
# Assessment Report

on

# "Predict Club Participation"

submitted as partial fulfillment for the award of

# BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

# CSE(AI)

By

Name : Harsh Kumar Mishra

Roll Number : 202401100300114

Section: B

Under the supervision of

# "Mr. Shivansh Prasad"

KIET Group of Institutions, Ghaziabad

May, 2025

## 1. Introduction

This project explores how machine learning can help in understanding student involvement in extracurricular activities. By predicting club participation using demographic and academic features, the model aids institutions in better planning and engagement strategies.

## 2. Problem Statement

To predict whether a student is likely to participate in a club based on their academic and demographic information.

## 3. Objectives

- Preprocess the dataset for training a machine learning model.

- Train a Random Forest model to classify club participation.

- Evaluate model performance using accuracy.

- Predict user input for live inference.

## 4. Methodology

Data Collection: Dataset downloaded via Google Drive using gdown.

Data Preprocessing:

- Column names cleaned.

- Categorical values encoded using LabelEncoder.

Model Building:

- Random Forest Classifier used for prediction.

- 80% training, 20% testing split.

Evaluation:

- Model accuracy calculated on test set.

- User input taken for real-time prediction.

## 5. Data Preprocessing

- Leading/trailing spaces removed from column names.

- Categorical columns encoded.

- Data split for training and testing.

## 6. Model Implementation

Random Forest Classifier is used due to its robustness and ability to handle mixed data types. The model is trained on the preprocessed dataset and used to predict club participation.

## 7. Evaluation Metrics

- Accuracy: Overall correctness of the model.

- Real-time prediction: Testing user input against trained model.

## 8. Results and Analysis

The model provided an accuracy score of approximately [Insert Accuracy]. The interactive prediction system allows users to input student details and receive predicted club affiliation.

## 9. Conclusion

The project demonstrates the potential of using machine learning in academic settings to predict student behavior. With further enhancements and additional data, the model could be expanded to support broader student analytics.

## 10. References

- scikit-learn documentation

- pandas documentation

- gdown library

- Google Drive Dataset Link:
https://drive.google.com/uc?id=1d_stwG7PnM509FwEjDcFcvZml4RdYzYN

## 11. Code

```
# Importing necessary libraries
import pandas as pd
import gdown
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# URL of the dataset to download
url = "https://drive.google.com/uc?id=1d_stwG7PnM509FwEjDcFcvZml4RdYzYN"
output = "club_participation.csv"
# Download the dataset
gdown.download(url, output, quiet=False)

# Load the dataset into a DataFrame
df = pd.read_csv("club_participation.csv")

# Strip any leading/trailing spaces from the column names
df.columns = df.columns.str.strip()

# Print the available column names in the dataset
print("Available columns:", df.columns.tolist())

# Define the target column for prediction
```

```
target_column = 'club_participation'

# Check if the target column exists in the dataset, raise an error if not
if target_column not in df.columns:
    raise ValueError(f"Target column '{target_column}' not found in dataset.")

# Create a dictionary to store label encoders for categorical columns
label_encoders = {}
for column in df.columns:
    # Check if the column has categorical data (object type)
    if df[column].dtype == 'object':
        # Initialize LabelEncoder for this column
        le = LabelEncoder()
        # Convert the categorical values to numeric values
        df[column] = le.fit_transform(df[column])
        # Store the encoder in the dictionary
        label_encoders[column] = le

# Separate the features (X) and target variable (y)
X = df.drop(target_column, axis=1)
y = df[target_column]

# 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)

# Initialize and train a RandomForestClassifier model
```

```
model = RandomForestClassifier()
model.fit(X_train, y_train)

# Make predictions using the trained model on the test data
y_pred = model.predict(X_test)

# Calculate and print the model's accuracy on the test data
print("Model Accuracy:", round(accuracy_score(y_test, y_pred) * 100, 2), "%")

# Request user input to predict the club participation of a student
print("\nEnter student details for prediction")
user_data = {}

# Loop through all columns in the feature set (X)
for col in X.columns:
    # If the column is categorical, ask the user to choose from available options
    if col in label_encoders:
        options = list(label_encoders[col].classes_)
        print(f"\nOptions for {col}: {options}")
        val = input(f"Enter {col}: ")
        # Ensure the input is valid, prompt again if not
        while val not in options:
            print("Invalid input! Please choose from:", options)
            val = input(f"Enter {col}: ")
        # Encode the input value into a numeric format
        encoded_val = label_encoders[col].transform([val])[0]
```

```
        user_data[col] = encoded_val
    else:
        # For numeric columns, directly ask for the value
        val = float(input(f"Enter {col} (numeric): "))
        user_data[col] = val

# Create a DataFrame with the user's input data
input_df = pd.DataFrame([user_data])

# Make a prediction based on the user's input
prediction = model.predict(input_df)[0]

# If the target column is categorical, convert the prediction back to its original label
if target_column in label_encoders:
    result = label_encoders[target_column].inverse_transform([prediction])[0]
else:
    result = prediction

# Output the predicted result
print(f"\nPredicted Club Participation: {result}")
```