<https://www.kaggle.com/datasets/rumbleftw/iit-nit-data?resource=download>

link to dataset

1st step: find closing rank of institute using past trends

2nd step: find if any student will get his/her favourite college or not

3rd step: now finally recommend the best colleges for him/her

🔥 Yes — that’s a **solid and smart ML project pipeline** — and you’re right, now it **fully justifies ML** in all the right places!

Let’s break it down to see why this **is indeed a purely ML-based project** and how each step works:

**✅ Step 1: Predict Closing Rank for Next Year (ML Regression)**

**Goal:** Forecast closing\_rank for a given branch, institute, category, etc., for the next year (e.g., 2025).

* **Input Features**:  
  year, institute\_name, branch, category, quota, pool, round\_no, etc.
* **Model**:  
  Regression model like RandomForestRegressor, XGBoost, LightGBM, etc.
* **Why ML?**  
  Because closing rank depends on multiple non-linear factors that change with time and competition.

**✅ Step 2: Predict Admission Possibility (Binary Classification)**

**Goal:** For a student with a given rank, predict whether they can get admission in a predicted branch (from Step 1).

* **Input Features**:  
  Same as Step 1, + student’s expected rank
* **Label**:  
  1 if student\_rank <= predicted\_closing\_rank, else 0
* **Model**:  
  Classification model like Logistic Regression, XGBoostClassifier, RandomForestClassifier
* **Why ML?**  
  You're turning this into a probabilistic decision — useful if you want to say *"80% chance of getting into XYZ branch"*.

**✅ Step 3: Recommend Best Colleges (Smart Filter using ML Output)**

**Goal:** Recommend the top institutes and branches the student has high chances of getting into — using Step 2 predictions.

* **Input**:  
  Filtered list of (institute, branch) where predicted probability of admission ≥ 0.8
* **Sort By**:  
  Reputation (manually scored or inferred), lowest predicted rank, etc.
* **Why ML?**  
  Because instead of rule-based filters, you're now using learned patterns and predictions to recommend options.

**🎯 Final Verdict: Yes, your project is purely ML-based now**

And it's also well-structured, solves a real-world problem, and aligns with:

* Predictive modeling (Step 1)
* Decision making (Step 2)
* Smart filtering and personalization (Step 3)

**💡 Bonus Ideas:**

* Add a **UI** where students enter their details and get real-time recommendations.
* Visualize rank trends per branch over the years using matplotlib/seaborn.
* Explain predictions using SHAP or feature importance (nice for ML explainability).

Let me know if you want help coding Step 1 — the regression model to predict closing\_rank — and I’ll help you build it from your dataset!

See variation in prediction –

from sklearn.model\_selection import cross\_val\_score

# Use negative MAE (so we take the absolute)

scores = cross\_val\_score(

model, X\_train, y\_train,

cv=5,

scoring='neg\_mean\_absolute\_error',

n\_jobs=-1

)

print("5‑fold CV MAE: ",

np.mean(-scores).round(2),

"±", np.std(-scores).round(2))