MACHINE LEARNING

Machine Learning Model Builder System

Martin NIYONZIMA (220446611)

BIGWI NGOGA Raul (220446609)

Fred MUGABO (220446612)

DECEMBER 25, 2025

Table of Contents

Machine Learning Model Builder System

1. Introduction & Problem Definition

Machine Learning (ML) has become a critical tool for data-driven decision making in domains such as finance, healthcare, and security. However, building ML models typically requires strong programming skills, understanding of data preprocessing techniques, and familiarity with evaluation metrics. This creates a barrier for students and non-expert users who wish to experiment with ML models.

The problem addressed in this project is the **lack of simple, interactive systems that allow users to build, train, and evaluate ML classification models without writing code**.

To solve this problem, this project implements a **web-based Machine Learning Model Builder System** that guides users through the complete ML workflow using a graphical interface connected to a modular backend.

2. Dataset Description

The system accepts **tabular datasets** in CSV or Excel format.
Datasets are assumed to contain:

- Multiple feature columns (numeric and/or categorical),
- One target (label) column selected by the user.

Upon upload, the system:

- Reads the dataset,
- Displays dataset dimensions (rows and columns),
- Shows a preview of the first five records,
- Automatically extracts column names.

Missing values in feature columns are handled during preprocessing, while rows with missing target values are safely removed to ensure valid model training.

3. GUI Design and Features

The graphical user interface (GUI) is implemented using **React** and follows a **step-by-step wizard (stepper) design** to ensure usability and clarity.

Key GUI Features:

- **Dataset Upload Page**: Allows file upload and displays dataset preview.
- **Preprocessing Page**: Enables selection of target column and preprocessing method.
- **Model Selection Page**: Allows users to select one or more ML models.
- **Training Page**: Provides training configuration options, including hyperparameters for MLP.
- **Results Page**: Displays performance metrics and allows saving trained models.

The stepper design ensures users cannot skip essential stages of the ML pipeline, reducing user error.

4. Preprocessing Choices

Preprocessing is a critical step in ML model performance. This system provides two preprocessing options:

Normalization

- Numeric features are scaled using standardization.
- Missing numeric values are imputed using the median.
- Categorical features are one-hot encoded.

One-Hot Encoding

- Numeric features are imputed but not scaled.
- Categorical features are converted into binary vectors.

The system automatically:

- Identifies numeric and categorical columns,
- Separates features from the target column,
- Build a preprocessing pipeline dynamically based on user choice.

5. Model Descriptions

The system supports three classification models:

5.1 Perceptron

A linear classifier suitable for linearly separable datasets. It is simple, fast, and serves as a baseline model.

5.2 Decision Tree

A tree-based classifier that splits data based on feature values. It is interpretable and can handle both numeric and categorical features effectively.

5.3 Multilayer Perceptron (MLP – Backpropagation)

A neural network model with one or more hidden layers. The system allows users to configure:

- Number of hidden layers,
- Neurons per layer,
- Learning rate,
- Number of training iterations.

All models are trained using a unified pipeline that combines preprocessing and classification.

6. Experimental Results

6.1 Evaluation Metrics

Each trained model is evaluated using the following metrics:

- Accuracy
- Precision (weighted)
- Recall (weighted)
- F1-score (weighted)

6.2 Confusion Matrices

For each model, a confusion matrix is generated to visualize:

- Correct classifications,
- False positives,
- False negatives.

These matrices provide deeper insight beyond accuracy, especially for imbalanced datasets.

8. Conclusion & Future Work

This project successfully delivers a **full-stack Machine Learning Model Builder System** that simplifies ML experimentation through an intuitive graphical interface and a robust backend.

Achievements:

- End-to-end ML workflow automation,
- Multiple preprocessing and model options,
- Clear visualization of results,
- Model persistence for reuse.

9. References

1. Géron, A. (2019). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow* (2nd ed.). O'Reilly Media.

2. Flask Documentation. (n.d.). *Flask: Web development, one drop at a time*. https://flask.palletsprojects.com

3. React Documentation. (n.d.). *React – A JavaScript library for building user interfaces*. https://react.dev

4. scikit-learn Documentation. (n.d.). *scikit-learn: Machine learning in Python*. https://scikit-learn.org

5. scikit-learn. (n.d.). *Confusion matrix visualization*. https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html

6. Kaggle. (n.d.). *Kaggle datasets*. https://www.kaggle.com/datasets