i want to use add multiple buttons for each model and dependin upon the clicked button use that model give pridiction

**ChatGPT said:**

Perfect! You can absolutely do that — in fact, it's a great idea for giving flexibility in your app. Here's how you can structure it:

**✅ Step-by-step Plan:**

**1. Train & Save All Models Separately**

In your .ipynb or Python script, save each trained model to its own file:

python

CopyEdit

import pickle

# Save Decision Tree

pickle.dump(ds, open('dt\_model.pkl', 'wb'))

# Save Naive Bayes

pickle.dump(NaiveBayes, open('nb\_model.pkl', 'wb'))

# Save SVM

pickle.dump(SVM, open('svm\_model.pkl', 'wb'))

# Save Logistic Regression

pickle.dump(LogReg, open('logreg\_model.pkl', 'wb'))

# Save Random Forest

pickle.dump(RF, open('rf\_model.pkl', 'wb'))

# Save the Label Encoder too

pickle.dump(encode\_ferti, open('fertilizer.pkl', 'wb'))

**2. Frontend (React / HTML + JS)**

Create multiple buttons for each model:

html

CopyEdit

<button onclick="getPrediction('rf')">Random Forest</button>

<button onclick="getPrediction('dt')">Decision Tree</button>

<button onclick="getPrediction('nb')">Naive Bayes</button>

<button onclick="getPrediction('svm')">SVM</button>

<button onclick="getPrediction('logreg')">Logistic Regression</button>

Each button calls a function like getPrediction(modelName) and sends a request to the backend with that model name.

**3. Backend (Python Flask or Node.js with Python)**

Create an API that:

* Accepts model\_name and input features as POST data
* Loads the right model file (\*.pkl) using pickle.load()
* Makes the prediction and returns the result

Example using Flask:

python

CopyEdit

from flask import Flask, request, jsonify

import pickle

import numpy as np

app = Flask(\_\_name\_\_)

@app.route('/predict', methods=['POST'])

def predict():

data = request.get\_json()

model\_name = data['model'] # 'rf', 'dt', etc.

features = np.array(data['features']).reshape(1, -1)

model\_file\_map = {

'rf': 'rf\_model.pkl',

'dt': 'dt\_model.pkl',

'nb': 'nb\_model.pkl',

'svm': 'svm\_model.pkl',

'logreg': 'logreg\_model.pkl'

}

model = pickle.load(open(model\_file\_map[model\_name], 'rb'))

prediction = model.predict(features)

label\_encoder = pickle.load(open('fertilizer.pkl', 'rb'))

fertilizer\_name = label\_encoder.inverse\_transform(prediction)[0]

return jsonify({'prediction': fertilizer\_name})

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

**4. React or JS Frontend (API Call Example)**

javascript

CopyEdit

function getPrediction(modelName) {

const input = [25, 78, 43, 4, 1, 22, 26, 38]; // replace with actual input

fetch("http://localhost:5000/predict", {

method: "POST",

headers: {

"Content-Type": "application/json"

},

body: JSON.stringify({

model: modelName,

features: input

})

})

.then(res => res.json())

.then(data => alert("Predicted Fertilizer: " + data.prediction))

.catch(err => console.error(err));

}