SmartLearn AI: AI-driven Personalized Learning

Adaptive Content Recommendations Using Machine Learning

Student Name: *Ashmita Luthra*

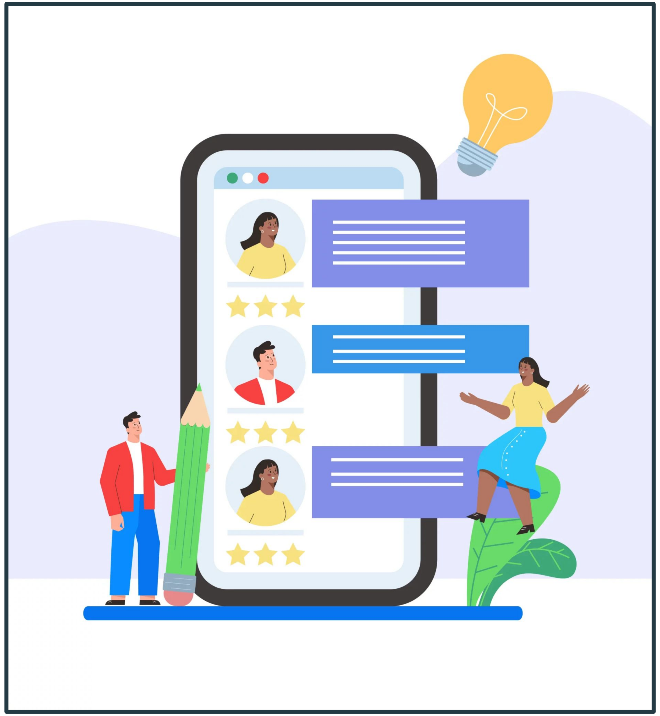
Roll No: *iitrpr_ai_25010035*

Mentor Name: Niranjan Deshpandey

Problem Overview

- Students learn at different paces and styles.
- Traditional platforms deliver uniform content → poor engagement.
- Educators need tools to personalize learning without extra effort.

Objective & AI Task



- Recommend next learning level (Easy / Medium / Hard).
- Multi-class classification problem.
- Support educators in content personalization, not replace them.

System Design

Input: Student features (quiz score, engagement, attempts, time spent, learning style)

Processing: Preprocessing, encoding, scaling

Model: Random Forest classifier

Output: Predicted next learning level

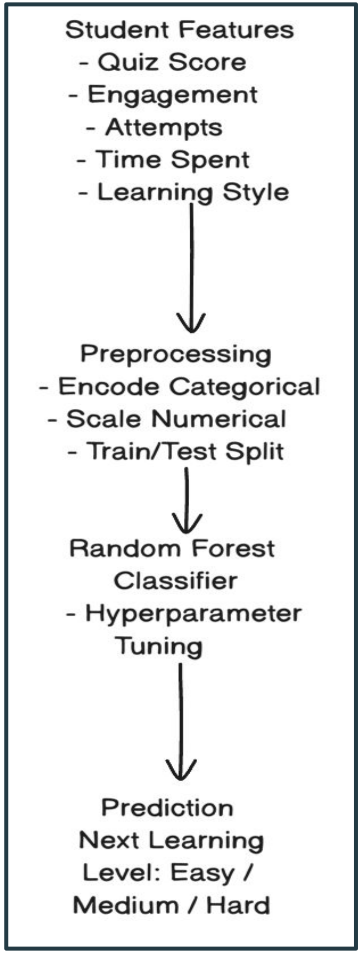


Fig1: Workflow

Data Overview

Synthetic dataset of 10,000 student records

Features include:

- quiz_score
- engagement_score
- attempts
- time_spent
- learning_style
- current_difficulty_level

	student_id	topic	learning_style	difficulty_level	quiz_score	time_spent_minutes	attempts	engagement_score	next_content_level
0	1	Biology	Visual	Hard	78.3	35.1	4	51.7	Medium
1	2	Chemistry	Practice	Easy	51.8	49.3	1	45.7	Medium
2	3	Computer Science	Visual	Easy	65.7	24.2	2	44.1	Medium
3	4	Chemistry	Visual	Medium	69.3	30.8	3	46.9	Medium
4	5	Chemistry	Textual	Medium	69.6	45.8	4	50.5	Medium

Fig2: Dataset (10K entries)

Preprocessing steps:

- Encode categorical features using Label Encoding
- Standardize numerical features with StandardScaler

Train-test split: 80-20 stratified

Model Design

Algorithm: Random Forest Classifier

Reason for choice:
Handles non-linear relationships

- Robust to overfitting
- Interpretability through feature importance

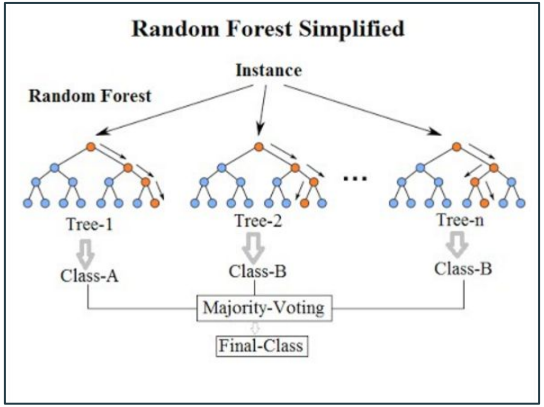
Hyperparameter tuning:

- Number of trees (`n_estimators`)
- Maximum depth (`max_depth`)
- Minimum samples per split (`min_samples_split`)

Key predictors: `quiz_score`, `engagement_score`

Explainable AI: Feature importance used to understand which factors influence predictions

```
RandomForestClassifier
RandomForestClassifier(n_jobs=-1, random_state=42)
```




```
# Initialize model
rf_model = RandomForestClassifie
    n_estimators=100,
    random_state=42,
    n_jobs=-1
)

# Train model
rf_model.fit(X_train, y_train)
```

Demo Snapshots

Preprocessing & Dataset Overview

	topic	learning_style	difficulty_level	quiz_score	time_spent_minutes	attempts	engagement_score
0	0	2	1	78.3	35.1	4	51.7
1	1	0	0	51.8	49.3	1	45.7
2	2	2	0	65.7	24.2	2	44.1
3	1	2	2	69.3	30.8	3	46.9
4	1	1	2	69.6	45.8	4	50.5

Model Training

```
rf_tuned = RandomForestClassifier(
    n_estimators=300,
    max_depth=12,
    min_samples_split=5,
    min_samples_leaf=2,
    random_state=42,
    n_jobs=-1
)

rf_tuned.fit(X_train, y_train)

y_pred_tuned = rf_tuned.predict(X_test)

accuracy_score(y_test, y_pred_tuned)
```

```
0.6295
```

```
print(classification_report(
    y_test,
    y_pred_tuned,
    target_names=target_encoder.classes_
))
```

```
              precision    recall  f1-score   support

        Easy       0.69      0.30      0.42       583
        Hard       0.73      0.61      0.66       411
      Medium       0.59      0.83      0.69      1006

    accuracy                           0.63      2000
   macro avg       0.67      0.58      0.59      2000
weighted avg       0.65      0.63      0.61      2000
```

Prediction Example

```
# Sample new student input
sample_student = pd.DataFrame([{
    "topic": "Computer Science",
    "learning_style": "Visual",
    "difficulty_level": "Medium",
    "quiz_score": 68,
    "time_spent_minutes": 55,
    "attempts": 2,
    "engagement_score": 72
}])
```

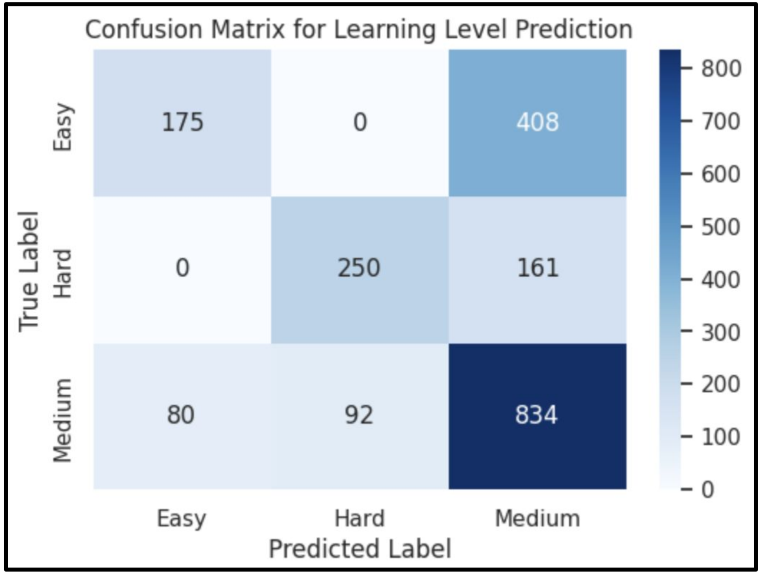
```
def explain_recommendation(student, recommendation):
    reasons = []

    if student["quiz_score"].values[0] < 70:
        reasons.append("moderate quiz performance")
    if student["engagement_score"].values[0] > 70:
        reasons.append("good engagement")
    if student["attempts"].values[0] > 2:
        reasons.append("multiple attempts required")

    explanation = (
        f"The system recommends **{recommendation}** level content due to "
        + ", ".join(reasons) + "."
    )
    return explanation


explain_recommendation(sample_student, recommended_level[0])
```

```
'The system recommends **Easy** level content due to moderate quiz performance, good engagement.'
```

Results & Evaluation

- **Overall Accuracy: 62.95%**
- Most misclassifications occur between adjacent levels (**Easy** ↔ **Medium**)
- Key predictors: `quiz_score` and `engagement_score`
- Confusion matrix (optional visual) shows realistic learning uncertainty

	precision	recall	f1-score	support
Easy	0.69	0.30	0.42	583
Hard	0.73	0.61	0.66	411
Medium	0.59	0.83	0.69	1006
accuracy			0.63	2000
macro avg	0.67	0.58	0.59	2000
weighted avg	0.65	0.63	0.61	2000



Key Learnings

- AI can **realistically personalize learning** while accounting for uncertainty.
- **Feature importance** helps make the model explainable to educators.
- Synthetic datasets are effective for **prototyping educational AI systems**.
- **Human-in-the-loop** approach is crucial: AI supports, not replaces, teachers.
- Misclassifications between adjacent levels reflect **natural learning progression**.

LLM Evaluation & Future Work

LLM Integration (Future Scope):

- Generate **personalized content** for each student
- Suggest **explanations, examples, or exercises** tailored to learning style

Future Improvements:

- Implement **reinforcement learning** for continuous adaptation
- Use **real-world educational datasets** for more accurate recommendations
- Extend recommendations to **content format** (text, video, exercises)
- Integrate **interactive dashboards** for educators