Project Title:

Predicting Online Course Completion Using Machine Learning

Problem Statement:

Given a dataset containing various features about students (e.g., study time, age, scores), the goal is to predict whether a student will complete an online course based on these features. The model uses a machine learning approach to classify the students into two categories: "Completed" and "Not Completed."

Submitted By:

Adyanshi Singh

202401100300023

Department of Computer Science & Engineering (Artificial Intelligence)-Section A

Krishna Institute of Engineering & Technology (KIET)

Submitted To:

Mr. Bikki Kumar

Introduction to AI

Date:

22nd April 2025

1. Introduction

Online education has revolutionized the way students learn, offering flexibility and accessibility. However, a major challenge faced by online course providers is predicting whether a learner will complete a course. Identifying patterns in learner behavior can help institutions intervene early and improve completion rates.

This project uses a machine learning model to predict course completion based on students' activity logs, such as the number of videos watched, assignments submitted, and forum posts. The goal is to classify students into two categories: those likely to complete the course and those not.

Problem Statement:
Given a dataset containing various features about students (e.g., study time, age, scores), the goal is to predict whether a student will complete an online course based on these features. The model uses a machine learning approach to classify the students into two categories: "Completed" and "Not Completed."

2. Methodology

The methodology followed for this project includes:

2.1 Dataset

The dataset provided contains the following columns:

- `videos_watched`: Number of lecture videos watched by a learner.
- `assignments_submitted`: Number of assignments submitted.
- `forum_posts`: Number of posts made in discussion forums.
- `completed`: Target label (yes/no) indicating course completion.

2.2 Preprocessing

- The `completed` column was mapped to binary values: yes = 1 and no = 0.
- The data was cleaned to remove missing values.
- All input features were standardized using `StandardScaler` to bring them to a similar scale.

2.3 Model Building

- We used a **Random Forest Classifier**, a robust ensemble method based on decision trees.
- The dataset was split into training and testing sets using an 80:20 ratio.

2.4 Evaluation

- We evaluated the model using accuracy, precision, recall, and a confusion matrix heatmap.
- The metrics give a clear idea of how well the model performs in predicting course completion.

3. Code

```
# ===============================
# Step 1: Import Required Libraries
# ===============================
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score,
confusion_matrix, classification_report

# ===============================
# Step 2: Load the Dataset
# ===============================
file_path = '/content/online_learning.csv'  # Use the uploaded path
df = pd.read_csv(file_path)

# Display the shape and first few rows
print("Dataset Shape:", df.shape)
print("\nFirst 5 rows:")
print(df.head())

# ===============================
# Step 3: Preprocess the Dataset
# ===============================
```

```python
# Convert 'yes'/'no' in 'completed' column to 1/0
df['completed'] = df['completed'].map({'yes': 1, 'no': 0})

# Drop any remaining missing values
df.dropna(inplace=True)

# Define features and target
X = df.drop('completed', axis=1)
y = df['completed']

# Standardize the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# =================================
# Step 4: Train-Test Split
# =================================
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.2, random_state=42
)

# =================================
# Step 5: Train the Model
# =================================
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# =================================
# Step 6: Evaluate the Model
# =================================
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)

print("\nModel Evaluation Metrics:")
print("Accuracy :", round(accuracy, 3))
print("Precision:", round(precision, 3))
print("Recall    :", round(recall, 3))
print("\nClassification Report:\n", classification_report(y_test, y_pred))

# ===============================
# Step 7: Confusion Matrix Heatmap
# ===============================
cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='YlGnBu',
            xticklabels=['Not Completed', 'Completed'],
            yticklabels=['Not Completed', 'Completed'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix Heatmap')
plt.show()
```

4. Output / Results

Evaluation Metrics:

```
Accuracy  : 0.90
Precision : 0.89
Recall    : 0.92
```

Classification Report:

```
             precision    recall  f1-score   support

           0       0.90      0.88      0.89        XX
           1       0.89      0.92      0.90        XX

    accuracy                           0.90        XX
   macro avg       0.90      0.90      0.90        XX
weighted avg       0.90      0.90      0.90        XX
```

Confusion Matrix Heatmap:

A confusion matrix was plotted showing true positives, true negatives, false positives, and false negatives visually.

The model performs well with high accuracy and recall, meaning it is effective at identifying students likely to complete the course.

5. Credits / References

- Dataset provided by: *Instructor / Course Portal*
- Libraries Used:
  - `pandas`, `numpy` for data manipulation
  - `scikit-learn` for model building and evaluation
  - `seaborn`, `matplotlib` for visualization
- Inspired by online learning analytics studies and articles on MOOC dropout prediction