Machine Learning Model Builder - User Manual (Version 1.0)

Welcome to the Machine Learning Model Builder!

This user manual will guide you through every step of using the **Machine Learning Model Builder** application. This application is designed to be a user-friendly tool for building and evaluating machine learning models without needing to write complex code. Whether you are a student, researcher, or data enthusiast, this tool aims to simplify the process of model building for both **classification** and **regression** tasks.

Developed By: Mukund Narayanan / Indian Institute of Technology Roorkee **Website:** https://www.mukundnarayanan.org/

Let's get started and explore the power of machine learning model building at your fingertips!

1. Getting Started

1.1 Installation (For .exe File)

As you are using the `.exe` file version of the Machine Learning Model Builder, you **do not need to install** anything! This is a standalone application.

To run the application, simply **double-click** the `.exe` file. The application should launch immediately.

1.2 Launching the Application

Once you double-click the `.exe` file, the **Machine Learning Model Builder** application window will appear on your screen.

1.3 Initial Interface Overview

When you first launch the application, you will see a window divided into two main sections:

- **Left Frame:** This section is primarily for **input and selection**. You will use this area to load your data, choose your variables (both independent and dependent), select the type of machine learning task (classification or regression), and choose the model you want to use.
- **Right Frame:** This section is focused on **model configuration and execution**. Here, you will set the hyperparameters for your chosen model, train the model, optimize hyperparameters, save and load models, load prediction data, make predictions, and visualize the results.

Let's explore each section in detail as we walk through the process of building a machine learning model.

2. Using the Application - Step-by-Step Guide

This section will guide you through the process of building a machine learning model using the application, step-by-step.

Step 1: Input Data

Action: Click the **"Input Data"** button located in the top left corner of the application window.

What Happens: Clicking this button will open a standard file explorer window. This window allows you to browse your computer and select the data file you want to use for building your machine learning model.

Supported File Types: The application supports the following file types for your data:

- **CSV files (.csv):** Comma Separated Values files are plain text files where data is organized in rows and columns, separated by commas. This is a very common format for data.
- **Excel files (.xlsx, .xls):** If your data is stored in Microsoft Excel format, you can load both `.xlsx` and older `.xls` file formats.

Choosing Your Data File: Navigate through the file explorer window to the location where your data file is saved. Select your data file (either a `.csv`, `.xlsx`, or `.xls` file) and click the **"Open"** button in the file explorer.

Data Loading and Encoding:

- The application will attempt to load your data. It tries to automatically detect the character encoding used in your file, starting with `utf-8` encoding which is the most common.
- If the application encounters an error while loading the data using `utf-8` (which might happen if your file uses a different encoding, like `latin1`), it will automatically retry loading the data using `latin1` encoding. `latin1` is another common encoding, especially for older files.
- If the data loading is successful, the application will populate the **"Independent Variables"** and **"Dependent Variables"** listboxes in the left frame with the column names from your data file.

Error Handling: If the application fails to load your data file (for example, if the file is corrupted, not in a supported format, or if there's an unexpected encoding issue), an **"Error"** message box will pop up on your screen. This message box will tell you that "Failed to load data" and will often include a technical error message to help in troubleshooting. If you see an error, please ensure that your file is a valid `.csv`, `.xlsx`, or `.xls` file and try again.

Step 2: Variable Selection

After successfully loading your data, the next step is to tell the application which columns in your data are your **independent variables** (features) and which are your **dependent variables** (the target you want to predict).

Interface Elements:

- **"Select Independent and Dependent Variables" Label:** This label simply indicates the purpose of this section.
- **"Independent Variables" Label:** A label above the listbox for independent variables.

- **"Independent Variables" Listbox:** This listbox displays all the column names from your loaded data. You need to select the columns that you want to use as **independent variables** for your model.
- **"Dependent Variables" Label:** A label above the listbox for dependent variables.
- **"Dependent Variables" Listbox:** Similar to the independent variables listbox, this displays all column names. You need to select the column(s) that you want to predict - these are your **dependent variables**.
- **"Confirm Variables" Button:** After you have selected your independent and dependent variables from the listboxes, you need to click this button to confirm your selections.

Selecting Variables:

1. **Independent Variables:** In the "Independent Variables" listbox, **click on the name of each column** that you want to use as an independent variable (predictor). You can select **multiple** independent variables. To select multiple items, you can typically **click and drag** to select a range, or **Ctrl+Click** (or Cmd+Click on Mac) to select individual items non-consecutively. Selected items will be highlighted.
2. **Dependent Variables:** In the "Dependent Variables" listbox, **click on the name of the column** that you want to predict. You should generally select **one** dependent variable for most standard machine learning tasks within this application. Although the listbox allows multiple selections, for typical regression and classification, you'll usually have a single target variable.

Important Note: Carefully choose your independent and dependent variables based on your understanding of your data and the problem you are trying to solve. Independent variables are the factors that you believe influence or predict the dependent variable.

Confirming Selections: Once you have selected your independent and dependent variables in the respective listboxes, click the **"Confirm Variables"** button.

Information Message: After clicking "Confirm Variables," an **"Info"** message box will appear. This message box will display:

- **"Independent Variables:"** Followed by a comma-separated list of the independent variables you selected.
- **"Dependent Variables:"** Followed by a comma-separated list of the dependent variables you selected (usually just one in most cases).

This message box is a confirmation that the application has registered your variable selections correctly.

Step 3: Feature Scaling (Optional but Recommended)

Interface Element:

- **"Enable Feature Scaling" Checkbutton:** This is a small checkbox with the label "Enable Feature Scaling". It is located in the left frame, below the variable selection section.

What is Feature Scaling?

Feature scaling is a preprocessing step in machine learning where you transform the numerical features of your dataset to a similar scale. This is often important because many machine learning algorithms perform better or converge faster when features are on a similar scale. For example, algorithms like Support Vector Machines (SVM) and Neural Networks are sensitive to the scale of input features.

When to Use Feature Scaling:

It is generally a good practice to enable feature scaling when:

- You are using algorithms that are sensitive to feature scaling, such as:
  - Neural Networks (MLP)
  - Support Vector Machines (SVM/SVR)
  - K-Nearest Neighbors (KNN - *not available directly in this application, but a good general guideline*)
- Your independent variables have significantly different ranges of values. For instance, if one variable ranges from 0 to 1, and another ranges from 1000 to 100000.

When Feature Scaling Might Be Less Critical:

- Tree-based models like Decision Trees and Random Forests are generally less sensitive to feature scaling. Linear Regression is also less directly affected by feature scaling in terms of coefficient correctness, though scaling can help with numerical stability and convergence during training with gradient descent-based solvers if regularization is used.

Using the Checkbutton:

- **To Enable Feature Scaling:** Click on the "Enable Feature Scaling" checkbutton to place a checkmark in the box. This tells the application that you want to apply feature scaling to your independent variables during model training and prediction.
- **To Disable Feature Scaling:** If the checkbutton is already checked, click it again to remove the checkmark. This disables feature scaling. By default, the checkbutton is likely **unchecked** (feature scaling is disabled).

How Feature Scaling is Applied:

If you enable feature scaling, the application uses the **StandardScaler** from the `scikit-learn` library. Standard Scaler standardizes features by removing the mean and scaling to unit variance. This means for each feature, it subtracts the mean of that feature across all data points and then divides by the standard deviation. This results in features with a mean of 0 and a standard deviation of 1.

Information Message: If you have enabled feature scaling and proceed to train your model (in Step 7), after feature scaling is applied successfully, an **"Info"** message box will appear stating: "Feature Scaling Applied Successfully".

Step 4: Task Selection

Interface Elements:

- **"Select Task" Label:** Indicates the purpose of this section.
- **Task Dropdown (OptionMenu):** This dropdown menu allows you to choose the type of machine learning task you want to perform. The available options are:
  - **Classification:** Choose this if you want to predict a categorical dependent variable (i.e., the dependent variable represents categories or classes). For example, predicting whether an email is "spam" or "not spam," or predicting the "type of flower" from its measurements.

    - **Regression:** Choose this if you want to predict a continuous numerical dependent variable. For example, predicting the "price of a house," or the "temperature tomorrow."

Selecting the Task:

1. **Click on the Task Dropdown Menu.** This will open the dropdown list showing "Classification" and "Regression".
2. **Select either "Classification" or "Regression"** based on the nature of your dependent variable and the problem you are trying to solve. **Click on your choice.**

Impact of Task Selection:

The task you select here is crucial because it determines:

- **The types of machine learning models** that will be available for you to choose from in the next step (Step 5). Different models are suited for different types of tasks. For example, while both Random Forest Classifier and Random Forest Regressor exist, the "Classifier" is used for classification tasks, and "Regressor" for regression.
- **The evaluation metrics** that will be used to assess the performance of your model. For classification, metrics like accuracy, classification report, and confusion matrix are used. For regression, metrics like R-squared (R^2) are used.
- **The plotting type** for visualizing results. For classification, a confusion matrix is plotted. For regression, a scatter plot of actual vs. predicted values is plotted.

After selecting a task, the application will automatically update the **"Model Selection"** dropdown menu in Step 5 to show only the models that are appropriate for the task you have chosen.

Step 5: Model Selection

Interface Elements:

- **"Select Model" Label:** Indicates the purpose of this section.
- **Model Dropdown (OptionMenu):** This dropdown menu allows you to choose the specific machine learning model algorithm that you want to use for your chosen task (classification or regression). The models available in this dropdown will change depending on whether you selected "Classification" or "Regression" in Step 4.

Available Models:

The models available for selection are:

- **For Classification Task:**
    - **MLP:** Multi-Layer Perceptron (a type of Neural Network Classifier)
    - **Random Forest:** Random Forest Classifier
    - **SVM:** Support Vector Machine Classifier (SVC)
    - **Decision Tree:** Decision Tree Classifier
- **For Regression Task:**
    - **MLP:** Multi-Layer Perceptron (a type of Neural Network Regressor)

    - **Random Forest:** Random Forest Regressor
    - **SVR:** Support Vector Regression
    - **Decision Tree:** Decision Tree Regressor
    - **Linear Regression:** Standard Linear Regression

Selecting a Model:

1. **Click on the "Model Selection" Dropdown Menu.** This will open the dropdown list showing the available machine learning models for the task you selected in Step 4.
2. **Select the Model you want to use.** Click on your model of choice from the list. For example, if you are performing a classification task and want to use a Random Forest, select "Random Forest" from the list.

Impact of Model Selection:

Choosing a model is a critical step. Different models have different strengths and weaknesses and may be more or less suitable for your specific data and problem. Factors to consider when choosing a model might include:

- **Complexity of the data:** Some models (like Neural Networks) can learn very complex relationships, while others (like Linear Regression) are simpler and assume linear relationships.
- **Amount of data:** Some complex models might require a lot of data to train effectively. Simpler models can sometimes perform better with less data.
- **Interpretability:** Some models (like Decision Trees and Linear Regression) are easier to interpret and understand why they make certain predictions. Others (like Neural Networks and SVMs) can be more like "black boxes".
- **Performance:** Ultimately, you want to choose a model that performs well on your specific task. This often requires experimentation and potentially comparing the performance of different models.

After selecting a model, the application will automatically update the **"Hyperparameter Setting"** section in the right frame (Step 6). It will display the hyperparameters that are relevant for the model you have selected.

Step 6: Hyperparameter Setting

Interface Elements:

- **"Set Hyperparameters" Label:** Indicates the purpose of this section.
- **Hyperparameter Frame:** This is a rectangular area in the right frame where the specific hyperparameter settings for the selected model are displayed.
- **Hyperparameter Labels and Input Fields:** Within the Hyperparameter Frame, you will see labels describing each hyperparameter and corresponding input fields to set their values. The type of input field depends on the hyperparameter type:
    - **Entry Fields (for Numerical and Tuple Parameters):** For hyperparameters that take numerical values (like integers or floats) or tuples (like `hidden_layer_sizes` for MLP), you'll see a text entry field where you can type in the value.
    - **Dropdown Menus (OptionMenus) (for Categorical Parameters):** For hyperparameters that take categorical values (i.e., choices from a predefined set, like `activation` function or `kernel` type), you will see a dropdown menu.

What are Hyperparameters?

Hyperparameters are settings of a machine learning model that are set *before* the learning process begins. They control how the model learns from the data. Choosing appropriate hyperparameters is crucial for achieving good model performance. Different models have different sets of hyperparameters.

Hyperparameters for Each Model (in this Application):

The hyperparameters you can set will vary depending on the model you selected in Step 5. Here's a breakdown of the hyperparameters available for each model in this application:

- **MLP (Multi-Layer Perceptron):**
  - **`hidden_layer_sizes` (tuple):** Defines the size of the hidden layers in the neural network. You need to enter a tuple (e.g., `(100,)` for one hidden layer with 100 neurons, or `(100, 50)` for two hidden layers with 100 and 50 neurons respectively). **Type: Tuple.**
  - **`activation` (categorical):** The activation function for the hidden layers. Options: "identity", "logistic", "tanh", "relu". **Type: Dropdown.**
  - **`solver` (categorical):** The solver for weight optimization. Options: "lbfgs", "sgd", "adam". **Type: Dropdown.**
  - **`alpha` (float):** L2 regularization term (strength of regularization). **Type: Entry Field (Float).**
  - **`learning_rate` (categorical):** Learning rate schedule for weight updates. Options: "constant", "invscaling", "adaptive". **Type: Dropdown.**
- **Random Forest:**
  - **`n_estimators` (int):** The number of trees in the forest. **Type: Entry Field (Integer).**
  - **`max_depth` (int):** The maximum depth of the trees. **Type: Entry Field (Integer).**
  - **`min_samples_split` (int):** The minimum number of samples required to split an internal node. **Type: Entry Field (Integer).**
  - **`min_samples_leaf` (int):** The minimum number of samples required to be at a leaf node. **Type: Entry Field (Integer).**
- **SVM/SVR (Support Vector Machines for Classification/Regression):**
  - **`C` (float):** Regularization parameter. **Type: Entry Field (Float).**
  - **`kernel` (categorical):** Specifies the kernel type to be used. Options: "linear", "poly", "rbf", "sigmoid". **Type: Dropdown.**
  - **`degree` (int):** Degree of the polynomial kernel function (only relevant if `kernel='poly'`). **Type: Entry Field (Integer).**
  - **`gamma` (categorical):** Kernel coefficient. Options: "scale", "auto". "scale" uses 1 / (n_features * X.var()), "auto" uses 1 / n_features. 'rbf', 'poly' and 'sigmoid'. **Type: Dropdown.**
- **Decision Tree:**
  - **`max_depth` (int):** The maximum depth of the tree. **Type: Entry Field (Integer).**
  - **`min_samples_split` (int):** The minimum number of samples required to split an internal node. **Type: Entry Field (Integer).**

    - **`min_samples_leaf` (int):** The minimum number of samples required to be at a leaf node. **Type: Entry Field (Integer).**

- **Linear Regression:**

    - **Linear Regression has no tunable hyperparameters** exposed in this application interface. It's a straightforward model that fits a linear equation to the data.

Setting Hyperparameter Values:

1. **For Entry Fields:** Click in the entry field next to the hyperparameter label. Type in the numerical value you want to set. Ensure you are entering the value in the correct format (integer or float as indicated in parentheses next to the parameter name). For `hidden_layer_sizes`, remember to enter a tuple like `(100,)` or `(100, 50)`.
2. **For Dropdown Menus:** Click on the dropdown menu for the hyperparameter. A list of available options will appear. Click on your desired option to select it.

Default Values and Optimization:

- If you leave a hyperparameter field blank, the model will often use its **default value** as defined in the `scikit-learn` library. However, to have explicit control and to fully utilize the hyperparameter optimization feature (Step 9), it's generally recommended to at least consider setting some key hyperparameters.
- Step 9, **"Optimize Hyperparameters,"** can help you automatically find good hyperparameter values if you are unsure what values to set manually.

Step 7: Train Model

Interface Element:

- **"Train Model" Button:** Located in the right frame, below the Hyperparameter Frame.

Action: After you have selected your variables, chosen your task, selected a model, and (optionally) set hyperparameters, click the **"Train Model"** button to start the model training process.

What Happens When You Click "Train Model":

1. **Data Preparation:**

    - The application retrieves the independent and dependent variables you selected in Step 2 from your loaded data.
    - It splits your data into two sets: a **training set** and a **testing set**. By default, it uses an 80/20 split, meaning 80% of your data is used for training the model, and 20% is used for testing its performance.
    - If you enabled **feature scaling** in Step 3, it will apply the StandardScaler to your independent variables in both the training and testing sets. The scaling is fit only on the training data and then applied to both training and test data to prevent data leakage.

2. **Model Initialization:**

    - The application initializes the machine learning model you selected in Step 5 (e.g., MLPClassifier, RandomForestRegressor, SVC, etc.).

    - It uses the hyperparameter values that you set in Step 6. If you left any hyperparameter fields blank, default values for those parameters will be used by the underlying `scikit-learn` model.

3. **Model Training:**

    - The initialized model is trained using the **training data** (independent variables and dependent variable targets from the training set). This is where the model learns the relationships between the independent variables and the dependent variable based on your training data.
    - A progress bar might briefly appear, although training for most models in this application might be very fast unless you are using very large datasets or complex models like deep MLPs.

4. **Model Evaluation:**

    - After training, the application **evaluates the trained model's performance** on both the **training data** and the **testing data**.
    - **For Classification Tasks:** It calculates the **training accuracy** and **testing accuracy**. It also generates a **classification report** (showing precision, recall, F1-score for each class) and a **confusion matrix**. However, the classification report and confusion matrix are only displayed when you click "Plot Accuracies" (Step 8).
    - **For Regression Tasks:** It calculates the **training R-squared (R²)** score and **testing R-squared (R²)** score.

5. **Information Message:**

    - Upon successful model training, an **"Info"** message box will appear. This message box will display:
        - **"Model Trained Successfully"**
        - **For Classification:** "Training Accuracy: [accuracy value]", "Testing Accuracy: [accuracy value]"
        - **For Regression:** The "classification report" and "confusion matrix" are not displayed in this info box for classification. Instead, for regression, it will show: "Training R^2 Score: [R² value]", "Testing R^2 Score: [R² value]". The full report text is displayed.

Error Handling: If any error occurs during model training or prediction (for example, due to incompatible hyperparameter settings, issues with data types, or problems during the model fitting process), an **"Error"** message box titled "Model Training Error" will pop up. It will display an error message indicating the problem. Please review the error message to understand the issue, and check your data, variable selections, model choice, and hyperparameters.

After successful training, the **"Plot Accuracies"** button (Step 8) will become enabled, allowing you to visualize the model's performance.

Step 8: Plot Accuracies (Visualize Model Performance)

Interface Element:

- **"Plot Accuracies" Button:** Located in the right frame, below the "Train Model" button. **This button is typically disabled until you have successfully trained a model in Step 7.**

Action: After you have trained a model, click the **"Plot Accuracies"** button to visualize the performance of your model.

What Happens When You Click "Plot Accuracies":

The type of plot that is generated depends on whether you selected "Classification" or "Regression" as your task in Step 4.

- **For Classification Tasks:**
    - **Confusion Matrix:** A **Confusion Matrix** plot will be displayed using `matplotlib` and `seaborn`. The confusion matrix is a table that visualizes the performance of a classification model by showing the counts of:
        - **True Positives (TP):** Correctly predicted positive instances.
        - **True Negatives (TN):** Correctly predicted negative instances.
        - **False Positives (FP):** Incorrectly predicted positive instances (Type I error).
        - **False Negatives (FN):** Incorrectly predicted negative instances (Type II error).
        - The heatmap visualization of the confusion matrix makes it easy to see patterns in your model's predictions, including which classes it is confusing with each other.
        - The plot will have a title "Confusion Matrix," labels for "True Label" (y-axis) and "Predicted Label" (x-axis), and annotations showing the counts in each cell of the matrix.
        - The color scheme (colormap) will be "Blues".

- **For Regression Tasks:**
    - **Actual vs. Predicted Scatter Plot:** A **Scatter Plot** will be displayed. This plot visualizes the relationship between the actual (true) values of your dependent variable and the values predicted by your regression model.
        - **Scatter Points:** Each point on the scatter plot represents a data point from your dataset. The x-coordinate of the point is the actual value of the dependent variable, and the y-coordinate is the value predicted by the model for that data point.
        - **Red Line (Ideal Prediction Line):** A red, dashed diagonal line is plotted on the graph. This line represents perfect predictions, where the predicted value is exactly equal to the actual value. If your model is performing well, the scatter points should be clustered closely around this red line.
        - The plot will have a title "Actual vs Predicted", x-axis label "Actual Values," and y-axis label "Predicted Values."

Displaying the Plot: The plot (either confusion matrix or scatter plot) will be displayed in a separate window using `matplotlib`. You can typically interact with these plot windows (zoom, pan, save the plot image, etc., depending on your `matplotlib` backend). Close the plot window to return to the main application window.

Error Handling: If there's an issue during the plotting process (e.g., if the model is not trained, or there's a problem generating the plot), an **"Error"** message box titled "Plotting Error" will appear, indicating that there was an error during plotting and possibly providing a technical error message. Ensure you have trained a model before trying to plot accuracies.

Step 9: Optimize Hyperparameters (Automated Tuning)

Interface Element:

- **"Optimize Hyperparameters" Button:** Located in the right frame, below the "Plot Accuracies" button.

Action: Click the **"Optimize Hyperparameters"** button to automatically search for better hyperparameter settings for your chosen model using a technique called hyperparameter optimization.

What is Hyperparameter Optimization?

Hyperparameter optimization (or hyperparameter tuning) is the process of finding the set of hyperparameters for a machine learning model that results in the best performance on a given dataset. Manually trying out different hyperparameter values can be time-consuming and inefficient. This feature automates that process.

How Hyperparameter Optimization Works in this Application:

- **Optuna Library:** This application uses the `Optuna` library, which is a powerful framework for automated hyperparameter optimization.
- **Objective Function:** The application defines an "objective function" that `Optuna` will try to optimize.
    - **For Classification:** The objective is to **maximize** the **testing accuracy** of the model.
    - **For Regression:** The objective is to **minimize** the **negative testing R-squared (R^2)** score (or maximize R-squared). Note that in the code, it's actually set up to maximize R-squared directly. The direction setting is important for Optuna to understand if it's trying to find the highest or lowest value.
- **Search Space:** `Optuna` will automatically explore a range of possible values for the hyperparameters that are relevant to your selected model. If you have already set some hyperparameter values in Step 6, and *leave other hyperparameter fields blank*, the application is set up to *suggest* values for those blank fields using `Optuna`'s trial suggestions. The search ranges and suggestion types are predefined in the code for each hyperparameter.
- **Trial Runs:** `Optuna` will run multiple "trials." In each trial, it will:
    1. **Suggest Hyperparameter Values:** `Optuna` will intelligently suggest a new set of hyperparameter values to try, based on the results of previous trials.
    2. **Train a Model:** The application will train a new model using the suggested hyperparameters on your training data.
    3. **Evaluate Performance:** It will evaluate the performance of the trained model on your testing data (calculating accuracy for classification, or R^2 for regression).
    4. **Record Results:** `Optuna` will record the hyperparameter values and the resulting performance metric.
- **Progress Bar:** During the optimization process, a **progress bar** labeled "Optimization Progress" will be displayed in the right frame. This progress bar

shows the percentage of optimization trials that have been completed. By default, the application is set to run **100 optimization trials**.
- **Best Parameters:** After all the trials are completed, `Optuna` will identify the set of hyperparameters that resulted in the best performance (highest accuracy for classification, highest R² for regression) across all trials.

Using the Optimization Feature:

1. **Optionally Set Initial Hyperparameters:** You can optionally set some initial hyperparameter values in Step 6 as a starting point. If you leave some hyperparameter fields blank, these are the parameters `Optuna` will try to optimize. If you fill in all hyperparameter fields, the optimization might not effectively change them, as the code is designed to suggest values for *empty* hyperparameter fields. For the most effective optimization, it's usually best to leave the hyperparameter fields you want to optimize *blank*.
2. **Click "Optimize Hyperparameters".** The optimization process will begin. You will see the progress bar advancing as trials are completed.

Information Message: Once the hyperparameter optimization process is complete, an **"Info"** message box will appear:

- **"Optimization Complete"**
- **"Best Parameters:"** Followed by a dictionary showing the best hyperparameter values found by `Optuna`. The application will automatically update the hyperparameter input fields in Step 6 to these best values.

Updating the Model: After optimization, the `self.model` in the application is updated to the best model found during optimization (the model trained with the best parameters). So, subsequent actions like "Plot Accuracies," "Save Model," and "Make Prediction" will use this optimized model.

Error Handling: If an error occurs during hyperparameter optimization (e.g., issues with `Optuna`, errors during model training within a trial), an **"Error"** message box titled "Optimization Error" will be displayed. It will indicate that there was an error during hyperparameter optimization and may include a technical error message.

Step 10: Save Model

Interface Element:

- **"Save Model" Button:** Located in the right frame, below the "Optimize Hyperparameters" button.

Action: Click the **"Save Model"** button if you want to save your trained (or optimized) machine learning model to a file. Saving the model allows you to reuse it later without having to retrain it from scratch.

What Happens When You Click "Save Model":

1. **File Explorer Dialog:** A standard file explorer "Save As" dialog window will open. This window allows you to choose where you want to save the model file on your computer and what you want to name the file.
2. **File Type:** The default file extension for saved models is `.pkl` (for "Pickle file"). Pickle is a Python-specific format for serializing objects (saving Python objects to files). You can also choose to save as "All files (.)" if you prefer a different extension or no extension, but `.pkl` is recommended.

3. **Choose Save Location and Filename:** Navigate to the folder where you want to save the model file, enter a filename for your model (e.g., `my_trained_model`), and click the **"Save"** button in the file dialog.

Saving Process: The application uses the `joblib` library to save your trained model. `joblib` is efficient for saving and loading Python objects, especially those containing NumPy arrays, which are common in machine learning models.

Information Message: After the model is successfully saved, an **"Info"** message box will appear:

- **"Model saved to [file path]"** - where `[file path]` is the full path to the file where you saved your model (e.g., "Model saved to C:\Users\YourName\Documents\my_trained_model.pkl").

Using Saved Models: Once you have saved a model, you can load it later using the "Load Model" button (Step 11) to make predictions on new data without retraining.

Step 11: Load Model

Interface Element:

- **"Load Model" Button:** Located in the right frame, below the "Save Model" button.

Action: Click the **"Load Model"** button when you want to load a previously saved machine learning model from a `.pkl` file. This is useful if you have trained a model earlier, saved it, and now want to use it again without retraining.

What Happens When You Click "Load Model":

1. **File Explorer Dialog:** A standard file explorer "Open" dialog window will open. This window allows you to browse your computer and select the `.pkl` model file that you want to load.
2. **File Type Filter:** The file explorer dialog will initially filter for files with the `.pkl` extension ("Pickle files"). You can also choose "All files (.)" to see all file types if you have saved your model with a different extension or no extension.
3. **Select Model File:** Navigate to the location where you saved your `.pkl` model file previously. Select the model file and click the **"Open"** button in the file explorer.

Loading Process: The application uses the `joblib.load()` function to load the model object from the selected `.pkl` file back into the application's memory. This restores your trained model, including its learned parameters and structure.

Information Message: After the model is successfully loaded, an **"Info"** message box will appear:

- **"Model loaded from [file path]"** - where `[file path]` is the full path to the file from which you loaded the model (e.g., "Model loaded from C:\Users\YourName\Documents\my_trained_model.pkl").

Using Loaded Models: After loading a model, the `self.model` in the application is updated to the model loaded from the file. You can now use this loaded model to:

- **Make Predictions on new data** (Step 13) using the "Make Prediction" button.
- **Plot the accuracies** (Step 8) of the loaded model (although, plotting accuracies might not be very informative for a *pre-trained* model without retraining or

having access to the original training data context).

Important Note: When you load a model, ensure that:

- The model file is a valid `.pkl` file that was created by saving a model from this application or a compatible process.
- The data you use for prediction (Step 13) has the same features (independent variables) that the model was trained on. If the features are different, the model might not work correctly or may produce meaningless predictions.

Step 12: Load Prediction Data

Interface Element:

- **"Load Prediction Data" Button:** Located in the right frame, below the "Load Model" button.

Action: Click the **"Load Prediction Data"** button when you want to load a new dataset for which you want to make predictions using a trained or loaded model.

What Happens When You Click "Load Prediction Data":

1. **File Explorer Dialog:** A standard file explorer "Open" dialog window will appear, similar to the "Input Data" button. This window allows you to browse your computer and select the data file containing the new data for prediction.
2. **Supported File Types:** Just like with the "Input Data" button, the "Load Prediction Data" button supports the same file types:
    - **CSV files (.csv)**
    - **Excel files (.xlsx, .xls)**
3. **Choose Prediction Data File:** Navigate to the location of your prediction data file, select the file, and click the **"Open"** button in the file explorer.

Data Loading for Prediction:

- The application will attempt to load your prediction data file, handling encodings (`utf-8`, then `latin1`) in the same way as when you loaded the training data (Step 1).
- **Important:** Your prediction data file **must** contain columns that correspond to the **independent variables** that your model was trained on. The column names in your prediction data file should ideally match the names of the independent variables you selected in Step 2. The prediction data file should **not** contain the dependent variable column (as you are trying to predict it!).

Information Message: If the prediction data is loaded successfully, an **"Info"** message box will appear:

- **"Prediction data loaded successfully."**

Error Handling: If there's an error loading the prediction data file (e.g., file not found, incorrect format, encoding issues), an **"Error"** message box titled "Error" will appear, indicating "Failed to load prediction data" and possibly including a technical error message. Check that your prediction data file is valid and in a supported format.

After loading prediction data, you can proceed to Step 13, "Make Prediction," to generate predictions using your trained or loaded model on this new data.

Step 13: Make Prediction

Interface Element:

- **"Make Prediction" Button:** Located in the right frame, below the "Load Prediction Data" button.

Action: After you have loaded prediction data (Step 12) and have a trained or loaded model (from Step 7, 9, or 11), click the **"Make Prediction"** button to generate predictions for your new prediction data.

Prerequisites: Before clicking "Make Prediction," ensure that:

1. **A model is loaded or trained:** You must have either trained a model in the current session (by clicking "Train Model" or "Optimize Hyperparameters") or loaded a previously saved model using "Load Model." If no model is available, clicking "Make Prediction" will result in an error message.
2. **Prediction data is loaded:** You must have loaded a new dataset for prediction using the "Load Prediction Data" button. If no prediction data is loaded, you will also get an error message.
3. **Independent Variables in Prediction Data:** Your prediction data file must contain columns with the same names as the **independent variables** that your model was trained with.

What Happens When You Click "Make Prediction":

1. **Data Preparation for Prediction:**

    - The application retrieves the independent variables from your loaded prediction data (`self.prediction_df`).
    - If **feature scaling** was enabled during model training (Step 3) *and* a scaler object was saved (which happens if feature scaling was used during training or optimization), the application will apply the *same* feature scaling transformation (using the saved scaler) to your prediction data's independent variables. This is crucial to ensure that the prediction data is in the same scale as the training data that the model was trained on.

2. **Prediction Generation:**

    - The application uses your trained or loaded model to make predictions on the prepared prediction data (independent variables).

3. **Adding Predictions to Prediction Data:**

    - The predictions generated by the model are added as a new column named **"Predictions"** to your prediction data DataFrame (`self.prediction_df`).

4. **Saving Predictions to File:**

    - **File Explorer Dialog (Save As):** A standard file explorer "Save As" dialog window will open. This window allows you to choose where you want to save the prediction results as a file.
    - **File Type:** The default file extension for saving prediction results is `.csv` (Comma Separated Values). You can also choose "All files (.)" if you prefer a different format.
    - **Choose Save Location and Filename:** Navigate to the folder where you want to save the predictions, enter a filename (e.g., `predictions_output`), and click the **"Save"** button.

    - **Saving Prediction File:** The application will save your `self.prediction_df` (which now includes the "Predictions" column) to a `.csv` file at the location you specified. The index of the DataFrame is not saved to the file.

Information Message: After the predictions are successfully generated and saved, an **"Info"** message box will appear:

- **"Predictions saved to [prediction file path]"** - where `[prediction file path]` is the full path to the `.csv` file where your predictions were saved (e.g., "Predictions saved to C:\Users\YourName\Documents\predictions_output.csv").

Error Handling: If there is an error during the prediction process (e.g., no model loaded, no prediction data loaded, issues with feature scaling during prediction, or problems during prediction generation), an **"Error"** message box titled "Prediction Error" will appear. It will indicate that there was an error during prediction and may include a technical error message. Check that you have loaded a model and prediction data, and that your prediction data contains the correct independent variables.

3. Help Menu and About Dialog

3.1 Help Menu

In the menu bar at the very top of the application window, you will find a **"Help"** menu.

Click on the "Help" menu to open it. Currently, it contains one option:

- **About:** Selecting this option will open the "About" dialog.

3.2 About Dialog

Action: Click on **"Help"** in the menu bar, and then select **"About"**.

What is the About Dialog?

The "About" dialog provides information about the **Machine Learning Model Builder** application. It is a standard way for applications to provide users with details about the software.

Content of the About Dialog:

When you open the "About" dialog, a new window will pop up, typically centered on your main application window. It displays the following information:

- **App Name: "Machine Learning Model Builder"** - The name of the application, in bold and a larger font size.
- **Version: "Version: 1.0"** - Indicates the current version of the application (Version 1.0 in this case).
- **Publisher: "Publisher: Mukund Narayanan / Indian Institute of Technology Roorkee"** - Identifies the developer or organization that created and published the application.
- **Publisher URL:** A clickable link to the website: **"https://www.mukundnarayanan.org/"** - Clicking on this blue, underlined text will open your default web browser and take you to the website of the publisher. This URL might provide more information about the application, the

developer, or other related projects. The cursor will change to a hand icon when you hover over this link to indicate it's clickable.

Closing the About Dialog:

- **"Close" Button:** At the bottom of the "About" dialog, there is a button labeled **"Close"**. Click this button to close the About dialog window and return to the main application window.

Purpose of the About Dialog:

The About dialog is primarily for informational purposes. It helps users understand:

- **What application they are using.**
- **Which version they are using.**
- **Who developed it.**
- **Where to find more information** (through the publisher's URL).

It's a standard element in well-designed applications to provide basic application identity and developer credits.

4. Troubleshooting

This section provides solutions to common problems you might encounter while using the Machine Learning Model Builder.

4.1 Data Loading Errors

Problem: "Error" message box pops up when clicking "Input Data" or "Load Prediction Data," indicating "Failed to load data" or "Failed to load prediction data."

Possible Causes and Solutions:

- **Incorrect File Type:** Ensure you are trying to load a file of a supported type: `.csv`, `.xlsx`, or `.xls`. Check the file extension.
- **Corrupted File:** The data file itself might be corrupted or incomplete. Try opening the file directly with Excel or a text editor to check if it is readable and properly formatted.
- **Incorrect File Path:** Double-check that you have selected the correct file in the file explorer dialog and that the file path is valid.
- **Encoding Issues (for CSV files):** Although the application attempts to handle `utf-8` and `latin1` encodings, very rarely, a CSV file might use a different, less common encoding. If you suspect this, try opening the CSV file in a text editor that allows you to specify encoding (like Notepad++ or Sublime Text). Save the file as `utf-8` encoded CSV and then try loading it again in the application.
- **Excel File Issues (for .xlsx, .xls):** Ensure that the Excel file is not password-protected or corrupted. Try opening it in Microsoft Excel or another spreadsheet program to verify it's readable.
- **Memory Issues (for very large files):** If you are trying to load an extremely large data file that exceeds your computer's memory capacity, the application might fail to load it. Try using a smaller subset of your data for initial testing, or if you are comfortable with programming, consider processing very large datasets in a more memory-efficient way outside of this GUI tool.

4.2 Variable Selection Issues

Problem: Independent or dependent variable listboxes are empty after loading data.

Possible Causes and Solutions:

- **Data Not Loaded Correctly:** If the data loading itself failed (see Section 4.1), the listboxes will not be populated. Go back to Step 1 and ensure data is loaded successfully without errors.
- **File Has No Columns:** In rare cases, the loaded file might be empty or not contain column headers in the first row as expected. Open the file in a text editor (for CSV) or Excel (for Excel files) to verify that it contains data columns with headers.
- **Data Format Issues:** If the file is in a very unusual format that the pandas library (used by the application) cannot recognize, it might fail to extract column names. Ensure your data is in a standard CSV or Excel format.

Problem: Cannot select variables in listboxes or "Confirm Variables" button does nothing.

Possible Causes and Solutions:

- **User Interface Glitch (Rare):** In extremely rare cases, there might be a temporary UI issue. Try restarting the application and reloading your data.
- **Incorrect Selection Method:** Ensure you are using the correct method to select items in the listboxes. You can typically **click and drag** for a range, or **Ctrl+Click** (or Cmd+Click on Mac) for individual items.

4.3 Model Training Errors

Problem: "Error" message box with title "Model Training Error" appears when clicking "Train Model" or "Optimize Hyperparameters."

Possible Causes and Solutions:

- **No Variables Selected:** Make sure you have completed Step 2 and selected both independent and dependent variables and clicked "Confirm Variables." If variables are not selected, the model cannot be trained.
- **Incorrect Hyperparameter Values:** You might have entered invalid hyperparameter values in Step 6. For example:
    - Entering non-numeric values in numeric fields.
    - Entering values outside the expected range for certain parameters (though less common in this UI).
    - Entering tuple values incorrectly for `hidden_layer_sizes` (e.g., not using parentheses or incorrect tuple syntax).
    - Ensure you are entering hyperparameters in the correct format and type as indicated in the interface (e.g., "int", "float", "categorical", "tuple").
- **Incompatible Hyperparameter Combinations:** For some models, certain combinations of hyperparameters might be incompatible or lead to errors. Try using default hyperparameters first, or refer to the documentation for the specific model you are using in `scikit-learn` to understand hyperparameter constraints and dependencies.
- **Data Issues:** In some cases, problems in your data (e.g., missing values that were not properly handled, or data of an unexpected type in columns) might cause training errors. While this application doesn't explicitly include

extensive data cleaning, ensure your data is reasonably clean and appropriate for the chosen model.

- **Algorithm-Specific Issues:** Rarely, there might be numerical instability or other issues with the specific algorithm implementation in `scikit-learn`. This is less common, but if you consistently encounter errors with a particular model, try a different model.
- **Memory Issues (for very complex models or large datasets):** Training very complex models like deep MLPs on large datasets can be memory-intensive. If you suspect memory issues, try using a simpler model or reducing the dataset size for testing.

4.4 Plotting Errors

Problem: "Error" message box with title "Plotting Error" appears when clicking "Plot Accuracies."

Possible Causes and Solutions:

- **Model Not Trained Yet:** You must train a model successfully (Step 7) before you can plot accuracies. If you haven't trained a model, the application won't have any predictions to plot. Ensure you click "Train Model" (or "Optimize Hyperparameters") and receive a "Model Trained Successfully" message before trying to plot.
- **Data or Model Issues:** In very rare cases, there might be issues with the data or model output that prevent the plotting functions from working correctly. If you see a plotting error after successful training, try retraining the model. If the error persists, consider simplifying your data or trying a different model.
- **`matplotlib` Backend Issues:** The plotting functionality relies on the `matplotlib` library. In rare cases, there might be issues with the `matplotlib` backend configuration on your system. This is less likely to be a problem with the `.exe` version, but if you encounter persistent plotting issues, you might need to investigate your `matplotlib` setup if you were using the Python script directly.

4.5 Prediction Errors

Problem: "Error" message box with title "Prediction Error" appears when clicking "Make Prediction."

Possible Causes and Solutions:

- **No Model Loaded or Trained:** You must have a trained model (from Step 7 or 9) or have loaded a saved model (Step 11) before you can make predictions. Ensure you have completed one of these steps.
- **No Prediction Data Loaded:** You must load prediction data using "Load Prediction Data" (Step 12) before clicking "Make Prediction."
- **Prediction Data Mismatch:** Crucially, your prediction data file **must** contain columns with the same names as the **independent variables** that your model was trained on. If the column names in your prediction data do not match the expected independent variables, the model will not be able to process the data for prediction. Double-check that the column names in your prediction data file are correct and match the independent variables you selected in Step 2.
- **Feature Scaling Mismatch:** If you enabled feature scaling during training (Step 3), the application attempts to apply the same scaling to your prediction data.

If for some reason, the scaling process fails during prediction, it might cause an error. This is less common, but if you suspect scaling issues, try disabling feature scaling for testing (if it was enabled).

- **Model-Specific Issues (Rare):** In very rare cases, there might be numerical issues or other problems specific to the model that prevent it from making predictions on new data. If you consistently encounter prediction errors after verifying the points above, try retraining the model or using a different model for prediction.
- **Memory Issues (for very large prediction datasets or complex models):** Making predictions on extremely large datasets or using very complex models might be memory-intensive. If you suspect memory problems, try using a smaller subset of your prediction data for testing.

5. Frequently Asked Questions (FAQ)

Q: What file types are supported for input data and prediction data? A: The application supports `.csv` (Comma Separated Values) files and Excel files (`.xlsx` and `.xls`).

Q: What types of machine learning tasks can I perform with this tool? A: You can perform both **classification** and **regression** tasks. Choose the task type using the "Task Selection" dropdown.

Q: What machine learning models are available? A: The application offers a selection of popular models: * **Classification:** MLP Classifier (Neural Network), Random Forest Classifier, Support Vector Machine (SVC), Decision Tree Classifier. * **Regression:** MLP Regressor (Neural Network), Random Forest Regressor, Support Vector Regression (SVR), Decision Tree Regressor, Linear Regression. The available models in the "Model Selection" dropdown will change depending on whether you select "Classification" or "Regression" for the task.

Q: What are hyperparameters and do I need to set them? A: Hyperparameters are settings that control how a machine learning model learns. They are set before training. You can set hyperparameters in Step 6. If you are unsure, you can try using default hyperparameters (by leaving the fields blank) or use the "Optimize Hyperparameters" feature (Step 9) to automatically find good values. Setting hyperparameters, especially for models like Neural Networks and SVMs, can significantly impact model performance.

Q: What is feature scaling and should I enable it? A: Feature scaling is a preprocessing technique to scale numerical features to a similar range. It is often beneficial, especially for models like Neural Networks and SVMs. It is generally recommended to enable feature scaling (Step 3) if your independent variables have very different ranges of values. Tree-based models (Random Forest, Decision Tree) are generally less sensitive to feature scaling.

Q: What is hyperparameter optimization and how does it help? A: Hyperparameter optimization (Step 9) is an automated process to find the best hyperparameter settings for your chosen model. It uses the `Optuna` library to intelligently search through different hyperparameter values and find the set that gives the best performance (highest accuracy for classification, highest R^2 for regression). This feature can save you a lot of time and effort compared to manually trying out different hyperparameter combinations.

Q: How do I save and load a trained model? A: Use the "Save Model" button (Step 10) to save your model to a `.pkl` file. Use the "Load Model" button (Step 11) to load a previously saved model from a `.pkl` file. Saving and loading models allows you to reuse trained models without retraining them every time.

Q: How do I make predictions on new data? A: First, load your new prediction data using the "Load Prediction Data" button (Step 12). Then, ensure you have a trained or loaded model. Finally, click the "Make Prediction" button (Step 13). The predictions will be saved to a `.csv` file.

Q: What are accuracy and R-squared, and where can I see model performance metrics? A: * **Accuracy** is a metric used for classification tasks. It represents the percentage of predictions that were correct. * **R-squared (R^2)** is a metric used for regression tasks. It represents the proportion of variance in the dependent variable that is predictable from the independent variables. R^2 values range from 0 to 1 (and sometimes can be negative for very poor models), with higher values generally indicating a better fit. After training a model (Step 7), the application will display training and testing accuracy (for classification) or R^2 scores (for regression) in an "Info" message box. You can also visualize model performance by clicking "Plot Accuracies" (Step 8), which generates a confusion matrix for classification and an actual vs. predicted scatter plot for regression.

Q: What is a confusion matrix and how to interpret it? A: A confusion matrix (plotted in Step 8 for classification tasks) is a table that summarizes the performance of a classification model. It shows the counts of True Positives, True Negatives, False Positives, and False Negatives. By examining the confusion matrix, you can understand how well your model is classifying each class and identify potential types of errors (e.g., if it is often confusing one class with another).

Q: What does the "Actual vs. Predicted" plot show for regression tasks? A: The "Actual vs. Predicted" scatter plot (plotted in Step 8 for regression tasks) visualizes the performance of a regression model. Each point on the plot represents a data point, with the x-coordinate being the actual value and the y-coordinate being the model's prediction. The red diagonal line represents perfect predictions. If the points are clustered closely around the red line, it indicates that the model's predictions are close to the actual values, and the model is performing well.

6. Conclusion

Congratulations! You have reached the end of the user manual for the Machine Learning Model Builder. You are now equipped with the knowledge to use this application to load data, build, train, optimize, evaluate, save, and load machine learning models for both classification and regression tasks.

This tool is designed to simplify the model building process and make machine learning more accessible. Experiment with different datasets, models, and hyperparameters to explore the world of machine learning!

For any further questions, feedback, or bug reports, please refer to the publisher's website: https://www.mukundnarayanan.org/.

Thank you for using the Machine Learning Model Builder! ```