

```

#
=====
# Step 1: Import Libraries and Load Kaggle Dataset
#
=====
import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.impute import SimpleImputer

# Machine Learning Models
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier

# Feature Selection Methods
from sklearn.feature_selection import SelectKBest, f_classif, RFE

# Evaluation Metrics
from sklearn.metrics import accuracy_score, classification_report

# Load the Kaggle dataset
df = pd.read_csv('xAPI-Edu-Data.csv')
print("Dataset Loaded Successfully. Shape:", df.shape)

#
=====
# Step 2: Handle Missing Values & Preprocessing
#
=====
# 1. Impute missing numerical values using Mean
num_cols = ['raisedhands', 'VisITedResources', 'AnnouncementsView',
            'Discussion']
imputer_num = SimpleImputer(strategy='mean')
df[num_cols] = imputer_num.fit_transform(df[num_cols])

# 2. Impute missing categorical values using Most Frequent (Mode)
cat_missing = ['NationalITY']
imputer_cat = SimpleImputer(strategy='most_frequent')
df[cat_missing] = imputer_cat.fit_transform(df[cat_missing])

# Separate Features (X) and Target (y)
X = df.drop(columns=['Class'])
y = df['Class']

```

```

# Encode target column 'Class' (Low, Medium, High -> 0, 1, 2)
le = LabelEncoder()
y_encoded = le.fit_transform(y)

# Convert categorical features into numeric using One-Hot Encoding
X_encoded = pd.get_dummies(X, drop_first=True)

# Standardize feature values
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_encoded)

# Split dataset (80% Train, 20% Test)
X_train, X_test, y_train, y_test = train_test_split(X_scaled,
y_encoded, test_size=0.2, random_state=42)
print("Data preprocessing completed successfully.")

#
=====
# Step 3: Apply Machine Learning Models with GridSearchCV
#
=====
print("\n--- Step 3: Base Machine Learning Models with GridSearch
---")

# 1. Random Forest Classifier
rf = RandomForestClassifier(random_state=42)
rf_params = {'n_estimators': [50, 100], 'max_depth': [None, 10, 20]}
grid_rf = GridSearchCV(rf, rf_params, cv=5, scoring='accuracy')
grid_rf.fit(X_train, y_train)

best_rf = grid_rf.best_estimator_
y_pred_rf = best_rf.predict(X_test)
print("Random Forest Accuracy:", round(accuracy_score(y_test,
y_pred_rf), 4))

# 2. Logistic Regression
lr = LogisticRegression(max_iter=1000, random_state=42)
lr_params = {'C': [0.1, 1.0, 10.0]}
grid_lr = GridSearchCV(lr, lr_params, cv=5, scoring='accuracy')
grid_lr.fit(X_train, y_train)

best_lr = grid_lr.best_estimator_
y_pred_lr = best_lr.predict(X_test)
print("Logistic Regression Accuracy:", round(accuracy_score(y_test,
y_pred_lr), 4))

#

```

```

=====
# Step 4: Use Two Feature Selection Methods
#
=====
print("\n--- Step 4 & 5: Feature Selection & Model Evaluation ---")

# Method 1: SelectKBest (Filter Method using ANOVA F-value)
selector_kbest = SelectKBest(score_func=f_classif, k=10)
X_train_kbest = selector_kbest.fit_transform(X_train, y_train)
X_test_kbest = selector_kbest.transform(X_test)

# Method 2: Recursive Feature Elimination (RFE - Wrapper Method)
rfe = RFE(estimator=DecisionTreeClassifier(random_state=42),
n_features_to_select=10)
X_train_rfe = rfe.fit_transform(X_train, y_train)
X_test_rfe = rfe.transform(X_test)

#
=====
# Step 5: Evaluate Models on Selected Features
#
=====
# Evaluating Random Forest on SelectKBest Features
grid_rf.fit(X_train_kbest, y_train)
best_rf_kbest = grid_rf.best_estimator_
y_pred_kbest = best_rf_kbest.predict(X_test_kbest)
print("Random Forest Accuracy (SelectKBest Features):",
round(accuracy_score(y_test, y_pred_kbest), 4))

# Evaluating Random Forest on RFE Features
grid_rf.fit(X_train_rfe, y_train)
best_rf_rfe = grid_rf.best_estimator_
y_pred_rfe = best_rf_rfe.predict(X_test_rfe)
print("Random Forest Accuracy (RFE Features):",
round(accuracy_score(y_test, y_pred_rfe), 4))

# Classification Report for Final Selected Features
print("\nFinal Model Classification Report (SelectKBest):")
print(classification_report(y_test, y_pred_kbest,
target_names=le.classes_))

Dataset Loaded Successfully. Shape: (480, 17)
Data preprocessing completed successfully.

--- Step 3: Base Machine Learning Models with GridSearch ---
Random Forest Accuracy: 0.5833

```

Logistic Regression Accuracy: 0.6042

--- Step 4 & 5: Feature Selection & Model Evaluation ---

Random Forest Accuracy (SelectKBest Features): 0.6562

Random Forest Accuracy (RFE Features): 0.6458

Final Model Classification Report (SelectKBest):

	precision	recall	f1-score	support
H	0.74	0.54	0.62	26
L	0.70	0.64	0.67	25
M	0.61	0.73	0.67	45
accuracy			0.66	96
macro avg	0.68	0.64	0.65	96
weighted avg	0.67	0.66	0.65	96