**Assessment Report**

On

“Predict Online Learning Completion”

submitted as partial fulfillment for the award of

**BACHELOR OF TECHNOLOGY**

**DEGREE**

SESSION 2024-25

in

**CSE(AI)**

By

Name : Rishabh Sharma

Roll Number : 202401100300200

Section: C

**KIET Group of Institutions, Ghaziabad**

**1. Introduction**

In recent years, online learning has become an increasingly popular and flexible way to acquire new skills and knowledge. However, despite its widespread adoption, one of the challenges faced by online education platforms is the high dropout rate among learners. Many students begin online courses but fail to complete them, often due to various factors such as lack of engagement, insufficient motivation, or difficulties in managing time.

To address this challenge, predicting which learners are likely to complete a course can help educational institutions or online platforms implement timely interventions. By analyzing learner activity patterns, we can forecast course completion based on key factors such as videos watched, assignments submitted, and forum participation.

This report explores how predictive analytics can be applied to identify learners at risk of dropping out, thus allowing instructors to offer targeted support to improve course completion rates.

**3. Methodology**

This section outlines the methodology used for analyzing the data and building a predictive model for online learning completion.

**Data Collection:**

The dataset used for this analysis contains several key features that reflect learner activity on an online course:

* videos\_watched: Number of videos watched by the learner during the course.
* assignments\_submitted: Number of assignments the learner submitted during the course.
* forum\_posts: Number of forum posts made by the learner.
* completed: Target variable indicating whether the learner successfully completed the course ("yes" or "no").

**Data Preprocessing:**

1. **Data Cleaning:**  
   The dataset was cleaned to handle missing or irrelevant data. We ensured that all entries were complete and relevant for the analysis.
2. **Target Variable Encoding:**  
   The target variable completed (which is categorical) was encoded into binary values:
   * "yes" = 1
   * "no" = 0
3. **Feature Selection:**  
   We selected three key features for the prediction model: videos\_watched, assignments\_submitted, and forum\_posts.
4. **Data Splitting:**  
   The dataset was divided into two subsets: a training set (80%) used to train the model, and a testing set (20%) used to evaluate the model's performance.

**Modeling:**

A **Logistic Regression** model was selected for this binary classification problem. Logistic regression is well-suited for predicting binary outcomes (such as "yes" or "no") and can provide insights into which factors most significantly affect course completion.

**4. Code**

The following Python code was used to preprocess the data, train the logistic regression model, and evaluate its performance.

# Import pandas for handling tabular data

import pandas as pd

# Import numpy for numerical operations

import numpy as np

# Import matplotlib and seaborn for visualization

import matplotlib.pyplot as plt

import seaborn as sns

# Import scikit-learn modules for model training and evaluation

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification\_report, confusion\_matrix, accuracy\_score

# Step 1: Load the CSV file into a DataFrame

# Make sure the file 'online\_learning.csv' is in the same folder as this script

df = pd.read\_csv("online\_learning.csv")

# Step 2: Display the first 5 records to understand the structure

print("First 5 records in the dataset:")

print(df.head())

# Step 3: Convert the target column 'completed' from 'yes'/'no' to 1/0

df['completed'] = df['completed'].map({'yes': 1, 'no': 0})

# Step 4: Define input features (X) and target variable (y)

X = df[['videos\_watched', 'assignments\_submitted', 'forum\_posts']]  # Features

y = df['completed']  # Target variable

# Step 5: Split the dataset into training and testing sets (80% training, 20% testing)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Step 6: Create and train a Random Forest Classifier

model = RandomForestClassifier(n\_estimators=100, random\_state=42)  # 100 trees in the forest

model.fit(X\_train, y\_train)  # Train the model using training data

# Step 7: Predict on the test data

y\_pred = model.predict(X\_test)  # Get predictions (0 or 1)

# Step 8: Convert predictions and actual labels from 0/1 to 'no'/'yes'

y\_pred\_labels = np.where(y\_pred == 1, "yes", "no")

y\_test\_labels = np.where(y\_test == 1, "yes", "no")

# Step 9: Print accuracy of the model

accuracy = accuracy\_score(y\_test\_labels, y\_pred\_labels)

print(f"\nModel Accuracy: {accuracy:.2f}")

# Step 10: Print classification report (precision, recall, F1-score)

print("\nClassification Report:")

print(classification\_report(y\_test\_labels, y\_pred\_labels))

# Step 11: Print confusion matrix

print("Confusion Matrix:")

print(confusion\_matrix(y\_test\_labels, y\_pred\_labels))

# Step 12: Visualize the importance of each feature in the prediction

importances = model.feature\_importances\_  # Get feature importance scores

features = X.columns  # Get feature names

# Plot a bar chart for feature importance

plt.figure(figsize=(8, 5))

sns.barplot(x=importances, y=features)

plt.title("Feature Importance from Random Forest")

plt.xlabel("Importance Score")

plt.ylabel("Features")

plt.tight\_layout()

plt.show()

**5. Output**

The logistic regression model was trained and tested on the dataset, and the following results were obtained:

First 5 records in the dataset:

videos\_watched assignments\_submitted forum\_posts completed

0 11 6 5 yes

1 43 1 11 no

2 37 1 8 no

3 18 4 14 yes

4 6 4 15 yes

Model Accuracy: 0.35

Classification Report:

precision recall f1-score support

no 0.25 0.43 0.32 7

yes 0.50 0.31 0.38 13

accuracy 0.35 20

macro avg 0.38 0.37 0.35 20

weighted avg 0.41 0.35 0.36 20

Confusion Matrix:

[[3 4]

[9 4]]

