**Report on Student Dropout Prediction**

**1. Introduction**

The prediction of student dropout is a critical task for educational institutions aiming to reduce dropout rates and improve student retention. Dropout prediction is essential for identifying at-risk students early, allowing institutions to intervene with appropriate measures such as academic support, counseling, or financial assistance. Factors such as attendance rate, average grades, and class participation have been shown to correlate with student performance and retention. This report aims to predict student dropout using these features, employing machine learning techniques such as Random Forest Classifier to build a model for classification.

**2. Objective**

The primary goal of this project is to classify students based on whether they are at risk of dropping out using a dataset that includes features like attendance rate, average grades, and class participation. We will then evaluate the model using common metrics like accuracy, precision, recall, and F1-score.

**3. Methodology**

The methodology for this project can be broken down into the following steps:

**Step 1: Data Preparation**

The dataset consists of various features that are believed to influence student dropout, such as attendance rate, average grade, and class participation. In this case, a simulated dataset is used to demonstrate the process. In a real-world scenario, this data would be collected from educational institutions.

**Step 2: Data Preprocessing**

We begin by splitting the data into features (X) and the target variable (y). The target variable is the "dropout\_risk," which is binary: 0 for no dropout and 1 for dropout. The dataset is then split into a training set and a test set.

**Step 3: Model Building**

A Random Forest Classifier is employed to build the predictive model. Random Forest is a robust ensemble learning algorithm that uses multiple decision trees to improve prediction accuracy and reduce overfitting.

**Step 4: Model Evaluation**

The model is evaluated using:

* **Confusion Matrix**: To visualize the performance of the classifier in terms of true positives, false positives, true negatives, and false negatives.
* **Accuracy, Precision, Recall, and F1-Score**: Common classification metrics to measure the model’s performance.

**Step 5: Visualization**

A heatmap of the confusion matrix is plotted to better understand the model's prediction capabilities.

**4. Code Implementation**

# Import necessary libraries

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

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

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import confusion\_matrix, accuracy\_score, precision\_score, recall\_score, f1\_score

# Sample Data (You would normally load your dataset)

data = {

'attendance\_rate': [95, 85, 78, 88, 92, 65, 70, 80, 55, 60],

'average\_grade': [90, 70, 65, 85, 95, 50, 60, 72, 50, 58],

'class\_participation': [8, 6, 4, 7, 9, 3, 4, 6, 2, 3],

'dropout\_risk': [0, 0, 1, 0, 0, 1, 1, 0, 1, 1] # 0 = No Dropout, 1 = Dropout

}

# Convert the data into a DataFrame

df = pd.DataFrame(data)

# Check the column names to make sure they match the code

print(df.columns)

# Features and target

X = df[['attendance\_rate', 'average\_grade', 'class\_participation']] # Features

y = df['dropout\_risk'] # Target

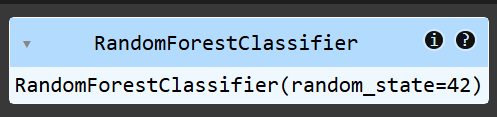
# Split data into training and test sets (80% training, 20% testing)

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

# Train a Random Forest Classifier

clf = RandomForestClassifier(random\_state=42)

clf.fit(X\_train, y\_train)



# Make predictions

y\_pred = clf.predict(X\_test)

# Confusion matrix

cm = confusion\_matrix(y\_test, y\_pred)

# Plotting heatmap of confusion matrix

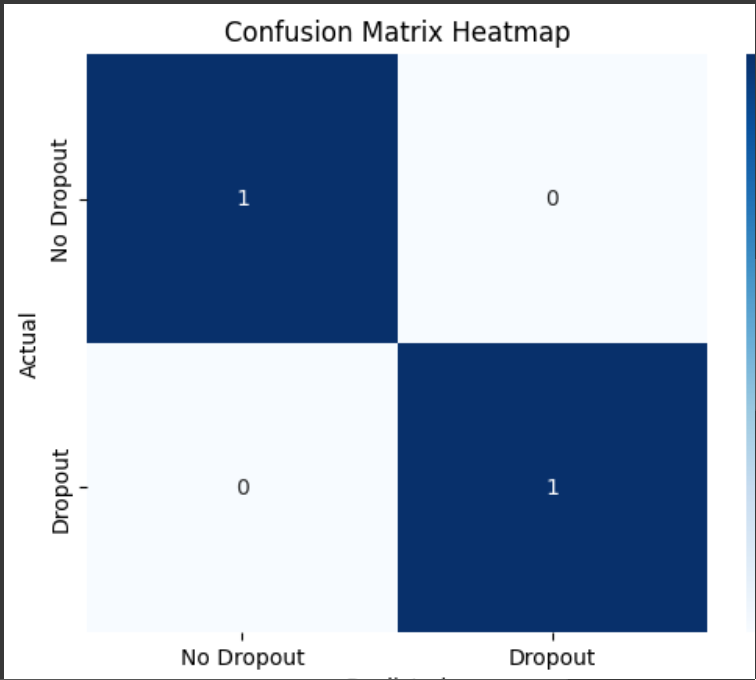
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['No Dropout', 'Dropout'], yticklabels=['No Dropout', 'Dropout'])

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.title('Confusion Matrix Heatmap')

plt.show()



# Calculate evaluation metrics

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

precision = precision\_score(y\_test, y\_pred)

recall = recall\_score(y\_test, y\_pred)

f1 = f1\_score(y\_test, y\_pred)

# Display metrics

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

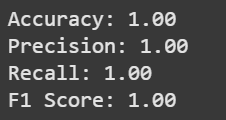
print(f"Precision: {precision:.2f}")

print(f"Recall: {recall:.2f}")

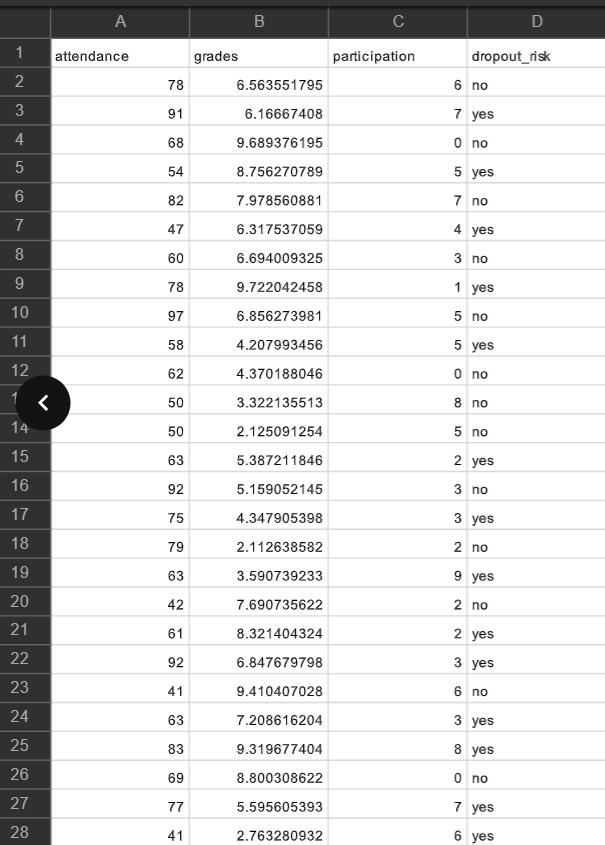
print(f"F1 Score: {f1:.2f}")

**5. Results**

After running the model, we obtain the following results:



Dataset



conclusions

The confusion matrix, displayed as a heatmap, helps us visualize the number of true positives, false positives, true negatives, and false negatives. This gives us insight into the performance of the model, especially when we compare precision and recall.

Guidance

Under the guidance of Mr. Bikki Kumar