1. Frontend (React)

- **User Interface:**
  - **Dataset Upload:** Create a form where users can upload their datasets (e.g., CSV files).
  - **Algorithm Selection:** Provide options for users to select from the available ML algorithms.
  - **Parameters Configuration:** Allow users to configure hyperparameters for the selected algorithms.
  - **Results Display:** Show the results of the ML model's predictions or evaluations.
- **Components:**
  - **UploadForm:** Handles file uploads and sends the file to the backend.
  - **AlgorithmSelector:** Lets users choose from the available algorithms.
  - **ParameterConfig:** Allows users to set hyperparameters.
  - **ResultsDisplay:** Shows the model's results or evaluation metrics.
- **API Calls:**
  - Use `fetch` or `Axios` to send data to the Flask backend and receive results.

2. Backend (Flask + scikit-learn)

- **Endpoints:**
  - **Upload Endpoint:** Receives and processes the uploaded dataset.
    - Parse the dataset and possibly store it temporarily or in a database.
  - **Train Model Endpoint:** Accepts user-selected algorithm and parameters.
    - Load the dataset, train the selected model, and evaluate its performance.
  - **Predict Endpoint:** Accepts new data and uses the trained model to make predictions.
- **Processing Flow:**
  - **Receive Dataset:**
    - Save the uploaded dataset.
    - Perform initial checks (e.g., data format, completeness).
  - **Algorithm Training:**
    - Based on user selection, instantiate the appropriate scikit-learn model.
    - Configure the model with the provided hyperparameters.
    - Train the model on the dataset.
    - Evaluate the model and return performance metrics (e.g., accuracy, precision).
  - **Prediction:**
    - Use the trained model to make predictions on new input data.

- **Integration:**
  - Use Flask routes to handle requests from the React frontend.
  - Ensure proper data validation and error handling.

3. ML Algorithms (Initial 5 Algorithms)

- **Classification Algorithms:**
  - **Logistic Regression**
  - **Decision Tree Classifier**
  - **Random Forest Classifier**
  - **Support Vector Machine (SVM)**
  - **K-Nearest Neighbors (KNN)**
- **Considerations:**
  - Each algorithm may have different hyperparameters that users can configure.
  - Implement model training and evaluation for each selected algorithm.

4. Additional Features

- **Model Persistence:** Optionally save and load trained models for reuse.
- **Visualization:** Provide visualizations of results (e.g., confusion matrix, ROC curves).
- **Error Handling:** Implement robust error handling and user feedback mechanisms.

Deployment:

- **Frontend:** Build and deploy your React application.
- **Backend:** Deploy your Flask application (e.g., using Heroku, AWS, or another cloud service).

Workflow Example:

1. **User uploads a dataset** via the React frontend.
2. **Frontend sends the dataset** to the Flask backend.
3. **Flask saves the dataset** and returns a success message.
4. **User selects an algorithm** and configures parameters.
5. **Frontend sends the algorithm choice and parameters** to the Flask backend.
6. **Flask trains the model** using scikit-learn and returns evaluation metrics.
7. **Frontend displays the results** and possibly allows the user to make predictions with new data.