Case Study: Trek Recommendation using Decision Trees

Nepal Tourism

Course: AI and Tourism – MIT-AI @ Gandaki University

Bidur Devkota, PhD
GCES Pokhara

January 20, 2026

1) Case Study

Brief Scenario

A Nepal tourism startup builds a web/mobile app to help tourists find suitable Himalayan treks from a catalog dataset.

DS Pipeline Tasks

- **Preprocess raw fields** (`Time`, `Max Altitude`, `Cost`) into numeric features.
- **Create target label** `suitability_level` (Beginner/Intermediate/Advanced) using rule-based logic.
- **Train a Decision Tree** classifier with a train-test split using `stratify=y`.
- **Evaluate the classifier** using accuracy and classification report.
- **Recommend trek names** using a two-step method: (1) predict level, (2) rank matching treks by Euclidean distance.

2) Preprocessing

Tasks

- **Clean Time** (e.g., "10 Days") → extracts digits to integer days.
- **Clean Max Altitude** (e.g., "5416m") → strips non-digits to integer meters.
- **Handle** `difficulty_encoded` → maps from Trip Grade keywords, else fallback encoding.
- **Create target** `suitability_level` → rule-based label using grade/altitude/time thresholds.

2) Preprocessing

Domain rules (def assign_suitability())

Threshold	Appears in Code	Interpretation
8 days	`time` $\leq$ 8	Short-duration treks, typically suitable for beginner trekkers
3500 m	`altitude` $\leq$ 3500	Below common altitude sickness risk zone; minimal acclimatization required
14 days	`time` $\leq$ 14	Medium-length treks requiring moderate endurance
5000 m	`altitude` $\leq$ 5000	High-altitude treks commonly attempted by experienced trekkers

Note

These thresholds are **heuristic, domain-driven rules** used to generate labels when no ground-truth suitability data is available.

(2) Preprocessing: suitability level?

Key Idea

The suitability level (**Beginner / Intermediate / Advanced**) is created using **rule-based heuristics** based on trek duration, maximum altitude, and trip grade. These labels are later used to train the Decision Tree.

Rule Logic Used in the Code

- **Beginner:** Easy trip grade **OR** duration $\leq$ 8 days **AND** altitude $\leq$ 3500 m
- **Intermediate:** Moderate trip grade **OR** duration $\leq$ 14 days **AND** altitude $\leq$ 5000 m
- **Advanced:** All remaining treks (longer duration or higher altitude)

Condition	Assigned Level
Easy grade OR (short + low altitude)	Beginner
Moderate grade OR (medium + high altitude)	Intermediate
Else (very long / very high altitude)	Advanced

Important Note These thresholds are **heuristic and domain-driven**, not learned from data. They provide interpretable labels when no ground-truth suitability

2) Preprocessing

Do we need to do the following Tasks ?

- **Cost cleaning robustness:** Does the parser correctly handle ranges like "$1200–$1500"?
  Hint: use regex to extract all numbers, then average; handle missing values safely.
- **Justify rule thresholds:** Why choose 8 days / 3500m / 5000m cutoffs?
  Hint: cite trekking practice + validate thresholds with domain experts or data distributions.

2) Model + Evaluation + Recommendation Logic

Tasks the code answers () and what it does

- **Feature set used:** `Time`, `Max Altitude`, `Cost_numeric`, `difficulty_encoded`.
- **Train/test split with stratify:** uses `stratify=y` to preserve class proportions.
- **Evaluation:** prints accuracy + `classification_report` (precision/recall/F1 by class).
- **Two-step recommendation:** predict suitability → filter treks by predicted level → rank by Euclidean distance.

Two-step recommendation: The code first uses a Decision Tree to predict the user's suitability level (Beginner/Intermediate/Advanced), then filters the trek database to only treks of that level, and finally ranks those treks by Euclidean distance from the user's preferences (days, altitude, cost, difficulty) to return the closest matches.

2) Model + Evaluation + Recommendation Logic

Example of Two-step recommendation (sample only)
The system first predicts the user's **suitability level** using a Decision Tree, then filters treks of that level and ranks them by **Euclidean distance** from the user's preferences.

Step	What the Code Does	Toy Example
1	Predict suitability level using Decision Tree	User input: 10 days, 4200 m, $1200, Moderate ⇒ **Intermediate**
2	Filter treks with predicted level	Filtered treks: ABC, Langtang Valley, Manaslu Circuit
3	Compute Euclidean distance on numeric features	$d_{\text{user,ABC}} = 0.8$ $d_{\text{user,Langtang}} = 0.4$ $d_{\text{user,Manaslu}} = 1.6$
4	Rank treks by smallest distance	**Recommendation:** Langtang Valley, then ABC Trek

2) Model + Evaluation + Recommendation Logic

DO we still need to do the following Tasks?

- **? Why Decision Tree (3 advantages) over RF/LR/SVM/NN?**
  Hint: interpretability (rule-like), nonlinearity, simple deployment; explainable to tourists/agents.
- **? Is 82–90% accuracy sufficient for production? What else to check?**
  Hint: confusion matrix, per-class recall (Beginner safety), misclassification costs, user feedback.
- **? Why Euclidean distance is "similarity"?**
  Hint: closer in (days, altitude, cost, difficulty) ⇒ closer to user preference profile.
- **? Limitations + improvements of similarity ranking?**
  Hint: scale dominance → standardize/weight features; use cosine/kNN; add categorical prefs.

3) Nepal Tourism Implications

What the app can now do (practical impact)

- **Faster shortlist for tourists:** converts preferences into *Beginner/Intermediate/Advanced* + returns 3–5 trek names.
- **Safer matching:** beginners can be guided away from very high altitude/long duration treks.
- **Higher conversion potential:** showing top matches reduces confusion and drop-off during planning.
- **Explainable recommendations:** decision tree splits can be explained as simple rules (days/altitude/budget).

Caution (important in Nepal trekking context)

Euclidean ranking without scaling/weights may over-emphasize altitude or cost; wrong ranking may frustrate users.

4) Summary: Done vs. To-Do

What has been done in the code (deliverables)

1. Cleaned numeric features: **Time**, **Max Altitude**, **Cost** (partially), and **Trip Grade** encoding.
2. Created a target label: **suitability_level** using rules.
3. Trained and evaluated a **Decision Tree** with stratified splitting.
4. Produced **top-N trek recommendations** via classification + distance-based ranking.

4) Summary: Done vs. To-Do

What YOU need to do (as part of the assignment)

1. **Fix/verify cost parsing** for ranges/currencies; handle missing/invalid values.
2. **Write justification** for using Decision Trees (3+ reasons) in tourism context.
3. **Add evaluation beyond accuracy:** confusion matrix, per-class recall, misclassification cost.
4. **Improve similarity ranking:** standardize/weight features; consider categorical prefs (season, accommodation).
5. **Discuss deployment impact & extra features:** user profile, past trips, group type, season/month, reviews, safety/weather.

End

Questions?