimised hyperparameters.

02. Literature Survey

5 6 A crucial stage in the creation of machine learning models is hyperparameter optimisation, which has a direct impact on the model's performance and capacity for generalisation. Despite their widespread use, traditional search techniques like Grid Search and Random Search frequently suffer from computational inefficiency, particularly when dealing with larger search spaces. In order to address this, more sophisticated methods have emerged, such as Bayesian Optimisation (e.g., using frameworks like Optuna and Hyperopt), which cleverly explores the parameter space based on prior evaluations.
In order to improve hyperparameter settings over iterations, evolutionary algorithms such as Genetic Algorithms and Particle Swarm Optimisation further imitate natural selection processes. In addition to optimisation techniques, AutoML frameworks such as Auto-Sklearn, Google AutoML, and TPOT have been created to automate model selection, feature engineering, pipeline optimisation, and hyperparameter tuning, thereby facilitating machine learning for non-experts.Lightweight, explainable deployments are hampered by the fact that current systems frequently demand large amounts of computational power, operate less transparently, and occasionally lose model interpretability.
Despite significant progress, there is still a need for a straightforward, CPU-efficient AutoML system that prioritises giving users a user-friendly interface for fine-tuning hyperparameters while preserving thorough transparency at every stage. Existing systems are either black boxes that provide little insight into model behaviour or require a high level of machine learning expertise to customise. By developing an intuitive AutoML solution that facilitates dataset uploading, intelligent model selection, automatic preprocessing, hyperparameter optimisation through contemporary methods, and comprehensive evaluation reporting, our project fills this gap. It also ensures scalability for small to medium datasets without relying on

GPUs.The goal of this work is to help researchers, students, and early-stage practitioners create more effective machine learning models by combining the best features of current approaches with their shortcomings.

03. System Design

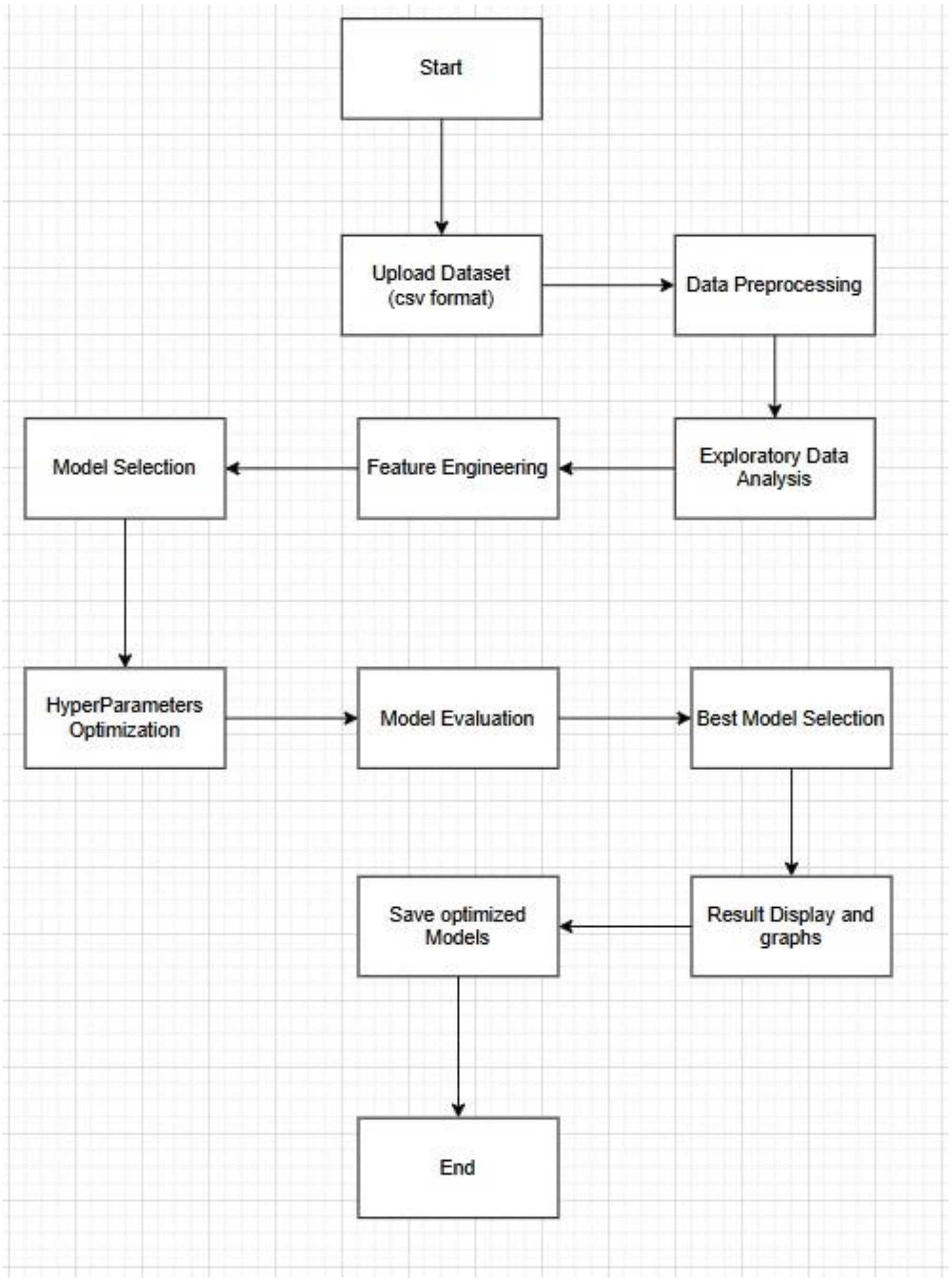


Fig. 1 : Flowchart of Work

04. Project Implementation

4.1 Overview of Project Modules

Upload Data : enables users to submit CSV files, or custom datasets, for modelling and analysis.

Target Column Selection : gives users the option to choose which column (classification or regression target) needs to be predicted.

Data Cleaning : automatically prepares data for modelling by handling scaling, encoding, and missing values.

Model Comparison : evaluates several machine learning models to determine which one performs best based on initial accuracy.

Hyperparameter Tuning : uses Optuna to automatically adjust key hyperparameters, optimising model performance.

Accuracy Reporting : shows the results of the evaluation, including performance graphs, confusion matrix, recall, accuracy, and precision.

4.2 Tools and Technologies Used

1.Python 3.11

2.Scikit-Learn

3.Optuna (for hyperparameter optimization)

4.Pandas, NumPy

5.Matplotlib, Seaborn (for graph generation)

6.Html , CSS , Javascript

4.3 Algorithm Details

4.3.1 Algorithm : Randomized Search

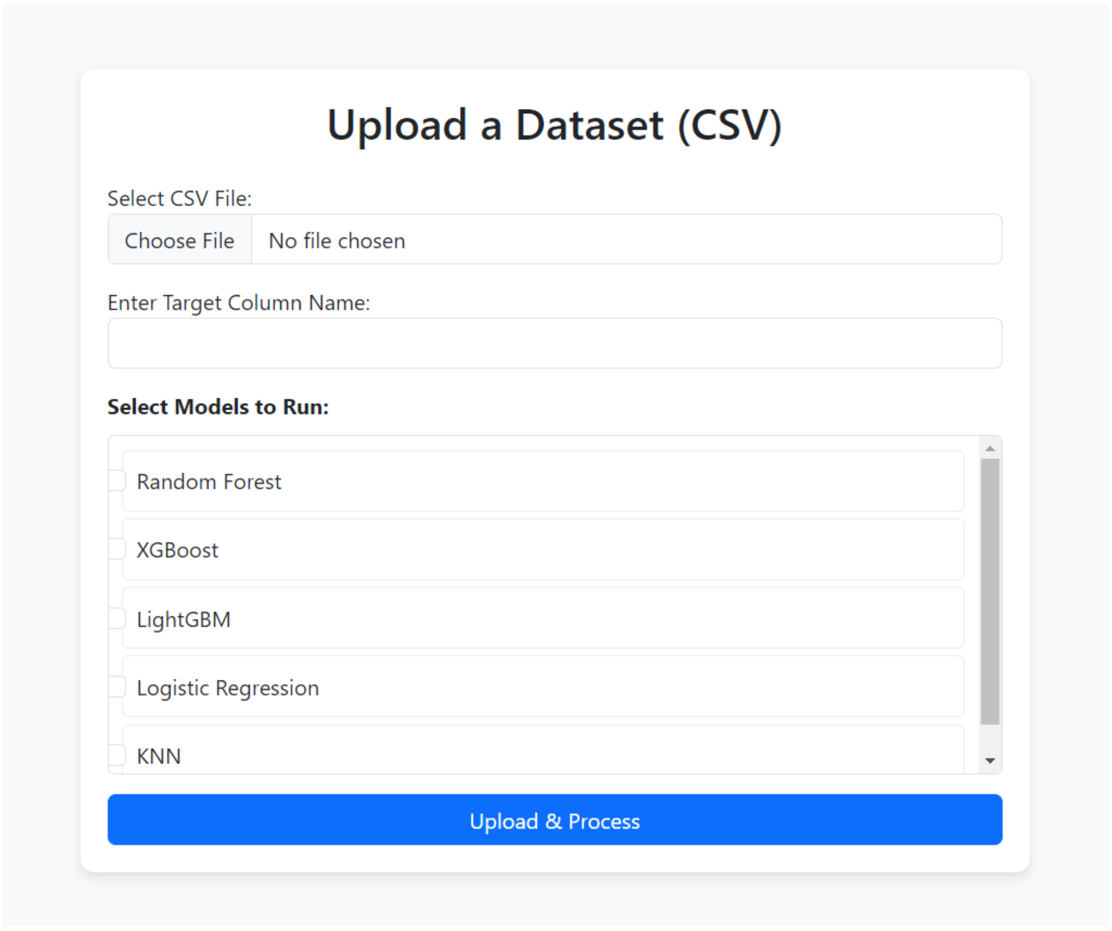
2 Instead of trying every possible combination, as in Grid Search, Randomised Search is a hyperparameter optimisation technique that chooses combinations of hyperparameter values at random from a given range or distribution. This method samples a predetermined number of parameter settings and assesses them using model performance metrics like recall, accuracy, and precision. Because Randomised Search can explore a larger range of hyperparameter space with a smaller budget of iterations, it is computationally more efficient. Compared to exhaustive approaches, it allows for faster convergence towards good results and is especially helpful when only a small number of hyperparameters have a significant impact on the model's performance. To find promising model configurations fast, this project uses Randomised Search as a baseline optimisation algorithm.

05. Results

5.1 Outcomes

System automatically recommends the best model with accuracy up to 95% (varies based on dataset.

5.2 Screen Shots



Upload a Dataset (CSV)

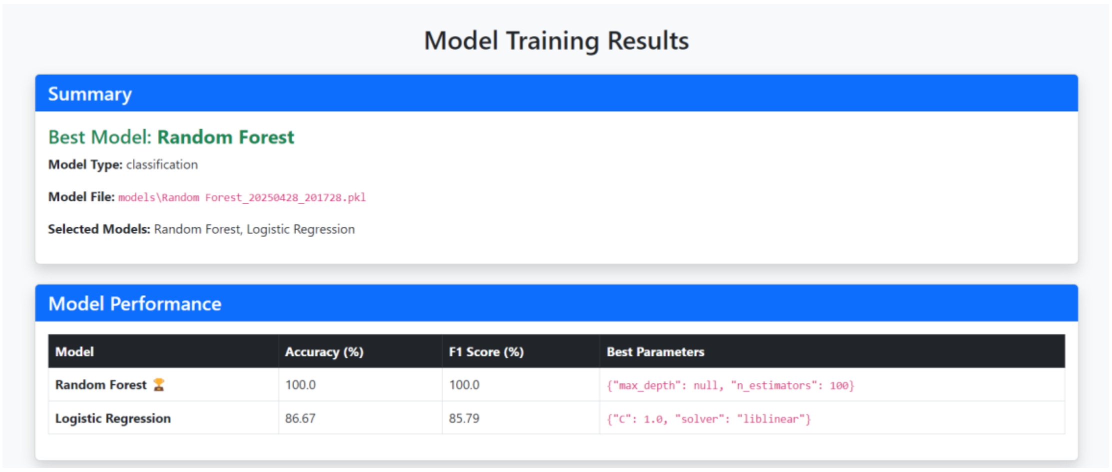
Select CSV File:

Enter Target Column Name:

Select Models to Run:

- Random Forest
- XGBoost
- LightGBM
- Logistic Regression
- KNN

Fig.2: Input GUI



Model Training Results

Summary

Best Model: **Random Forest**

Model Type: classification

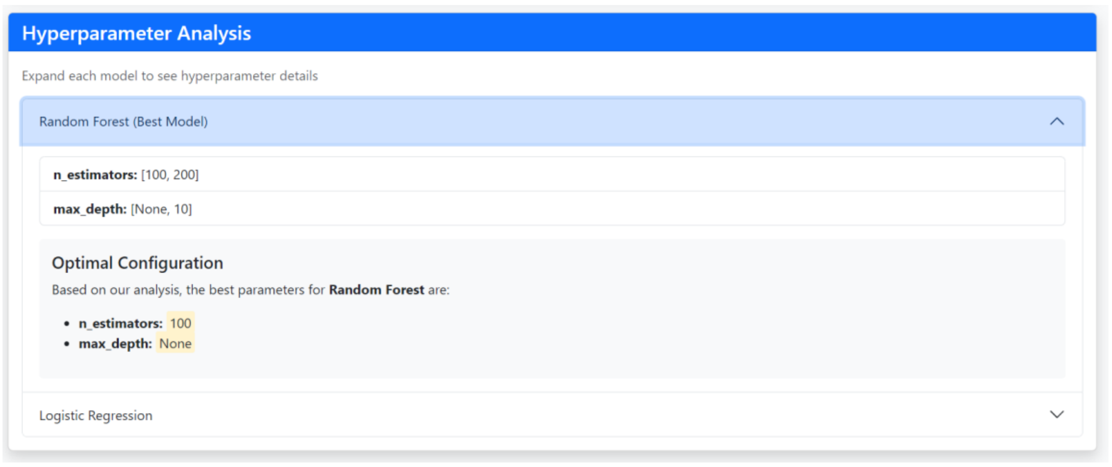
Model File: `models\Random Forest_20250428_201728.pkl`

Selected Models: Random Forest, Logistic Regression

Model Performance

Model	Accuracy (%)	F1 Score (%)	Best Parameters
Random Forest 🏆	100.0	100.0	`{"max_depth": null, "n_estimators": 100}`
Logistic Regression	86.67	85.79	`{"C": 1.0, "solver": "liblinear"}`

Fig. 3: Model Training Results



Hyperparameter Analysis

Expand each model to see hyperparameter details

Random Forest (Best Model)

n_estimators: [100, 200]

max_depth: [None, 10]

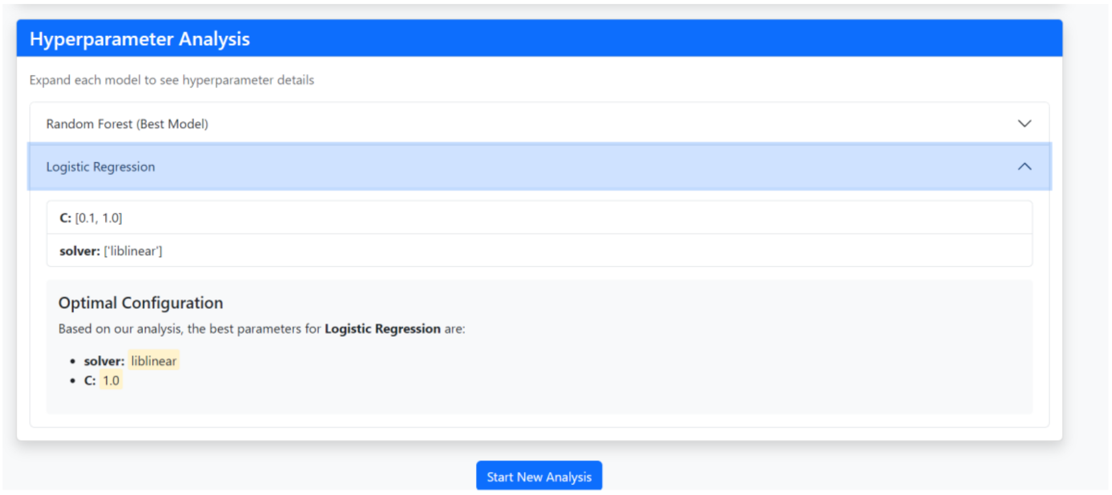
Optimal Configuration

Based on our analysis, the best parameters for **Random Forest** are:

- **n_estimators:** 100
- **max_depth:** None

Logistic Regression

Fig. 4 : Hyperparameter for Best Model



Hyperparameter Analysis

Expand each model to see hyperparameter details

Random Forest (Best Model)

Logistic Regression

C: [0.1, 1.0]

solver: ['liblinear']

Optimal Configuration

Based on our analysis, the best parameters for **Logistic Regression** are:

- **solver:** liblinear
- **C:** 1.0

Fig. 5 : Hyperparameters for other selected model

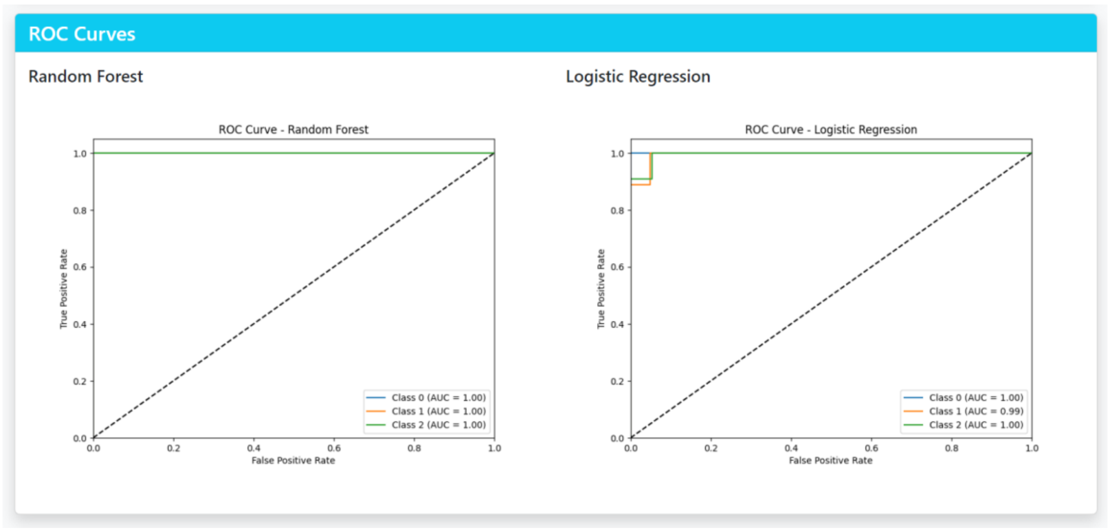


Fig. 6 : ROC Curve

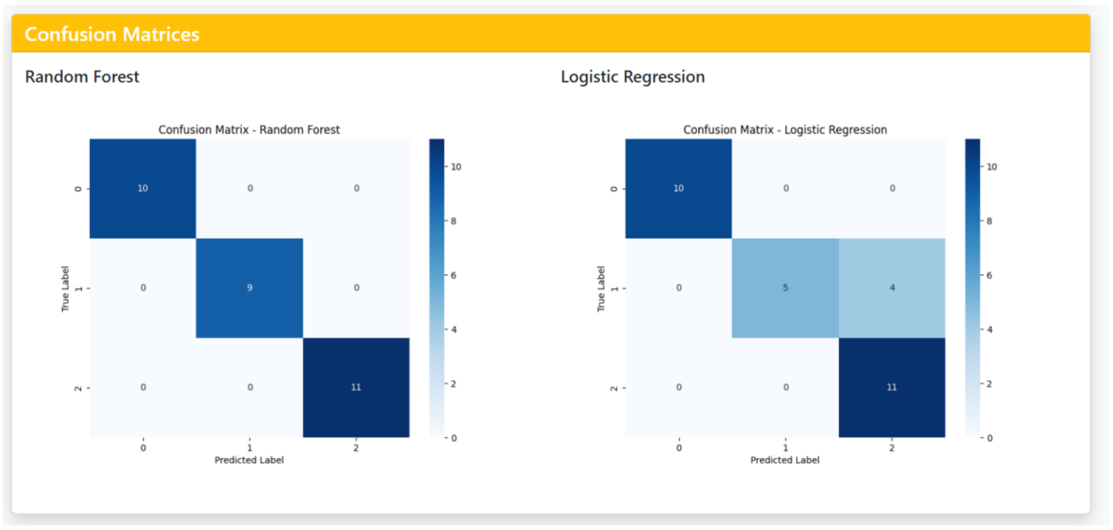


Fig. 7 : Confusion Matrix

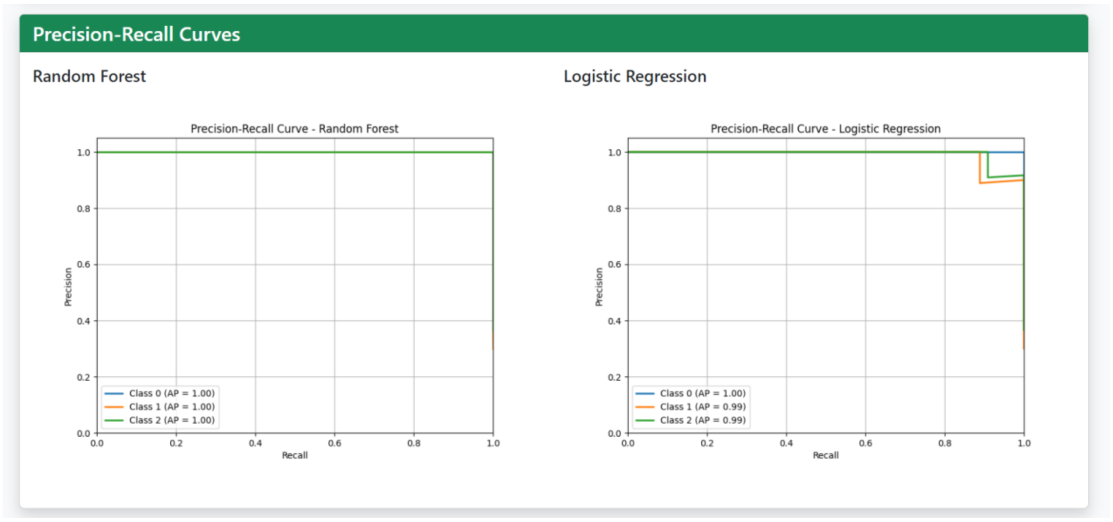


Fig. 8 : Precision and recall Curve

06. Conclusions

6.1 Conclusions

We successfully developed and put into use an AutoML system with an intelligent hyperparameter optimisation focus in this project. Users can upload datasets, choose the target variable, perform automatic data cleaning, compare various machine learning models, and refine them using sophisticated optimisation techniques like Randomised Search and Optuna-based Bayesian Optimisation. The system offers an end-to-end automated solution. Our approach prioritises usability, necessitates little expertise in machine learning, and is built to function effectively even on typical CPU-based computers with low processing demands.Improved accuracy, precision, recall, and F1-scores across tested datasets show that hyperparameter tuning dramatically improves model performance. Our system balances automation and transparency by automating time-consuming processes like model selection and preprocessing while providing fine-grained control over optimisation. Additionally, users can gain a comprehensive understanding of model behaviour through the system's graphical outputs, confusion matrices, and evaluation metrics. All things considered, the project demonstrates how lightweight, modular AutoML platforms can democratise the adoption of machine learning in small business, research, and educational settings.

6.2 Future Work

1.Add deep learning models (e.g., CNNs, RNNs).

2.Expand to handle time-series datasets.

6.3 Applications

1.Automated Machine Learning for startups.

2.Assisting non-technical industries in predictive analytics.

3.Educational tools for ML students.

Appendix A

Problem statement feasibility assessment:

Complexity: Polynomial time solvable (P-Type).

Satisfiability Analysis: No unsolvable conditions for standard datasets.

NP-Hardness: Not applicable directly since tuning can be optimized heuristically.

Modern Algebra: Parameter search space treated as a vector space; optimization seeks minima/maxima within the set.

Appendix B

Plagiarism Report:

References

[1]Thomas Noltey, Hans Hanssony, Lucia Lo Belloz, "Communication Buses for Automotive Applications", Proceedings of the 3rd Information Survivability Workshop (ISW-2007), Boston, Massachusetts, USA, October 2007. IEEE Computer Society.

[2]Feurer, M., Klein, A., Eggensperger, K., Springenberg, J.T., Blum, M., & Hutter, F. (2015). Auto-sklearn: Efficient and Robust Automated Machine Learning.

[3]Bergstra, J., & Bengio, Y. (2012). Random Search for Hyper-Parameter Optimization. JMLR.

[4]Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna: A Next-generation Hyperparameter Optimization Framework.